COMMUNICATIONS IN INFORMATION AND SYSTEMS
Vol. 4, No. 3, pp. 235-252, 2004

003

OPTIMIZATION BASED FLOW CONTROL WITH IMPROVED PERFORMANCE*

HAO ZHANG†, ZHONG-PING JIANG†, YI FAN†, AND S. PANWAR†

Abstract. Optimization based flow control has been proposed in [2] to improve the network performance with congested bottle links. This rate-based technique has advantages over traditional window based heuristic algorithms in that the optimal performance in terms of maximal aggregate utility function can be achieved when each source adaptively adjusts its data rate. Several decentralized optimization algorithms have been applied to the flow control. However, one of most important features of these algorithms: the relation between the convergence speed and network parameters is not well studied, experimentally or theoretically. The contribution of this paper is two-fold. The first contribution is that we propose Aitken-extrapolation to accelerate the convergence process. Secondly, we compare the convergence speed of various algorithms by theoretic analysis and simulations. Based on the results, the network parameters can be appropriately chosen to improve network performance.

Key words: Optimization, Communication Networks, Decentralization, Flow Control, Gradient Projection Method, Aitken-Extrapolation.

1. Introduction. Effective rate control is required in order to control network flow and avoid congestion. A recent approach to flow control is based on optimization methods, e.g., [2∼14]. In optimization-based flow control, each user is associated with a utility function, which suggests the portion of sharing of bandwidth with other users. A constraint is that the aggregate rate in one specific link should be within the link capacity [1]. The rate control objective is to achieve traffic rates that maximize the sum of the user utilities. A decentralized algorithm based on the dual model was proposed by S. Low and his coworkers ina nice paper [2] (also see [3]). They divided the primal problem into two sub-problems. One is an optimization problem of choosing the source rates to maximize the total benefit—-subtract the total bandwidth cost from the total utility function. This part is computed under a given price. On the contrary, the dual problem is to choose prices that can achieve the minimum of the dual objective function. This dual method has led the primal problem to a decentralized solution. A gradient projection method has been proposed in order to minimize the dual objective function. Two models are discussed: one is a synchronous distributed model; the other is an asynchronous distributed model. The gradient projection method was proved to converge under these two circumstances, but the speed of convergence is very slow. In [3], a Newton-like method was proposed which uses the diagonal of Hessian matrix to form a scaled algorithm. From experiments, we

*This work has been supported in part by NSF Grants ANI-0081527, ECS-0093176 and OISE-0408925.

†Department of Electrical and Computer Engineering, Polytechnic University, Brooklyn, NY 11201, U.S.A. E-mail: zjiang@control.poly.edu, hzhang08@utopia.poly.edu

can see that it has a faster convergence speed than the unscaled algorithm. But the authors of [2] and [3] have not given any theoretical analysis about the convergence speed of these algorithms. Motivated by this observation and the fact that the speed of convergence is important in network flow control, we make a further step toward the problem of optimization-based flow control.

Firstly, we would like to introduce a modified Aitken-Extrapolation algorithm for flow control in telecommunications networks. Supported by theoretical analysis and computer simulations, it is shown that the proposed Aitken-Extrapolation algorithm yields faster convergence than the previous gradient projection algorithm in [2] and improved performance for various step sizes than the Newton-like algorithm in [3]. In particular, it is shown that the gradient projection algorithm is of the first order and with geometric convergence speed if the step size is properly chosen. In addition, the Newton-like algorithm is a derivative from convergence acceleration methodology and, like the Aitken-extrapolation method, is superlinearly convergent.

The rest of the paper is presented as follows. Section 2 describes the (convex) optimization problem for a distributed network. Section 3 recalls the gradient projection method and a Newton-like algorithm proposed in the recent literature, and presents our modified Aitken-Extrapolation algorithm. Our main theorems on the theoretical analysis and rate of convergence of these algorithms are also stated in Section 3. Computer simulations shown in Section 4 validate the effectiveness of our modified Aitken-Extrapolation algorithm. Some brief concluding remarks are contained in Section 5.

2. The Optimization Problem. Optimization of communication networks has been studied in various contexts through the consideration of different objective functions and constraints. In this section we will use a popular optimization problem model, which leads to the techniques in this paper.

A. Notation

Suppose a network is composed of a set $\mathbf{L}$=$\{1,\ldots,\mathrm{L}\}$ of unidirectional links of capacity $c_l, l \in \mathbf{L}$. A set of sources $\mathbf{S}$=$\{1,\ldots,\mathrm{S}\}$ are sharing these links. We introduce the following notation for this system:

$L(s)$:a subset of $\mathbf{L}$, which is a set of links which form the path for source s

U_s :a utility function for source s which is positive, strictly concave and is of class C^2

M_s :the maximum possible transmission rate for source s

m_s :the minimum possible transmission rate for source s

p_l :price of link l

$$p: \quad [p_1, p_2, ..., p_L]^T$$

x_s :transmission rate of source s

I_s =[m_s,M_s]: the range in which source rate x_s must lie

$S(l) = \{s \in \mathbf{S} | l \in L(s)\}$: the set of sources that use link l

B. Problem Statement

The objective function

$$\text{P} : \max_{x_s \in I_s} \quad \sum_s U_s(x_s) \tag{1}$$

subject to

$$\sum_{s \in S(l)} x_s \leq c_l, l = 1, ..., L. \tag{2}$$

The constraint (2) means the aggregate source rate at any link l should not exceed the capacity. Since the objective function $U_s(x_s)$ is strictly concave and hence, a unique feasible optimal solution exists and should be a global solution.

Solving the primal problem requires coordination among all sources and it is hard to implement a centralized solution in real networks. This observation leads the authors of [2], among others, to consider a dual approach.

C. Dual Model

The Lagrangian of the constrained problem is defined as

$$\begin{aligned} L(x,p) &= \sum_s U_s(x_s) - \sum_l p_l(\sum_{s \in S(l)} x_s - c_l) \\ &= \sum_s (U_s(x_s) - x_s \sum_{l \in L(s)} p_l) + \sum_l p_l c_l. \end{aligned} \tag{3}$$

Necessary conditions of the optimization problem are obtained as follows:

$$\begin{aligned} &(a) \qquad U_s'(x_s) = p^s \\ &(b) \qquad p_l(x^l - c_l) = 0 \\ &(c) \qquad p_l \geq 0 \\ &(d) \qquad x_s \in I_s[m_s, M_s] \end{aligned}$$

where

$$p^s = \sum_{l \in L(s)} p_l, \tag{4}$$

and

$$x^l = \sum_{s \in S(l)} x_s. \tag{5}$$

$U_s'(x_s)$denotes the first derivative of U. We can see it is hard to implement distributed algorithms under the primal model. So a practical dual method is proposed in [2].

$$D(p) = \max_{x_s \in I_s} L(x,p) = \sum_s B_s(p^s) + \sum_l p_l c_l \tag{6}$$

where

$$B_s(p^s) = \max_{x_s \in I_s} U_s(x_s) - x_s p^s, \tag{7}$$

the dual problem is:

$$D: \quad \min_{p \geq 0} \quad D(p). \tag{8}$$

Dual model is actually coordination between different users (sources) and routers (links).

The purpose of the source algorithm is try to achieve maximum profit for each user and the link algorithm is for the social welfare by adjusting the link price p_l. So the dual problem
can be solved by decentralized algorithm.

3. Synchronous Distributed Algorithms.

A. Discussion on the gradient projection method

Based on this dual method, we will discuss some synchronous distributed algorithms. In [2], gradient projection algorithms are used to solve the synchronous problem. The basic algorithm is divided into two parts: one is a link algorithm, which adjusts prices in opposite direction to the gradient $\nabla D(p)$:

$$p_l(t+1) = [p_l(t) - \gamma \nabla D(p(t))]^+. \tag{9}$$

We denote $[a]^+ = \max\{0, a\}$, $\nabla D(p)$is a $L \times 1$ vector, the lth element of which is $\frac{\partial D}{\partial p_l}(p) = c_l - x^l(p)$, so we have the dual algorithm which is developed as follows:

Algorithm A1: gradient projection algorithms

Link l's algorithm:

At time t=1,2..., link l:

1. Receives rates $x_s(t)$ from all sources $s \in S(l)$that share link l.

2. Computes the aggregate rates at the link l: $x^l(t) = \sum_{s \in S(l)} x_s(t)$, then calculates a new price

$$p_l(t+1) = [p_l(t) + \gamma(x^l(t) - c_l)]^+. \tag{10}$$

3. Communicates the new price to all sources that use link l.

Source s's algorithm:

At time t=1,2..., source s:

1. Receives the prices $p_l(t)$from each link the source shares and then get the sum of the prices

$$p^s(t) = \sum\nolimits_{l \in L(s)} p_l(t). \tag{11}$$

2. From the necessary condition (a), the new transmission rates should be computed as

$$x_s(t+1) = \min\{M_s, \max\{m_s, (U_s')^{-1}(p^s(t))\}\}. \tag{12}$$

$(U_s')^{-1}$denotes the inverse function of U_s'.

3. Communicates new rates $x_s(t+1)$ to links that are used by source s.

This algorithm is implemented in the REM algorithm discussed in [4]. But we can see from simulations that the convergence of gradient projection algorithm is slow. Instead of the gradient projection algorithm as used in [2][13], a simplified Hessian matrix $\nabla^2(D)$ is employed to improve the convergence speed [3]. This Newton-like algorithm is in fact a scaled algorithm that neglects the off-diagonal elements in the Hessian matrix, which is practical for implementation of decentralized algorithms. More specifically, in [3], the derivative $[\nabla^2 D(p(t))]_{ll}$ is approximated by $-\frac{x^l(t)-x^l(t-1)}{p_l(t)-p_l(t-1)}$. The source's algorithm is the same as the gradient projection algorithm; the link algorithm is revised as:

$$p_l(t+1) \approx [p_l(t) + \gamma H_{ll}^{-1}(x^l(t) - c_l)]^+ \tag{13}$$

where $H_{ll} = \max\{\varepsilon, -\frac{x^l(t)-x^l(t-1)}{p_l(t)-p_l(t-1)}\}$, and ε is a positive parameter used to make $H = diag(H_{ll})$ positive definite. This parameter should be chosen carefully, because in case some diagonal elements in H are non-positive, ε is activated as the scaled elements. This method was justified by simulation in [3] to converge faster than gradient projection method. As said, this Newton-like algorithm is obtained from neglecting the off-diagonal elements and thus it is not necessary to analyze it in the same way as the general Newton algorithm. So we will derive this method from another point of view—-convergence acceleration. Using this way, it is easier and more natural for us to understand this scaled algorithm.

B. Aitken Extrapolation [16]

The link algorithm can be thought of as a root finding process. Define $f_l(p) = c_l - x^l(p)$ and recall $p = [p_1, p_2, ..., p_L]^T$. From the necessary condition (b), the optimization procedure is to find p to make $p_l f_l(p) = 0$. This formula can be interpreted as follows: when $f_l(p) > 0$, $p_l = 0$; and when $f_l(p) = 0$, p_l has a non-negative value. The iterative formula that satisfies the above requirements is described as $p_l(t+1) = [p_l(t) - \gamma f_l(p)]^+$ in (10).

Defining $\phi_l(p) = [p_l - \gamma f_l(p)]^+$, we can solve a fixed-point problem of the form $p_l = \phi_l(p)$ (or $p = \phi(p)$). Now we consider a sequence of p_l:

$$p_l(t) = \phi_l(p(t-1)), \tag{14}$$

$$\bar{p}_l(t+1) = \phi_l(p(t)), \tag{15}$$

$p_l(t)$is the actual price of link l at time t computed by the above gradient projection algorithm. $\bar{p}_l(t+1)$is an intermediate price of link l at time $t+1$ computed by the

gradient projection algorithm, but we will change it to $p_l(t+1)$— an actual price at time $t+1$.

Denote (x^*, p^*) as any pair of primal-dual optimal solution to the dual problem. As noticed in [2,3,17], p^* is not unique although x^* is unique because of the convexity assumptions made in the optimization problem. Then $p^* = \phi(p^*)$, $x^* = \phi(x^*)$. p_l^* and x_s^* are the lth and sth element of p^* and x^*, respectively.

From Mean Value Theorem or Taylor expansions, we have

$$p_l(t) - p_l^* = \phi_l(p(t-1)) - \phi_l(p^*) = \phi_l'(\theta_1)(p_l(t-1) - p_l^*), \tag{16}$$

$$\bar{p}_l(t+1) - p_l^* = \phi_l(p(t)) - \phi_l(p^*) = \phi_l'(\theta_2)(p_l(t) - p_l^*). \tag{17}$$

Whenever $p(+\infty) = p^*$ (we assume this gradient projection method is convergent), we can take

$$\theta_1 = p(t_1), t-1 \le t_1 \le +\infty$$

and

$$\theta_2 = p(t_2), t \le t_2 \le +\infty.$$

According to Aitken's formula, if p is near the optimal point p^*, $\phi_l^{'}(\theta_1)$ is almost equal to $\phi_l^{'}(\theta_2)$, i.e. $\phi_l^{'}(\theta_1) \approx \phi_l^{'}(\theta_2)$. So eliminate $\phi_l^{'}(\theta_1), \phi_l^{'}(\theta_2)$ from the above two equations to arrive at:

$$\frac{p_l(t) - p_l^*}{\bar{p}_l(t+1) - p_l^*} \approx \frac{p_l(t-1) - p_l^*}{p_l(t) - p_l^*}. \tag{18}$$

Then we obtain

$$p_l^* \approx \bar{p}_l(t+1) - \frac{(\bar{p}_l(t+1) - p_l(t))^2}{\bar{p}_l(t+1) - 2p_l(t) + p_l(t-1)}. \tag{19}$$

So we have the following Aitken's iteration formula based on the dual model:

Link l's algorithm at time $t-1, t = 1, 3, ..., 2n+1, ...n \in Z, n \ge 0$:

1. Receive rates $x_s(t-1)$ from all sources $s \in S(l)$that share link l.

2. Compute the aggregate rates at the link l $\quad x^l(t-1) = \sum_{s \in S(l)} x_s(t-1)$, then calculate a new price

$$p_l(t) = [p_l(t-1) + \gamma(x^l(t-1) - c_l)]^+. \tag{20}$$

3.Communicate new price $p_l(t)$to all sources that use link l.

Source s'**s** algorithm at time $t-1, t = 1, 3, ..., 2n+1, ...n \in Z, n \ge 0$:

1. Receive the prices $p_l(t)$from each link the source shares and them get the sum of the prices $p^s(t-1) = \sum_{l \in L(s)} p_l(t-1)$.

2. From the necessary condition (a), the new transmission rates should be computed as

$$x_s(t) = \min\{M_s, \max\{m_s, (U_s')^{-1}(p^s(t-1))\}\}. \tag{21}$$

3. Communicate new rates $x_s(t)$to links that are used by source s.

Link l's algorithm at time at time $t, t = 1, 3, ..., 2n+1, ...n \in Z, n \geq 0$:

4. Receive rates $x_s(t)$ from all sources $s \in S(l)$that share link l.

5. Compute the aggregate rates at the link $l \quad x^l(t) = \sum_{s \in S(l)} x_s(t)$, then calculate a new price

$$\bar{p}_l(t+1) = [p_l(t) + \gamma(x^l(t) - c_l)]^+, \tag{22}$$

$$p_l(t+1) = [\bar{p}_l(t+1) - \frac{(\bar{p}_l(t+1) - p_l(t))^2}{\bar{p}_l(t+1) - 2p_l(t) + p_l(t-1)}]^+. \tag{23}$$

6. Communicate new price $p_l(t)$ to all sources that use link l.

Source s's algorithm at time $t, t = 1, 3, ..., 2n+1, ...n \in Z, n \geq 0$:

4. Receive the prices $p_l(t)$ from each link the source shares and them get the sum of the prices $p^s(t) = \sum_{l \in L(s)} p_l(t)$.

5. From the necessary condition (a), the new transmission rates should be computed as

$$x_s(t+1) = \min\{M_s, \max\{m_s, (U_s')^{-1}(p^s(t))\}\}. \tag{24}$$

6. Communicate new rates $x_s(t+1)$to links that are used by source s.

It can be seen from the computation procedure that at discrete time $2n+1$ ($n \geq 0$), we only implement the usual gradient projection algorithm; while at discrete time $2n$ ($n \geq 1$) we use not only gradient projection algorithm again, but the combination of the former two results. This methodology is very effective in improving convergence speed and reducing the fluctuation as well, as confirmed by our simulations in Section 4.

C. Newton-Like Algorithm

Steven Low and his coworkers [3] proposed a Newton-like algorithm for flow control, which was illustrated to be superior to the gradient projection algorithm by experiments.

The link l's algorithm is scaled by H^{-1}, an approximation of Hessian matrix $(\nabla^2 D)^{-1}$ as follows.

$$p(t+1) = [p(t) - \gamma H(p(t))^{-1} \nabla D(p(t))]^+, \tag{25}$$

which can be written in a separate way:

$$\begin{aligned} p_l(t+1) &= [p_l(t) - \gamma[H_{ll}(p(t))]^{-1} \frac{\partial D(p)}{\partial p_l}|_{p(t)}]^+ \\ &= [p_l(t) + \gamma[H_{ll}(p(t))]^{-1}(x^l(t) - c_l)]^+ \\ &= [p_l(t) - \gamma[\frac{\partial x^l(p)}{\partial p_l}|_{p(t)}]^{-1}(x^l(t) - c_l)]^+. \end{aligned} \tag{26}$$

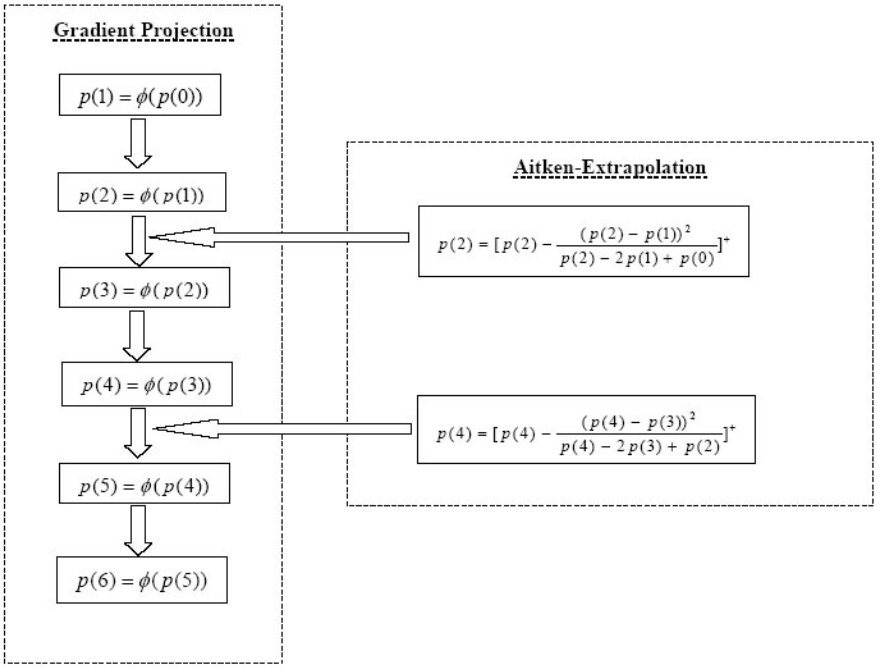

FIG. 1. *Flow Chart of Gradient Projection Algorithm and Aitken-Extrapolation.*

Note that in order to form a decentralized method, the off-diagonal elements of Hessian matrix $\nabla^2 D$ were arbitrarily set to zero, the diagonal elements $\frac{\partial x^l(p)}{\partial p_l}|_{p(t)}$ $(l = 1, ..., L)$ are first-order derivatives being approximated by $-\frac{x^l(t)-x^l(t-1)}{p_l(t)-p_l(t-1)}$. Now we will show that the Newton-like algorithm in (26) can be deduced from (17). The short proof we provide below sheds light on the connection between the gradient algorithm and the Newton-like algorithm.

To keep the technicality to a minimum, we consider the simplified case without projection $[\ ]^+$, that guarantees the differentiability of $\phi(p)$. Then, it follows from (17) that

$$\begin{aligned}(\phi_l^{'}(\theta_2) - 1)p_l^* &= \phi_l^{'}(\theta_2)p_l(t) - \bar{p}_l(t+1) \\ &= \phi_l^{'}(\theta_2)p_l(t) - [p_l(t) + \gamma(x^l - c_l)] \\ &= (\phi_l^{'}(\theta_2) - 1)p_l(t) - \gamma(x^l - c_l).\end{aligned} \tag{27}$$

The range of θ_2 is $\theta_2 = p(t_2), t \le t_2 \le +\infty$, here we choose $t_2 = t$, then we have

$$\begin{aligned}\phi_l^{'}(\theta_2) &= \frac{\partial[p_l + \gamma(x^l - c_l)]}{\partial p_l}|_{p(t)} \\ &= 1 + \gamma\frac{\partial x^l(p)}{\partial p_l}|_{p(t)},\end{aligned} \tag{28}$$

$$p_l^* = p_l(t) - (\frac{\partial x^l(p)}{\partial p_l}|_{p(t)})^{-1}(x^l(t) - c_l). \tag{29}$$

So the result is

$$p_l^* = p_l(t) - (\frac{\partial x^l(p)}{\partial p_l}|_{p(t)})^{-1}(x^l(t) - c_l).$$

Usually, we can still add the step size γ. So we get the improved iterative algorithm

$$p_l(t+1) = [p_l(t) - \gamma(\frac{\partial x^l(p)}{\partial p_l}|_{p(t)})^{-1}(x^l(t) - c_l)]^+. \tag{30}$$

This is the Newton-like algorithm proposed by Steven Low and his coworkers, which is the same as in (26) and can be considered as a convergence acceleration methodology of the original gradient projection algorithm. Another point of view is to consider this algorithm as a method of false position, which will be discussed in the Appendix.

D. Speed of Convergence

The convergence of the Aitken-Extrapolation algorithm in scalar form is proved in [16]. Here we deal with the case where the mapping function ϕ is vector-valued, leading to some technical complication.

THEOREM I: *The gradient projection algorithm reviewed in Section 3.A is convergent with linear convergence speed if the stepsize is appropriately chosen.*

THEOREM II: *The Newton-like algorithm reviewed in Section 3.C is superlinearly convergent if* $\gamma = 1$. *It is approximately linearly convergent if* $0 < \gamma < 1$ and diverges if $\gamma >> 1$.

THEOREM III: *The Aitken-Extrapolation algorithm proposed in Section 3.B is superlinearly convergent.*

The proofs of these main results will be given in the Appendix.

REMARK 1: As said previously, the gradient projection algorithm was first proposed in [2] for network flow control but the authors did not analyze the rate of convergence. Theorem I complements the theoretical result of [2] by providing explicit sufficient conditions for geometric convergence.

4. Simulation Results. We have based our simulation on the same model as in [3]. In Fig.2, five connections $S_i - D_i$ (i=1,2...,5) with source S_i and destination D_i share four links. Connection $S_1 - D_1$ spanned links 1,2,3,4; $S_2 - D_2$ spanned link 1; $S_3 - D_3$ spanned link 2; $S_4 - D_4$ spanned link 3; $S_5 - D_5$ spanned link 4. All links were identical with capacity equal to 220 packets per second measuring interval. Source S_1 transmitted data from time 0s to time 300s. The start times of the other sources are staggered with 40s interval. Sources S_2, S_3, S_4, S_5 remain active until 120s, 160s, 200s, 240s, respectively. The utility functions of the sources were set to $\omega_s \log(1 + x_s$, with ω_s equal to 4×10^4 for source S_1 and 1×10^4 for source S_2,S_3, S_4, S_5 respectively. The target bandwidth (c_l) was set at 200 packets per 1s measuring

interval, while the actual bandwidth is set to 220 packets per second, leading to zero equilibrium buffer occupancy.

The initial conditions of this algorithm are set as follows:

$\gamma = 0.15$for gradient projection algorithm in order to achieve convergence; $\gamma = 1$ or $\gamma = 0.5$ for both Newton-like and Aitken-Extrapolation algorithms.

$$\begin{aligned} x_{s\in S} &= 0, s = 1, ..., S, \\ p_{l\in L} &= 0, l = 1, ..., L, \\ m_s &= 0, \\ M_s &= 300 \text{ packets/ sec}. \end{aligned}$$

In Fig.4 and Fig.5, we set $\gamma = 1$ for Newton-Like and Aitken-Extrapolation algorithms. Their convergence speeds are faster than the gradient projection algorithm, and the sizes of their buffer occupancy are much smaller (cf. Fig. 6). However, the value of step size γ affects performance of Newton-Like algorithm enormously. In Fig. 7, when the step size $\gamma = 0.5$ is used, Newton-Like algorithm leads to almost the same buffer occupancy as the gradient projection algorithm. Simulation results have shown that Aitken-Extrapolation algorithm yield smaller buffer occupancy than Newton-Like algorithm, especially under smaller step sizes.

5. Conclusion. The faster convergence rate implies less overloading and hence much less buffer requirement at the links [3]. In this paper, we have examined the application of a practical Aitken-Extrapolation algorithm in network flow control. Its superiority over gradient projection algorithms is illustrated by theoretical analysis. We also show by simulations that Aitken-Extrapolation is less sensitive to step sizes than the Newton-like method, although both of which are of superlinear convergence speed. The convergence speed of Newton-like algorithm is shown to be nearly 1.618 by method of false position.

For the proposed Aitken-Extrapolation algorithm, we use three values at time t-1, t, t+1 to extrapolate a new value that is closer to the optimal one. Employing this methodology, it is not difficult to get a more precise algorithm based on the extrapolation of more than three points. Of course more memory is needed to restore the old values (prices and sources rates) and more computation task is required. Another advantage of the Aitken-Extrapolation algorithm is that, *even if the original projection algorithm is not convergent but of the first order, the Aitken-Extrapolation algorithm is still convergent.* We are currently investigating *asynchronous* distributed algorithms with this method, and will report on new findings separately.

Appendix – Proofs.

Proof of Theorem I:

As said previously, the authors of [2] have already proved that any limit point

(x^*, p^*) of the sequence $(x(t), p(t))$ is primal-dual optimal. In the sequel, we focus on the rate of convergence and provide conditions for the step size.

Take a sufficiently small $\varepsilon \in R^L$. From (14), we have

$$\begin{aligned}\phi_l(p^*+\varepsilon) &= [p_l^* + \varepsilon_l + \gamma(x^l(p^*+\varepsilon) - c_l)]^+ \\ &= \Big[p_l^* + \varepsilon_l + \gamma\big\{x^l(p^*) + \nabla x^l(p^*)^T\varepsilon \\ &\quad + \frac{1}{2}\varepsilon^T\nabla^2 x^l(p^*+\theta\varepsilon)\varepsilon - c_l\big\}\Big]^+, \quad 0<\theta<1.\end{aligned} \tag{31}$$

If $x^l(p^*) < c_l$, then using the necessary condition (b) we have $p_l^* = 0$, and then from (31) $\phi_l(p^*+\varepsilon) = 0$. If $x^l(p^*) = c_l$, then $\phi_l(p^*+\varepsilon) = [p_l^* + \varepsilon_l + \gamma E_l^T\varepsilon + \varepsilon^T V_l^T\varepsilon]^+$. Here $E_l = \nabla x^l(p^*)$, $V_l = \nabla^2 x^l(p^*+\theta\varepsilon)/2$.

So the gradient projection algorithm is of the first order in the sense that

$$\frac{\left\|\gamma E_l^T\varepsilon + \varepsilon_l\right\|}{\|\varepsilon\|} \le k, l = 1, 2, ..., L,$$

where k is a positive constant.

Set

$$E = [E_1 E_2 ... E_L]^T.$$

Here we can use natural norm, so the former expression can be rewritten as:

$$\frac{\left\|\gamma E^T\varepsilon + \varepsilon\right\|_\infty}{\|\varepsilon\|_\infty} \le k. \tag{32}$$

When $k < 1$, the algorithm is geometrically convergent (or more precisely, linearly convergent). Note that

$$E = R\frac{\partial x(p^*)}{\partial p} = -RW(p^*)R^T \tag{33}$$

where$R$ is a routing matrix whose (l,s)th entry is $R_{ls} = 1$ if $l \in L(s)$ (or $s \in S(l)$), and 0 otherwise [2]. We have the similar procedure as that in [2]:

$W(p) = Diag(\beta_s(p), s \in S)$is a $S \times S$ diagonal matrix with diagonal elements

$$\beta_s(p) = \begin{cases} \frac{1}{-U_s''(x_s(p))} & \text{if} \quad U_s'(M_s) \le p^s \le U_s'(m_s), \\ 0 & \text{otherwise.} \end{cases} \tag{34}$$

Suppose $0 \le \beta_s(p) \le \bar{\alpha}_s(p)$. Define $\bar{L} := \max_{s\in S}|L(s)|$, $\bar{S} := \max_{l\in L}|S(l)|$, $\bar{\alpha} := \max\{\bar{a}_s(p^*), s \in S\}$. So $\|R\|_\infty = \bar{S}$, $\left\|R^T\right\|_\infty = \bar{L}$, $\|W(p^*)\|_\infty = \bar{\alpha}$. From (32), we have

$$\begin{aligned}\frac{\left\|\varepsilon + \gamma E^T\varepsilon\right\|_\infty}{\|\varepsilon\|_\infty} = \frac{\left\|(I+\gamma E^T)\varepsilon\right\|_\infty}{\|\varepsilon\|_\infty} &= \frac{\left\|(I-\gamma RW(p^*)R^T)\varepsilon\right\|_\infty}{\|\varepsilon\|_\infty} \\ &\le \left\|I - \gamma RW(p^*)R^T\right\|_\infty.\end{aligned} \tag{35}$$

Since $RW(p^*)R^T$ is symmetric and positive definite, we have

$$\left\|I - \gamma RW(p^*)R^T\right\|_\infty = |\gamma\lambda_m - 1|$$

where $\lambda_m > 0$ is an eigenvalue of $RW(p^*)R^T$with maximum eigenvalue not large than $\bar{S}\bar{L}\bar{\alpha}$. Given that $-1 < \gamma\bar{S}\bar{L}\bar{\alpha} - 1 < 1$, i.e., $\gamma < 2/\bar{S}\bar{L}\bar{\alpha}$, we obtain both $\left\|I - \gamma RW(p^*)R^T\right\|_\infty < 1$and $k¡1$. Finally, the algorithm is geometrically convergent. **Q.E.D.**

Proof of Theorem II:

Note that the convergence of the Newton-like has been proved in [3,17]. We devote ourselves to the rate analysis below.

From (13) we see that there are two major differences between Newton-like algorithm and Newton algorithm. One is that the scaled matrix in Newton-like algorithm retains only the diagonal terms and has zero off-diagonal terms. The second difference is that the diagonal terms are approximated by finite differences [3]. From another point of view, we can consider this algorithm as a method of false position. In [16], convergence speed of false position has been analyzed for one-dimensional variables. But the algorithm used in [3] is in a vector form, whose convergence analysis is shown below:

Recall that $f_l(p) = c_l - x^l(p)$, the iterative algorithm in (13) can be written as

$$p_l^{v+1} = p_l^v - \gamma f_l(p^v)\frac{p_l^v - p_l^{v-1}}{f_l(p^v) - f_l(p^{v-1})}. \tag{36}$$

Here we neglect the parameter ε and projection function for simplicity. We also assume that $f_l(p^*) = 0$ for all l=1,2,... L.

(36) can be rewritten as

$$\begin{aligned} p_l^* - p_l^{v+1} &= p_l^* - p_l^v + \gamma f_l(p^v)\frac{p_l^v - p_l^{v-1}}{f_l(p^v) - f_l(p^{v-1})} \\ &= \gamma(p_l^* - p_l^v)\frac{f_l[p^{v-1}, p^v] - f_l[p^v, p^*]}{f_l[p^{v-1}, p^v]} + (1-\gamma)(p_l^* - p_l^v) \end{aligned} \tag{37}$$

where we define $f_l[p^{v-1}, p^v] \equiv [f_l(p^v) - f_l(p^{v-1})]/(p_l^v - p_l^{v-1})$, $f_l[p^v, p^*]$ also has the similar form. By introducing

$$f_l[p^{v-1}, p^v, p^*] \equiv \frac{f_l[p^{v-1}, p^v] - f_l[p^v, p^*]}{p_l^{v-1} - p_l^*}. \tag{38}$$

(37) can be simplified to the form

$$p_l^* - p_l^{v+1} = -\gamma(p_l^* - p_l^v)(p_l^* - p_l^{v-1})\frac{f_l[p^{v-1}, p^v, p^*]}{f_l[p^{v-1}, p^v]} + (1-\gamma)(p_l^* - p_l^v). \tag{39}$$

By Mean Value Theorem,

$$f_l[p^{v-1}, p^v] = \frac{\partial f_l(\zeta_l^v)}{\partial p_l}, \tag{40}$$

$$f_l[p^{v-1}, p^v, p^*] = \frac{1}{2}\frac{\partial^2 f_l(\eta_l^v)}{\partial p_l^2}. \tag{41}$$

So

$$p_l^* - p_l^{v+1} = -\gamma\frac{\partial^2 f_l(\eta_l^v)/\partial p_l^2}{2\partial f_l(\zeta_l^v)/\partial p_l}(p_l^* - p_l^v)(p_l^* - p_l^{v-1}) + (1-\gamma)(p_l^* - p_l^v). \tag{42}$$

If we choose $\gamma = 1$, then assuming that $\gamma\left|\frac{\partial^2 f_l(\eta_l^v)/\partial p_l^2}{2\partial f_l(\zeta_l^v)/\partial p_l}\right| \leq M_l$, the vector form of (42) is bounded as follows:

$$\left\|p^* - p^{v+1}\right\|_\infty \leq M\left\|p^* - p^v\right\|_\infty \left\|p^* - p^{v-1}\right\|_\infty, \tag{43}$$

where $M = \left\|\begin{bmatrix} M_1 & M_2 & \dots & M_L \end{bmatrix}^T\right\|_\infty$.

Then by setting $M\left\|p^* - p^v\right\|_\infty \equiv e^v$, we obtain

$$e^{v+1} \leq e^v e^{v-1}; v = 1, 2, ... \tag{44}$$

The following procedure is the same as in [16].

If we define $\max(e^0, e^1) = \delta$, from the inequalities (44) we have

$$e^i \leq \delta^{m_i}, i = 0, 1, 2..., \tag{45}$$

where $m_0 = m_1 = 1$ and $m_{i+1} = m_i + m_{i-1}, (i = 1, 2, ...,)$. Obviously, m_i forms a well known Fibonacci sequence, which has the solution

$$m_i = \frac{1}{\sqrt{5}}(r_+^{i+1} - r_-^{i+1}), \qquad r_\pm = \tfrac{1\pm\sqrt{5}}{2}.$$

For large $i, m_i \approx \frac{1}{\sqrt{5}}(r_+^{i+1}) \cong 0.447(1.618)^{i+1}$

$$\frac{e^{v+1}}{(e^v)^{1.618}} \cong 0.447.$$

That is to say, the order of Newton-like algorithm is approximately 1.618 for $\gamma = 1$. It is superlinearly convergent.

In the case of $\gamma << 1$, $1 - \gamma \approx 1$, from (42), the coefficient before $p_l^* - p_l^v$ is $-\gamma\frac{\partial^2 f_l(\eta_l^v)/\partial p_l^2}{2\partial f_l(\zeta_l^v)/\partial p_l}(p_l^* - p_l^{v-1}) << 1 - \gamma$ given that $p_l^* - p_l^{v-1}$ is small. So we have

$$p_l^* - p_l^{v+1} \approx (1-\gamma)(p_l^* - p_l^v),$$

which means the Newton-like algorithm is of approximately linear convergence.

In the case of $\gamma >> 1$, we suppose $p_l^* - p_l^{v-1}$ is still small (means that p_l^v is convergent). Then $\left|p_l^* - p_l^{v+1}\right| \approx (\gamma - 1)\left|(p_l^* - p_l^v)\right| > |(p_l^* - p_l^v)|$, it is divergent, which leads to a contradiction. So if $\gamma >> 1$, the Newton-like algorithm is not convergent. **Q.E.D.**

Proof of Theorem III:

From (14), (15), we have $\bar{p}_l(t+1) = \phi_l\{\phi[p(t-1)]\}$. Denote $\phi_l[\phi(p)] = \phi_l\phi(p)$ so (19) can be written as

$$p_l(t+1) = [\phi_l\phi(p(t-1)) - \frac{(\phi_l\phi(p(t-1)) - \phi_l(p(t-1)))^2}{\phi_l\phi(p(t-1)) - 2\phi_l(p(t-1)) + p_l(t-1)}]^+. \tag{46}$$

Here $t = 1, 3, 5, \ldots, p(0)$ is the initial value, $p(0), p(1), \bar{p}(2), p(3), \bar{p}(4), ...$ are produced by the gradient projection algorithm, i.e., $p(2n+1) = \phi(\bar{p}(2n)), \bar{p}(2n) = \phi(p(2n-1))$, $p(2n)$ is obtained by this Aitken-Extrapolation method. Usually, Aitken-Extrapolation is a separate procedure, but here, we use $p(2n)$ to substitute $\bar{p}(2n)$ as the input of gradient projection algorithm, i.e., $p(2n+1) = \phi(p(2n))$. We make this modification because $p(2n)$ is closer to the optimal point than $\bar{p}(2n)$. For the sake of simplicity, we still analyze the traditional method instead of the modified one.

Using $p$to denote the sequence $\bar{p}(2n)$, and $g_l(p)$ the sequence $p(2n)$, we can rewrite (46) as follows:

$$g_l(p) = [\phi_l\phi(p) - \frac{(\phi_l\phi(p) - \phi_l(p))^2}{\phi_l\phi(p) - 2\phi_l(p) + p_l}]^+. \tag{47}$$

Now we will show that the rate of convergence of $g_l(p)$ is superlinear.

Without loss of generality, we can neglect the positive projection $[\ ]^+$, that is, instead of (47), we can consider

$$\begin{aligned} g_l(p) &= \phi_l\phi(p) - \tfrac{(\phi_l\phi(p) - \phi_l(p))^2}{\phi_l\phi(p) - 2\phi_l(p) + p_l} \\ &= \tfrac{p_l\phi_l\phi(p) - \phi_l^2(p)}{\phi_l\phi(p) - 2\phi_l(p) + p_l}. \end{aligned} \tag{48}$$

Indeed, this simplification will not affect the result because of the following formula [17]:

$$\left\| [g_l(p)]^+ - [g_l(q)]^+ \right\|_2 \leq \|g_l(p) - g_l(q)\|_2, \tag{49}$$

where $\|\bullet\|_2$ denotes the Euclidean norm.

Recall that p^* is the vector optimizer, and p_l^* is the scalar optimizer. In [16], we know that the Aitken-extrapolation is of the second order in the scalar form ($S = L = 1$). In the general case, recall that we have $\phi_l(p) = p_l - \gamma f_l(p)$ and $\phi(p) = p - \gamma f(p)$, so

$$\phi_l\phi(p) = p_l - \gamma f_l(p) - \gamma f_l(\phi(p)).$$

It is not difficult to simplify (48) to the following form:

$$g_l(p) = p_l - f_l(p)\frac{\phi_l(p) - p_l}{f_l(\phi(p)) - f_l(p)}, \tag{50}$$

which is the continuous form of (36) with $\gamma = 1.$, i.e. false position method. From the previous context and [2], any optimal solution of this dual algorithm (may have

multiple optimal solution p^*, but unique solution x^*) is also the solution of the primal optimization problem (1) and (2). The convergence speed analysis is the similar with Theorem . The difference between this modified Aitken-Extrapolation method and Newton-like method is that the latter employs extrapolation (or false position) each step instead of every two steps in the former method. **Q.E.D.**

REFERENCES

[1] Frank Kelly, Aman Maulloo, and David Tan,*Rate control in communication networks: shadow prices, proportional fairness and stability*, Journal of the Operational Research Society, 49(1998), pp. 237–252.

[2] S. H. Low and D. E. Lapsley, *Optimization Flow Control, I: Basic Algorithm and Convergence*, IEEE/ACM Transactions on Networking, 7:6(1999), pp. 861–75.

[3] S. Athuraliya and S. H. Low, *Optimization Flow Control with Newton-like Algorithm*, Journal of Telecommunication Systems, 15:3/4(2000), pp. 345–358.

[4] S. Athuraliya, S. Low, and D. Lapsley, *Random early marking*, in: Proceedings of the First International Workshop on Quality of future Internet Services (QofIS'2000), Berlin, Germany, September 2000.

[5] Sanjeewa Athuraliya, David Lapsley, and Steven Low, *An Enhanced Random Early Marking Algorithm for Internet Flow Control*, in: Proceedings of IEEE Infocom, March 2000.

[6] Yair Bartal, J. Byers, and D. Raz, *Global optimization using local information with applications to flow control*, in: STOC, October, 1997.

[7] C. Courcoubetis, V. A. Siris, and G. D. Stamoulis. *Integration of Pricing and Flow Control for Available Bit Rate Services in ATM Networks*, in: Proceedings of the IEEE GLOBECOM, pp. 644–648, 1996.

[8] R. J. Gibbens and F. P. Kelly, *Resource pricing and the evolution of congestion control*, Automatica, 35, 1999.

[9] Jamal Golestani and Supratik Bhattacharyya, *End-to-end congestion control for the Internet: A global optimization framework*, in: Proceedings of International Conf. On Network Protocols (ICNP), October 1998.

[10] F. P. Kelly, *Charging and rate control for elastic traffic*, European Transactions on Telecommunications, 8(1997), pp. 33–37.

[11] S. H. Low, *Optimization flow control with on-line measurement*, in: Proceedings 16th International Teletraffic Congress, volume 3a, pp. 237–249, Elsevier, 1999.

[12] Srisankar Kunniyur and R. Srikant, *End-to-end congestion control schemes: utility functions, random losses and ECN marks*, in: Proceedings of IEEE Infocom, March 2000.

[13] David E. Lapsley and Steven H. Low, *An optimization approach to ABR control*, in: Proceedings of the ICC, June 1998.

[14] L Massoulie and J. Roberts, *Bandwidth sharing: objectives and algorithms*, in: Infocom'99, March 1999.

[15] David G. Luenberger, *Linear and Nonlinear Programming*, 2nd Ed. Addison-Wesley, Publishing Company, 1984.

[16] Eugene Isaacson and Herbert Bishop Keller, *Analysis of Numerical Methods*, John Wiley, & Sons, Inc., New York, 1996.

[17] Dimitri P. Bertsekas and John N. Tsitsiklis, *Parallel and Distributed Computation*, Prentice-Hall, 1989.

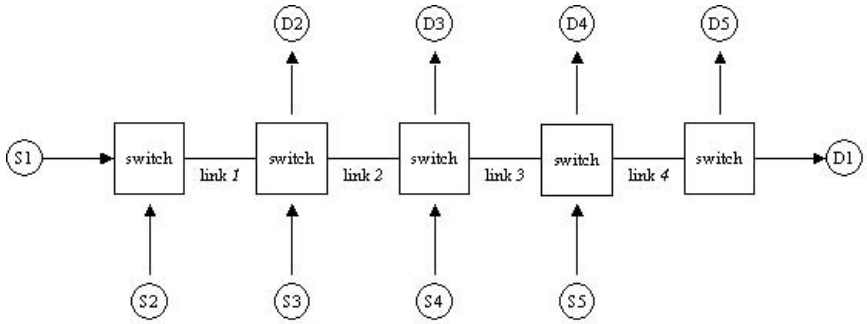

FIG. 2. *Network Topology [3].*

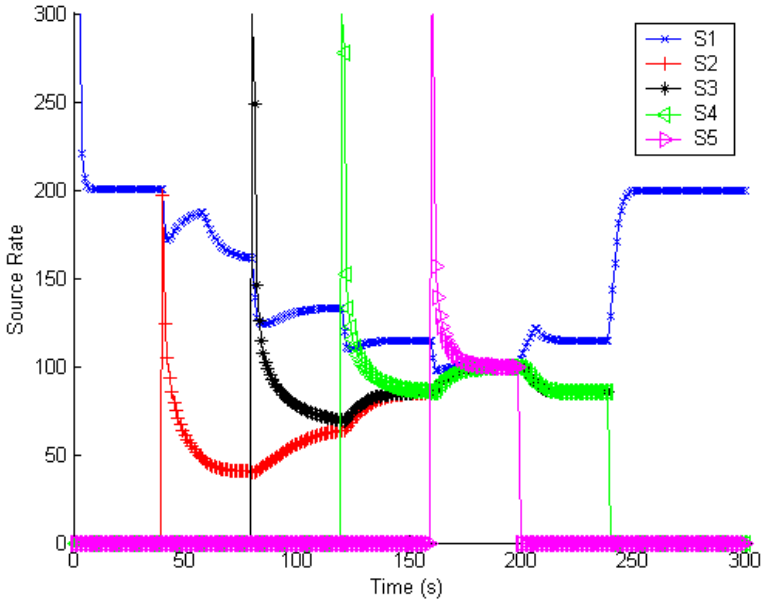

FIG. 3. *Source rates under gradient projection algorithm.*

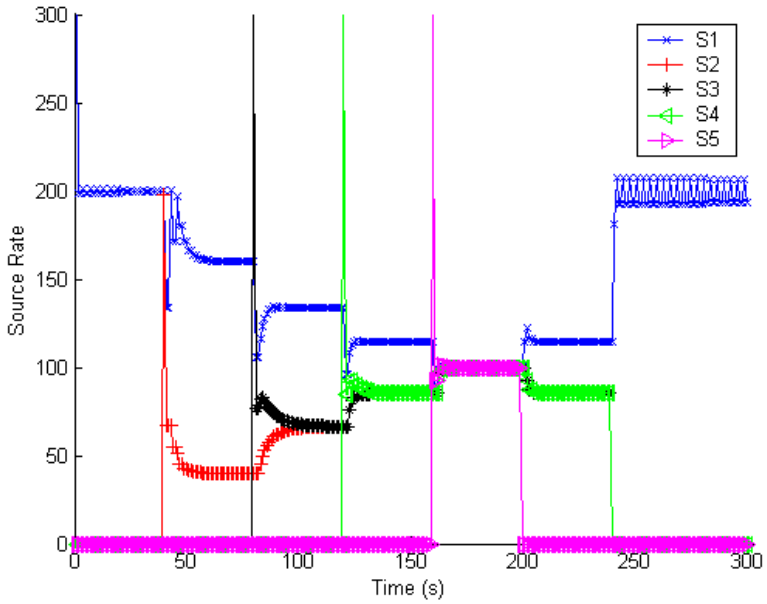

FIG. 4. *Source rates under Newton-like algorithm.*

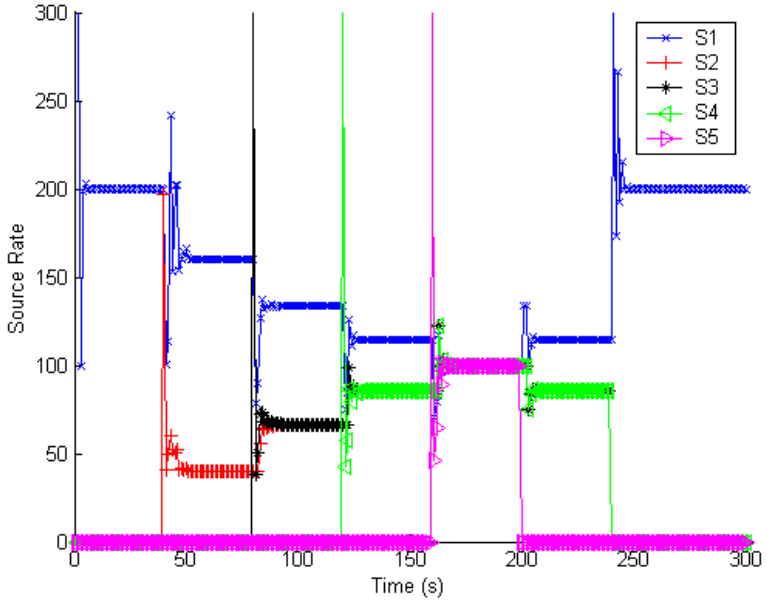

FIG. 5. *Source rates under Aitken-Extrapolation algorithm.*

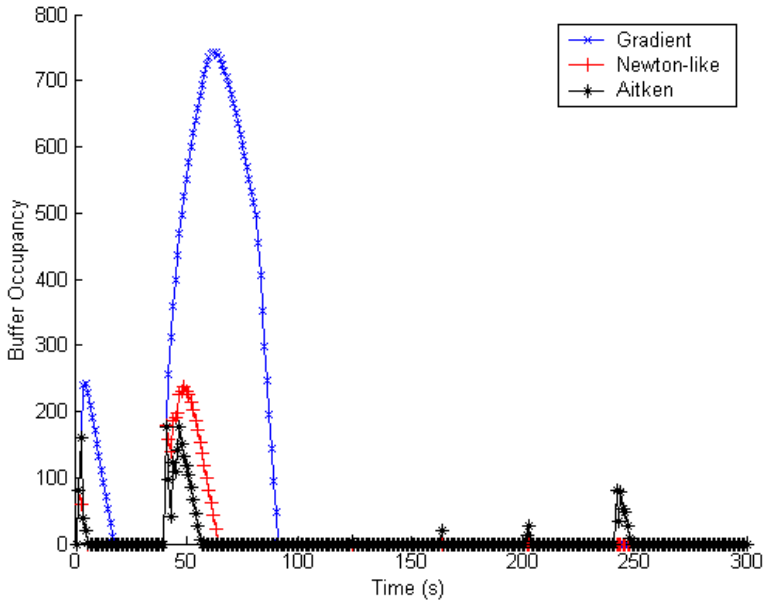

FIG. 6. *Comparison of Buffer Occupancy.*

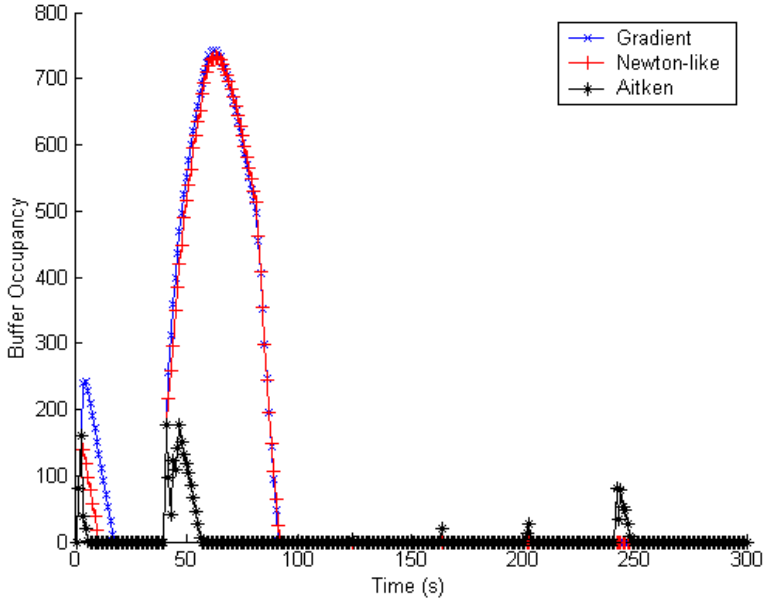

FIG. 7. *Comparison of Buffer Occupancy.*