From previous lecture, we need to find the flows

Find $\underline{x}$ that maximizes

$$L(\underline{x}, \underline{\lambda}) = f(\underline{x}) - \sum \lambda_j \theta_j$$

$$= u(x_1) + u(x_2) + u(x_3) - \lambda_1 x_1 - \lambda_1 x_2 + \lambda_1 - \lambda_2 x_1 - \lambda_2 x_3 + \lambda_2$$

find flows x_1, x_2, x_3 that maximize

$$\begin{aligned} u(x_1) &- (\lambda_1 + \lambda_2) x_1 \\ u(x_2) &- \lambda_1 x_2 \\ u(x_3) &- \lambda_2 x_3 \end{aligned}$$

$\Rightarrow$ x_1 adjusts flow independently of x_2 and x_3

Do this for a given value of $\underline{\lambda}$ obtaining

$\underline{x}(\underline{\lambda})$

Then minimize with respect to λ i.e.

find shadow prices for links 1, 2

$$\underline{\lambda}^* = \underset{\underline{\lambda}}{\operatorname{argmin}} \; D(\underline{\lambda}) = \underset{\underline{\lambda}}{\operatorname{argmin}} \max_{\underline{x}} L(\underline{x}, \underline{\lambda})$$

- This corresponds to finding the saddle point where the Lagrangian is minimum with respect to price and max with respect to flow.

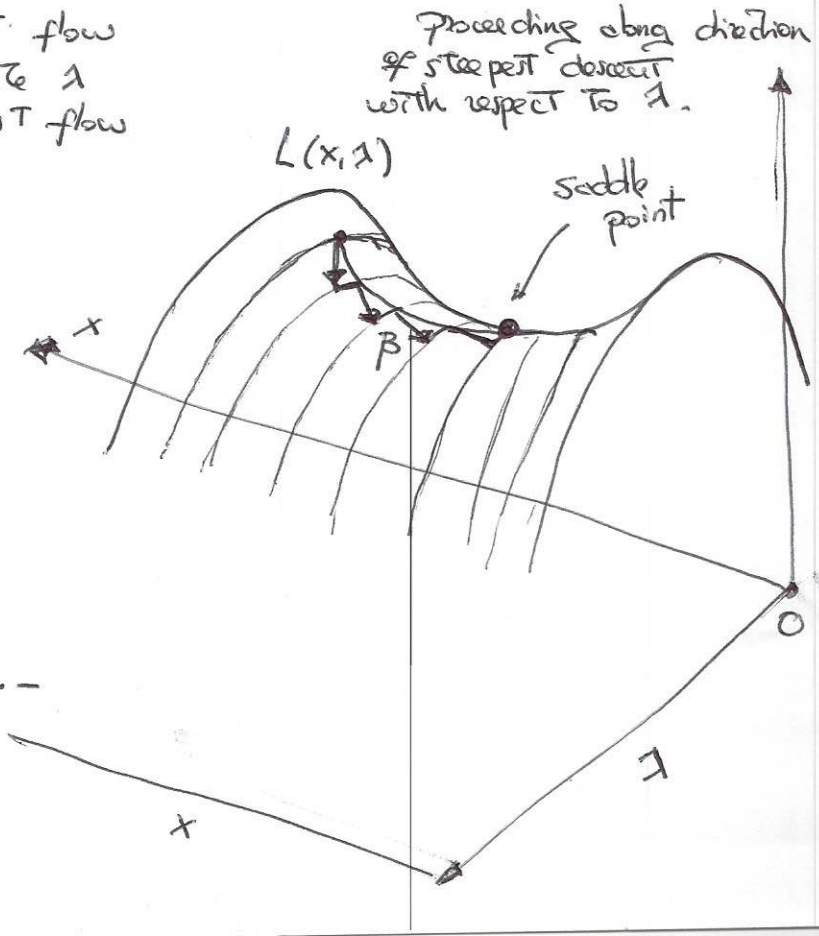
How can we do this distributely.

ITERATIVE ALGORITHM

- adjust flow
- update λ
- adjust flow

$$\lambda_j(n+1) = \lambda_j(n) - \beta \frac{\partial}{\partial \lambda_j} L(x, \lambda)$$

This corresponds to taking a step of length β along the direction of steepest descent with respect to λ. The direction is tangential to the curve of steepest descent passing through te points where x is maximum.



At each step adjust the flows to be on the max point for the given $\underline{\lambda}(n)$

$$\begin{cases} x_1(n) = \underset{x_1}{\arg\max}\ u(x_1) - [\lambda_1(n) - \lambda_2(n)]x_1 \\ x_2(n) = \underset{x_2}{\arg\max}\ u(x_2) - \lambda_2(n) x_2 \\ x_3(n) = \underset{x_3}{\arg\max}\ u(x_3) - \lambda_2(n) x_3 \end{cases}$$

Then proceed along the line of steepest descent

$$\begin{cases} \lambda_1(n+1) = \lambda_1(n) - \beta \dfrac{\partial}{\partial \lambda_1} L(\underline{x}, \lambda_1) \\ \lambda_2(n+1) = \lambda_2(n) - \beta \dfrac{\partial}{\partial \lambda_2} L(\underline{x}, \lambda_2) \end{cases}$$

Repeat the steps

How are Step 1 & Step 2 achieved ?

Look evolution of queue length for flow j over an interval of time τ

$$\boxed{q_j[(n+1)\tau] = q_j[n\tau] + \tau g_j}$$

where for example $g_1 = x_1 + x_2 - 1$

= arrival rate − service rate.

= excess rate = increment in queue length

This is remarkably similar to eq. for $\lambda(n)$ that we need.

So we set

$$\lambda_j = \frac{\beta}{\tau} q_j$$

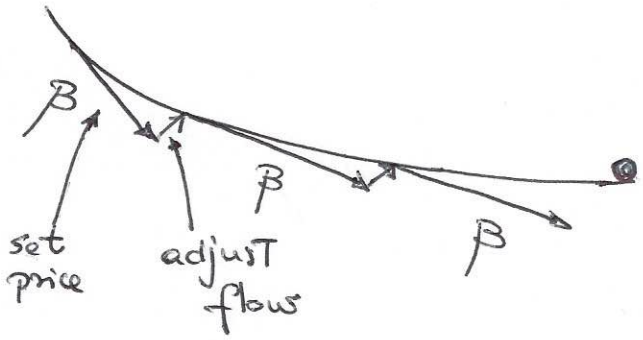
To have 1 step of gradient descent algorithm be executed every τ time units of queue evolution.

This either increases the price if $g_j > 0$ which forces the flow to decrease (via AIMD)

or decreases the price if $g_j < 0$ forcing the flow to increase.

We only need knowledge of queue length.
With it we can <u>set prices</u> and then let flow be readjusted via AIMD -

Setting prices corresponds to taking a step β towards saddle point.
Adjusting flow corresponds to climbing up to maximum with respect to x for the given value of λ



Queue length is estimated via RED
measuring packet losses at host that are proportional to queue length decaye router drops packets with probability proportional to queue length.