Lecture-24: Primal algorithm

1 Distributed algorithms: primal solution

Definition 1.1. Consider the set of sources $\mathcal{S}$, the set of links $\mathcal{L}$, fixed routing matrix $R \in \{0,1\}^{\mathcal{L}\times\mathcal{S}}$, and link capacity vector $c \in \mathbb{R}_+^{\mathcal{L}}$. The set of source rate vectors $x \in \mathbb{R}_+^{\mathcal{S}}$ that satisfy the the link capacity constraint, is written as

$$\mathcal{D} \triangleq \left\{ x \in \mathbb{R}_+^{\mathcal{S}} : y \triangleq Rx \leqslant c \right\}.$$

Assuming that each source $r \in \mathcal{S}$ has a concave increasing and continuously differentiable utility function $U_r : \mathbb{R}_+ \to \mathbb{R}_+$, the sum network utility maximizing resource allocation problem is formulated as the following convex optimization problem

$$x^* \triangleq \arg\max \left\{ \sum_{r\in\mathcal{S}} U_r(x_r) : x \in \mathcal{D} \right\}.$$

Remark 1. The techniques used to solve the optimization problem assume that we have complete knowledge of the topology and routes. Clearly this is infeasible in a giant network such as the Internet. We will study distributed algorithms which only require limited information exchange among the sources and the network for implementation. We will first study the primal solution, where instead of imposing a strict capacity constraint on each link, we append a cost to the sum network utility.

Definition 1.2 (Cost function). For each link $\ell \in \mathcal{L}$, a map $B_\ell : \mathbb{R}_+ \to \mathbb{R}_+$ is called a *cost function* if it is continuously differentiable and convex with $B_\ell(0) = 0$.

Definition 1.3 (Congestion price function). The continuous derivative of cost function B_ℓ is denoted by $f_\ell : \mathbb{R}_+ \to \mathbb{R}_+$, defined as $f_\ell(y) \triangleq B'_\ell(y)$ for all $y \in \mathbb{R}_+$, and called *congestion price function* or simply the price function at link ℓ, since it associates a price $p_\ell \triangleq f_\ell(y_\ell)$ with the level of congestion y_ℓ on link ℓ.

Lemma 1.4. *A map $B_\ell : \mathbb{R}_+ \to \mathbb{R}_+$ is a cost function iff the associate price function $f_\ell : \mathbb{R}_+ \to \mathbb{R}_+$ is increasing and*

$$B_\ell(y_\ell) = \int_0^{y_\ell} f_\ell(u)du. \tag{1}$$

Proof. Let B_ℓ be a continuously differentiable convex function. Then we denote its continuous derivative as price function $f_\ell \triangleq B'_\ell$ to write (1). Conversely, if f_ℓ is continuous then it follows from (1) that $B_\ell(0) = 0$ and is continuously differentiable.

Next, we let $\lambda \in (0,1]$ and $x < y \in \mathbb{R}_+$. Using (1) and change of variables, we can write

$$B_\ell(\bar{\lambda}x + \lambda y) - B_\ell(x) = \lambda \int_0^{(y-x)} f_\ell(x + \lambda u)du, \qquad \lambda \int_0^{y-x} f_\ell(x+u)du = \lambda(B_\ell(y) - B_\ell(x)).$$

We observe that $B_\ell(\bar{\lambda}x + \lambda y) - B_\ell(x) \leqslant \lambda(B_\ell(y) - B_\ell(x))$ if and only if $f_\ell(x + \lambda u) \leqslant f_\ell(x+u)$ for all $u \in [0, y-x]$. Since the choice of x, y, λ was arbitrary, it follows that B_ℓ is convex iff f_ℓ is increasing. □

Definition 1.5 (Modified network utility). Let $x \in \mathbb{R}_+^{\mathcal{S}}$ be rate vector of all sources, $R \in \{0,1\}^{\mathcal{L}\times\mathcal{S}}$ be the routing matrix for sources across links, and load vector $y \triangleq Rx \in \mathbb{R}_+^{\mathcal{L}}$, and $B_\ell : \mathbb{R}_+ \to \mathbb{R}_+$ is the map that determines the cost or price of sending data on link $\ell \in \mathcal{L}$. We can write the *modified network utility function* $W : \mathbb{R}_+^{\mathcal{S}} \to \mathbb{R}$ defined for any allocation $x \in \mathbb{R}_+^{\mathcal{S}}$ as

$$W(x) \triangleq \sum_{r\in\mathcal{S}} U_r(x_r) - \sum_{\ell\in\mathcal{L}} B_\ell(y_\ell). \tag{2}$$

Remark 2. Modified network utility $W(x)$ represents a tradeoff. Increasing the data rates x results in increased utility, but there is a price to be paid for the increased data rates at the links.

Corollary 1.6. *The modified network utility function W in* (2) *is concave in allocation* $x \in \mathbb{R}_+^{\mathcal{S}}$.

Proof. From the convexity of B_ℓ and the fact that sum of convex functions is convex, we get the result. □

Remark 3. If B_ℓ is interpreted as a *barrier* function associated with link ℓ, it should be chosen so that $\lim_{y \to c_\ell} B_\ell(y) = \infty$. Thus, for any $x \in \mathbb{R}_+^{\mathcal{S}}$ that leads to finite $W(x)$ will have $Rx \leqslant c$.

Remark 4. If B_ℓ is interpreted as a *penalty* function which penalizes the arrival rate for exceeding the link capacity, rates slightly larger than the link capacity may be allowable, but this will result in packet losses over the link. One can also interpret c_ℓ as a virtual capacity of the link which is smaller than the real capacity, in which case, even if the arrival rate exceeds c_ℓ, one may still be operating within the link capacity.

Remark 5. While it is not apparent in the deterministic formulation here, later in the book we will see that, even when the arrival rate on a link is less than its capacity, due to randomness in the arrival process, packets in the network will experience delay or packet loss. The function B_ℓ may thus be used to represent average delay, packet loss rate, etc.

1.1 Primal algorithm

Clearly, it is not practically feasible to solve (3) offline and implement the resulting data rates in the network since, as mentioned earlier, the topology of the network is unknown. Therefore, we will develop a decentralized algorithm under which each user can collect limited information from the network and solve for its own optimal data rate.

Remark 6. A natural candidate for such an algorithm is the so-called gradient ascent algorithm from optimization theory. The basic idea behind the gradient ascent algorithm is intuitive, especially if the concave function is a function of one variable: since a concave function has a derivative which is a decreasing function and the optimal solution is obtained at the point where the derivative is zero, it makes sense to seek a solution by moving in the direction of the derivative. More generally, for a function of many variables, the gradient ascent algorithm suggests moving in the direction of the gradient.

Definition 1.7. We will assume that utility functions $(U_r : r \in \mathcal{S})$ and price functions $(f_\ell : \ell \in \mathcal{L})$ are such that the maximization of (2) results in a solution with allocation $x_r > 0$ for all sources $r \in \mathcal{S}$. Then, the first-order condition for optimality states that the maximizer of (2) must satisfy for all $r \in \mathcal{S}$,

$$\frac{\partial}{\partial x_r} W(x) = U_r'(x_r) - \sum_{l \in \mathcal{L}} \Big(\frac{\partial}{\partial x_r} y_\ell \Big) B_\ell'(y_\ell) = U_r'(x_r) - \sum_{l \in \mathcal{L}} R_{\ell,r} f_\ell(y_\ell) = 0. \tag{3}$$

Definition 1.8. We will consider an algorithm where the source rate vector and link price vector evolve over time, and at time t are denoted by $x(t) \in \mathbb{R}_+^{\mathcal{S}}$ and $p(t) \in \mathbb{R}_+^{\mathcal{L}}$ respectively. Consequently, the load at each link $\ell \in \mathcal{L}$ and route price aggregated over all links traversed by a route also evolve over time, and at time t are denoted by $y(t) \in \mathbb{R}_+^{\mathcal{L}}$ and $q(t) \in \mathbb{R}_+^{\mathcal{S}}$ respectively. Recall that

$$y(t) = Rx(t), \qquad p_\ell(t) \triangleq f_\ell(y_\ell(t)), \quad \ell \in \mathcal{L}, \qquad q(t) = R^\top p(t).$$

Definition 1.9 (Primal algorithm). Consider the *primal algorithm* where the evolution of source rate allocation $x : \mathbb{R}_+ \to \mathbb{R}_+^{\mathcal{S}}$ is an autonomous dynamic system governed by $\dot{x} = g(x)$ for a map $g : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$ such that for each $r \in \mathcal{S}$

$$g_r(x) \triangleq k_r(x_r) \frac{\partial}{\partial x_r} W(x) = k_r(x_r) \Big(U_r'(x_r) - \sum_{l \in \mathcal{L}} R_{\ell,r} f_\ell(y_\ell) \Big) = k_r(x_r)(U_r'(x_r) - q_r). \tag{4}$$

The right-hand side of the above differential equation is simply the derivative of (2) with respect to x_r, while $k_r(x_r)$ is simply a step-size parameter which determines how far one moves in the direction of the gradient.

Remark 7. The scaling function $k_r : \mathbb{R}_+ \to \mathbb{R}_+$ must be chosen such that the equilibrium of the differential equation is the same as the optimal solution to the resource allocation problem. For example, if $k_r(x_r) > 0$, setting $\dot{x}_r = 0$ for all $r \in \mathcal{S}$ yields the same set of equations as (3).

Remark 8. Algorithm (4) is called a primal algorithm since it arises from the primal formulation of the utility maximization problem. Note that the primal algorithm is a congestion control algorithm for the following reasons. When the route price $q_r = \sum_{\ell \in \mathcal{L}} R_{\ell,r} f_\ell(y_\ell)$ is large, the congestion controller decreases its transmission rate. Further, if rate x_r is large then $U'(x_r)$ is small, since $U_r(x_r)$ is concave. Thus the rate of increase is small, and the network can be viewed as a control system with the network providing the feedback to allow the sources to adjust their rates.

Definition 1.10. For the strictly concave modified network utility function W, we denote its unique maximizer by $\hat{x}$. Further, we define a non-negative function $V : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}_+$ as $V(x) \triangleq W(\hat{x}) - W(x)$ for any $x \in \mathbb{R}_+^{\mathcal{S}}$.

Lemma 1.11. *Function V defined in Definition 1.10 is non-negative with unique zero at $x = \hat{x}$.*

Proof. Since W is strictly concave, it has a unique maximizer denoted by $\hat{x}$ and $W(\hat{x}) \geqslant W(x)$ for all $x \in \mathbb{R}_+^{\mathcal{S}}$ with equality iff $x = \hat{x}$. □

Theorem 1.12 (Stability of primal algorithm). *Consider a network in which all sources adjust their data rates according to the primal control algorithm* (4)*, and consider the function $V : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}_+$ defined in Definition 1.10. Assume that the functions U_r, k_r and f_ℓ are such that*
(a) utility function $U_r : \mathbb{R}_+ \to \mathbb{R}_+$ is concave increasing and continuously differentiable for each source $r \in \mathcal{S}$,
(b) scaling function $k_r : \mathbb{R}_+ \to \mathbb{R}_+$ is continuous for each source $r \in \mathcal{S}$,
(c) function $V : \mathbb{R}_+^{\mathcal{S}} \to \mathbb{R}$ is radially unbounded,
(d) optimal allocation $\hat{x}_r > 0$ for each source $r \in \mathcal{S}$, and
(e) the equilibrium point of (4) *is the maximizer of* (2)*.*
Then, the controller in (4) *is globally asymptotically stable.*

Proof. Since U_r' exists and is continuous, f_ℓ is continuous and V is radially unbounded, we observe that V is a potential function. Differentiating V with respect to time t, we get

$$\dot{V}(x(t)) = \langle \nabla V(x), \dot{x} \rangle = -\sum_{r \in \mathcal{S}} \frac{\partial}{\partial x_r} W(x_r) g_r(x) = -\sum_{r \in \mathcal{S}} k_r(x_r)(U_r'(x_r) - q_r)^2.$$

It follows that $\dot{V}(x(t)) \leqslant 0$ for all $x \in \mathbb{R}^{\mathcal{S}}$ with equality iff $x = \hat{x}$. Further, $V(x) \geqslant 0$ for all $x \in \mathbb{R}^{\mathcal{S}}$ with equality iff $x = \hat{x}$ from Lemma 1.11. Thus, all the conditions of the Lyapunov theorem are satisfied, and so the system state $x(t)$ will converge to $\hat{x}$, starting from any initial condition. □

Example 1.13. In the proof of Theorem 1.12, we have assumed that the utility, price, and scaling functions are such that modified network utility W satisfies the conditions required to apply the Lyapunov stability theorem. It is easy to find functions that satisfy these properties. For example, if $U_r(x_r) = w_r \ln x_r$ and $k_r(x_r) = x_r$, the primal congestion control algorithm for source r becomes

$$\dot{x}_r = k_r(x_r) \frac{\partial}{\partial x_r} W(x) = x_r \Big(\frac{w_r}{x_r} - \sum_{\ell \in \mathcal{L}} R_{\ell,r} f_\ell(y_\ell) \Big) = w_r - x_r \sum_{\ell \in \mathcal{L}} R_{\ell,r} f_\ell(y_\ell),$$

and thus the unique equilibrium point can be obtained by solving $\frac{w_r}{x_r} = \sum_{\ell \in \mathcal{L}} R_{\ell,r} f_\ell(y_\ell)$. Further, if price function f_ℓ is such that B_ℓ is a polynomial function, then $\lim_{\|x\| \to \infty} W(x) = -\infty$, and thus $\lim_{\|x\| \to \infty} V(x) = \infty$.

1.2 Congestion feedback and distributed implementation

For a congestion control algorithm to be useful in practice, it should be amenable to decentralized implementation. We now present one possible manner in which the primal algorithm could be implemented in a distributed fashion.

1.2.1 Setting price field

We first note that each source $r \in \mathcal{S}$ simply needs to know $q_r \triangleq \sum_{\ell \in \mathcal{L}} R_{\ell,r} p_\ell$ the sum of the link prices on its route to adjust its data rate $x_r = (U_r')^{-1}(q_r)$ as suggested by the algorithm. Suppose that every packet has a field (a certain number of bits) set aside in its header to collect the price of its route. When the source releases a packet into the network, the price field can be set to zero. Then, each link on the route can add its price to the price field so that, by the time the packet reaches its destination, the price field will contain the route price. This information can then be fed back to the source to implement the congestion control algorithm. A noteworthy feature of the congestion control algorithm is that the link prices p depend only on the total arrival rate $y_\ell = \sum_{r \in \mathcal{S}} R_{\ell,r} x_r$ to the link ℓ, and not on the individual arrival rates $(x_r : r \in \mathcal{S}_\ell)$ of each source $\mathcal{S}_\ell \triangleq \left\{ r \in \mathcal{S} : R_{\ell,r} = 1 \right\}$ using the link. Thus, each link ℓ has only to keep track of the total arrival rate y_ℓ to compute the link price p_ℓ. If the algorithm required each link to keep track of individual source arrival rates, it would be infeasible to implement since the number of sources using high-capacity links could be prohibitively large. Thus, the primal algorithm is both amenable to a distributed implementation and has low overhead requirements.

1.2.2 Setting single bit congestion field

Packet headers in the Internet are already crowded with a lot of other information, such as source and destination addresses to facilitate routing, so Internet practitioners do not like to add another field in the packet header to collect congestion information. In view of this, the overhead required to collect congestion information can be further reduced to accommodate practical realities. Consider the extreme case where there is only one bit available in the packet header to collect congestion information. Suppose that each packet is marked independently with probability $1 - e^{-p_\ell}$ when the packet passes through link ℓ. We denote this congestion indicator by $\xi_{n,\ell}$ for packet n for link ℓ. Marking simply means that a bit in the packet header is flipped from a 0 to a 1 to indicate congestion. Then, a packet n is marked along a route r indicated by

$$\xi_{n,r} \triangleq 1 - \prod_{\ell \in \mathcal{L}_r} (1 - \xi_{n,\ell}), \text{ where } \mathcal{L}_r \triangleq \left\{ \ell \in \mathcal{L} : R_{\ell,r} = 1 \right\}.$$

The marking probability of a packet n along route r is given by

$$\mathbb{E}\xi_{n,r} = 1 - \prod_{\ell \in \mathcal{L}_r} e^{-p_\ell} = 1 - e^{-\sum_{\ell \in \mathcal{L}_r} p_\ell} = 1 - e^{-\sum_{\ell \in \mathcal{L}} R_{\ell,r} p_\ell} = 1 - e^{-q_r}.$$

If the acknowledgement for each packet contains one bit of information to indicate whether a packet is marked or not, then, by computing the fraction of marked packets, the source can estimate the route price $q_r = \sum_{\ell \in \mathcal{L}} R_{\ell,r} p_\ell$. The assumption here is that source rates x change slowly so that each link price p_ℓ remains roughly constant over many packets. Thus, one can estimate p_ℓ approximately.

1.2.3 Estimating congestion via packet drops

While marking, as mentioned above, has been widely studied in the literature, the pre-dominant mechanism used for congestion feedback in the Internet today is through packet drops. Buffers used to store packets at a link have finite capacity, and therefore a packet that arrives at a link when its buffer is full is dropped immediately. If a packet is dropped, the destination of the packet will not receive it. So, if the destination then provides feedback to the source that a packet was not received, this provides an indication of congestion. Clearly, such a scheme does not require even a single bit in the packet header to collect congestion information. However, strictly speaking, this type of congestion feedback cannot be modeled using our framework since we assume that a source's data rate is seen by all links on the route, whereas, if packet dropping is allowed, some packets will not reach all links on a source's route.

However, if we assume that the packet loss rate at each link is small, we can approximate the rate at which a link receives a source's packet by the rate at which the source is transmitting packets. Further, the end-to-end drop probability on a route can be approximated by the sum of the drop probabilities on the links along the route if the drop probability at each link is small. That is, let p_ℓ be the drop probability of packets at link ℓ, then the drop probability along the route r is

$$q_r = 1 - \prod_{\ell \in \mathcal{L}_r} (1 - p_\ell) \approx \sum_{\ell \in \mathcal{L}_r} p_\ell = \sum_{\ell \in \mathcal{L}} R_{\ell,r} p_\ell.$$

Thus, the optimization formulation approximates reality under these assumptions. To complete the connection to the optimization framework, we have to specify the price function at each link. A crude approximation to the drop probability also known as packet loss rate at link $\ell \in \mathcal{L}$ is

$$p_\ell \triangleq \left(\frac{(y_\ell - c_\ell)}{y_\ell}\right)_+,$$

which is non-zero only if $y_\ell = \sum_{r \in \mathcal{S}} R_{\ell,r} x_r$ is larger than c_ℓ. This approximate formula for the packet loss rate can serve as the price function for each link.