Communications Network Design

lecture 08

Matthew Roughan
<matthew.roughan@adelaide.edu.au>

Discipline of Applied Mathematics
School of Mathematical Sciences
University of Adelaide

April 1, 2009

Routing (continued)

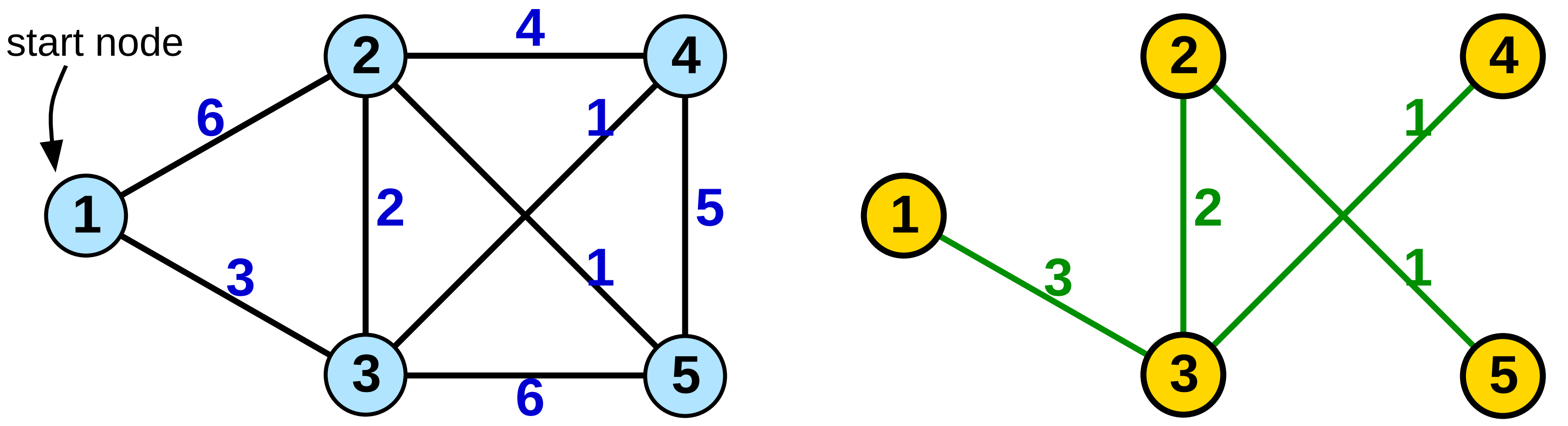
The simple routing considered so far has fixed distances, but if we consider a more queueing view of networks, then packets are delayed when a link is heavily loaded, and so this increases delays. Minimum delay routing forms a non-linear, **convex** optimization problem with **separable** costs. We present two simple gradient descent methods for solution of such problems including the Frank Wolfe method.

Recap link-state routing

- topology is flooded
  - including the link weights α
- calculate shortest paths
  - assumption of linear costs, based on weights
  - not automatically based on congestion
    - capacity constraints are ignored in the optimization
  - so too much traffic can be routed along any one route
- note that the link weights are arbitrary
  - how can we use this to avoid congestion?
- recap notation in lecture 6

Link loads

Once we know shortest paths, we can compute link loads



Costs are linear in the costs/distances, and loads

$$C(\mathbf{f}) = \sum_{e \in E} \alpha_e f_e = \sum_{(p,q) \in K} \hat{l}_{pq} t_{pq}$$

either link or path costs and loads can be used.

Example cost calculation

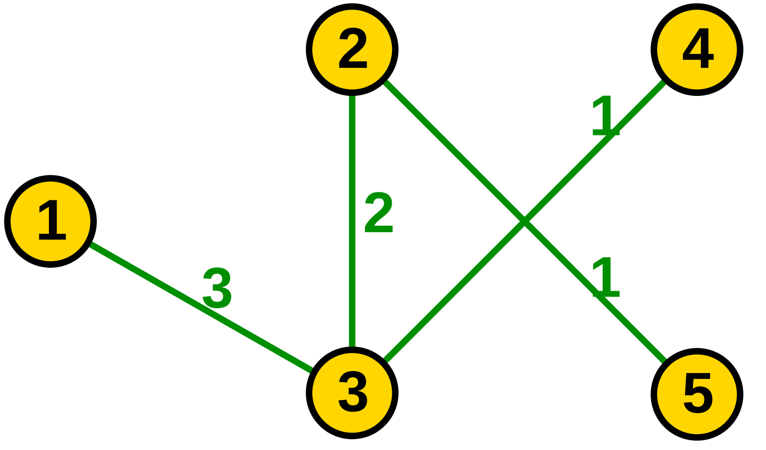
OD pair	load t_{pq}	path	path length	$\hat{l}_{pq} t_{pq}$
$(1,2)$	$t_{12} = 1$	$1-3-2$	$\hat{l}_{12} = 5$	5
$(1,3)$	$t_{13} = 2$	$1-3$	$\hat{l}_{13} = 3$	6
$(1,4)$	$t_{14} = 3$	$1-3-4$	$\hat{l}_{14} = 4$	12
$(1,5)$	$t_{15} = 4$	$1-3-2-5$	$\hat{l}_{15} = 6$	24
$(2,3)$	$t_{23} = 2$	$3-2$	$\hat{l}_{23} = 2$	4
$(2,4)$	$t_{24} = 3$	$2-3-4$	$\hat{l}_{24} = 3$	9
$(2,5)$	$t_{25} = 3$	$2-5$	$\hat{l}_{25} = 1$	3
$(3,4)$	$t_{34} = 2$	$3-4$	$\hat{l}_{34} = 1$	2
$(3,5)$	$t_{35} = 1$	$3-2-5$	$\hat{l}_{35} = 3$	3
$(4,5)$	$t_{45} = 2$	$4-3-2-5$	$\hat{l}_{45} = 4$	8
			total cost	76

Example loads on links

OD pair	t_{pq}	links			
		$(1,3)$	$(2,3)$	$(2,4)$	$(3,5)$
$(1,2)$	$t_{12} = 1$	1	1		
$(1,3)$	$t_{13} = 2$	2			
$(1,4)$	$t_{14} = 3$	3			3
$(1,5)$	$t_{15} = 4$	4	4	4	
$(2,3)$	$t_{23} = 2$		2		
$(2,4)$	$t_{24} = 3$		3		
$(2,5)$	$t_{25} = 3$			3	
$(3,4)$	$t_{34} = 2$				2
$(3,5)$	$t_{35} = 1$		1	1	
$(4,5)$	$t_{45} = 2$		2	2	2
	total load	10	13	10	10

Alternative cost calculation

link	α_e	f_e	cost $\alpha_e \times f_e$
$(1,3)$	3	10	30
$(2,3)$	2	13	26
$(2,4)$	1	10	10
$(3,5)$	1	10	10
total			76



This also tells us the link loads, from which we could estimate congestion.

Link loads

Why should this result in low cost network?

- link weights relate to link cost
- higher weight results in less traffic
- hence less cost
- relationship between link loads and shortest paths
  - shorter paths result in fewer hops
  - so less resources used
  - less cost

But is a linear model the right approach?

Non-linear cost functions

Non-linear functions could be anything: we will restrict ourselves to

- continuous functions
  - no breaks in the function
- differentiable
  - no corners or edges in the function
  - assume its differentiable enough
  - can define gradient and Hessian
- convex functions
  - chords lie above the function

Differentiable functions

The **gradient** $\nabla C(\mathbf{f}) = \left(\frac{\partial C(\mathbf{f})}{\partial f_e} : e \in E \right)$ is the vector of first partial derivatives of C.

For example

$$C(\mathbf{f}) = \sum_{e \in E} \frac{f_e}{r_e - f_e} = \sum_{e \in E} \left[\frac{r_e}{r_e - f_e} - 1 \right]$$

has gradient

$$\frac{\partial C(\mathbf{f})}{\partial f_e} = \frac{r_e}{(r_e - f_e)^2}$$ and $$\nabla C(\mathbf{f}) = \begin{bmatrix} \frac{r_{e_1}}{(r_{e_1} - f_{e_1})^2} \\ \frac{r_{e_2}}{(r_{e_2} - f_{e_2})^2} \\ \vdots \\ \frac{r_{e_m}}{(r_{e_m} - f_{e_m})^2} \end{bmatrix}$$

Differentiable functions

The **Hessian** $\nabla^2 C(\mathbf{f}) = \left(\frac{\partial^2 C(\mathbf{f})}{\partial f_e \partial f_g} : e, g \in E \right)$ is the square matrix of all second partial derivatives of C.

Example above has

$$\nabla^2 C(\mathbf{f}) = \begin{bmatrix} \frac{2 r_{e_1}}{(r_{e_1} - f_{e_1})^3} & 0 & \cdots & 0 \\ 0 & \frac{2 r_{e_2}}{(r_{e_2} - f_{e_2})^3} & \cdots & 0 \\ \vdots & & & \\ 0 & 0 & \cdots & \frac{2 r_{e_m}}{(r_{e_m} - f_{e_m})^3} \end{bmatrix}$$

Note that in this example, the Hessian is a diagonal matrix. This will always be the case when C is separable in f_e. i.e. $C(\mathbf{f}) = \sum_{e \in E} c_e(f_e)$.

Linear cost example

$$C(\mathbf{f}) = \sum_{e \in E} \alpha_e f_e$$

$$\nabla C(\mathbf{f}) = (\alpha_1, \alpha_2, \ldots \alpha_m)^T$$

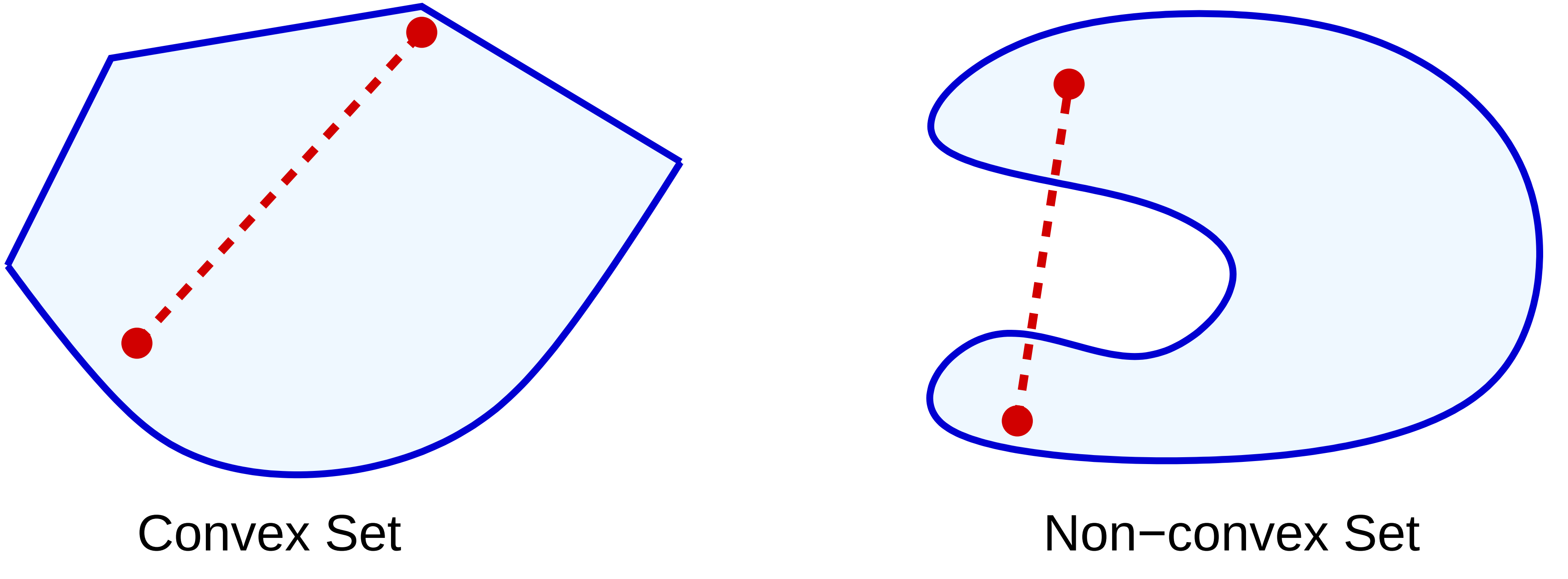
$$\nabla^2 C(\mathbf{f}) = [0]$$

a matrix of 0's, since $C(\mathbf{f})$ is linear

Convex sets

Definition: *A* set Ω is a convex set in R^m if for all $\mathbf{x}, \mathbf{y} \in \Omega$, $t\mathbf{x} + (1-t)\mathbf{y} \in \Omega$ for all $t \in [0,1]$.

i.e. chords between points in the set lie inside the set.

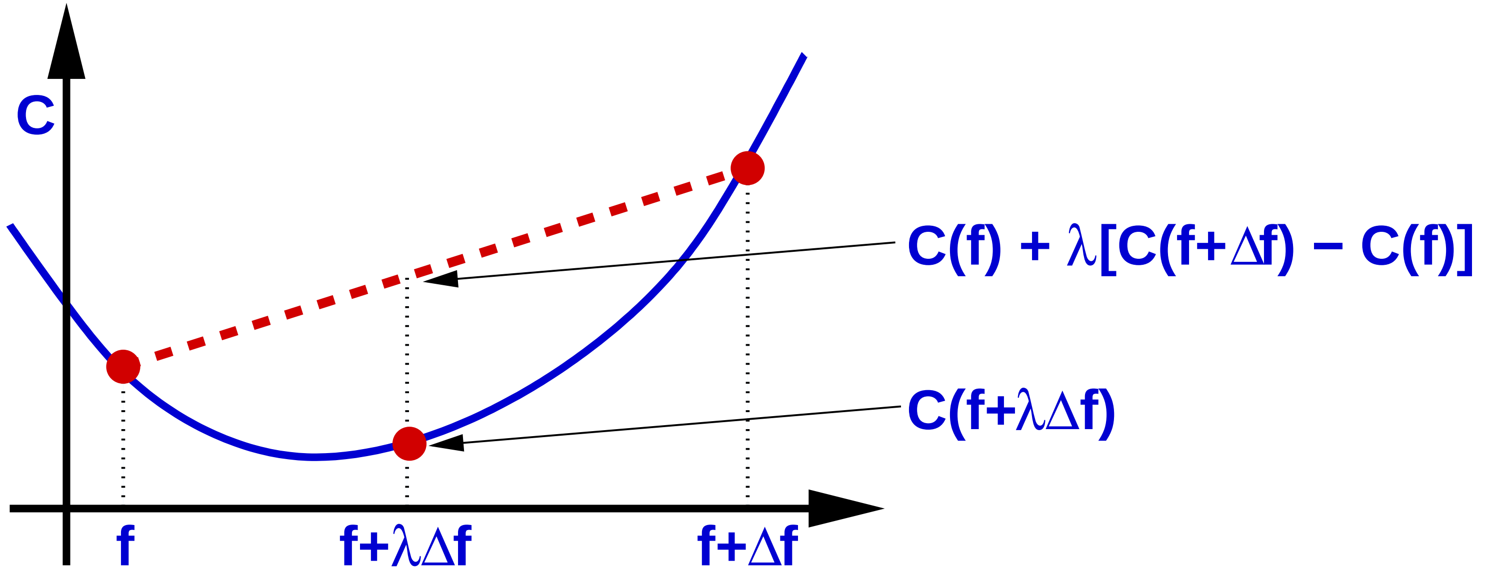


Convex functions

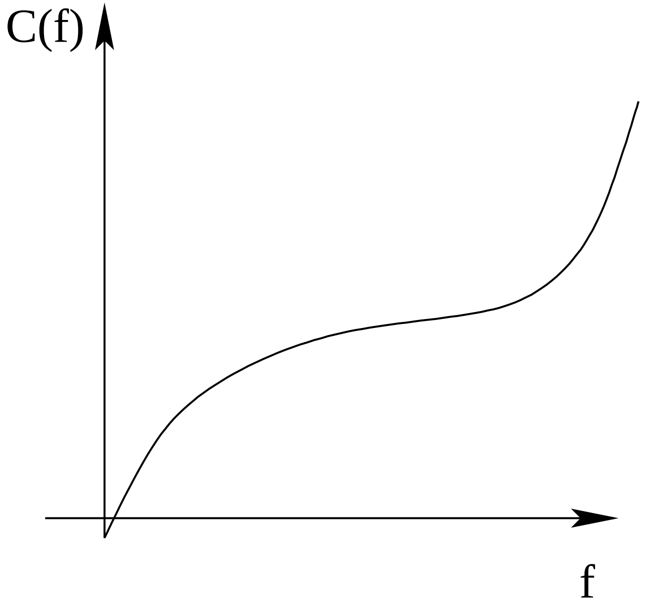
Definition: Let Ω be a convex set in R^m. A function $f:\Omega\to R$ is a convex function if for all $\lambda\in(0,1)$,

$$C(\mathbf{f}+\lambda\Delta\mathbf{f})\leq C(\mathbf{f})+\lambda(C(\mathbf{f}+\Delta\mathbf{f})-C(\mathbf{f})),$$

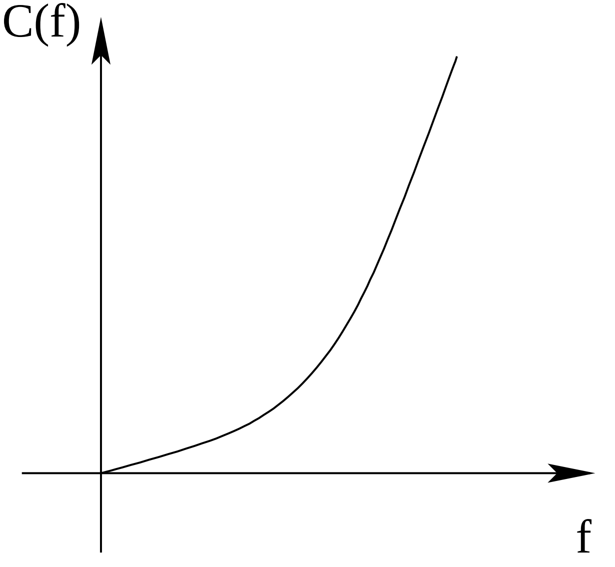
for all $\mathbf{f},\mathbf{f}+\Delta\mathbf{f}\in\Omega$. In 2-D, one can picture this as the chord joining $(f,C(f))$ and $(f+\Delta f,C(f+\Delta f))$ sitting above the curve $y=C(f)$.



Convex functions



not convex



convex

Convex differentiable functions

Theorem: Let Ω be a convex set in R^m. A differentiable function $C : \Omega \to R$ is convex iff

$$C(\mathbf{f} + \Delta\mathbf{f}) \geq C(\mathbf{f}) + \nabla C(\mathbf{f})^T \Delta\mathbf{f}.$$

Proof: Omitted. Proof uses a Taylor Series approach.

Thus a differentiable function is convex iff

$$C(\mathbf{f} + \Delta\mathbf{f}) - C(\mathbf{f}) \geq \nabla C(\mathbf{f})^T \Delta\mathbf{f}.$$

Says that tangents will lie below the convex function.

Convex differentiable functions

Theorem: *A* differentiable function C is convex on the convex set Ω iff the Hessian $\nabla^2 C(\mathbf{f})$ is positive semidefinite on Ω i.e. C is convex iff $\mathbf{z}^T \nabla^2 C(\mathbf{f})\mathbf{z} \geq 0$ for all vectors $\mathbf{z} \in \Omega$

i.e. C is convex iff $\Delta\mathbf{f}^T \nabla^2 C(\mathbf{f})\Delta\mathbf{f} \geq 0$ for all $\Delta\mathbf{f} \in \Omega$.

Example

A separable, differentiable function $C(\mathbf{f}) = \sum_e c_e(f_e)$ is convex iff $c''_e(f_e) = \frac{\partial^2 c_e(f_e)}{\partial f_e^2} \geq 0$ for all $e \in E$.

Explanation:

To be positive semi-definite we must have $\mathbf{z}^T \nabla^2 C(\mathbf{f})\mathbf{z} = \sum_e \frac{\partial^2 c_e(f_e)}{\partial f_e^2} z_e^2 \geq 0$ for all $\mathbf{z}$.

$(\Rightarrow)$ clearly if $c''_e(f_e) \geq 0$ then the sum above is ≥ 0

$(\Leftarrow)$ Also, recall that in this example,

$$\nabla^2 C(\mathbf{f}) \;\; = \left[\text{diag}\{c''_{e_1}(f_{e_1}), \ldots, c''_{e_m}(f_{e_m})\}\right]$$

If $\mathbf{z} = (0.\ldots.0, 1, 0, \ldots 0)^T$ with the '1' in the i-th spot, then $\mathbf{z}^T \nabla^2 C(\mathbf{f})\mathbf{z} = c''_{e_i}(f_{e_i})$ and hence we must have c_{e_i} convex for all i

Simple queueing model

Imagine we wish to minimize delays caused by queueing

- simple queueing model M/M/1 queue
- average queueing delay on a link is given by

$$c(f_e; r_e) = \frac{f_e}{r_e - f_e}$$

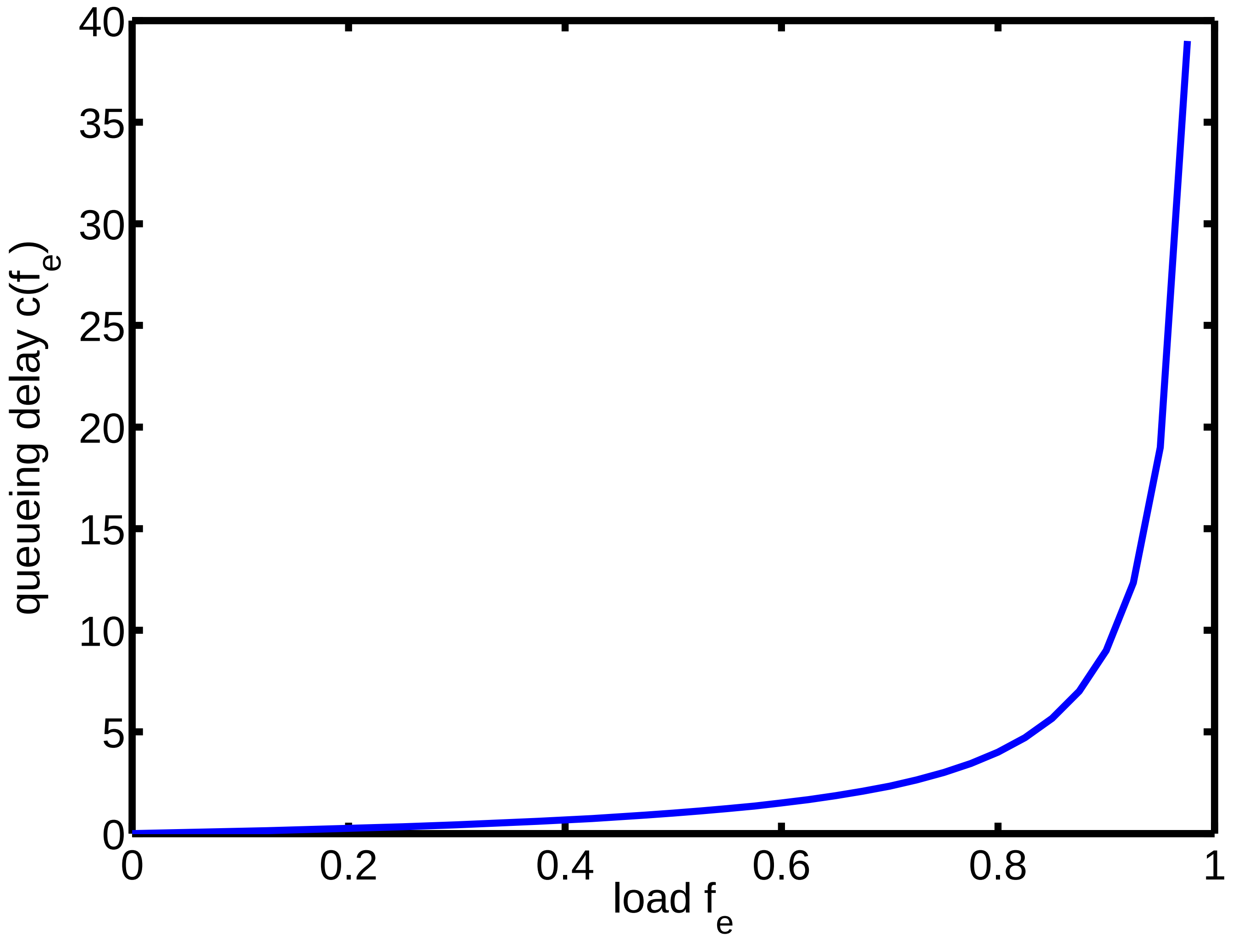
where f_e is the link load, and r_e is the capacity

Assume that the interactions between queues are weak

- **Kleinrock's Independence Approximation**

$$C(\mathbf{f}; \mathbf{r}) = \sum_{e \in E} c(f_e; r_e) = \sum_{e \in E} \frac{f_e}{r_e - f_e}$$

Simple queueing model



The function is increasing, convex and differentiable (except at r_e), with an asymptote at r_e

Minima

- convex functions have a unique minimum
- non-convex functions can have non-unique minima, and local minima
- by definition, at the minima $\hat{\mathbf{f}}$ we get

$$C(\hat{\mathbf{f}}) \leq C(\hat{\mathbf{f}} + \Delta\mathbf{f})$$

- if differentiable, for all **feasible** routing changes

$$\nabla C(\hat{\mathbf{f}})^T \Delta\mathbf{f} \geq 0$$

reason lies in Taylor's theorem

$$C(\mathbf{f} + \lambda\Delta\mathbf{f}) = C(\mathbf{f}) + \lambda\nabla C(\mathbf{f})^T \Delta\mathbf{f} + O(\lambda^2)$$

If $\nabla C(\hat{\mathbf{f}})^T \Delta\mathbf{f} < 0$, for small $\lambda > 0$ then $C(\hat{\mathbf{f}}) > C(\hat{\mathbf{f}} + \lambda\Delta\mathbf{f})$

Feasible routing changes

Feasible change in routing $\Delta\mathbf{x}$

- no path traffic can go negative

$$x_\mu + \Delta x_\mu \geq 0, \ \forall \mu \in P_{pq}$$

- traffic must be conserved

$$\sum_{\mu \in P_{pq}} \Delta x_\mu = 0, \ \forall [p,q] \in K,$$

- note that the change in link loads will be

$$\Delta f_e = \sum_{\mu \in P : e \in \mu} \Delta x_\mu \quad \forall e \in E$$

Separable cost functions

- if we have cost function $C(\mathbf{f})$

$$
\begin{aligned}
\nabla C(\mathbf{f})^T \Delta \mathbf{f} &= \sum_{e \in E} \frac{\partial C(\mathbf{f})}{\partial f_e} . \Delta f_e \\
&= \sum_{e \in E} \frac{\partial C(\mathbf{f})}{\partial f_e} . \left(\sum_{\mu \in P : e \in \mu} \Delta x_\mu \right) \\
&= \sum_{\mu \in P} \left(\sum_{e \in \mu} \frac{\partial C(\mathbf{f})}{\partial f_e} \right) . \Delta x_\mu \\
&= \sum_{\mu \in P} l_\mu(\mathbf{f}) \Delta x_\mu
\end{aligned}
$$

- $\sum_{e \in \mu} \frac{\partial C(\mathbf{f})}{\partial f_e} = l_\mu(\mathbf{f})$ is called path length (again)
- note that path length now depends on the loads $\mathbf{f}$

Shortest path with non-linear costs

$l_\mu(\mathbf{f})$ is called the length of path μ, and

$$\nabla C(\mathbf{f})^T \Delta \mathbf{f} = \sum_{\mu \in P} l_\mu(\mathbf{f}) \Delta x_\mu.$$

For a load $\mathbf{f}$ and any O-D pair $[p,q] \in K$, let

$$\hat{l}_{pq}(\mathbf{f}) = \text{min}\{l_\mu(\mathbf{f}) : \mu \in P_{pq}\}$$

As before, we call a path $\mu = \hat{\mu} \in P_{pq}$ for which $l_{\hat{\mu}}(\mathbf{f}) = \hat{l}_{pq}(\mathbf{f})$ a **shortest path** for $[p,q]$.

Note that this is consistent with the previous example where $\frac{\partial C}{\partial f_e} = \alpha_e$.

Shortest path with non-linear costs

Theorem: A minimum cost routing implies a shortest path routing (though the reverse is not necessarily true).

Proof: Suppose the routing is NOT a shortest path routing. In particular, assume some traffic for the O-D pair $[p,q] \in K$ is assigned to a path $\mu' \in P_{pq}$ which is NOT of shortest length. That is,

$$l_{\mu'}(\mathbf{f}) > \hat{l}_{pq}(\mathbf{f}) \qquad \text{and} \qquad x_{\mu'} > 0.$$

Let $\hat{\mu} \in P_{pq}$ be a shortest path for $[p,q]$. So $l_{\hat{\mu}}(\mathbf{f}) = \hat{l}_{pq}(\mathbf{f})$.

Shortest path with non-linear costs

Proof (continued): Reroute as follows:

$$\begin{aligned} \Delta x_{\mu'} &= -\delta \\ \Delta x_{\hat{\mu}} &= \delta \\ \Delta x_{\mu} &= 0 \quad \forall \text{ other } \mu \in P, \end{aligned}$$

where $0 < \delta \leq x_{\mu'}$. Then note $l_{\mu'}(\mathbf{f}) > l_{\hat{\mu}}(\mathbf{f})$

$$\begin{aligned} \nabla C(\mathbf{f})^T \Delta \mathbf{f} &= \sum_{\mu \in P} l_{\mu}(\mathbf{f}) \Delta x_{\mu} \\ &= -l_{\mu'}(\mathbf{f})\delta + l_{\hat{\mu}}(\mathbf{f})\delta \\ &= (-l_{\mu'}(\mathbf{f}) + l_{\hat{\mu}}(\mathbf{f}))\delta \\ &\quad \text{(something -ve). (something +ve)} \\ &< 0. \end{aligned}$$

Thus if the routing is not a shortest path routing, $\nabla C(\mathbf{f})^T \Delta \mathbf{f} < 0$ which means it cannot be minimum cost.

Shortest path with convex costs

Theorem: If $C(\mathbf{f})$ is convex and differentiable, then $\mathbf{x}$ is a minimum cost routing **iff** $\mathbf{x}$ is a shortest path routing.

Proof: $\Rightarrow$ from previous theorem
$\Leftarrow$ from properties of convex functions:

- assume we have shortest path routing, e.g.

  $x_\mu = 0, \forall \mu \in P_{pq}$ **not** a shortest path

- for a routing change $\Delta\mathbf{x}$, then $\Delta x_\mu \geq 0, \forall \mu \in P_{pq}$ which are **not** shortest paths, i.e.

  $\Delta x_\mu \geq 0$ when $l_\mu(\mathbf{f}) > \hat{l}_{pq}(\mathbf{f})$

- Also, for all $\mu \in P_{pq}$ which are shortest paths,

  $\Delta x_\mu \geq -x_\mu$ when $l_\mu(\mathbf{f}) = \hat{l}_{pq}(\mathbf{f})$.

Shortest path with convex costs

Proof: (cont) $\Rightarrow (l_\mu(\mathbf{f}) - \hat{l}_{pq}(\mathbf{f}))\Delta x_\mu \geq 0, \; \forall [p,q], \mu \in P_{pq}$

- either first term > 0 and second ≥ 0
- or first term =0, so second term is irrelevant

So $l_\mu(\mathbf{f})\Delta x_\mu \geq \hat{l}_{pq}(\mathbf{f})\Delta x_\mu$. Therefore

$$\begin{aligned}
\nabla C(\mathbf{f})^T \Delta \mathbf{f} &= \sum_{\mu \in P} l_\mu(\mathbf{f})\Delta x_\mu \\
&= \sum_{[p,q] \in K} \sum_{\mu \in P_{pq}} l_\mu(\mathbf{f})\Delta x_\mu \\
&\geq \sum_{[p,q] \in K} \sum_{\mu \in P_{pq}} \hat{l}_{pq}(\mathbf{f})\Delta x_\mu \\
&= \sum_{[p,q] \in K} \hat{l}_{pq}(\mathbf{f}) \left(\sum_{\mu \in P_{pq}} \Delta x_\mu \right) = 0, \quad \text{since } \sum_{\mu \in P_{pq}} \Delta x_\mu = 0.
\end{aligned}$$

Shortest path with convex costs

Proof: (cont)
Thus $\nabla C(\mathbf{f})^T \Delta\mathbf{f} \geq 0$ for all feasible changes in load $\Delta\mathbf{f}$.

Now one of the properties of a convex differentiable function $C(\mathbf{f})$ is that

$$C(\mathbf{f} + \Delta\mathbf{f}) - C(\mathbf{f}) \geq \nabla C(\mathbf{f})^T \Delta\mathbf{f}.$$

If $C(\hat{\mathbf{f}})^T \Delta\mathbf{f} \geq 0$ then

$$C(\hat{\mathbf{f}} + \Delta\mathbf{f}) - C(\hat{\mathbf{f}}) \geq 0$$

or alternatively $C(\hat{\mathbf{f}} + \Delta\mathbf{f}) \geq C(\hat{\mathbf{f}})$, which means that $C(\hat{\mathbf{f}})$ takes its minimum value at $\hat{\mathbf{f}}$. $\square$

Descent Methods

Definition: *A* vector $\mathbf{u} \in R^{|P|}$ is said to be a **descent direction** for the routing $\mathbf{x}$, with induced load $\mathbf{f}$, if

(i) $u_\mu < 0 \Rightarrow x_\mu > 0.$

we can only subtract traffic from a path μ if there is some traffic on it in the first place!

(ii) $\sum_{\mu \in P_{pq}} u_\mu = 0 \qquad \forall$ O-D pairs $(p,q) \in K$

any traffic we take from one path μ must be added to the traffic on some other path(s)

(iii) $\sum_{\mu \in P} l_\mu(\mathbf{f}) u_\mu < 0$

it is a descent vector, i.e., the change in C by going a small distance in this direction is negative.

Descent Methods: notes

- The change in C for a small change $\lambda\mathbf{u}$ will be

$$C(\mathbf{f}+\lambda\Delta\mathbf{f})-C(\mathbf{f})=\lambda\sum_{\mu\in P} l_\mu(\mathbf{f})u_\mu+O(\lambda^2)$$

  and we require that $\sum_{\mu\in P} l_\mu(\mathbf{f})u_\mu<0$

- The change in routing will be $\Delta\mathbf{x}=\lambda\mathbf{u}$, for some small $\lambda>0$. λ must be chosen with two things in mind:
  (a) $\mathbf{x}+\Delta\mathbf{x}$, the new routing, must still be feasible.
  (b) we only go as far in the direction $\mathbf{u}$ as we need to, to get maximum decrease in $C(\mathbf{f})$, in that direction.

Descent Methods

Broadly, the method consists of the following steps:

1. Choose a feasible descent direction $\mathbf{u} \in R^{|P|}$.
2. Given that the new routing will be $\mathbf{x} + \lambda\mathbf{u}$, choose a step length $\lambda > 0$ so that
   (i) $\mathbf{x} + \lambda\mathbf{u}$ is feasible (i.e. $\geq \mathbf{0}$)
   (ii) $\mathbf{x} + \lambda\mathbf{u}$ minimises the cost of the induced load.
3. Change the routing and the induced load
4. Unless you have a minimum, goto step 1.
   (i) For convex costs, when we have a shortest path routing, we have reached the minima.

Calculating the new cost

Take the change in routing to be $\Delta\mathbf{x} = \lambda\mathbf{u}$

$$\begin{aligned}\Delta f_e &= \sum_{\mu:e\in\mu} \Delta x_\mu \\ &= \lambda \sum_{\mu:e\in\mu} u_\mu \\ &= \lambda v_e\end{aligned}$$

where we define $v_e = \sum_{\mu:e\in\mu} u_\mu$ and $\mathbf{v} = (v_e : e \in E) \in R^m$.

More succinctly $\Delta\mathbf{f} = \lambda\mathbf{v}$ and the new cost is $C(\mathbf{f} + \lambda\mathbf{v})$.

Descent Method 1

Simple exchange method:

- transfer some traffic from a longer path $\mu^* \in P_{pq}$ to a shortest path $\hat{\mu} \in P_{pq}$, i.e. $l_{\mu^*}(\mathbf{f}) > l_{\hat{\mu}}(\mathbf{f}) = l_{\hat{\mu}}(\mathbf{f})$
- descent direction $\mathbf{u}$ has components

$$\begin{aligned} u_{\mu^*} &= -1 && \text{transfer off } \mu^* \\ u_{\hat{\mu}} &= +1 && \text{transfer onto } \hat{\hat{\mu}}^* \\ u_{\mu} &= 0 && \forall \text{ other } \mu \in P \end{aligned}$$

Note that with $\mathbf{u}$ as above

$$\sum_{\mu} l_\mu u_\mu = +l_{\hat{\mu}}(\mathbf{f}) - l_{\mu^*}(\mathbf{f}) < 0$$

and therefore $\mathbf{u}$ is a descent direction.

Descent Method 1

Simple exchange method:

- to maintain feasibility we require

$$0 \leq \lambda \leq x_{\mu^*}$$

- the vector $\mathbf{v}$ has components

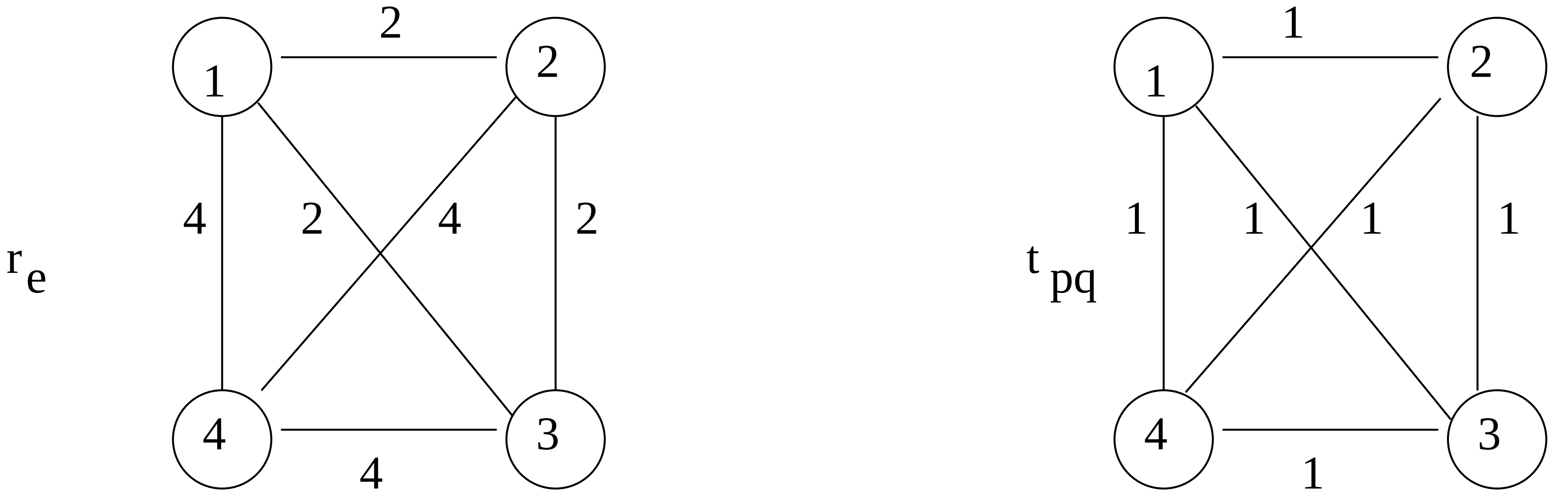
$$v_e = \begin{cases} 1 & \text{if } e \in \hat{\mu} \text{ and } e \notin \mu^* \\ -1 & \text{if } e \in \mu^* \text{ and } e \notin \hat{\mu} \\ 0 & \text{otherwise} \end{cases}$$

- We wish to determine $\lambda^* \in [0, x_{\mu^*}]$ which minimises $C(\mathbf{f} + \lambda\mathbf{v})$

Descent Method 1: example

An example network

Capacities r_e Traffic demands t_{pq}

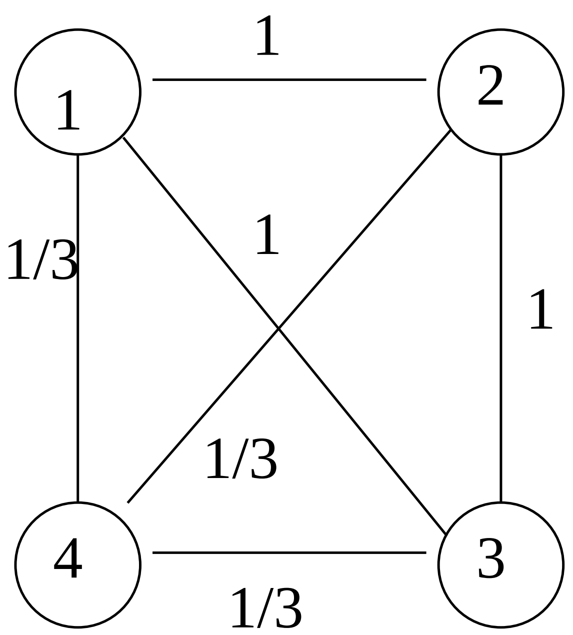


Descent Method 1: example

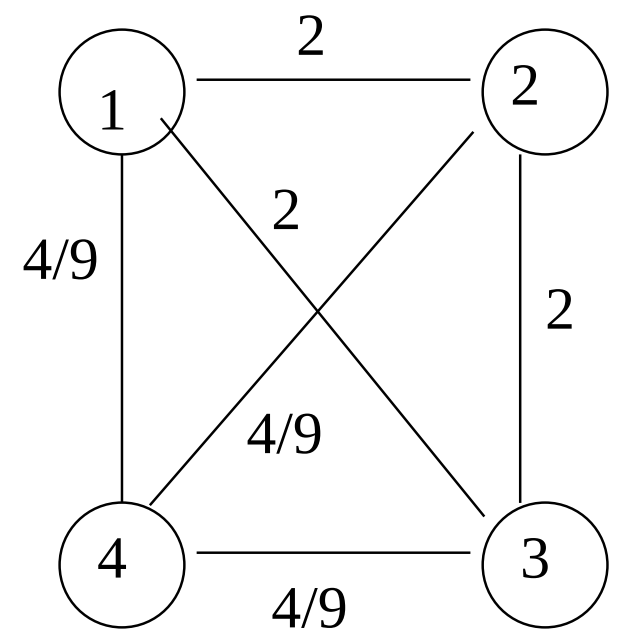
Assume direct routing of the traffic

Costs $c_e(f_e) = \dfrac{f_e}{r_e - f_e}$ $\qquad \dfrac{dc_e}{df_e} = \dfrac{r_e}{(r_e - f_e)^2}$

$c_e\ (f_e)$



$\dfrac{dc_e}{df_e}$



Total cost $C(\mathbf{f}) = \sum_e c_e(f_e) = 3.\frac{1}{2-1} + 3.\frac{1}{4-1} = 4$

Descent Method 1: example

shortest paths are as follow:

OD pair	direct path	shortest path
$1,2$	$1-2$	$1-4-2$
$1,3$	$1-3$	$1-4-3$
$1,4$	$1-4$	$1-4$
$2,3$	$2-3$	$2-4-3$
$2,4$	$2-4$	$2-4$
$3,4$	$3-4$	$3-4$

- not all traffic is routed on the shortest path!
- For example: O-D pair [1,3], the shortest route would be 1-4-3 (length of $\frac{8}{9}$), but at present the traffic is routed on 1-3 (length of 2)

Descent Method 1: example

We transfer some load from a direct path, to a shortest path e.g. transfer some flow from path $\mu = 1–2$ to $\mu = 1–4–2$.

In this problem, there are 30 paths in this network. So $\mathbf{x}$ and $\mathbf{u}$ have 30 entries. Listing all paths lexicographically, e.g. paths

$$\begin{gathered}1-2, 1-2-3, 1-2-4, 1-2-3-4, 1-2-4-3,\\ 1-3, 1-3-2, 1-3-4, 1-3-2-4, 1-3-4-2,\\ 1-4, 1-4-2, 1-4-3, 1-4-2-3, 1-4-3-2, \ldots\end{gathered}$$

$$\mathbf{x}^T = (1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0)$$

and

$$\mathbf{u}^T = (-1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$$

Descent Method 1: example

We need to calculate v_e given the above descent direction.

- $u_{1-2} = -1$ which says
  - we reduce the traffic on path $1-2$
  - and hence on link $1-2$
  - so this gives us $v_{1-2} = -1$
- $u_{1-4-2} = 1$ which says
  - we increase the traffic on path $1-4-2$
  - and hence on links $1-4$ and $4-2$
  - so this gives us $v_{1-4} = v_{4-2} = 1$

Net effect is

$$\mathbf{v} = (v_{1-2}, v_{1-3}, v_{1-4}, v_{2-3}, v_{2-4}, v_{3-4})^T = (-1, 0, 1, 0, 1, 0)^T$$

Descent Method 1: example

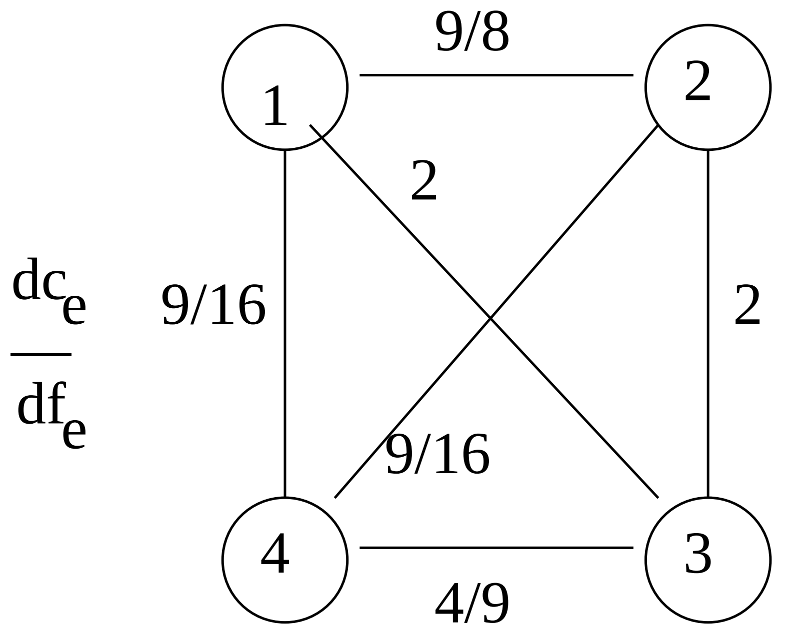
We move $\lambda \in [0,1]$ in the descent direction (above), so recalculating the costs we get

$$
\begin{aligned}
C(\mathbf{f}+\lambda\mathbf{v}) &= \sum_e c_e(f_e+\lambda v_e) \\
&= \sum_e \frac{f_e+\lambda v_e}{r_e-(f_e+\lambda v_e)} \\
&= c+\frac{f_{1\text{–}2}-\lambda}{r_{1\text{–}2}-(f_{1\text{–}2}-\lambda)}+\frac{f_{1\text{–}4}+\lambda}{r_{1\text{–}4}-(f_{1\text{–}4}+\lambda)}+\frac{f_{4\text{–}2}+\lambda}{r_{4\text{–}2}-(f_{4\text{–}2}+\lambda)} \\
&= c+\frac{1-\lambda}{2-1+\lambda}+2\frac{1+\lambda}{4-1-\lambda} \\
\frac{dC}{d\lambda} &= 2\left(\frac{-1}{(1+\lambda)^2}+\frac{4}{(3-\lambda)^2}\right)
\end{aligned}
$$

Descent Method 1: example

$$\frac{dC}{d\lambda} = 2\left(\frac{-1}{(1+\lambda)^2} + \frac{4}{(3-\lambda)^2}\right)$$

which is equal to zero for $\lambda = 1/3$, so this gives us out optimal step size λ. The new "distances" are shown below. Note it is still not a shortest path graph.



Descent Method 2

Frank-Wolfe method:

- we know we are aiming for a shortest path
- why not try to get there in one step
  - given a feasible routing $\mathbf{x}$, find shortest path routing $\mathbf{z}$
  - set $\mathbf{u} = \mathbf{u} - \mathbf{x}$, and $\lambda \in [0, 1]$
  - Find λ to minimize the new cost $C(\mathbf{f} + \lambda \mathbf{v})$
  - Continue
- don't really get there in one step, as shortest paths change when load changes
  - but iterations converge
  - proof on following slide

Descent Method 2

Lemma: If $\mathbf{z}$ is a shortest path routing *wrt* $l_\mu(\mathbf{f})$ (where $\mathbf{f}$ is the load induced by current routing $\mathbf{x}$) then $\mathbf{u} = \mathbf{z} - \mathbf{x}$ is a descent direction.

Proof of Lemma: (recall the definition)

1. if $x_\mu = 0$ then $u_\mu = z_\mu \geq 0$
2. $\sum_{\mu \in P_{pq}} u_\mu = \sum_{\mu \in P_{pq}} z_\mu - \sum_{\mu \in P_{pq}} x_\mu = t_{pq} - t_{pq} = 0$
3. $\sum_{\mu \in P} l_\mu(\mathbf{f}) u_\mu = \sum_{[[p,q] \in K} \sum_{\mu \in P_{pq}} (l_\mu(\mathbf{f}) z_\mu - l_\mu(\mathbf{f}) x_\mu) < 0$

   since $\mathbf{z}$ being shortest path routing implies second sum is larger than first sum.

Hence $\mathbf{z} - \mathbf{x}$ is a descent direction. $\square$

Methods: Dynamic feedback

ARPANET's earliest methods [1, 2].

- the M/M/1 model is not really a good model for the Internet
  - we don't a priori know the best model
- want a distributed algorithm
- what can we do?
- bright idea
  - measure delays (two different methods)
  - use these in a SPF routing
- problem: oscillation
  - the network and traffic are not static
  - doesn't take much to cause oscillation

Greedy vs Hill Climbing

- We have discussed hill-climbing today
  - actually we described descent methods, but hill-climbing is just the reverse
  - follow the path up (down) a hill (optimization function)
- Greedy algorithms are similar
  - choose the next best step at each point
  - like going up a hill, but
  - only a partial solution at each step until the end
  - Dijkstra is a good example of a greedy algorithm

Traffic Engineering

Modern IGP routing protocols are almost all based on simple SPF algorithms with linear costs, but real costs are non-linear. It works fine most of the time, but when congestion occurs, there is a problem. Traffic engineering is the process of rebalancing traffic loads on a network to avoid congestion.

Now a'days

Modern IGP routing protocols are almost all based on simple linear cost SPF algorithms!

- link costs are static: no dependence on congestion
- mainly used for rerouting in failures
- how can we optimize if the cost function is really non-linear
- optimization becomes choice of the best weights α_e
- NP-hard so need heuristics [3, 4, 5]

Planning horizons

More generally

- real way to optimize network is to change its design (which we consider next)
- planning horizon for network redesign is months
  - ordering and delivery of equipment
  - test and verification of equipment
  - waiting for planned maintenance windows
  - availability of technical staff
  - capital budgeting cycles.
- need a process to allow rebalancing of traffic on shorter time scale: traffic engineering

Traffic Engineering

- Traffic engineering fills the gap
- Planning horizon of hours/days: only need to change router configuration (the link weights)
- Two methods
  - link weight optimization (as above)
  - MPLS: full optimization of all routing using tunnels
- But a lot of traffic engineering is still done in a very ad hoc way.

References

[1] J. M.McQuillan, G. Falk, and I. Richer, "A review of the development and performance of the ARPANET routing algorithm," IEEE Transactions on Communication, vol. COM-26, pp. 60–74, December 1978.

[2] J. M.McQuillan, I. Richer, and E. C. Rosen, "A new routing algorithm for the ARPANET," IEEE Transactions on Communication, vol. COM-28, pp. 711–719, May 1980.

[3] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," IEEE Communications Magazine, vol. 40, no. 10, pp. 118–124, 2002.

[4] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," IEEE Journal on Selected Areas in Communications, vol. 20, no. 4, pp. 756–767, 2002.

[5] M. Roughan, M. Thorup, and Y. Zhang, "Performance of estimated traffic matrices in traffic engineering," in ACM SIGMETRICS, (San Diego, CA, USA), pp. 326–327, 2003.