An Optimization Based Approach for Quality of Service Routing in High-Bandwidth Networks

Xiaojun Lin and Ness B. Shroff

School of Electrical and Computer Engineering, Purdue University

West Lafayette, IN 47907, U.S.A.

{linx, shroff}@ecn.purdue.edu. *

Abstract

In this paper, we propose an optimization based approach for Quality of Service routing in high-bandwidth networks. We view a network that employs QoS routing as an entity that distributively optimizes some global utility function. By solving the optimization problem, the network is driven to an efficient operating point. In earlier work, it has been shown that when the capacity of the network is large, this optimization takes on a *simple form*, and once the solution to this optimization problem is found, simple proportional QoS routing schemes will suffice. However, this optimization problem requires global information. We develop a distributed and adaptive algorithm that can efficiently solve the optimization online. Compared with existing QoS routing schemes, the proposed optimization based approach has the following advantages: (1) The computation and communication overhead can be greatly reduced without sacrificing performance; (2) The operating characteristics of the network can be analytically studied; and (3) The desired operating point can be tuned by choosing appropriate utility functions.

*This work has been partially supported by the NSF grant ANI-0099137 and the Indiana 21st Century Fund 651-1285-074.

1 Introduction

Future telecommunication networks are expected to support applications with diverse Quality of Service requirements. Quality of Service (QoS) routing is an important component of such networks and has received considerable attention over the past decade (for a good survey, see [4] and the reference therein). The objective of QoS routing is two-fold: to find a feasible path for each incoming connection; and to optimize the usage of the network by balancing the load.

In this paper, as in the majority of studies on QoS routing, we assume a source routing model where routing decisions are made at the point where connection requests originate. In most of these studies, researchers take the following view of the QoS routing problem: The links are "dumb" and they advertise their status. The intelligence lies in the end-systems (sources or edge routers) to compute paths based on the current knowledge of the link states.

The above paradigm would have worked well if the link states were stable. However, not all link state metrics are stable. In particular, the available bandwidth metric of a link is inherently dynamic and changes frequently as connections enter and leave the network. Therefore, the link state advertisement and the QoS routing algorithm have to be executed frequently in order to keep up with the changes in link states. This leads to a significant amount of computation and communication overhead. To reduce the computation and communication burden, the frequency of the computation and the link state updates then need to be contained. This could, however, result in staleness of the link state information and inaccuracy in the routing decisions. Hence, there is a fundamental tradeoff between the amount of computation and communication resources consumed and the quality of the routing decisions. This tradeoff is usually difficult to analyze and researchers have had to resort to simulation studies [1, 14, 21, 22]. These studies reveal that the performance of existing QoS routing schemes degrades when computation and link state updates become infrequent. However, the extent to which the performance degrades depends not only on how infrequently the computation and link state updates are made, but also on a large number of other factors that include: the specifics of the path computation algorithm, the topology and the demand pattern of the network, the cost metrics assigned for each link, the link state update strategy, and the strategy to handle routing failures, etc. In general, the exact level of performance degradation is hard to predict.

In this paper, we take a different view of the QoS routing problem. We view the network (including the end-systems *and* the links) that employs QoS routing as an integral entity that jointly optimizes some global utility function. Once the solution to this optimization problem is found, the network will be driven to an efficient operating point, and the routing performance will be close to optimal. No further computation and communication are needed as long as the prevailing network condition remains essentially unchanged.*

We refer to our proposed scheme as the optimization based approach for QoS routing. When the capacity of the network is large, this optimization takes on a simple form. Our proposal is based on a known result: *simple proportional routing schemes* can approach the performance of the *optimal dynamic routing schemes* when the capacity of the network is large [8, 12, 16]. In a *proportional routing scheme*, calls are routed to alternate paths based on pre-determined probabilities. The right routing probabilities can be derived from the solution of a simple optimization problem that depends only on the *average* demand and capacity of the network.

We develop an online, distributed algorithm that can efficiently solve the optimization problem. Fig. 1 provides a high-level view of the optimization based approach. Each link in the network is associated with an implicit cost. The implicit cost summarizes the congestion level at the link and can be updated by the observed demand and capacity at the link. Thus, we equip the link with only a minimal amount of intelligence (i.e., to update the implicit cost). It turns out that the implicit cost is the only information that the end-system needs to solve the optimization problem. The end-system has three components: a path-finding component that maintains a set of alternate paths; an optimization component that solves for the optimal routing

*In practice, some computation and communication will still be required to track changes in the network condition. However, a nice feature of our work is that computation and communication intensive operations can be done at very long time-scales, with a negligible impact on performance.

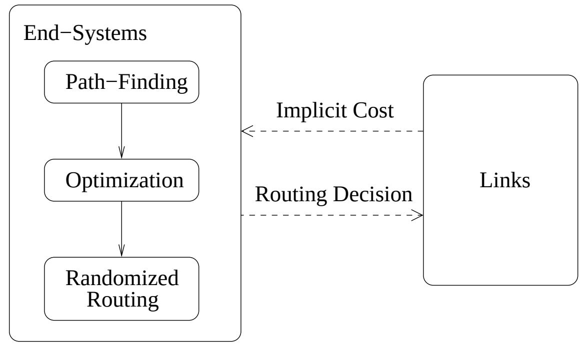

Figure 1: Our Optimization Based Approach

probabilities; and a randomized routing component that routes each incoming connection based on the precomputed routing probabilities.

Compared with existing QoS routing schemes, our optimization based approach has the following advantages:

(1) The computation and communication overhead can be greatly reduced without sacrificing performance. Once the optimal operating point is found, the same routing parameters can be used by a large number of future arrivals, as long as the *average* network condition remains unchanged. Infrequent computation and link state updates will only affect the speed of convergence of the distributed algorithm, but not the end-result that the algorithm converges to.

In practical networks, the *average* network condition can also change gradually over time (non-stationary behavior), e.g., during the course of a day. Our distributed algorithm will track the changes in the average network condition and adjust the operating point accordingly. Note that in a control system, there has always been the issue of the right *time scale of control*. A nice feature of our proposed solution is that, the control that needs to be done at a fast time scale, i.e., the randomized routing, is very simple; while the control that requires a large amount of computation, i.e., the optimization of routing probabilities and the search for new alternate paths, can be carried out over a much slower time scale. Using the right separation of control time scales, our optimization based approach ensures near optimal performance even when the computation and communication become infrequent.

(2) The operating characteristics of the network can be analytically studied. Given the network model, we can easily predict the operating point by solving the optimization problem. In contrast, due to the complexity of the system, the analysis of existing QoS routing schemes appears to be intractable, especially under inaccurate link state information and infrequent computation.

(3) The desired operating point can be tuned by appropriately choosing the utility functions. The optimization based approach allow us not only to *predict* the operating point of the network, but also to *control* it. By choosing different utility functions for different classes and source-destination pairs, we can achieve the desired balance among the service levels offered to different groups of users. For example, when the network becomes congested, connections with a larger number of hops could suffer significantly more blocking than shorter connections. In our optimization based approach, this can be avoided by assigning longer connections a utility function that has a higher marginal utility.

1.1 Related Work

The optimal control of loss networks has been studied extensively in the past. Both off-line [6, 7, 17] and simulation based schemes [15] have been proposed. Our contribution is to propose an *online* solution for QoS routing. Our online scheme exploits the fact that simplicities arise in high-bandwidth networks, e.g., as long as the loads at all links are less than or equal to one, the blocking probability of a high-bandwidth system will be close to zero. This property results in a much simpler and easily decomposable optimization problem.

Our proposed solution employs a proportional routing scheme. The asymptotic optimality of the proportional routing scheme in large systems has been known for some time [8, 16]. However, a major criticism of proportional routing schemes has been the following: if the demand is incorrectly estimated, the computed routing probabilities could lead to poor performance [15]. We solve this problem by using an *adaptive* algorithm that does not rely on any prior knowledge of the demand. The Adaptive Proportional Routing scheme proposed in [18] is also related to our work. In their scheme, each class measures the amount of blocking along each alternate paths, and uses the inverse Erlang formula to estimate a "virtual capacity" grabbed by the class along each path. Then each class locally optimizes the routing probabilities based on the demand and these virtual capacities. The advantage of our optimization based approach is that the optimality of the resulting operating point and the convergence of the algorithm can be rigorously shown. Further, the implicit costs provide additional information for discovering new alternate paths.

The structure of our optimization problem is similar to that of multi-path flow control problems in [5, 23]. Our implicit cost based solution is similar to the one in [23]. In [23], the authors claim that their algorithm is one of the Arrow-Hurwicz algorithms [2]. However, the convergence of the Arrow-Hurwicz algorithm was established in [2] only for the case when the objective function is strictly concave, which is not true for the problem at hand. In this paper, we present a new result that characterizes the convergence correctly.

The rest of the paper is organized as follows: In Section 2, we present the asymptotic optimality of the static proportional routing scheme. In Section 3, we derive the distributed algorithm for computing the optimal routing probabilities and obtain the proposed QoS algorithm. We discuss implementation issues in Section 4, present simulation results in Section 5, and then conclude.

2 Simplification of QoS Routing in Large Networks

2.1 The Model

We adopt a multi-class loss network model. There are L links in the network. Each link $l \in \{1, ..., L\}$ has capacity R^l. There are I classes of users. Each class is associated with a source-destination pair, and some given QoS requirements. Flows of class i arrive to the network according to a Poisson process with rate λ_i. Once admitted, a flow of class i will hold r_i amount of bandwidth. (For the moment we assume that bandwidth is the only QoS metric. *The extension* *to multiple QoS metrics will be addressed in Section 3.4.)* The service times within a class are i.i.d. and independent of the arrival process. The service time distribution is general with mean $1/\mu_i$. Each admitted flow of class i generates v_i amount of revenue per unit time. The objective of the network is to maximize the revenue from all flows admitted into the network.

Such a network model could represent the backbone of an ISP serving applications with different QoS requirements. The revenue v_i could either be actual money, or simply an assigned weight that represents the network's preference for each class. There could be multiple classes associated with each source-destination pair, differing in their bandwidth requirement r_i and revenue v_i.

In this section, we assume that each class i has set up $\theta(i)$ alternate paths using, for example, MPLS [20] (we will address how these alternate paths can be found in Section 3.3). The alternate paths are represented by a matrix $[H_{ij}^l]$ such that $H_{ij}^l = 1$ if path j of class i uses link l, and $H_{ij}^l = 0$ otherwise. We denote the state of the system by a vector $\vec{n} = [n_{ij}, i = 1, ...I, j = 1, ..., \theta(i)]$, where n_{ij} is the number of flows of class i currently using path j. The bandwidth requirements and the capacity constraints then determine the set of feasible states $\mathbf{\Omega_n} = \{\vec{n} : \sum_{i=1}^{I}\sum_{j=1}^{\theta(i)} n_{ij} r_i H_{ij}^l \leq R^l \quad \forall l\}$.

We denote the routing decision (which can be time varying) for class i by a vector

$$\begin{aligned} \vec{p}_i &= [p_{i1}, p_{i2}, ..., p_{i,\theta(i)}], \\ \vec{p}_i &\in \Omega_i \triangleq \{p_{ij} \geq 0, \sum_{j=1}^{\theta(i)} p_{ij} \leq 1, \text{ for all } j\}, \end{aligned}$$

where

$$p_{ij} = \mathbf{Pr}\{\text{an incoming flow of class } i \text{ is routed to path } j\}.$$

Hence, an incoming flow of class i will be admitted with probability $\sum_{j=1}^{\theta(i)} p_{ij}$, and, if admitted, it will be routed to path j with probability $p_{ij}/\sum_{k=1}^{\theta(i)} p_{ik}$. Let $\vec{p} = [\vec{p}_1, ..., \vec{p}_I]$.

A *dynamic routing scheme* is one where routing decisions can adapt to the changing utilization level of the network. For example, $\vec{p}(t)$ can be a function of the current state of the network, i.e.,

$\vec{p}(t) = g(\vec{n}(t))$. Note that this model can characterize virtually any QoS routing proposals that select paths based on the current snapshot of the network. Alternatively, $\vec{p}(t)$ can be a function of some past history of network states $\vec{n}(s), s \in [t-d, t]$, where d is the length of the history information. The network can use the past history to predict the future, and use prediction to improve the routing decision. $\vec{p}(t)$ can also depend on the service time T of the incoming connection, if this information is available. The routing policy can then be written, in a most general form, as

$$\vec{p}(t) \;=\; g(\vec{n}(s), s \in [t-d, t]; T). \tag{1}$$

Each admitted flow of class i will generate v_i amount of revenue per unit time. The dynamic routing scheme that maximizes the long term average revenue is then

$$J^* \triangleq \max_g \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \mathbf{E}_g \left[n_{ij}(t) \right] v_i,$$

where $\mathbf{E}_g$ denotes the expectation taken with respect to the stationary distribution under policy g. It can be shown that the system under g will always converge to a stationary version, and the stationary version is ergodic [12].

Finally, in a *static scheme*, the routing policy is represented by a time-invariant vector $\vec{p}$. This corresponds to a proportional routing scheme. The performance of the static scheme is:

$$J_0 \triangleq \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} p_{ij} v_i [1 - \mathbf{P}_{Loss,ij}],$$

where $\mathbf{P}_{Loss,ij}$ is the blocking probability experienced by flows of class i routed to path j.

2.2 Asymptotic Optimality of Static Schemes

The drawback of dynamic schemes is that the optimal schemes are difficult to find, and the implementation will consume a large amount of computation and communication resources. *When the capacity of the system is large, simple static schemes can approach the performance of the optimal dynamic scheme.* This has been the central theme of our earlier work [12]. Here, we rephrase the main result under the context of QoS routing. We scale the capacity and the demand proportionally by $c > 1$, i.e., in the c-scaled network, the capacity at each link l is $R^{l,c} = cR^l$, and the arrival rate of each class i is $\lambda_i^c = c\lambda_i$. It turns out that when c is large[†], a simple static scheme will suffice. The static scheme is constructed as follows:

Step 1: Solve the following optimization problem:

$$
\begin{aligned}
J_{ub} = \max_{\vec{p} \in \Omega} \quad & \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} \sum_{j=1}^{\theta(i)} p_{ij} v_i \\
\text{subject to} \quad & \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} r_i p_{ij} H_{ij}^l \le R^l \quad \text{for all } l,
\end{aligned}
\tag{2}
$$

where $\Omega = \bigotimes_{i=1}^{I} \Omega_i$.

Step 2: Use the optimal point $\vec{p}$ in (2) as the static policy. Let J_s be its performance.

The following proposition can be shown as in [12].

Proposition 1 *Let $J^{*,c}$ and J_s^c be the revenue of the optimal dynamic scheme and the revenue of the static scheme constructed above, respectively, in the c-scaled system, then*

$$\lim_{c \to \infty} J_s^c / c = \lim_{c \to \infty} J^{*,c} / c = J_{ub}.$$

We sketch the main ideas behind Proposition 1. Firstly, one can show that cJ_{ub} is an upper bound of $J^{*,c}$ *under any dynamic routing policy* g [8, 16]. Secondly, the static revenue J_s^c differs from the upper bound cJ_{ub} only by the term $(1 - \mathbf{P}_{Loss,ij})$. Now since $\vec{p}$ satisfies the constraint of (2), the traffic load at each link is no greater than 1. Lemma 2 in [12] then ensures that the blocking probability goes to zero as $c \to \infty$. Finally, because $J_s^c \le J^{*,c} \le cJ_{ub}$, Proposition 1 then follows. The detailed proof is available in Appendix A. Readers can refer to [12] for a thorough treatment of the various simplicities that arise in the control of large-bandwidth networks.

[†]Note that here largeness does not imply over-provisioning.

3 The Optimization Based Approach to QoS Routing

There is a continuing trend to deploy routers with larger and larger link capacities in the Internet. Therefore, the results in the last section offer important insights on the QoS routing problem in the high-bandwidth networks of today and the future. Firstly, by solving a simple upper bound, we can obtain a simple *time-invariant* scheme that is *close to optimal*. Once we *precompute* the routing probabilities according to (2), this QoS routing result can be used for future arrivals. Thus, *the computation overhead can be greatly reduced.* Secondly, the upper bound (2) replaces the instantaneous capacity constraint $\sum_{i=1}^{I}\sum_{j=1}^{\theta(i)} n_{ij} r_i H_{ij}^l \le R^l$ by an average load constraint $\sum_{i=1}^{I}\sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} r_i p_{ij} H_{ij}^l \le R^l$. Hence, the precomputation only needs to react to the *average congestion level* in the network rather than the *instantaneous congestion level*. The staleness of the link state information is no longer a major issue!

Therefore, if we are able to solve the upper bound (2) efficiently, we can obtain a QoS routing algorithm that is close to optimal in large networks and that can tolerate infrequent computation and infrequent link state updates. However, we still need to consider the following issues.

- The upper bound is a global optimization problem. A distributed solution is desired.
- Some parameters, such as λ_i and μ_i, could be unknown a priori and changing gradually over time. A solution is needed that can automatically adapt to these changes.

We next present an adaptive, distributed algorithm for solving the upper bound. Before we proceed, we note that in many scenarios, it is also desirable to modify the upper bound to improve fairness. We can view the upper bound (2) as a constrained optimization problem that maximizes some aggregate utility functions:

$$\begin{aligned} \max_{\vec{p}\in\Omega} \quad & \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} U_i\Big(\sum_{j=1}^{\theta(i)} p_{ij}\Big) v_i \qquad (3) \\ \text{subject to} \quad & \sum_{i=1}^{I}\sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} r_i p_{ij} H_{ij}^l \le R^l \quad \text{for all } l, \end{aligned}$$

where the utility function U_i is linear: $U_i(p) = p$. A linear utility function, however, does not possess good fairness properties: for example, connections with a larger number of hops could be completely blocked to give way to connections with fewer hops. To improve fairness, we can use a strictly concave utility function U_i, as in flow control problems [13]. The derivative $U_i'(\sum_{j=1}^{\theta(i)} p_{ij})$ represents the amount of marginal utility lost if the overall admission probability for class i is further reduced. The desired balance among different classes can be achieved by tuning the revenue v_i and the utility function U_i. Proposition 1 can be generalized to the case with concave utility functions (see Appendix A). In the simulation results that follow, we will use utility function that satisfies $U_i'(1) = 1$. This choice of the utility function ensures that the revenue v_i is correctly reflected by the marginal utility when all flows of class i can be admitted, i.e., $v_i U_i'(\sum_{j=1}^{\theta(i)} p_{ij}) = v_i$ when $\sum_{j=1}^{\theta(i)} p_{ij} = 1$. As long as the utility function follows this rule, our simulation results indicate that the revenue is usually not affected much by changing the utility functions.

3.1 A Distributed Algorithm

Let $\vec{p^*}$ be the maximizer of the modified upper bound (3). Because the objective function is concave and the constraint set is convex and compact, a maximizer always exists. However, it is generally not unique, since the objective function is not strictly concave. (Note that even if U_i is strictly concave, the overall problem is not, because of the linear operation $\sum_{j=1}^{\theta(i)} p_{ij}$.)

The form of the upper bound motivates us to study its dual. However, when the objective function of the primal problem is not strictly concave, the dual problem may not be differentiable. To circumvent this difficulty, we use ideas from Proximal Optimization Algorithms [3, Chapter 3.4]. The idea is to add a quadratic term to the objective function, so that the objective function becomes strictly concave. We introduce an auxiliary variable y_{ij} for each p_{ij}. Let $\vec{y}_i = [y_{ij}, j = 1, ..., \theta(i)]$ and $\vec{y} = [\vec{y}_1, .., \vec{y}_I]$. The optimization becomes:

$$\max_{\vec{p}\in\Omega, \vec{y}\in\Omega} \quad \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} U_i(\sum_{j=1}^{\theta(i)} p_{ij}) v_i$$

$$-\sum_{i=1}^{I}\sum_{j=1}^{\theta(i)}\frac{\lambda_i}{\mu_i}\frac{\nu_i}{2}(p_{ij}-y_{ij})^2 v_i \tag{4}$$

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

where ν_i is some positive number chosen for each class i. It is easy to show that the optimal value of (4) coincides with that of (3). In fact, if $\vec{p} = \vec{p^*}$ is the maximizer of (3), then $\vec{p} = \vec{p^*}, \vec{y} = \vec{p^*}$ is the maximizer of (4).

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 minimize the augmented objective function with respect to $\vec{p}$. To be precise, this step solves:

$$\max_{\vec{p}\in\Omega} \quad \sum_{i=1}^{I}\frac{\lambda_i}{\mu_i}U_i(\sum_{j=1}^{\theta(i)}p_{ij})v_i$$

$$-\sum_{i=1}^{I}\sum_{j=1}^{\theta(i)}\frac{\lambda_i}{\mu_i}\frac{\nu_i}{2}(p_{ij}-y_{ij})^2 v_i \tag{5}$$

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

  Since the primal objective is now strictly concave, the maximizer exists and is unique. Let $\vec{p}(t)$ be the solution to this optimization.

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

Step P1) can now be solved through its dual. Let $q^l, l = 1, ..., L$ be the Lagrange Multiplier for the constraints in (5). Let $\vec{q} = [q^1, ..., q^L]$. Define the Lagrangian as:

$$L(\vec{p},\vec{q},\vec{y}) = \sum_{i=1}^{I}\frac{\lambda_i}{\mu_i}U_i(\sum_{j=1}^{\theta(i)}p_{ij})v_i$$

$$-\sum_{l=1}^{L}q^l(\sum_{i=1}^{I}\sum_{j=1}^{\theta(i)}\frac{\lambda_i}{\mu_i}r_i p_{ij}H_{ij}^l - R^l)$$

$$-\sum_{i=1}^{I}\sum_{j=1}^{\theta(i)}\frac{\lambda_i}{\mu_i}\frac{\nu_i}{2}(p_{ij}-y_{ij})^2 v_i$$

$$= \sum_{i=1}^{I}\frac{\lambda_i}{\mu_i}\left\{U_i(\sum_{j=1}^{\theta(i)}p_{ij})v_i - r_i\sum_{j=1}^{\theta(i)}p_{ij}\sum_{l=1}^{L}H_{ij}^l q^l\right.$$
$$\left.-\sum_{j=1}^{\theta(i)}\frac{\nu_i}{2}(p_{ij}-y_{ij})^2 v_i\right\} + \sum_{l=1}^{L}q^l R^l. \tag{6}$$

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

$$D(\vec{q},\vec{y}) = \max_{\vec{p}\in\Omega} L(\vec{p},\vec{q},\vec{y}) = \sum_{i=1}^{I} B_i(\vec{q}_i,\vec{y}_i)\frac{\lambda_i}{\mu_i} + \sum_{l=1}^{L}q^l R^l, \tag{7}$$

where

$$B_i(\vec{q}_i,\vec{y}_i) \quad = \quad \max_{\vec{p}_i\in\Omega_i}\left\{U_i(\sum_{j=1}^{\theta(i)}p_{ij})v_i - r_i\sum_{j=1}^{\theta(i)}p_{ij}q_{ij}\right.$$
$$\left.-\sum_{j=1}^{\theta(i)}\frac{\nu_i}{2}(p_{ij}-y_{ij})^2 v_i\right\}. \tag{8}$$

Note that in the definition of the dual objective function $D(\vec{q},\vec{y})$ in (7), we have decomposed the original problem into I separate subproblems. Given $\vec{q}$, each class can solve the routing probabilities $\vec{p}_i$ via its local subproblem (8) independently. If we interpret q^l as the implicit cost per unit bandwidth at link l, then q_{ij} is the total cost per unit bandwidth for all links in the path j of class i. Thus the q_{ij} captures all the information each subproblem needs about the path class i traverses. We note that an important feature of this decomposition is that the subproblem (8) is independent of the parameters λ_i and μ_i. This makes online implementation particularly easy.

The dual problem of (5), given $\vec{y}$, is:

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

Since the objective function of the primal problem (5) is strictly concave, the dual is always differentiable. The gradient of D is

$$\frac{\partial D}{\partial q^l} = R^l - \sum_{i=1}^{I}\sum_{j=1}^{\theta(i)}\frac{\lambda_i}{\mu_i}p_{ij}^0 r_i H_{ij}^l, \tag{9}$$

where p_{ij}^0 solves the local subproblem (8). Then step P1) can be solved by using the gradient descent iteration on the dual variable, i.e.,

$$q^l(t+1) = \left[q^l(t) - \alpha^l (R^l - \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} p_{ij}^0 r_i H_{ij}^l) \right]^+ , \tag{10}$$

where $[.]^+$ denotes the projection to $[0, +\infty)$.

The class of distributed algorithms we will use in this paper can be summarized as follows:

Algorithm $\mathcal{A}$:

- A1) Fix $\vec{y}(t)$ and use the gradient descent iteration (10) on the dual variable $\vec{q}$. Depending on the number of times the descent iteration is executed, we will obtain a dual variable $\vec{q}(t+1)$ that either exactly or approximately minimizes $D(\vec{q}, \vec{y}(t))$ (and, equivalently, solves (5)). Let K be the number of times the dual descent iteration is executed.

- A2) Let $\vec{p}(t)$ be the primal variable that maximizes, over all $\vec{p} \in \Omega$, the Lagrangian $L(\vec{p}, \vec{q}(t+1), \vec{y}(t))$ corresponding to the new dual variable $\vec{q}(t+1)$. Set $\vec{y}(t+1) = \vec{p}(t)$.

From now on, we will refer to (10) as the dual update, and step A2) as the primal update.

A stationary point of the algorithm $\mathcal{A}$ can be defined as a primal-dual pair $(\vec{y^*}, \vec{q^*})$ such that

$$\vec{y^*} \text{ maximizes } L(\vec{p}, \vec{q^*}, \vec{y^*}) \text{ over all } \vec{p} \in \Omega$$
$$\vec{q^*} \text{ is also a stationary point of (10)} .$$

By standard duality theory, any stationary point $(\vec{y^*}, \vec{q^*})$ of the algorithm $\mathcal{A}$ solves the augmented problem (4). Hence $\vec{p} = \vec{y^*}$ solves the upper bound (3).

An important question is how large K (in step A1) needs to be for algorithm $\mathcal{A}$ to converge to a stationary point. The standard proximal optimization theory [3] requires $K = \infty$, i.e., at each iteration the optimization (5) has to be solved exactly. When $K < \infty$, at best an approximate solution to (5) is obtained at each iteration. If the accuracy of the approximate solution can be controlled appropriately (see [19]), one can still show the convergence of the algorithm $\mathcal{A}$.

However, in this case the number of dual updates K has to depend on the required accuracy and usually needs to be large.

For online implementation, one cannot carry out the dual update infinitely many times *for one iteration of algorithm* $\mathcal{A}$. It is also difficult to distributively control the accuracy of the approximate solution to (5). Hence, in this work we use a different approach. The following result is new and shows that, by appropriately choosing the stepsize α^l, the algorithm $\mathcal{A}$ converges *for any choice of* $K \geq 1$. The proof is highly technical and is available in Appendix B.

Proposition 2 *Fix* $1 \leq K \leq \infty$. *As long as the stepsize* α^l *is small enough, the algorithm* $\mathcal{A}$ *will converge to a stationary point* $(\vec{y^*}, \vec{q^*})$ *of the algorithm, and* $\vec{p^*} = \vec{y^*}$ *solves the upper bound (3). The sufficient condition for convergence is:*

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

where $\mathcal{L} = \max\{\sum_{l=1}^{L} H_{ij}^l, i = 1, ..., I, j = 1, ...\theta(i)\}$ *is the maximum number of hops for any path, and* $\mathcal{S} = \max\{\sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} H_{ij}^l, l = 1, ..., L\}$ *is the maximum number of paths going through any link.*

Remark: The sufficient condition for $K = 1$ differs from that of $K = \infty$ only by a constant factor. For $K > 1$, our result requires that the stepsizes decrease on the order of $O(1/k^2)$. This is probably not the tightest possible result, and we conjecture that stepsizes of order $O(1)$ would work for any K. However, we leave this for future work. Finally, note that ν_i appears on the right hand side of the condition. Hence, by making the objective function more concave, we also relax the requirement on the stepsize α^l.

3.2 Distributed Implementation

Algorithm $\mathcal{A}$ lends naturally to online distributed implementation. The ingress router for each class is responsible for determining the routing probabilities for this class. To do so, the ingress router only needs to solve the local subproblem (8) using the implicit costs q^l at all core routers that class i traverses. An efficient algorithm can solve (8) in at most $O[\theta(i)\log\theta(i)]$ steps. (For details, see Appendix C.) The core routers bear the responsibility to update the implicit costs q^l according to the simple dual update rule (10). After every K dual updates, the ingress router executes the primal update.

We have mentioned earlier that the solution of each local subproblem (8) does not require knowledge of the demand parameters λ_i and μ_i. Next, we show that the dual update can also be carried out using online measurement at each link, again without prior knowledge of the demand parameters of each class. We then obtain an *adaptive* algorithm that can track changes in the network conditions.

Note that in the dual gradient (9), $\sum_{i=1}^{I}\sum_{j=1}^{\theta(i)}\frac{\lambda_i}{\mu_i}r_i p_{ij} H_{ij}^l$ is the average load per unit time at link l. This motivates us to estimate the gradient as follows: over a certain time window W, each link l collects the information of flow connection requests *from all classes* that arrive at the link. Let w be the total number of flow 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 requests.) Then we can use

$$G_t = R^l - \frac{\sum_{k=1}^{w} r_k T_k}{W} \tag{11}$$

to estimate the gradient. The interpretation is immediate: $\sum_{k=1}^{w} r_k T_k$ is the total amount of load brought to link l. One can verify that this estimate is unbiased, i.e., $\mathbf{E}[G_t] = \partial D/\partial q^l$. We can then update the implicit costs by

$$q^l(t+1) = \left[q^l(t) + \alpha^l\left(\frac{\sum_{k=1}^{w} r_k T_k}{W} - R^l\right)\right]^+ \tag{12}$$

When W is not large, the stepsize α^l has to be small to "average out" the noise in the estimate. This algorithm has the flavor of *stochastic approximation algorithms* [10] that have been used in many engineering problems. We have not yet been able to prove the convergence of this stochastic approximation algorithm, but our simulations seem to show good convergence properties when a small fixed stepsize is used. That is, according to the simulations, the stochastic approximation algorithm converges to a small neighborhood of the solution to the upper bound.

When the stepsize α^l is away from zero, our algorithm can track the nonstationary behavior of the network. As the demand (i.e., λ_i, μ_i) changes, it is reflected in the gradient estimate G_t. The network will then move towards the new optimal operating point.

3.3 How to Generate Alternate Paths

The set of alternate paths, denoted by the matrix $[H^l_{ij}]$ could potentially be the enumeration of all possible paths for each class. In practice, however, a much smaller set of alternate paths suffices. Maintaining this set of alternate paths is the role of the path-finding component in Fig. 1. There are several options to generate the candidate paths.

Option 1: Use paths that appear to be "heuristically good." For example, given a source-destination pair, we can use the set of minimum-hop paths, or, paths whose number of hops is no greater than h plus that of the minimum-hop path. Obviously, h should be small to avoid an explosion in the number of candidate paths.

Option 2: A better approach is to discover new paths online. The implicit costs q^l, which arise naturally as the Lagrangian Multipliers of the dual problem, give us guidelines on discovering potentially better alternate paths. Given a configuration of the alternate paths, the following properties can be easily verified that characterize any stationary point $(\vec{p^*}, \vec{q^*})$ of algorithm $\mathcal{A}$ (for details, see Appendix D): (1) when the utility functions are strictly concave, the admission probability $\sum_{j=1}^{\theta(i)} p^*_{ij}$ for each class i can be uniquely determined; (2) only paths that have the minimum cost see positive routing probabilities. (The cost of a path is the sum of the implicit costs for all links along the path.) Let $q_{i,0}$ denote the minimum cost among all alternate paths for class i, then for all j

$$p^*_{ij} > 0 \Rightarrow q^*_{ij} = q_{i,0} \triangleq \min_j \sum_{l=1}^{L} H^l_{ij} q^{l,*}$$

This is consistent with the *minimum first derivative path* discussed in [3, p417]. Therefore, adding paths whose costs are larger than the minimum cost will not yield any gain.

We can use the the above properties to iteratively generate the candidate paths online. Starting from any initial set of candidate paths, we execute the distributed algorithm $\mathcal{A}$ to solve the upper bound. Then based on the implicit costs at the (possibly approximate) stationary point, we can run *any* minimal cost routing algorithm using the implicit costs as the cost metric for each link. If the minimal cost is smaller than the minimal cost among the current set of candidate paths by a certain threshold, we add this new path into the set, and continue. Otherwise, we can conclude that no further alternate paths need to be added.

Option 3: Use historical data. This can be viewed as a traffic engineering step. We first take measurements of typical traffic demands at different times of the day. For each demand pattern, we can use the above procedure in Option 2 *offline* to find the optimal alternate paths. The union of the alternate paths under all demand patterns can then be used as the set of the candidate paths. The role of the distributed algorithm is to shift the traffic load among these candidate paths automatically as network condition changes.

3.4 Extensions to Multiple QoS Constraints

So far we have assumed that the bandwidth constraint is the only QoS constraint. We now address the extension to multiple QoS metrics and constraints. We can argue that link states metrics other than the available bandwidth, e.g., delay and overflow probabilities, etc., could be more stable in future high-bandwidth networks. When the link capacity of the network is large, the network can support a large number of connections at the same time. Due to the complexity in maintaining per-flow information, Quality of Service is likely to be provisioned on an aggregate basis. Each node in the network will provide a QoS guarantee on delay and/or packet loss probabilities *for all flows* belonging to the same class, rather than *for each individual flow.* Such guarantees will stay unchanged as new flows arrive at or old flows depart from the network.

Let each class be given some QoS requirements on *both* the bandwidth constraint *and* some other constraints such as delay or packet loss probabilities. We now assume that each link will provision certain QoS guarantees on these other QoS metrics. Such guarantees are constant over time and can be advertised to the entire network. The alternate paths for each class must now be constrained to those that satisfy these other QoS requirements. Given a set of alternate paths, the distributed algorithm in Section 3.1 can be used unchanged to find the optimal routing probabilities. In order to generate the alternate paths, we can use the options in Section 3.3, except that now we have to consider other constraints too. For example, in Option 2, we can still use the implicit cost as the cost metric for each link and execute any *constrained minimal cost* QoS routing algorithm to search for new alternate paths.

It is important to note that the path-finding step does not deal with the available bandwidth constraint directly. Instead, it is based on the implicit cost, which is a more stable parameter that depends on the average congestion level of the network. Hence, the path-finding step can be carried out infrequently. Note that the computation of optimal paths under multiple QoS constraints is usually a NP-complete problem. Hence, for any practical implementation of QoS routing solutions, the computation overhead has always been a key issue. Our optimization based approach does not directly reduce the computational complexity. Rather, it reduces the *frequency* of the computation. We emphasize that *the optimal performance is still preserved even though computation becomes infrequent*. This, as mentioned in the Introduction, is again due to the separation of control time-scales: the set of candidate paths needs to change only when the *average* demand and capacity of the network changes significantly. Hence, the intensive computations only need to be carried out infrequently.

4 Implementational Issues

In this section we address some implementational issues. The distributed algorithm requires communicating the implicit costs back to the ingress routers. There are two alternatives. One is to use the connection request packets sent by the ingress router. Each link can insert its own implicit costs when processing the connection request packets. When the response is sent back to the ingress router, the implicit costs are piggy-backed for free. The other approach is to periodically advertise the implicit costs throughout the network. In the latter case, even when the implicit costs are updated infrequently, while the speed of convergence of the distributed algorithm will be affected, the optimal routing probabilities that the algorithm converges to will remain the same.

For the link algorithm, the gradient estimate in (11) requires the information from all flow arrivals, including those that could have been rejected by the upstream links. In some network systems, once an intermediate link along the path rejects a connection request, the request will not be passed on to downstream links. Let $\mathbf{P}_{i,j}^{B,l}$ be the probability that a connection request of class i routed to path j is rejected by links that are upstream to link l. The true connection arrival rate of class i at a link l will be $\lambda_i p_{ij}(1 - \mathbf{P}_{i,j}^{B,l})$. In this case, the gradient estimate constructed in (11), by counting only actual arrivals, will be biased. However, when the system is large, this error will be small. This is due to two factors: 1. as long as the load at each link is less than or equal to 1, $\mathbf{P}_{i,j}^{B,l}$ will be close to zero (see [12]); 2. if some links have load greater than 1, the implicit costs at these links will be increased until the loads become less than or equal to 1. Therefore, in the end $\mathbf{P}_{i,j}^{B,l}$ will be close to zero and have a minimal impact on the gradient estimate.

In (11), When the service time T_k is not known at the time of connection arrival, it can also be replaced by the time average of past flows. The unbiasness of (11) is not affected. This time average can be calculated at the ingress router by measuring flows that have completed service.

The transient behavior of the distributed algorithm is sensitive to the choice of the stepsize α^l. A smaller stepsize will result in a smaller *misadjustment* (overshoot or undershoot) around the optimal solution, but takes a longer time to converge. A larger stepsize expedites the convergence at the cost of larger misadjustment. This tradeoff between misadjustment and speed of convergence is a fundamental one for stochastic approximation algorithms with constant stepsizes. A better approach is to use an adaptive stepsize scheme: a larger stepsize is used initially (or when sudden changes occur) to expedite convergence, followed by a smaller stepsize to reduce the misadjustment. This idea of stepsize adaptation has been used in many other applications, especially in adaptive filtering. Here we illustrate one such approach, borrowed from the idea in [9]:

Fix a link l. Let G_t be the estimate of the gradient at the t-th iteration. Let E_t be a weighted average of the past samples of G_t, i.e., upon a new sample G_t, and let

$$E_{t+1} = \epsilon^l G_t + (1 - \epsilon^l) E_t,$$

where ϵ^l is a small positive constant. Let α_t^l denote the stepsizes at the t-th iteration. We can update the stepsize based on the correlation between E_t and G_t, i.e.,

$$\alpha_{t+1}^l = \min\{[\alpha_t^l + \beta^l E_t G_t]^+, \alpha_{\max}\}, \tag{13}$$

where β^l is a small positive constant, and $\alpha_{\max}$ is a maximum allowable stepsize chosen to ensure the stability of the system.

5 Simulation Results

In this section, we present simulation results that illustrate our optimization based approach for QoS routing. We implement the distributed algorithm following the online measurement based scheme in Section 3.2. The topologies we use are shown in Fig. 2. We first demonstrate the convergence of the distributed algorithm using the "triangle" network in Fig. 2. There are three classes of flows (AB, BC, CA). For each class of flows, there are two alternate paths, i.e., a direct one-link path, and an indirect two-link path. The arrival rates for classes AB, BC, CA are 1, 1 and 3, respectively. Each flow consumes one bandwidth unit along the path(s) and holds the resources for a time that is exponentially distributed with mean of 100 units. Let the capacity of all links be 100. For all classes the revenue v_i is 1 unit and the utility function is $U_i(p) = \ln p$.

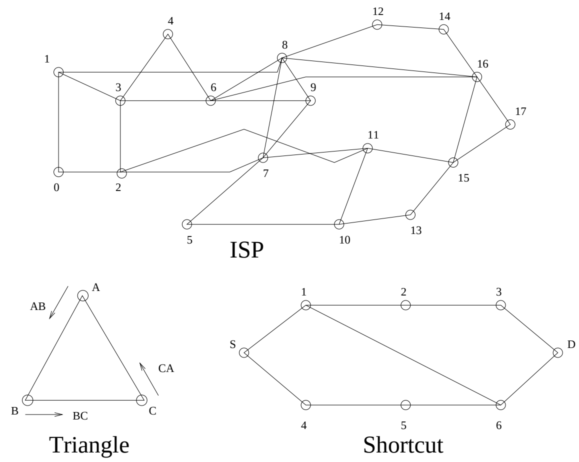

Figure 2: Network Topologies

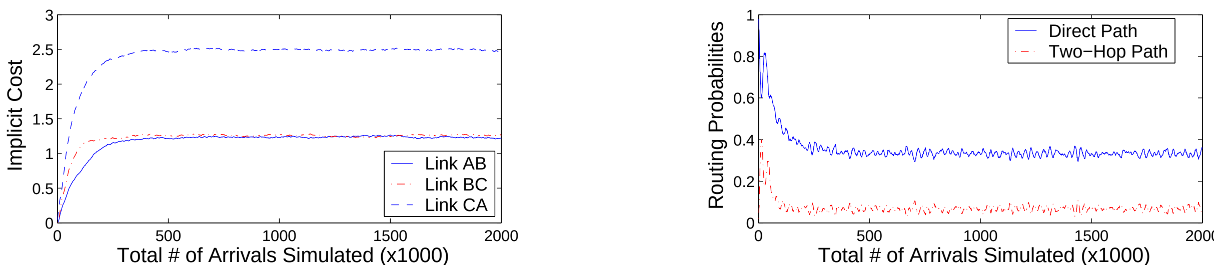

Figure 3: Evolution of the implicit costs (left) and the routing probabilities of class CA (right) with respect to the number of arrivals simulated. The unit of x-axis is 1000 arrivals. The solution to the upper bound is the following: the implicit costs are 1.25, 1.25, and 2.5, respectively, for link AB, BC and CA. The routing probability for class CA are 0.33 for the direct path and 0.067 for the two-hop path.

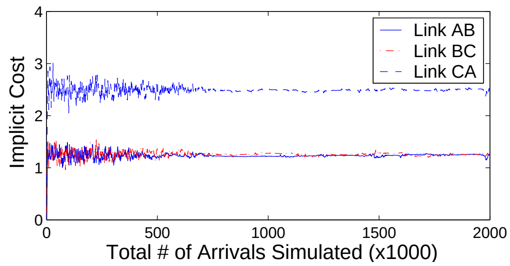

Figure 4: Evolution of the implicit costs when the adaptive stepsize scheme is used.

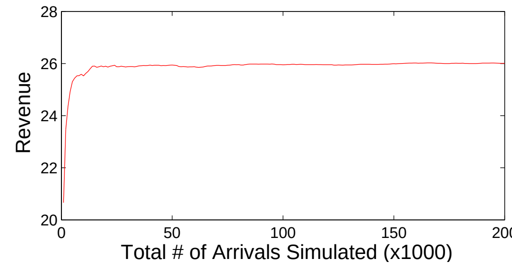

Figure 5: Evolution of the average revenue when the adaptive stepsize scheme is used.

Fig. 3 demonstrates the evolution over time of the implicit costs at all links and the evolution of the routing probabilities of class CA. The x-axis corresponds to the total number of arrivals simulated. Readers can verify that all quantities of interest converge to a small neighborhood of the solution to the upper bound. The parameters we use for the distributed algorithm are: $\alpha^l = 0.0001$, $\nu_i = 1$, $K = 1000$ and $W = 1$.

Fig. 4 demonstrates the convergence of the implicit costs when we use the adaptive stepsize scheme in Section 4. The parameters we use are: $\epsilon^l = 0.001, \alpha_{\max} = 0.1, \beta^l = 0.0001$ and $\alpha_0^l = 0$. The initial convergence is almost immediate: the implicit costs quickly jump to a small neighborhood of the solution to the upper bound, thanks to an increase in the stepsize initially. The evolution of the routing probabilities (not shown) follows the same trend. While the misadjustment takes time to die out (as the stepsize becomes smaller), Fig. 5 shows that the convergence of the revenue to its stationary value is achieved must faster (note that the range on the x-axis is smaller). As far as the overall revenue is concerned, the fluctuations of the implicit costs appear to cancel themselves out.

We next simulate a larger network, i.e., the "ISP" topology in Fig. 2, which is reconstructed from an ISP network and has been used in many simulation studies [1, 14, 21, 22, 18]. It has 18 nodes and 30 links. We simulate the case with a uniform demand matrix: flows arrive at each node according to a Poisson process with rate λ, and the destinations are chosen uniformly among all other nodes. Each connection consumes one unit of bandwidth. Revenue v_i is 1 unit.

We use a Pareto service time distribution with shape parameter 2.5, to capture the heavy-tailed characteristic of the traffics on the Internet. The mean service time is 100 units. The capacity of each link is 1000 bandwidth units.

There are a total of $18 \times 17 = 306$ source-destination pairs (i.e., classes). When the simulation is initialized, the set of alternate paths for each source-destination pair consists of all minimum-hop paths. Once simulation starts, new paths can be added following Option 2 in Section 3.3. To simplify the simulation, we adopt an upper limit of 10 on the number of alternate paths for each source-destination pair: when a new path is found, if there are already 10 alternate paths, the old path with the smallest routing probability will be replaced by the new path.

We choose the utility function of the following form

$$U_i(p) = h_i \ln p - (h_i - 1)p,$$

where h_i is the minimal number of hops between source-destination pair i. This utility function improves the admission probability for flows that traverse a larger number of hops. (At the same level of admission probability $p < 1$, the marginal utility $\frac{dU_i}{dp} = h_i/p - (h_i - 1)$ is larger for flows that traverse a long path.)

We simulate the optimization based approach using the distributed algorithm and compare, in Fig. 6, the revenue and total blocking probability over all classes against the values determined by the upper bound. We vary the per-node flow arrival rate λ from 1.0 to 10.0. As we can see from these figures, our distributed algorithm tracks the upper bound consistently over all loads. With a network of this size (each link can hold 1000 flows) the difference between the upper bound and the simulation of our distributed algorithm is already small.

We also compare the performance of the Widest-Shortest-Path (WSP) algorithm. WSP has been used in many simulation studies [1, 14, 18]. Among all feasible paths, the WSP algorithm will first choose paths that have the smallest number of hops. If there are multiple such paths, the WSP algorithm will choose the one with the largest available bandwidth. However, as shown in Fig. 6 the performance of a faithful implementation of WSP starts to taper off at $\lambda = 5.0$. The

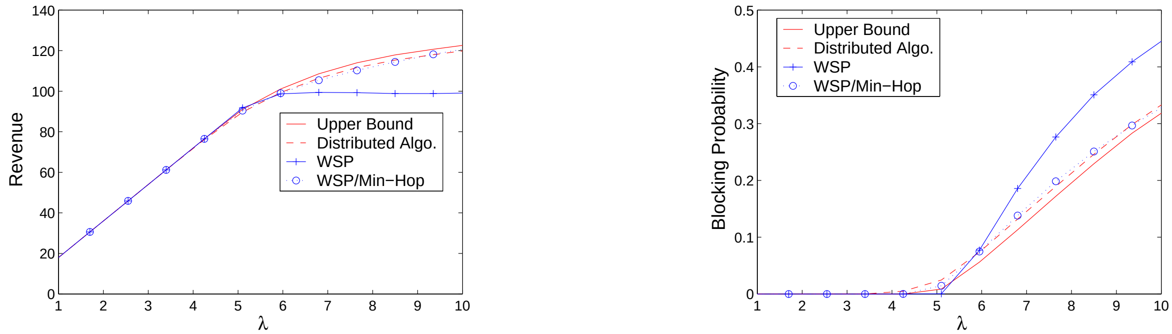

Figure 6: The revenue (left) and the blocking probability (right) of the distributed algorithm compared with the upper bound and WSP.

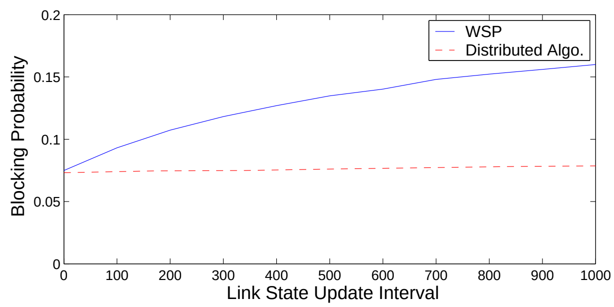

Figure 7: The blocking probability as the link state update interval increases.

performance degradation of WSP is due to its selection of non-minimal hop paths, which could result in sub-optimal configurations for the whole network. If we constrain WSP to minimum-hop paths only, the performance degradation will disappear in this example, as shown by the curve labeled "WSP/Min-Hop." However, from this, we should not draw the conclusion that such a practice is always better. By constraining WSP to minimum-hop paths, one also reduces the capability of WSP to use other potentially less congested paths. The end result depends on the topology of the network and the demand pattern. For example, in the "shortcut" topology in Fig. 2, assume that the capacities of all links are the same. If flows from S to D is to only use the minimum-hop path (S-1-6-D), once this path is full, no more flows can be admitted. However, if the flows use the non-minimum-hop paths S-1-2-3-D and S-4-5-6-D, twice as many flows can be admitted. Hence it is not always better to restrict on minimum-hop paths.

Our distributed algorithm, on the other hand, will always be able to find the right balance by solving the upper bound. It *consistently* tracks the upper bound under all load conditions. This provable optimality is an attractive feature of our optimization based approach as it ensures that the routing decision will always be close to optimal.

The strength of the optimization based approach is even more evident when the computation and link state updates become infrequent. To show this, we pick $\lambda = 6.0$ and simulate both the distributed algorithm and the WSP (with minimum-hop path only) when we vary the interval between link-state updates. For the distributed algorithm the implicit costs are advertised with each link state update. Computation is carried out after each link state update. In contrast to the suggestion given in [21], we do not allow WSP to *recompute* paths when a connection routed to a precomputed path is later rejected. The reason is that one cannot reduce the computational overhead too much if the recomputation is employed: for example, when the blocking probability is around 10%, on average 1 out of 10 arrivals will trigger recomputation! For a similar reason, we also do not use the *triggered* link state update strategy of [21] for WSP. When the triggered strategy is used, changes in available bandwidth that exceed certain percentage of the past advertised available bandwidth will trigger a new link state update. When the network operates at a high utilization level, the available bandwidth is small. Even small changes in available bandwidth will trigger frequent updates. Hence, one can not reduce the communication overhead too much using a triggered update strategy.

Simulation results are presented in Fig. 7. The performance of the distributed algorithm changes little as the link state update interval becomes larger and larger, while the performance of WSP decreases significantly. (The unit time on the x-axis is the mean inter-arrival time of flows at each node.) In the worst case, WSP blocks *twice as many connections* compared to the case when it has perfect link states. The exact level of this performance degradation is a complex function that depends on many factors, such as the topology and the demand of the network, etc. Again, the strength of the optimization based approach is that it consistently

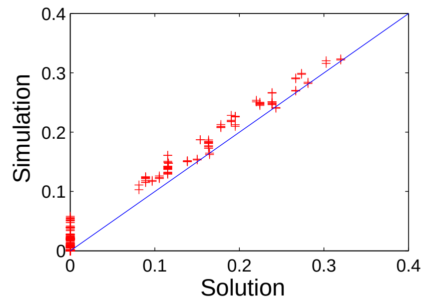

Figure 8: The blocking probability predicted by the upper bound compared with that collected from the simulation of the distributed algorithm. $\lambda = 6.0$ for this figure.

achieves near optimal performance, even when the computation and communication overhead are greatly reduced.

When our optimization based approach to QoS routing is used, designers can predict the operating point of the network by analytically solving the upper bound. This is shown in Fig. 8 where each point represents the blocking probability of one source-destination pair computed by the upper bound (along the x-axis) and that collected from the simulation of the distributed algorithm (along the y-axis). The points follow the diagonal line, which indicates that the simulation matchs the theory. In contrast, the analysis of dynamic QoS routing schemes (such as WSP) appears to be an intractable problem, especially when the computation becomes infrequent and the link state information becomes inaccurate. One usually has to resort to simulation to find out the operation of a QoS routing algorithm.

6 Conclusion and Future Work

In this paper, we developed an optimization based approach for Quality of Service routing in high-bandwidth networks. We view a network that employs QoS routing as an entity that carries out a distributed optimization. By solving the optimization problem, the network is driven to an efficient operating point. When the capacity of the network is large, this optimization takes on a simple form. We develop a distributed and adaptive algorithm that can efficiently solve the optimization online. The proposed optimization based approach has several advantages in reducing the computation and communication overhead, and in improving the predictability and controllability of the operating characteristics of the network.

We now briefly outline directions for future work: (1) In this paper we propose to update the implicit costs by measuring the arrived load. Other methods are possible, for example, by taking into account the utilization levels of the links. (2) A deeper understanding of the transient behavior of the distributed algorithm is important. The adaptive stepsize scheme in Section 4 that improves the speed of the convergence is of particular interest. (3) We assume that the capacity of the network is uniformly large. If some part of the network is not so large (for example, at the network edge), one then has to study a finer level of dynamics in these parts of the network. It would be interesting to study hybrid schemes that combine our results with some further details of the dynamics of smaller links. (4) In this paper we take a source routing model. Adapting our result to the distributed routing or hierarchical routing paradigms is also a possible direction for future work. (5) Finally, from a theoretical viewpoint, it would be important to prove the convergence of the distributed algorithm under more general settings, such as with asynchronous computation and stochastic approximation.

7 Appendix

A Proof of Proposition 1

We will focus on the case when the performance objective is the total utility. The case when the performance objective is the total revenue then follows as a special case where the utility function is the identify function.

When the performance objective is the total utility, the definition of J^*, J_0, J_s and J_{ub} needs to be modified accordingly. Given any dynamic routing policy g, one can show that the system under g will converge to a stationary version and the stationary version is ergodic ([12, 11]). Let $N_i(0,t)$ denote the number of arrivals of class i that arrive to the system from time 0 to t. Let $N_{ij}(0,t)$ denote the number of arrivals that are admitted and routed to path j from time 0 to t. Let

$$\lambda_{ij} = \lim_{t\to\infty} \frac{N_{ij}(0,t)}{t}.$$

The quantity λ_{ij} denotes the average rate of flows of class i routed to path j. It is well defined under a given policy g due to the stationary and ergodicity of the system. By definition

$$\lambda_i = \lim_{t\to\infty} \frac{N_i(0,t)}{t}$$

Further,

$$\sum_{j=1}^{\theta(i)} \frac{\lambda_{ij}}{\lambda_i} = \lim_{t\to\infty} \frac{\sum_{j=1}^{\theta(i)} N_{ij}(0,t)}{N_i(0,t)}$$

is the average proportions of flows of class i that are admitted.

Define the performance of policy g as the weighted total utility:

$$\sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} v_i U_i \left(\sum_{j=1}^{\theta(i)} \frac{\lambda_{ij}}{\lambda_i} \right) \tag{14}$$

When the utility function is the identify function, i.e., $U_i(p) = p$, then

$$\begin{aligned} \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} v_i U_i \left(\sum_{j=1}^{\theta(i)} \frac{\lambda_{ij}}{\lambda_i} \right) &= \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_{ij}}{\mu_i} v_i \\ &= \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \mathbf{E}_g[n_{ij}] v_i \end{aligned}$$

where n_{ij} is the random variable that denotes the number of flows of class i routed to path j that are in the system at any time, and $\mathbf{E}_g$ denote the expectation taken with respect to the stationary distribution under policy g. The last equality is by the Little's Law

$$\mathbf{E}_g[n_{ij}] = \frac{\lambda_{ij}}{\mu_i}$$

Hence, the definition of the performance in (14) is equal to the average revenue defined in Section 2. When $U_i(p)$ is a concave function, it represents the "utility" when the average admission probability of class i is p. In the simulation results in Section 5, we will use utility function that satisfies $U_i'(1) = 1$. This choice of the utility function ensures that the revenue v_i is correctly reflected by the marginal utility when all flows of class i can be admitted, i.e., $v_i U_i'(\sum_{j=1}^{\theta(i)} p_{ij}) = v_i$ when $\sum_{j=1}^{\theta(i)} p_{ij} = 1$. As long as the utility function follows this rule, our simulation results indicate that the revenue is usually not affected much by changing the utility functions.

The performance of the optimal dynamic scheme is defined as

$$J^* = \max_g \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} v_i U_i \left(\sum_{j=1}^{\theta(i)} \frac{\lambda_{ij}}{\lambda_i} \right)$$

Note that λ_{ij} is a function of g in the above formula.

The following Proposition establishes an upper bound for J^*.

Proposition 3 *Let J_{ub} be the solution of the following optimization problem:*

$$\begin{aligned} J_{ub} = \max_{\vec{p} \in \Omega} \quad & \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} v_i U_i (\sum_{j=1}^{\theta(i)} p_{ij}) \\ \textit{subject to} \quad & \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} p_{ij} r_i H_{ij}^l \le R^l \textit{ for all } l. \end{aligned} \tag{15}$$

Then

$$J^* \le J_{ub}$$

Proof: Fix a routing policy g. Let

$$p_{ij} = \frac{\lambda_{ij}}{\lambda_i}$$

Then

$$0 \le p_{ij} \le 1 \text{ and } \sum_{j=1}^{\theta(i)} p_{ij} \le 1$$

Hence, $\vec{p} \in \Omega$.

Let n_{ij} be the random variable that denotes the number of flows of class i routed to path j that are in the system at any time. Let $\mathbf{E}_g$ denote the expectation taken with respect to the stationary distribution under policy g. By Little's Law,

$$\mathbf{E}_g[n_{ij}] = \frac{\lambda_{ij}}{\mu_i}$$

Hence,

$$\frac{\lambda_i}{\mu_i} p_{ij} = \frac{\lambda_{ij}}{\mu_i} = \mathbf{E}_g[n_{ij}]$$

and

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

By definition of n_{ij}, at any time

$$\sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} n_{ij} r_i H_{ij}^l \leq R^l$$

Therefore,

$$\sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} p_{ij} r_i H_{ij}^l \leq R^l$$

i.e., $\vec{p}$ satisfies the constraint of (15). Therefore,

$$\sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} v_i U_i \left(\sum_{j=1}^{\theta(i)} \frac{\lambda_{ij}}{\lambda_i} \right) = \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} v_i U_i \left(\sum_{j=1}^{\theta(i)} p_{ij} \right) \leq J_{ub}$$

Since this is true for any policy g, hence,

$$J^* \leq J_{ub}$$

Q.E.D.

The performance of a static policy can be defined analogously to (14). That is, let $\vec{p}$ denote the static policy, the weighted utility under $\vec{p}$ is

$$J_0 = \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} v_i U_i \left(\sum_{j=1}^{\theta(i)} p_{ij} (1 - \mathbf{P}_{Loss,ij}) \right)$$

where $\mathbf{P}_{Loss,ij}$ is the blocking probability of flows of class i routed to path j. Let J_s denote the performance of the static policy induced by the solution to the upper bound, i.e., when $\vec{p}$ is the optimal point of (15).

Finally, we scale the capacity and the demand proportionally by $c > 1$, i.e., in the c-scaled network, the capacity at each link l is $R^{l,c} = cR^l$, and the arrival rate of each class i is $\lambda_i^c = c\lambda_i$. Let $J^{*,c}$ and J_s^c be the utility of the optimal dynamic scheme and the utility of the static scheme induced by the solution to the upper bound, respectively, in the c-scaled system, The following Lemma from [12] will be used in the main proof. Its own proof can be found in [11].

Lemma 4 *Assume that flows of class i arrive to path j according to a Poisson process with rate λ_{ij}. Let $1/\mu_i$ be the mean holding time. If the load at each resource is less than or equal to 1, i.e.,*

$$\sum_{i=1}^{I}\sum_{j=1}^{\theta(i)} \frac{\lambda_{ij}}{\mu_i} r_i H_{ij}^l \leq R^l \textit{ for all } l,$$

then under the aforementioned scaling where the arrival rate λ_{ij} and the capacity R^l are scaled proportionally by c, the blocking probability of each class goes to 0 as $c \to \infty$. The speed of convergence is at least $1/\sqrt{c}$.

Now we can proceed with the main proof of Proposition 1.

Proof of Proposition 1 : Firstly, note that the upper bound in the c-scaled system is obtained by

$$\begin{aligned} \max_{\vec{p}\in\Omega} \quad & \sum_{i=1}^{I} \frac{c\lambda_i}{\mu_i} v_i U_i(\sum_{j=1}^{\theta(i)} p_{ij}) \\ \text{subject to} \quad & \sum_{i=1}^{I}\sum_{j=1}^{\theta(i)} \frac{c\lambda_i}{\mu_i} p_{ij} r_i H_{ij}^l \leq cR^l \text{ for all } l. \end{aligned}$$

Hence, the upper bound is equal to cJ_{ub} where J_{ub} is the upper bound for the base system (i.e., $c = 1$). Further, the optimal point is independent of c.

Now consider J_s^c. Note that

$$J_s^c = \sum_{i=1}^{I} \frac{c\lambda_i}{\mu_i} v_i U_i \left(\sum_{j=1}^{\theta(i)} p_{ij}(1 - \mathbf{P}^c_{Loss,ij}) \right)$$

where $\vec{p} = [p_{ij}, i = 1, ..., I, j = 1, ..., \theta(i)]$ is the solution to the upper bound, and $\mathbf{P}^c_{Loss,ij}$ is the blocking probability of flows of class i on path j in the c-scaled system. Since the arrivals of each class i are Poisson with rate λ_i, the flows that are routed to path j by the static policy $\vec{p}$ also form a Poisson process with rate $\lambda_i p_{ij}$ and are independent of flows that are routed to other paths. Since $\vec{p}$ satisfies the constraint in (15), the load at each link is less that 1. Hence, by Lemma 4, the blocking probability $\mathbf{P}^c_{Loss,ij}$ goes to zero as $c \to \infty$. Note that U_i is continuous due to concavity, therefore,

$$\begin{aligned}
\lim_{c\to\infty} \frac{J_s^c}{c} &= \lim_{c\to\infty} \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} v_i U_i \left(\sum_{j=1}^{\theta(i)} p_{ij}(1 - \mathbf{P}^c_{Loss,ij}) \right) \\
&= \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} v_i U_i \left(\sum_{j=1}^{\theta(i)} p_{ij}\right) \\
&= J_{ub}
\end{aligned}$$

Finally, since $J_s^c \le J^{*,c} \le cJ_{ub}$, the result then follows. *Q.E.D.*

B Proof of Proposition 2

For simplicity of exposition, we will first prove the result for a simplified version of the problem. Next we will rephrase the problem and introdcue the notations. We will also rephrase algorithm $\mathcal{A}$ and the main proposition to be proved. The main proof then follows.

B.1 Preliminaries

We will first consider the following problem. Let d_i be a positive number for each i. Let $\vec{x}_i = [x_{ij}, j = 1, ...\theta(i)]$ be a vector in a convex set $C_i = \{x_{ij} \ge 0, \sum_{j=1}^{\theta(i)} x_{ij} \le d_i, j = 1, ..., \theta(i)\}$. Let

$\vec{x} = [\vec{x}_i, i = 1, ..., I]$ and $C = \bigotimes_{i=1}^{I} C_i$. Let f_i be a concave function on $\mathbf{R}$. The optimization problem that we are interested in is:

$$\max_{\vec{x} \in C} \qquad \sum_{i=1}^{I} f_i(\sum_{j=1}^{\theta(i)} x_{ij}) \tag{16}$$
$$\text{subject to} \qquad \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} H_{ij}^l x_{ij} \le R^l \quad \text{for all } l = 1, ..., L,$$

Compared with the original problem (3), (16) is of a simpler form since we remove other constants such as $\lambda_i, \mu_i, r_i, v_i$, etc. Once we prove the convergence for this problem, Proposition 2 follows by taking an appropriate mapping.

For convenience, we use vector notations wherever possible. Let H denote the matrix with L rows and $\sum_{i=1}^{I} \theta(i)$ columns such that the $(l, \sum_{k=1}^{i-1} \theta(k) + j)$ component is H_{ij}^l, i.e.,

$$H = \begin{bmatrix} H_{1,1}^1 & ... & H_{1,\theta(1)}^l & ... & H_{i,j}^l & ... & H_{I,\theta(I)}^l \\ H_{1,1}^2 & ... & H_{1,\theta(1)}^2 & ... & H_{i,j}^2 & ... & H_{I,\theta(I)}^2 \\ ... & ... & ... & ... & ... & ... & ... \\ H_{1,1}^l & ... & H_{1,\theta(1)}^l & ... & H_{i,j}^l & ... & H_{I,\theta(I)}^l \end{bmatrix}$$

Let $R = [R^1, R^2, ...R^l]^T$. Let A be the $L \times L$ diagonal matrix with diagonal terms being α^l.

In the sequel (except for the proof of Lemma 7), it will be convenient to refer to the objective function in (16) as a concave function of $\vec{x}$, i.e, $\mathbf{f}(\vec{x}) = \sum_{i=1}^{I} f_i(\sum_{j=1}^{\theta(i)} x_{ij})$. 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 (16) becomes:

$$\max_{\vec{x}} \qquad \mathbf{f}(\vec{x}) \tag{17}$$
$$\text{subject to} \qquad H\vec{x} \le R$$

Next we rephrase algorithm $\mathcal{A}$ for the problem (17). We will refer to the derivation in Section 3.1, and rewrite them using the new notations above. In the Proximal Optimization Algorithm, we will augment the objective function by a quadratic term. Let c_i be a positive number for each class i. Let V be the $\sum_{i=1}^{I} \theta(i) \times \sum_{i=1}^{I} \theta(i)$ diagonal matrix, with diagonal terms being c_i (each c_i is repeated $\theta(i)$ times). The augmented problem (5) in Section 3.1 becomes:

$$\begin{aligned} \max_{\vec{x}} \quad & \mathbf{f}(\vec{x}) - \frac{1}{2}(\vec{x}-\vec{y})^T V (\vec{x}-\vec{y}) \\ \text{subject to} \quad & H\vec{x} \le R \end{aligned} \tag{18}$$

Given $\vec{y}$, (18) can be solved through its dual. Let $\vec{q}$ be the vector of the Lagrange multipliers for the constraints in (18). The Lagrangian (6) in Section 3.1 becomes:

$$L(\vec{x},\vec{q},\vec{y}) = \mathbf{f}(\vec{x}) - \vec{x}^T H^T \vec{q} - \frac{1}{2}(\vec{x}-\vec{y})^T V (\vec{x}-\vec{y}) + \vec{q}^T R \tag{19}$$

The objective function of the dual problem is

$$D(\vec{q},\vec{y}) = \max_{\vec{x}} L(\vec{x},\vec{q},\vec{y})$$

and the dual problem of (18), given $\vec{y}$, is:

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

Since the objective function of the primal problem (18) is strictly concave, the dual is always differentiable. The gradient of D with respect to $\vec{q}$ is

$$\frac{\partial D}{\partial q^l} = R^l - H\vec{x}^0$$

where $\vec{x}^0$ optimizes $L(\vec{x},\vec{q},\vec{y})$ over all $\vec{x}$.

The augmented problem (18) can be solved by gradient descent iterations on the dual variables:

$$\vec{q}(t+1) = \left[\vec{q}(t) - A(R - H\vec{x}^0)\right]^+,$$

where $[.]^+$ denotes the projection to $[0,+\infty)$ component-wise.

This dual update then forms the step A1) in algorithm $\mathcal{A}$.

Finally, we allow a relaxation factor for the prime update in step A2). Let $0 < \beta_i \le 1$ for each class i. Let $\vec{z}(t)$ be the prime variable that corresponds to the new dual variable after the completion of K dual updates in algorithm $\mathcal{A}$. The new prime update rule is

$$y_{ij}(t+1) = y_{ij}(t) + \beta_i(z_{ij}(t) - y_{ij}(t))$$

The original prime update rule in Section 3.1 corresponds to $\beta_i = 1$ for all class i. Let B be the $\sum_{i=1}^{I} \theta(i) \times \sum_{i=1}^{I} \theta(i)$ diagonal matrix, with diagonal terms being β_i (each β_i is repeated $\theta(i)$ times).

To summarize, the algorithm $\mathcal{A}$ can be rewritten as:

- A1) Let $\vec{q}(t,0) = \vec{q}(t)$. Repeat for each $k = 0, 1, ...K-1$:

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

$$\vec{q}(t, k+1) = [\vec{q}(t,k) + A(H\vec{x}(t,k) - R)]^{+} \tag{20}$$

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

$$\vec{y}(t+1) = \vec{y}(t) + B(\vec{z}(t) - \vec{y}(t))$$

A stationary point of algorithm $\mathcal{A}$ is a primal-dual pair $(\vec{y^*}, \vec{q^*})$ such that

$$\vec{y^*} \text{ maximizes } L(\vec{x}, \vec{q^*}, \vec{y^*}) \text{ over all } \vec{x}$$

$$\vec{q^*} \text{ is also a stationary point of (20) .}$$

The main Proposition 2 can be rephrased as:

Proposition 5 *Fix $1 \le K \le \infty$. As long as the stepsize α^l is small enough, the algorithm $\mathcal{A}$ will converge to a stationary point $(\vec{y^*}, \vec{q^*})$ of the algorithm, and $\vec{x} = \vec{y^*}$ solves the problem (17). 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 $\mathcal{L} = \max\{\sum_{l=1}^{L} H_{ij}^l, i = 1, ..., I, j = 1, ...\theta(i)\}$ *and* $\mathcal{S} = \max\{\sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} H_{ij}^l, l = 1, ..., L\}$.

We next prove Proposition 5.

B.2 Proof of Proposition 5

We first collect some useful properties due to the concavity of the objective function $\mathbf{f}(\vec{x})$ and $f_i(x)$. Fix $\vec{y} = \vec{y}(t)$. Given an implicit cost vector $\vec{q}$, let $\vec{x} = \operatorname{argmax}_{\vec{x}} L(\vec{x}, \vec{q}, \vec{y}(t))$. By taking subgradients (see [19]) of the Lagrangian (19) with respect to $\vec{x}$, we have

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

where $\nabla \mathbf{f}(\vec{x})$ is an element of the subdifferential of $\mathbf{f}$ at $\vec{x}$.

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

$$\left[\nabla \mathbf{f}(\vec{x}_2) - \nabla \mathbf{f}(\vec{x}_1)\right]^T (\vec{x}_2 - \vec{x}_1) \leq 0 \tag{22}$$

An immediate consequence is that the mapping from $\vec{q}$ to $\vec{x}$ is continuous, as illustrated by the following Lemma: if $\vec{q}$ is changed little, so is $\vec{x} = \operatorname{argmax}_{\vec{x}} L(\vec{x}, \vec{q}, \vec{y}(t))$.

Lemma 6 *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 (19), 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) $(\vec{x}_2 - \vec{x}_1)^T V (\vec{x}_2 - \vec{x}_1) \leq (\vec{q}_2 - \vec{q}_1)^T H V^{-1} H^T (\vec{q}_2 - \vec{q}_1)$, and

2) $(\vec{q}_2 - \vec{q}_1)^T H (\vec{x}_2 - \vec{x}_1) \leq -(\vec{x}_2 - \vec{x}_1)^T V (\vec{x}_2 - \vec{x}_1)$

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

$$H^T(\vec{q}_2 - \vec{q}_1) \;=\; \left[\nabla \mathbf{f}(\vec{x}_2) - \nabla \mathbf{f}(\vec{x}_1)\right] - V(\vec{x}_2 - \vec{x}_1) \tag{23}$$

Hence,

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

where we have used (22). To show Part 2, note

$$
\begin{aligned}
& (\vec{q}_2 - \vec{q}_1)^T H(\vec{x}_2 - \vec{x}_1) \\
= \; & \left[\nabla \mathbf{f}(\vec{x}_2) - \nabla \mathbf{f}(\vec{x}_1)\right]^T (\vec{x}_2 - \vec{x}_1) - (\vec{x}_2 - \vec{x}_1)^T V(\vec{x}_2 - \vec{x}_1) \\
\leq \; & -(\vec{x}_2 - \vec{x}_1)^T V(\vec{x}_2 - \vec{x}_1)
\end{aligned}
$$

Q.E.D.

Next, let $(\vec{y^*}, \vec{q^*})$ denote a stationary point of algorithm $\mathcal{A}$. Then $\vec{y^*}$ maximizes $L(\vec{x}, \vec{q^*}, \vec{y^*})$ over all $\vec{x}$, and $\nabla \mathbf{f}(\vec{y^*}) - H^T \vec{q^*} = 0$ by (21). Using (22), we have

$$\left[\nabla \mathbf{f}(\vec{x}_1) - \nabla \mathbf{f}(\vec{y^*})\right]^T (\vec{x}_1 - \vec{y^*}) \leq 0. \tag{24}$$

The following Lemma can be viewed as an extension of the above inequality.

Lemma 7 *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 (19). Then,*

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

Remark: If $\vec{q}_2 = \vec{q}_1$, then $\vec{x}_2 = \vec{x}_1$ and we get back to (24). Lemma 7 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.

Proof: We need to use the facts that $\mathbf{f}(\vec{x})$ is of the form $\sum_{i=1}^{I} f_i(\sum_{j=1}^{\theta(i)} x_{ij})$, and C_i is of the form $C_i = \{x_{ij} \geq 0, \sum_{j=1}^{\theta(i)} x_{ij} \leq d_i, j = 1, ..., \theta(i)\}$. We rewrite the Lagrangian (19) as

$$L(\vec{x}, \vec{q}, \vec{y}) \;\; = \;\; \sum_{i=1}^{I} f_i(\sum_{j=1}^{\theta(i)} x_{ij}) - \vec{x}^T H^T \vec{q} - \frac{1}{2}(\vec{x} - \vec{y})^T V(\vec{x} - \vec{y}) + \vec{q}^T R \tag{25}$$

and the maximization of the Lagrangian has to be taken over $\vec{x}_i \in C_i$ for all i. We can incorporate the constraint $\sum_{j=1}^{\theta(i)} x_{ij} \leq d_i$ into the definition of the function f_i by setting $f_i(x) = -\infty$ when

$x > d_i$. Then the function f_i is still concave, and the maximization of the Lagrangian can be taken over all $\vec{x} \geq 0$.

Let $\vec{x} = \text{argmax}_{\vec{x}\geq 0} L(\vec{x}, \vec{q}, \vec{y})$ where $L(\vec{x}, \vec{q}, \vec{y})$ is as defined in (25). Let L_{ij} denote the Lagrange multiplier for the constraint $x_{ij} \geq 0$. Then, by taking subgradients of (25) with respect to x_{ij} and using the Karush-Kuhn-Tucker condition, we have, for all i, j:

$$\begin{aligned} &\partial f_i(\sum_{j=1}^{\theta(i)} x_{ij}) - \sum_{l=1}^{L} H_{ij}^l q^l - c_i(x_{ij} - y_{ij}) + L_{ij} = 0 \\ &L_{ij} \geq 0, x_{ij} \geq 0 \text{ and } L_{ij} x_{ij} = 0 \end{aligned} \tag{26}$$

where $\partial f_i(x)$ is an element of the subdifferential of f_i at x. Compared with (21), we see that

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

Define $L_{ij,1}$ and $L_{ij,2}$ analogously for the case when the implicit cost vectors are $\vec{q}_1$ and $\vec{q}_2$, respectively. Further, since $\vec{y^*} = \text{argmax}_{\vec{x}\geq 0} L(\vec{x}, \vec{q^*}, \vec{y^*})$, we again let L_{ij}^* denote the Lagrange multipliers for the constraint $x_{ij} \geq 0$. Then, we have, for all i, j,

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

Compared with $\nabla \mathbf{f}(\vec{y^*}) - H^T \vec{q^*} = 0$, we have

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

Therefore,

$$\begin{aligned} & \left[\nabla \mathbf{f}(\vec{x}_1) - \nabla \mathbf{f}(\vec{y^*})\right]^T (\vec{x}_2 - \vec{y^*}) \\ = & \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \left[(\nabla \mathbf{f}(\vec{x}_1))_{ij} - (\nabla \mathbf{f}(\vec{y^*}))_{ij}\right] (x_{ij,2} - y_{ij}^*) \\ = & \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \left[\partial f_i(\sum_{j=1}^{\theta(i)} x_{ij,1}) - \partial f_i(\sum_{j=1}^{\theta(i)} y_{ij}^*) + L_{ij,1} - L_{ij}^*\right] (x_{ij,2} - y_{ij}^*) \\ = & \sum_{i=1}^{I} \left[\partial f_i(\sum_{j=1}^{\theta(i)} x_{ij,1}) - \partial f_i(\sum_{j=1}^{\theta(i)} y_{ij}^*)\right] (\sum_{j=1}^{\theta(i)} x_{ij,2} - \sum_{j=1}^{\theta(i)} y_{ij}^*) \end{aligned} \tag{27}$$

$$+\sum_{i=1}^{I}\sum_{j=1}^{\theta(i)}(L_{ij,1}-L_{ij}^*)(x_{ij,2}-y_{ij}^*) \tag{28}$$

We will bound the two terms (27) and (28) seperately. Specificly, we will show next that for all i,

$$\begin{aligned}&\left[\partial f_i(\sum_{j=1}^{\theta(i)}x_{ij,1})-\partial f_i(\sum_{j=1}^{\theta(i)}y_{ij}^*)\right](\sum_{j=1}^{\theta(i)}x_{ij,2}-\sum_{j=1}^{\theta(i)}y_{ij}^*)\\ \leq\ & \frac{1}{4c_i}\sum_{j=1}^{\theta(i)}\left[\sum_{l=1}^{L}H_{ij}^l(q_2^l-q_1^l)\right]^2 \end{aligned}\tag{29}$$

and

$$\sum_{j=1}^{\theta(i)}(L_{ij,1}-L_{ij}^*)(x_{ij,2}-y_{ij}^*) \ \leq\ \frac{1}{4c_i}\sum_{j=1}^{\theta(i)}\left[\sum_{l=1}^{L}H_{ij}^l(q_2^l-q_1^l)\right]^2 \tag{30}$$

Lemma 7 then follows by substituting (29) and (30) into (27) and (28), respectively.

To show (29), apply equation (26) for $\vec{q}_1$ and $\vec{q}_2$, respectively, and take difference. We have, for each i,j,

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

Now fix i. Let J_i denote the set $\{j : x_{ij,2}>0 \text{ or } x_{ij,1}>0\}$. There are three cases for $j\in J_i$:

- For any j such that $x_{ij,2}>0$ and $x_{ij,1}=0$, we have $L_{ij,2}=0$ and $L_{ij,1}\geq 0$. Hence

  $$x_{ij,2}-x_{ij,1}>0 \text{ and } L_{ij,2}-L_{ij,1}\leq 0$$

  Let

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

  then,

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

or

$$\frac{1}{1+\gamma_{ij}} \sum_{l=1}^{L} H_{ij}^{l}(q_2^l - q_1^l) = \frac{1}{1+\gamma_{ij}} \left[\partial f_i(\sum_{j=1}^{\theta(i)} x_{ij,2}) - \partial f_i(\sum_{j=1}^{\theta(i)} x_{ij,1}) \right] - c_i(x_{ij,2} - x_{ij,1}) \quad (31)$$

- Similarly, for any j such that $x_{ij,2} = 0$ and $x_{ij,1} > 0$, we have $x_{ij,2} - x_{ij,1} < 0$ and $L_{ij,2} - L_{ij,1} \geq 0$. Let

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

  we obtain the same equation (31).

- Finally, for any j such that $x_{ij,2} > 0$ and $x_{ij,1} > 0$, we have $L_{ij,2} = L_{ij,1} = 0$. Let $\gamma_{ij} \triangleq 0$, then equation (31) also holds.

Summing (31) over all $j \in J_i$, we have

$$\sum_{j \in J_i} \left[\frac{1}{1+\gamma_{ij}} \sum_{l=1}^{L} H_{ij}^{l}(q_2^l - q_1^l) \right] = \frac{1}{\gamma_i^0} \left[\partial f_i(\sum_{j=1}^{\theta(i)} x_{ij,2}) - \partial f_i(\sum_{j=1}^{\theta(i)} x_{ij,1}) \right] - c_i(\sum_{j=1}^{\theta(i)} x_{ij,2} - \sum_{j=1}^{\theta(i)} x_{ij,1})$$

where we have used the fact that $x_{ij,2} = x_{ij,1} = 0$ for $j \notin J_i$, and γ_i^0 is defined as

$$\gamma_i^0 \triangleq \frac{1}{\sum_{j \in J_i} \frac{1}{1+\gamma_{ij}}}.$$

Since the function f_i is concave,

$$\left[\partial f_i(\sum_{j=1}^{\theta(i)} x_{ij,1}) - \partial f_i(\sum_{j=1}^{\theta(i)} y_{ij}^*) \right] (\sum_{j=1}^{\theta(i)} x_{ij,1} - \sum_{j=1}^{\theta(i)} y_{ij}^*) \leq 0 \quad (32)$$

Let

$$\gamma_i \triangleq -\frac{\gamma_i^0 c_i(\sum_{j=1}^{\theta(i)} x_{ij,1} - \sum_{j=1}^{\theta(i)} y_{ij}^*)}{\partial f_i(\sum_{j=1}^{\theta(i)} x_{ij,1}) - \partial f_i(\sum_{j=1}^{\theta(i)} y_{ij}^*)} \geq 0$$

(The inequality (29) trivially holds if $\partial f_i(\sum_{j=1}^{\theta(i)} x_{ij,1}) - \partial f_i(\sum_{j=1}^{\theta(i)} y_{ij}^*) = 0$.) Then

$$\left[\partial f_i(\sum_{j=1}^{\theta(i)} x_{ij,1}) - \partial f_i(\sum_{j=1}^{\theta(i)} y_{ij}^*) \right] (\sum_{j=1}^{\theta(i)} x_{ij,2} - \sum_{j=1}^{\theta(i)} y_{ij}^*)$$

$$\begin{aligned}
&= \frac{1}{1+\gamma_i}\left[\partial f_i(\sum_{j=1}^{\theta(i)} x_{ij,1}) - \partial f_i(\sum_{j=1}^{\theta(i)} y_{ij}^*) - c_i\gamma_i^0(\sum_{j=1}^{\theta(i)} x_{ij,1} - \sum_{j=1}^{\theta(i)} y_{ij}^*)\right](\sum_{j=1}^{\theta(i)} x_{ij,2} - \sum_{j=1}^{\theta(i)} y_{ij}^*) \\
&= \frac{1}{1+\gamma_i}\left[\partial f_i(\sum_{j=1}^{\theta(i)} x_{ij,1}) - \partial f_i(\sum_{j=1}^{\theta(i)} x_{ij,2}) - c_i\gamma_i^0(\sum_{j=1}^{\theta(i)} x_{ij,1} - \sum_{j=1}^{\theta(i)} x_{ij,2})\right](\sum_{j=1}^{\theta(i)} x_{ij,2} - \sum_{j=1}^{\theta(i)} y_{ij}^*) \\
&\quad + \frac{1}{1+\gamma_i}\left[\partial f_i(\sum_{j=1}^{\theta(i)} x_{ij,2}) - \partial f_i(\sum_{j=1}^{\theta(i)} y_{ij}^*) - c_i\gamma_i^0(\sum_{j=1}^{\theta(i)} x_{ij,2} - \sum_{j=1}^{\theta(i)} y_{ij}^*)\right](\sum_{j=1}^{\theta(i)} x_{ij,2} - \sum_{j=1}^{\theta(i)} y_{ij}^*) \\
&\leq \frac{1}{1+\gamma_i}\gamma_i^0 \sum_{j\in J_i}\left[\frac{1}{1+\gamma_{ij}}\sum_{l=1}^{L} H_{ij}^l(q_2^l - q_1^l)\right](\sum_{j=1}^{\theta(i)} x_{ij,2} - \sum_{j=1}^{\theta(i)} y_{ij}^*) \\
&\quad - \frac{1}{1+\gamma_i}c_i\gamma_i^0(\sum_{j=1}^{\theta(i)} x_{ij,2} - \sum_{j=1}^{\theta(i)} y_{ij}^*)^2 \\
&\leq \frac{\gamma_i^0}{1+\gamma_i}\frac{1}{4c_i}\left\{\sum_{j\in J_i}\left[\frac{1}{1+\gamma_{ij}}\sum_{l=1}^{L} H_{ij}^l(q_2^l - q_1^l)\right]\right\}^2 \\
&\leq \frac{1}{1+\gamma_i}\frac{1}{4c_i}\gamma_i^0\left\{\sum_{j\in J_i}(\frac{1}{1+\gamma_{ij}})^2\right\}\left\{\sum_{j\in J_i}\left[\sum_{l=1}^{L} H_{ij}^l(q_2^l - q_1^l)\right]^2\right\} \\
&\leq \frac{1}{4c_i}\sum_{j=1}^{\theta(i)}\left[\sum_{l=1}^{L} H_{ij}^l(q_2^l - q_1^l)\right]^2
\end{aligned}$$

where in the last inequality we have used $1/(1+\gamma_i) \leq 1$ and

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

This proves (29). To show (30), 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,

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

Hence

$$\sum_{j=1}^{\theta(i)}(L_{ij,1} - L_{ij}^*)(x_{ij,2} - y_{ij}^*) \leq -\sum_{j=1}^{\theta(i)}(x_{ij,2} - x_{ij,1})(L_{ij,2} - L_{ij,1})$$

Using the concavity of f_i again (see (32)),

$$
\begin{aligned}
& -\sum_{j=1}^{\theta(i)}(x_{ij,2}-x_{ij,1})(L_{ij,2}-L_{ij,1}) \\
\leq \quad & -\sum_{j=1}^{\theta(i)}\left[\partial f_i(\sum_{j=1}^{\theta(i)} x_{ij,2}) - \partial f_i(\sum_{j=1}^{\theta(i)} x_{ij,1}) + L_{ij,2} - L_{ij,1}\right](x_{ij,2}-x_{ij,1}) \\
\leq \quad & -\sum_{j=1}^{\theta(i)}\left[\sum_{l=1}^{L} H_{ij}^l(q_2^l - q_1^l) + c_i(x_{ij,2}-x_{ij,1})\right](x_{ij,2}-x_{ij,1}) \\
\leq \quad & \frac{1}{4c_i}\sum_{j=1}^{\theta(i)}\left[\sum_{l=1}^{L} H_{ij}^l(q_2^l - q_1^l)\right]^2
\end{aligned}
$$

This proves (30). The result of Lemma 7 then follows. *Q.E.D.*

Finally, the following lemma will be quite convenient later on.

Lemma 8 *If*

$$\max_l \alpha^l < \frac{a}{\mathcal{SL}} \min_i c_i \text{ for some positive number } a,$$

then

$$aV > H^T A H \text{ and } aA^{-1} > HV^{-1}H^T$$

Proof: To show the first part, let $\vec{x}$ be any vector,

$$
\begin{aligned}
& \vec{x}^T H^T A H \vec{x} = \sum_{l=1}^{L} \alpha^l \left[\sum_{i=1}^{I}\sum_{j=1}^{\theta(i)} H_{ij}^l x_{ij}\right]^2 \\
\leq \quad & \sum_{l=1}^{L}\alpha^l(\sum_{i=1}^{I}\sum_{j=1}^{\theta(i)} H_{ij}^l)\left[\sum_{i=1}^{I}\sum_{j=1}^{\theta(i)} H_{ij}^l x_{ij}^2\right] \leq \mathcal{S}\sum_{i=1}^{I}\sum_{j=1}^{\theta(i)}\left[\sum_{l=1}^{L}\alpha^l H_{ij}^l\right] x_{ij}^2 \\
\leq \quad & \mathcal{SL}\max_l \alpha^l \sum_{i=1}^{I}\sum_{j=1}^{\theta(i)} x_{ij}^2
\end{aligned}
$$

Hence a sufficent condition for $aV - H^T AH$ to be positive definite is

$$\max_l \alpha_l < \frac{a}{\mathcal{SL}} \min_i c_i$$

The second part can be shown analogously. *Q.E.D.*

Now we can proceed with the main proof. Define

$$
\begin{aligned}
||\vec{q}||_A &= \vec{q}^T A^{-1} \vec{q} = \sum_{l=1}^{L} \frac{1}{\alpha^l} (q^l)^2 \\
||\vec{y}||_V &= \vec{y}^T V \vec{y} = \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} c_i y_{ij}^2 \\
||\vec{y}||_{BV} &= \vec{y}^T B^{-1} V \vec{y} = \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{1}{\beta_i} c_i y_{ij}^2
\end{aligned}
$$

There are three cases:

B.2.1 $K = \infty$

We only need to show that the step (A1) converges. The convergence of the entire algorithm then follows the standard results on Proximal Optimization Algorithms [3]. Fix $\vec{y}(t)$. Let $\vec{q}_0$ denote a stationary point of (20) in step (A1). Let $\vec{x}_0$ be the corresponding prime variables. $\vec{x}_0$ is unique and $\vec{x}_0 = \text{argmax}_{\vec{x}} L(\vec{x}, \vec{q}_0, \vec{y}(t))$. Using the property of projection mapping [3, Proposition 3.2(c), p211], we have

$$
\begin{aligned}
& ||\vec{q}(t,k+1) - \vec{q}_0||_A \\
= \; & ||\, [\vec{q}(t,k) + A(H\vec{x}(t,k) - R)]^+ - [\vec{q}_0 + A(H\vec{x}_0 - R)]^+ \,||_A \\
\le \; & ||\, [\vec{q}(t,k) + A(H\vec{x}(t,k) - R)] - [\vec{q}_0 + A(H\vec{x}_0 - R)] \,||_A \\
= \; & ||\, [\vec{q}(t,k) - \vec{q}_0] + AH(\vec{x}(t,k) - \vec{x}_0)||_A \\
= \; & ||\vec{q}(t,k) - \vec{q}_0||_A + (\vec{x}(t,k) - \vec{x}_0)^T H^T A H (\vec{x}(t,k) - \vec{x}_0) \\
& +2(\vec{q}(t,k) - \vec{q}_0)^T H (\vec{x}(t,k) - \vec{x}_0)
\end{aligned}
$$

By Lemma 6,

$$
(\vec{q}(t,k) - \vec{q}_0)^T H (\vec{x}(t,k) - \vec{x}_0) \;\le\; -(\vec{x}(t,k) - \vec{x}_0)^T V (\vec{x}(t,k) - \vec{x}_0)
$$

Hence, if we can ensure that

$$C_0 = 2V - H^T AH$$

is positive definite, then

$$\begin{aligned} ||\vec{q}(t,k+1) - \vec{q}_0||_A &\leq ||\vec{q}(t,k) - \vec{q}_0||_A - (\vec{x}(t,k) - \vec{x}_0)^T C_0 (\vec{x}(t,k) - \vec{x}_0) \\ &\leq ||\vec{q}(t,k) - \vec{q}_0||_A \end{aligned} \tag{33}$$

Therefore, $||\vec{q}(t,k) - \vec{q}_0||_A, k = 1, 2, \ldots$ is a nonnegative and decreasing sequence, and hence must have a limit. Then by (33), $\vec{x}(t,k) \to \vec{x}_0$ as $k \to \infty$. By Lemma 8, a sufficent condition for C_0 to be positive definite is

$$\max_l \alpha_l < \frac{2}{\mathcal{SL}} \min_i c_i$$

B.2.2 $K = 1$

When $K = 1$,

$$\vec{q}(t+1) = [\vec{q}(t) + A(H\vec{x}(t) - R)]^+ \tag{34}$$

Let $(\vec{y^*}, \vec{q^*})$ be any stationary point of algorithm $\mathcal{A}$. Using the property of the projection mapping, ([3, Proposition 3.2(b), p211], we have

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

Hence,

$$\begin{aligned} ||\vec{q}(t+1) - \vec{q^*}||_A &= ||\vec{q}(t) - \vec{q^*}||_A - ||\vec{q}(t+1) - \vec{q}(t)||_A + 2(\vec{q}(t+1) - \vec{q^*})^T A^{-1}(\vec{q}(t+1) - \vec{q}(t)) \\ &\leq ||\vec{q}(t) - \vec{q^*}||_A - ||\vec{q}(t+1) - \vec{q}(t)||_A + 2(\vec{q}(t+1) - \vec{q^*})^T (H\vec{x}(t) - R) \end{aligned} \tag{35}$$

Since $(\vec{y^*}, \vec{q^*})$ is a stationary point of the algorithm, $H\vec{y^*} - R \leq 0$ and $\vec{q^*}^T (H\vec{y^*} - R) = 0$, hence

$$||\vec{q}(t+1) - \vec{q^*}||_A \leq ||\vec{q}(t) - \vec{q^*}||_A - ||\vec{q}(t+1) - \vec{q}(t)||_A + 2(\vec{q}(t+1) - \vec{q^*})^T H(\vec{x}(t) - \vec{y^*})$$

Since $y_{ij}(t+1) = (1-\beta_i)y_{ij}(t) + \beta_i z_{ij}(t)$, hence

$$\begin{aligned} (y_{ij}(t+1) - y^*_{ij})^2 &\leq (1-\beta_i)(y_{ij}(t) - y^*_{ij})^2 + \beta_i (z_{ij}(t) - y^*_{ij})^2 \\ ||\vec{y}(t+1) - \vec{y^*}||_{BV} - ||\vec{y}(t) - \vec{y^*}||_{BV} &\leq ||\vec{z}(t) - \vec{y^*}||_V - ||\vec{y}(t) - \vec{y^*}||_V \end{aligned} \tag{36}$$

Hence,

$$
\begin{aligned}
& ||\vec{q}(t+1)-\vec{q^*}||_A + ||\vec{y}(t+1)-\vec{y^*}||_{BV} - (||\vec{q}(t)-\vec{q^*}||_A + ||\vec{y}(t)-\vec{y^*}||_{BV}) \\
\leq\ & -||\vec{q}(t+1)-\vec{q}(t)||_A + 2(\vec{q}(t+1)-\vec{q^*})^T H(\vec{x}(t)-\vec{y^*}) + ||\vec{z}(t)-\vec{y^*})||_V - ||\vec{y}(t)-\vec{y^*}||_V \\
\leq\ & -||\vec{q}(t+1)-\vec{q}(t)||_A \\
& + \left\{||\vec{z}(t)-\vec{y^*})||_V - ||\vec{y}(t)-\vec{y^*}||_V - 2(\vec{z}(t)-\vec{y}(t))^T V(\vec{x}(t)-\vec{y^*})\right\} \qquad (37) \\
& +2\left[\nabla\mathbf{f}(\vec{z}(t)) - \nabla\mathbf{f}(\vec{y^*})\right]^T (\vec{x}(t)-\vec{y^*}) \qquad (38)
\end{aligned}
$$

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

$$
H^T(\vec{q}(t+1)-\vec{q^*}) = \nabla\mathbf{f}(\vec{z}(t)) - \nabla\mathbf{f}(\vec{y^*}) - V(\vec{z}(t)-\vec{y}(t)). \qquad (39)
$$

The second term (37) is equal to

$$
\begin{aligned}
& ||\vec{z}(t)-\vec{y^*}||_V - ||\vec{y}(t)-\vec{y^*}||_V - 2(\vec{z}(t)-\vec{y}(t))^T V(\vec{x}(t)-\vec{y^*}) \\
=\ & \left[||\vec{z}(t)-\vec{y^*}||_V + ||\vec{x}(t)-\vec{y^*}||_V - 2(\vec{z}(t)-\vec{y^*})^T V(\vec{x}(t)-\vec{y^*})\right] \\
& -\left[||\vec{y}(t)-\vec{y^*}||_V + ||\vec{x}(t)-\vec{y^*}||_V - 2(\vec{y}(t)-\vec{y^*})^T V(\vec{x}(t)-\vec{y^*})\right] \\
=\ & ||(\vec{z}(t)-\vec{x}(t)||_V - ||\vec{y}(t)-\vec{x}(t)||_V \qquad (40)
\end{aligned}
$$

Invoking Lemma 6,

$$
||\vec{z}(t)-\vec{x}(t)||_V \leq (\vec{q}(t+1)-\vec{q}(t))^T H V^{-1} H^T (\vec{q}(t+1)-\vec{q}(t)) \qquad (41)
$$

For the third term (38), we can invoke Lemma 7,

$$
2\left[\nabla\mathbf{f}(\vec{z}(t)) - \nabla\mathbf{f}(\vec{y^*})\right]^T (\vec{x}(t)-\vec{y^*}) \leq (\vec{q}(t+1)-\vec{q}(t))^T H V^{-1} H^T (\vec{q}(t+1)-\vec{q}(t)) \qquad (42)
$$

Therefore, by substituting (40-42) into (37-38), we have

$$
\begin{aligned}
& ||\vec{q}(t+1)-\vec{q^*}||_A + ||\vec{y}(t+1)-\vec{y^*}||_{BV} - (||\vec{q}(t)-\vec{q^*}||_A + ||\vec{y}(t)-\vec{y^*}||_{BV}) \\
\leq\ & -||\vec{q}(t+1)-\vec{q}(t)||_A - ||\vec{y}(t)-\vec{x}(t)||_V + 2(\vec{q}(t+1)-\vec{q}(t))^T H V^{-1} H^T (\vec{q}(t+1)-\vec{q}(t))
\end{aligned}
$$

Hence, if the matrix

$$C_1 = A^{-1} - 2HV^{-1}H^T$$

is positive definite, then

$$\begin{aligned} & ||\vec{q}(t+1) - \vec{q^*}||_A + ||\vec{y}(t+1) - \vec{y^*}||_{BV} - (||\vec{q}(t) - \vec{q^*}||_A + ||\vec{y}(t) - \vec{y^*}||_{BV}) \\ \leq & \ -(\vec{q}(t+1) - \vec{q}(t))^T C_1 (\vec{q}(t+1) - \vec{q}(t)) - ||\vec{y}(t) - \vec{x}(t)||_V \leq 0 \end{aligned} \tag{43}$$

Hence the sequence $\{\vec{y}(t), \vec{q}(t), t = 1, ...\}$ is bounded. There must exists 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. From (43), we have,

$$\lim_{h\to\infty} \vec{q}(t_h + 1) \ = \ \vec{q}_0 \text{ and } \lim_{h\to\infty} \vec{x}(t_h) = \vec{y}_0$$

Taking limits at both sides of (34) as $h \to \infty$, we have,

$$\vec{q}_0 = [\vec{q}_0 + A(H\vec{y}_0 - R)]^+$$

Further, note that $\vec{x}(t_h)$ maximizes $L(\vec{x}, \vec{q}(t_h), \vec{y}(t_h))$ over all $\vec{x}$. Similar to Lemma 6, one can show that the mapping from $(\vec{y}(t_h), \vec{q}(t_h))$ to $\vec{x}(t_h)$ is continuous. Hence taking limits as $h \to \infty$, we have

$$\vec{y}_0 \text{ maximizes } L(\vec{x}, \vec{q}_0, \vec{y}_0) \text{ over all } \vec{x}$$

Therefore $(\vec{y}_0, \vec{q}_0)$ is a stationary point of the algorithm. Substituting $\vec{q^*} = \vec{q}_0$ and $\vec{y^*} = \vec{y}_0$ in (43), we have

$$\lim_{t\to\infty} ||\vec{q}(t) - \vec{q}_0||_A + ||\vec{y}(t) - \vec{y}_0||_{BV} \ = \ \lim_{h\to\infty} ||\vec{q}(t_h) - \vec{q}_0||_A + ||\vec{y}(t_h) - \vec{y}_0||_{BV} = 0.$$

Hence $(\vec{y}(t), \vec{q}(t)) \to (\vec{y}_0, \vec{q}_0)$ as $t \to \infty$. Finally, by Lemma 8, a sufficent condition for C_1 to be positive definite is

$$\max_l \alpha^l < \frac{1}{2\mathcal{SL}} \min_i c_i$$

B.2.3 $K > 1$

Let $(\vec{y^*}, \vec{q^*})$ be any stationary point of algorithm $\mathcal{A}$. We only need to show that the following inequality holds: for sufficiently small stepsize $\alpha^l, l = 1, ..., L$,

$$\begin{aligned} & ||\vec{q}(t+1) - \vec{q^*}||_A + K||\vec{y}(t+1) - \vec{y^*}||_{BV} - (||\vec{q}(t) - \vec{q^*}||_A + K||\vec{y}(t) - \vec{y^*}||_{BV}) \\ \leq \; & -(\vec{q}(t+1) - \vec{q}(t))^T C_2 (\vec{q}(t+1) - \vec{q}(t)) - ||\vec{y}(t) - \vec{x}(t)||_V \end{aligned} \tag{44}$$

where C_2 is some positive definite matrix. This corresponds to the inequality (43) for the case when $K = 1$. The proof then follows along the same line as the case when $K = 1$.

To show (44), we start from (35). For each $k = 0, 1, ...K$,

$$\begin{aligned} & ||\vec{q}(t, k+1) - \vec{q^*}||_A - ||\vec{q}(t, k) - \vec{q^*}||_A \\ \leq \; & -||\vec{q}(t, k+1) - \vec{q}(t, k)||_A + 2(\vec{q}(t, k+1) - \vec{q^*})^T (H\vec{x}(t, k) - R) \end{aligned} \tag{45}$$

Since for any $k \leq K - 2$,

$$\begin{aligned} \vec{q}(t, k+1) - \vec{q^*} &= \vec{q}(t, K) - \vec{q^*} - (\vec{q}(t, K) - \vec{q}(t, k+1)) \\ &= \vec{q}(t, K) - \vec{q^*} - \sum_{m=k+1}^{K-1} \gamma_m A(H\vec{x}(t, m) - R) \end{aligned}$$

where γ_m reflects the "truncation" when the implicit costs are projected to $\mathbf{R}^+$ and $0 \leq \gamma_m \leq 1$. Hence,

$$\begin{aligned} & 2(\vec{q}(t, k+1) - \vec{q^*})^T (H\vec{x}(t, k) - R) \qquad (46) \\ = \; & 2(\vec{q}(t, K) - \vec{q^*})^T (H\vec{x}(t, k) - R) - 2 \sum_{m=k+1}^{K-1} \gamma_m (H\vec{x}(t, m) - R)^T A(H\vec{x}(t, k) - R) \end{aligned}$$

If we choose α^l such that

$$\max_l \alpha^l < \frac{1}{K\mathcal{S}\mathcal{L}} \min_i c_i \tag{47}$$

then by Lemma 8, $H^T A H < \frac{1}{K} V$. Hence,

$$-2 \sum_{m=k+1}^{K-1} \gamma_m (H\vec{x}(t, m) - R)^T A(H\vec{x}(t, k) - R)$$

$$
\begin{aligned}
&\leq \frac{1}{2}\sum_{m=k+1}^{K-1}\gamma_m(\vec{x}(t,m)-\vec{x}(t,k))^T H^T A H(\vec{x}(t,m)-\vec{x}(t,k)) \\
&\leq \frac{1}{2K}\sum_{m=k+1}^{K-1}||\vec{x}(t,m)-\vec{x}(t,k)||_V \\
&\leq \frac{1}{2K}\sum_{m=k+1}^{K-1}(\vec{q}(t,m)-\vec{q}(t,k))^T H V^{-1} H(\vec{q}(t,m)-\vec{q}(t,k)) \qquad (48)
\end{aligned}
$$

by Lemma 6. Further, Since $(\vec{y^*},\vec{q^*})$ is a stationary point of the algorithm, $H\vec{y^*}-R\leq 0$ and $\vec{q^*}^T(H\vec{y^*}-R)=0$. Combining this, (46) and (48) into (45), we have,

$$
\begin{aligned}
&||\vec{q}(t,k+1)-\vec{q^*}||_A-||\vec{q}(t,k)-\vec{q^*}||_A \\
\leq\ &-||\vec{q}(t,k+1)-\vec{q}(t,k)||_A+2(\vec{q}(t,K)-\vec{q^*})^T H(\vec{x}(t,k)-\vec{y^*}) \\
&+\frac{1}{2K}\sum_{m=k+1}^{K-1}(\vec{q}(t,m)-\vec{q}(t,k))^T H V^{-1} H(\vec{q}(t,m)-\vec{q}(t,k))
\end{aligned}
$$

Summing over all $k=0,1,...K-1$, we have

$$
\begin{aligned}
&||\vec{q}(t,K)-\vec{q^*}||_A-||\vec{q}(t,0)-\vec{q^*}||_A \\
\leq\ &-\sum_{k=0}^{K-1}||\vec{q}(t,k+1)-\vec{q}(t,k)||_A+2\sum_{k=0}^{K-1}(\vec{q}(t,K)-\vec{q^*})^T H(\vec{x}(t,k)-\vec{y^*}) \\
&+\frac{1}{2K}\sum_{k=0}^{K-2}\sum_{m=k+1}^{K-1}(\vec{q}(t,m)-\vec{q}(t,k))^T H V^{-1} H^T(\vec{q}(t,m)-\vec{q}(t,k))
\end{aligned}
$$

Therefore, using (36) and (39), we have,

$$
\begin{aligned}
&||\vec{q}(t,K)-\vec{q^*}||_A-||\vec{q}(t,0)-\vec{q^*}||_A+K(||\vec{y}(t+1)-\vec{y^*}||_{BV}-||\vec{y}(t)-\vec{y^*}||_{BV}) \\
\leq\ &-\sum_{k=0}^{K-1}||\vec{q}(t,k+1)-\vec{q}(t,k)||_A \\
&+\left\{K||\vec{z}(t)-\vec{y^*})||_V-K||\vec{y}(t)-\vec{y^*}||_V-2\sum_{k=0}^{K-1}(\vec{z}(t)-\vec{y}(t))^T V(\vec{x}(t,k)-\vec{y^*})\right\} \qquad (49) \\
&+2\sum_{k=0}^{K-1}(\nabla\mathbf{f}(\vec{z}(t)-\nabla\mathbf{f}(\vec{y^*}))^T(\vec{x}(t,k)-\vec{y^*}) \qquad (50) \\
&+\frac{1}{2K}\sum_{k=0}^{K-2}\sum_{m=k+1}^{K-1}(\vec{q}(t,m)-\vec{q}(t,k))^T H V^{-1} H^T(\vec{q}(t,m)-\vec{q}(t,k))
\end{aligned}
$$

The second term (49) on the right hand side can be bounded by

$$\begin{aligned}
& K||\vec{z}(t)-\vec{y^*})||_V - K||\vec{y}(t)-\vec{y^*}||_V - 2\sum_{k=0}^{K-1}(\vec{z}(t)-\vec{y^*})^T V(\vec{x}(t,k)-\vec{y}(t)) \\
= & -\sum_{k=0}^{K-1}||\vec{x}(t,k)-\vec{y}(t)||_V + \sum_{k=0}^{K-1}||\vec{z}(t)-\vec{x}(t,k)||_V \\
\leq & -\sum_{k=0}^{K-1}||\vec{x}(t,k)-\vec{y}(t)||_V + \sum_{k=0}^{K-1}(\vec{q}(t,K)-\vec{q}(t,k))^T HV^{-1}H^T(\vec{q}(t,K)-\vec{q}(t,k))
\end{aligned}$$

by Lemma 6. For the third term (50), by Lemma 7,

$$\begin{aligned}
& 2\sum_{k=0}^{K-1}(\nabla\mathbf{f}(\vec{z}(t)-\nabla\mathbf{f}(\vec{y^*}))^T(\vec{x}(t,k)-\vec{y^*}) \\
\leq & \sum_{k=0}^{K-1}(\vec{q}(t,k)-\vec{q}(t,K))^T HV^{-1}H^T(\vec{q}(t,k)-\vec{q}(t,K))
\end{aligned}$$

Substituting them back to (49) and (50), we have

$$\begin{aligned}
& ||\vec{q}(t,K)-\vec{q^*}||_A - ||\vec{q}(t,0)-\vec{q^*}||_A + K(||\vec{y}(t+1)-\vec{y^*}||_{BV} - ||\vec{y}(t)-\vec{y^*}||_{BV}) \\
\leq & -\sum_{k=0}^{K-1}||\vec{q}(t,k+1)-\vec{q}(t,k)||_A - \sum_{k=0}^{K-1}||\vec{x}(t,k)-\vec{y}(t)||_V \\
& +2\sum_{k=0}^{K-1}(\vec{q}(t,k)-\vec{q}(t,K))^T HV^{-1}H^T(\vec{q}(t,k)-\vec{q}(t,K)) \\
& +\frac{1}{2K}\sum_{k=0}^{K-2}\sum_{m=k+1}^{K-1}(\vec{q}(t,m)-\vec{q}(t,k))^T HC_1H^T(\vec{q}(t,m)-\vec{q}(t,k))
\end{aligned}$$

Since

$$\vec{q}(t,K)-\vec{q}(t,k) = \sum_{m=k}^{K-1}\vec{q}(t,m+1)-\vec{q}(t,m)$$

using the Schwarz Inequality, we can show that,

$$\begin{aligned}
& \sum_{k=0}^{K-1}(\vec{q}(t,k)-\vec{q}(t,K))^T HV^{-1}H^T(\vec{q}(t,k)-\vec{q}(t,K)) \\
\leq & \sum_{k=0}^{K-1}(K-k)\sum_{m=k}^{K-1}(\vec{q}(t,m+1)-\vec{q}(t,m))^T HV^{-1}H^T(\vec{q}(t,m+1)-\vec{q}(t,m)) \\
= & \sum_{m=0}^{K-1}\left[\sum_{k=0}^{m}(K-k)\right](\vec{q}(t,m+1)-\vec{q}(t,m))^T HV^{-1}H^T(\vec{q}(t,m+1)-\vec{q}(t,m)) \\
\leq & \frac{K(K+1)}{2}\sum_{m=0}^{K-1}(\vec{q}(t,m+1)-\vec{q}(t,m))^T HV^{-1}H^T(\vec{q}(t,m+1)-\vec{q}(t,m))
\end{aligned}$$

Similarly,

$$\begin{aligned}
& \frac{1}{K}\sum_{k=0}^{K-2}\sum_{m=k+1}^{K-1}(\vec{q}(t,m)-\vec{q}(t,k))^T HV^{-1}H^T(\vec{q}(t,m)-\vec{q}(t,k)) \\
\leq\ & \frac{1}{K}\sum_{k=0}^{K-2}\sum_{m=k+1}^{K-1}(m-k)\sum_{n=k}^{m-1}(\vec{q}(t,n+1)-\vec{q}(t,n))^T HV^{-1}H(\vec{q}(t,n+1)-\vec{q}(t,n)) \\
=\ & \frac{1}{K}\sum_{n=0}^{K-2}\left[\sum_{k=0}^{n}\sum_{m=n+1}^{K-1}(m-k)\right](\vec{q}(t,n+1)-\vec{q}(t,n))^T HV^{-1}H^T(\vec{q}(t,n+1)-\vec{q}(t,n)) \\
=\ & \frac{K(K+1)}{2}\sum_{n=0}^{K-2}(\vec{q}(t,n+1)-\vec{q}(t,n))^T HV^{-1}H^T(\vec{q}(t,n+1)-\vec{q}(t,n))
\end{aligned}$$

Hence,

$$\begin{aligned}
& ||\vec{q}(t,K)-\vec{q^*}||_A-||\vec{q}(t,0)-\vec{q^*}||_A+K(||\vec{y}(t+1)-\vec{y^*}||_{BV}-||\vec{y}(t)-\vec{y^*}||_{BV}) \\
\leq\ & -\sum_{k=0}^{K-1}||\vec{q}(t,k+1)-\vec{q}(t,k)||_A-\sum_{k=0}^{K-1}||\vec{x}(t,k)-\vec{y}(t)||_V \\
& +\frac{5K(K+1)}{4}\sum_{m=0}^{K-1}(\vec{q}(t,m+1)-\vec{q}(t,m))^T HV^{-1}H^T(\vec{q}(t,m+1)-\vec{q}(t,m))
\end{aligned}$$

Hence, in order to have

$$||\vec{q}(t,K)-\vec{q^*}||_A-||\vec{q}(t,0)-\vec{q^*}||_A+K(||\vec{y}(t+1)-\vec{y^*}||_{BV}-||\vec{y}(t)-\vec{y^*}||_{BV})\leq 0$$

it is sufficient to let

$$C_2=A^{-1}-\frac{5K(K+1)}{4}HV^{-1}H^T$$

be positive definite and to satisfy (47). By Lemma 8, a sufficient condition is

$$\max_l \alpha^l<\frac{4}{5K(K+1)\mathcal{S}\mathcal{L}}\min_i c_i$$

which also satisfies (47) automatically.

B.3 Mapping to Proposition 2

Finally, to map to the original problem (3), we only need to take the following mapping:

$$x_{ij} \quad = \quad \frac{\lambda_i r_i}{\mu_i}p_{ij}$$

$$
\begin{aligned}
c_i &= \frac{\mu_i v_i}{\lambda_i r_i^2}\nu_i \\
f_i(x) &= \frac{\lambda_i v_i}{\mu_i} U_i(\frac{\mu_i}{\lambda_i r_i} x)
\end{aligned}
$$

Then the problem (3) and algorithm $\mathcal{A}$ in Section 3.1 can be mapped to the problem (16) and algorithm $\mathcal{A}$ in this Section. By substituting c_i into the sufficient condition, Proposition 2 follows.

C An Efficient Algorithm for Solving the Local Subproblem (8)

Given the implicit costs $\vec{q}$, each class i solves its local subproblem (8) to obtain the routing probabilities. Recall that the local subproblem is:

$$
\begin{aligned}
B_i(\vec{q}_i, \vec{y}_i) &= \max_{\vec{p}_i \in \Omega_i} \left\{ U_i(\sum_{j=1}^{\theta(i)} p_{ij}) v_i - r_i \sum_{j=1}^{\theta(i)} p_{ij} q_{ij} \right. \\
&\qquad \left. - \sum_{j=1}^{\theta(i)} \frac{\nu_i}{2}(p_{ij} - y_{ij})^2 v_i \right\}. \qquad (51)
\end{aligned}
$$

where

$$
\Omega_i \triangleq \{p_{ij} \geq 0, \sum_{j=1}^{\theta(i)} p_{ij} \leq 1, \text{ for all } j\}.
$$

Let L_j be the Lagrangian multiplier for the constraint $p_{ij} \geq 0$, and let L_0 be the Lagrangian multiplier for the constraint $\sum_{j=1}^{\theta(i)} p_{ij} \leq 1$. Then, the Karush-Kuhn-Tucker condition becomes:

$$
\begin{aligned}
& L_0 \geq 0, L_j \geq 0, j = 1, ..., \theta(i) \\
& p_{ij} \geq 0, \sum_{j=1}^{\theta(i)} p_{ij} \leq 1, j = 1, ..., \theta(i) \\
& L_j p_{ij} = 0, L_0(\sum_{j=1}^{\theta(i)} p_{ij} - 1) = 0, j = 1, ..., \theta(i) \\
& U_i'(\sum_{j=1}^{\theta(i)} p_{ij}) v_i - r_i q_{ij} - \nu_i (p_{ij} - y_{ij}) v_i + L_j - L_0 = 0
\end{aligned}
$$

Let $Q_{ij} = \nu_i y_{ij} v_i - r_i q_{ij}$. The last equation becomes:

$$U_i'(\sum_{j=1}^{\theta(i)} p_{ij})v_i - \nu_i v_i p_{ij} + Q_{ij} + L_j - L_0 = 0 \tag{52}$$

Without loss of generality, assume that the alternate paths are ordered such that $Q_{i,1} \geq Q_{i,2} \geq ... \geq Q_{i,\theta(i)}$. Then we can show the following:

Lemma 9 *For any* $1 \leq k \leq j \leq \theta(i)$, $p_{ik} \geq p_{ij}$.

Proof: The result trivially holds if $p_{ij} = 0$. If $p_{ij} > 0$, we have $L_j = 0$. Then for any $k < j$,

$$\begin{aligned} 0 &= U_i'(\sum_{j=1}^{\theta(i)} p_{ij})v_i - \nu_i v_i p_{ij} + Q_{ij} - L_0 \\ &= U_i'(\sum_{j=1}^{\theta(i)} p_{ij})v_i - \nu_i v_i p_{ik} + Q_{ik} + L_k - L_0 \end{aligned}$$

Hence,

$$\nu_i v_i (p_{ik} - p_{ij}) = (Q_{ik} - Q_{ij}) + L_k \geq 0$$

i.e.,

$$p_{ik} \geq p_{ij}$$

Q.E.D.

By the above Lemma, there must exists a number J such that

$$p_{ij} > 0 \text{ for any } j \leq J \text{ and } p_{ij} = 0 \text{ for any } j > J$$

Note that p_{ij} can be easily found *if J is known.* To see this, let $L_j = 0$ and sum (52) for all $j \leq J$. We have,

$$JU_i'(\sum_{j=1}^{\theta(i)} p_{ij})v_i - \nu_i v_i \sum_{j=1}^{\theta(i)} p_{ij} + \sum_{j=1}^{J} Q_{ij} - JL_0 = 0$$

$$\sum_{j=1}^{\theta(i)} p_{ij} = 1 \text{ or } L_0 = 0$$

Let $f(x) = JU_i'(x)v_i - \nu_i v_i x$. It is equivalent to find

$$f(x) + \sum_{j=1}^{J} Q_{ij} - JL_0 = 0$$
$$x = 1 \text{ or } L_0 = 0$$

Note that $f(x)$ is decreasing in x due the concavity of U_i. If

$$f(1) + \sum_{j=1}^{J} Q_{ij} < 0$$

Then the solution of the above set of equantion should be some $x < 1$ and $L_0 = 0$, in which case x is the solution of

$$f(x) + \sum_{j=1}^{J} Q_{ij} = 0$$

Otherwise x should be equal to 1, and

$$L_0 = \frac{f(1) + \sum_{j=1}^{J} Q_{ij}}{J}$$

In both cases, each p_{ij} can be easily found via (52),

$$p_{ij} = \frac{U_i'(x)v_i + Q_{ij} - L_0}{\nu_i v_i}$$

for all $j \leq J$. In the last formula, p_{ij} is decreasing in j. Hence in order to check whether J is found correctly, we only need to check whether $p_{i,J}$ is no less than 0.

To find J, we start with $J = \theta(i)$ and reduce J if any p_{ij} solved above is less than 0. The algorithm proceeds as follows:

1) Let $J = \theta(i)$ and $Q = \sum_{j=1}^{\theta(i)} Q_{ij}$.

2) If $JU_i'(1)v_i - \nu_i v_i + Q < 0$, then solve

$$JU_i'(x)v_i - \nu_i v_i x + Q = 0$$

for x[‡] and let $L_0 = 0$. Otherwise, let $x = 1$ and

$$L_0 = \frac{JU_i'(1) - \nu_i v_i + Q}{J}$$

[‡]With the choice of the utility function in Section 5, the solution can be written explicitly.

3) Compute

$$p_{i,J} = \frac{U_i'(x)v_i + Q_{i,J} - L_0}{\nu_i v_i}$$

If $p_{i,J} \geq 0$,

4) then J is found correctly. Compute p_{ij} as

$$\begin{aligned} p_{ij} &= \frac{U_i'(x)v_i + Q_{ij} - L_0}{\nu_i v_i} \text{ for } j \leq J \\ p_{ij} &= 0 \text{ for } j > J \ . \end{aligned}$$

and the algorithm terminates. Otherwise,

5) let

$$\begin{aligned} J &= J - 1 \\ Q &= Q - Q_{i,J+1} \end{aligned}$$

If $J \geq 1$, go to step 2. Otherwise, set $p_{ij} = 0$ for all j and terminates.

Next we summarize the complexity of the algorithm. All steps except Step 4 are $O(1)$, and they may need to be executed $\theta(i)$ times in the worst case. Step 4 is $O(J)$ but it only needs to be executed once. The sorting of the $Q_{i,j}$s can be executed in $O(\theta(i) \log \theta(i))$ using an efficient sorting algorithm such as quicksort. Hence, the overall complexity is at most $O(\theta(i) \log \theta(i))$.

D Properties of the Stationary Point of Algorithm $\mathcal{A}$

From the definition of the stationary point of algorithm $\mathcal{A}$, one can establish the following properties that characterize any stationary point towards which the algorithm $\mathcal{A}$ converges:

Proposition 10 *Let $(\vec{p^*}, \vec{q^*})$ be a stationary point of algorithm $\mathcal{A}$. Define the cost of path j of class i to be the sum of the implicit costs over all links along the path, i.e., $q_{ij}^* = \sum_{l=1}^{L} H_{ij}^l q^{l,*}$. Let $q_{i,0} = \min_{j=1,\ldots,\theta(i)} q_{ij}^*$ denote the minimum cost among all alternate paths of class i. Then:*

1) $p^*_{ij} > 0 \Rightarrow q^*_{ij} = q_{i,0}$ *for all* i, j

2) Further, if the functions U_i *are strictly concave, then for any two stationary points* $\vec{p}^{*,1}, \vec{q}^{*,1}$ *and* $\vec{p}^{*,2}, \vec{q}^{*,2}$*, we have*

$$\begin{aligned} \sum_{j=1}^{\theta(i)} p^{*,1}_{ij} &= \sum_{j=1}^{\theta(i)} p^{*,2}_{ij} \text{ for all } i, \text{ and} \\ q^{*,1}_{i,0} &= q^{*,2}_{i,0} \text{ for all } i \text{ such that } 0 < \sum_{j=1}^{\theta(i)} p^{*,1}_{ij} < 1 \end{aligned}$$

Proof: To prove Part 1, assume in the contrary that there exists a $p^*_{ih} > 0$ and there exists another path k of the same class i such that $q^*_{ih} > q^*_{ik}$. Consider a small perturbation $\vec{p}_i$ around $\vec{p}^*_i$ such that:

$$\begin{aligned} p_{ih} &= p^*_{ih} - \delta \\ p_{ik} &= p^*_{ik} + \delta \\ p_{ij} &= p^*_{ij} \text{ if } j \neq h \text{ and } j \neq k \end{aligned}$$

where $0 < \delta < p^*_{ih}$. Then $\sum_{j=1}^{\theta(i)} p_{ij} = \sum_{j=1}^{\theta(i)} p^*_{ij}$, $\vec{p}_i \in \Omega_i$ and

$$\begin{aligned} & \left\{ U_i(\sum_{j=1}^{\theta(i)} p_{ij}) v_i - r_i \sum_{j=1}^{\theta(i)} p_{ij} q^*_{ij} - \sum_{j=1}^{\theta(i)} \frac{\nu_i}{2} (p_{ij} - p^*_{ij})^2 v_i \right\} \\ & - \left\{ U_i(\sum_{j=1}^{\theta(i)} p^*_{ij}) v_i - r_i \sum_{j=1}^{\theta(i)} p^*_{ij} q^*_{ij} - \sum_{j=1}^{\theta(i)} \frac{\nu_i}{2} (p^*_{ij} - p^*_{ij})^2 v_i \right\} \\ = \; & r_i \delta q^*_{ih} - r_i \delta q^*_{ik} - \nu_i \delta^2 v_i \end{aligned}$$

which is positive for small enough δ. This contradicts with the definition that $\vec{p}^*_i$ should maximize

$$U_i(\sum_{j=1}^{\theta(i)} p_{ij}) v_i - r_i \sum_{j=1}^{\theta(i)} p_{ij} q^*_{ij} - \sum_{j=1}^{\theta(i)} \frac{\nu_i}{2} (p_{ij} - p^*_{ij})^2 v_i$$

over all $\vec{p}_i \in \Omega_i$. This proves Part 1.

To prove Part 2, let $0 < \delta < 1$ and let

$$\vec{p} = \delta \vec{p}^{*,1} + (1 - \delta) \vec{p}^{*,2}$$

Then $\vec{p}$ also satisfies the constraints of (3) and

$$\sum_{j=1}^{\theta(i)} p_{ij} = \delta \sum_{j=1}^{\theta(i)} p_{ij}^{*,1} + (1-\delta) \sum_{j=1}^{\theta(i)} p_{ij}^{*,2}$$

Let u_0 be the optimal value of (3) achieved at both $\vec{p}^{*,1}$ and $\vec{p}^{*,2}$. Due to the concavity of U_i, we have

$$\sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} U_i(\sum_{j=1}^{\theta(i)} p_{ij}) v_i \geq u_0$$

However, since u_0 is the optimal value, only equality can hold. Hence

$$\sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} U_i(\sum_{j=1}^{\theta(i)} p_{ij}) v_i = u_0 = \delta \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} U_i(\sum_{j=1}^{\theta(i)} p_{ij}^{*,1}) v_i + (1-\delta) \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} U_i(\sum_{j=1}^{\theta(i)} p_{ij}^{*,2}) v_i$$

Since U_i is strictly concave, the above equality is possible only if

$$\sum_{j=1}^{\theta(i)} p_{ij}^{*,1} = \sum_{j=1}^{\theta(i)} p_{ij}^{*,2} \text{ for all } i$$

Finally, according to (8), $\vec{p}^{*,1}$ optimizes

$$U_i(\sum_{j=1}^{\theta(i)} p_{ij}) v_i - r_i \sum_{j=1}^{\theta(i)} p_{ij} q_{ij}^{*,1} - \sum_{j=1}^{\theta(i)} \frac{\nu_i}{2} (p_{ij} - p_{ij}^{*,1})^2 v_i$$

for all $\vec{p} \in \Omega$. If $0 < \sum_{j=1}^{\theta(i)} p_{ij}^{*,1} < 1$, then there exists some k such that $p_{ik}^{*,1} > 0$. Take derivative of the above function with respect to p_{ik} at $\vec{p}^{*,1}$, we have

$$U_i'(\sum_{j=1}^{\theta(i)} p_{ij}^{*,1}) v_i - r_i q_{ik}^{*,1} = 0$$

Note that no Lagrangian multipliers are needed because $p_{ik}^{*,1} > 0$ and $\sum_{j=1}^{\theta(i)} p_{ij}^{*,1} < 1$. Hence,

$$q_{ik}^{*,1} = U_i'(\sum_{j=1}^{\theta(i)} p_{ij}^{*,1}) \frac{v_i}{r_i}$$

As shown earlier, $q_{i,0}^{*,1} = q_{ik}^{*,1}$ since $p_{ik}^{*,1} > 0$. Further, $\sum_{j=1}^{\theta(i)} p_{ij}^{*,1}$ can be uniquely determined. Therefore $q_{i,0}^{*,1}$ can also be uniquely determined, i.e., $q_{i,0}^{*,1} = q_{i,0}^{*,2}$. *Q.E.D.*

References

[1] 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), pp. 17–28, September 1998.

[2] K. Arrow, L. Hurwicz, and H. Uzawa, *Studies in Linear and Nonlinear Programming*. Stanford, CA: Stanford University Press, 1958.

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

[4] S. Chen and K. Nahrstedt, "An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions," *IEEE Network, Special Issue on Transmission and Distribution of Digital Video*, vol. 12, pp. 64–79, November/December 1998.

[5] 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.

[6] F. Kelly, "Routing in Circuit Switched Networks: Optimization, Shadow Prices and Decentralization," *Advances in Applied Probability*, vol. 20, pp. 112–144, 1988.

[7] F. Kelly, "Routing and capacity Allocation in Networks with Trunk Reservation," *Mathematics of Operations Research*, vol. 15, no. 4, pp. 771–793, 1990.

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

[9] H. Kushner and J. Yang, "Analysis of Adaptive Step-Size SA Algorithms for Parameter Tracking," *IEEE Transactions on Automatic Control*, vol. 40, pp. 1403–1410, August 1995.

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

[11] X. Lin and N. B. Shroff, "Simplification of Network Dynamics in Large Systems," *Technical Report, Purdue University, http://min.ecn.purdue.edu/∼linx/papers.html*, 2001.

[12] X. Lin and N. B. Shroff, "Simplification of Network Dynamics in Large Systems," in *Tenth International Workshop on Quality of Service(IWQoS 2002)*, (Miami Beach, FL), May 2002.

[13] 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, 1999.

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

[15] P. Marbach, O. Mihatsch, and J. Tsitsiklis, "Call Admission Control and Routing in Integrated Service Networks Using Neuro-Dynamic Programming," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 197–208, February 2000.

[16] R. J. McEliece and K. N. Sivarajan, "Maximizing Marginal Revenue In Generalized Blocking Service Networks," in *Proc. 30th Annual Allerton Conference on Communication, Control, and Computing*, pp. 455–464, 1992.

[17] D. Mitra, J. Morrison, and K. Ramakrishnan, "ATM Network Design and Optimization: a Multirate Loss Network Framework," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 531–543, August 1996.

[18] S. Nelakuditi, Z.-L. Zhang, R. Tsang, and D. Du, "Adaptive Proportional Routing: a Localized QoS Routing Approach," *IEEE/ACM Transactions on Networking*, vol. 10, pp. 790–804, December 2002.

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

[20] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," *RFC 3031*, January 2001.

[21] A. Shaikh, J. Rexford, and K. Shin, "Efficient Precomputation of Quality-of-Service Routes," in *Proceedings of Workshop on Network and Operating Systems Support for Digital Audio and Video*, (Cambridge, United Kingdom), July 1998.

[22] 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, pp. 162–176, April 2001.

[23] W. Wang, M. Palaniswami, and S. H. Low, "Optimal Flow Control and Routing in Multi-Path Networks," *Performance Evaluation*, vol. 52, pp. 119–132, April 2003.