Fair Flows and Robustness in Infrastructure Networks

Rui Carvalho, School Mathematical Sciences, QMUL

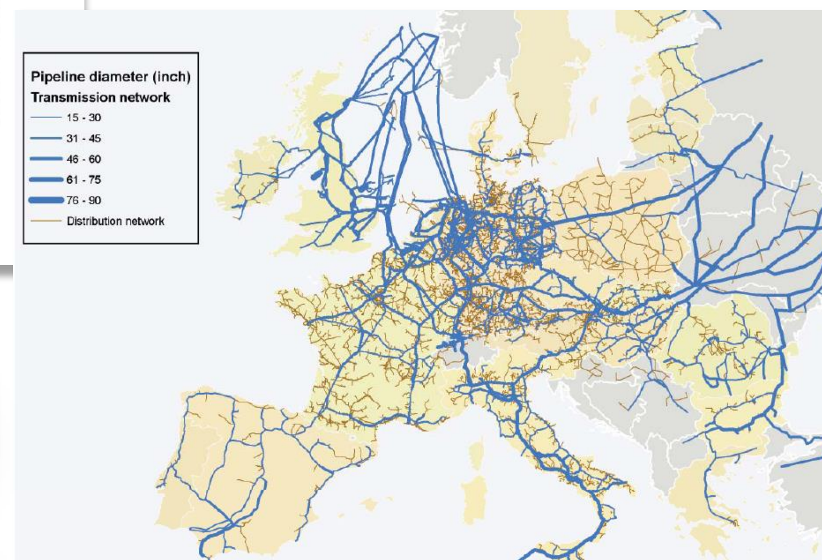
QMUL:
Wolfram Just
David Arrowsmith

University of Zilina:
Lubos Buzna

Joint Research Centre
Flavio Bono
Eugenio Gutierrez

ETHZ:
Dirk Helbing

raven
Resilience, Adaptability and Vulnerability of
Complex Energy Networks



R Carvalho, L Buzna, F Bono, E Gutiérrez, W Just, and D Arrowsmith,
Phys. Rev. E **80**, 016106 (2009)

R Carvalho, L Buzna, W Just, D Helbing, D Arrowsmith,
Phys. Rev. E **85**, 046101 (2012)

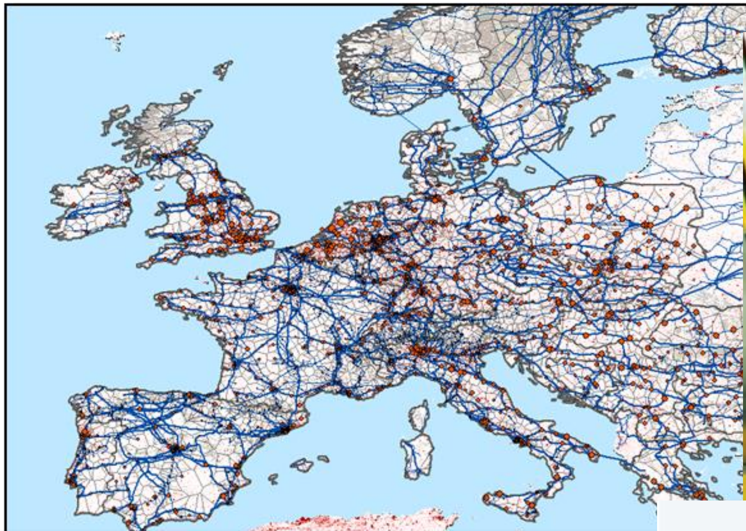
Queen Mary University of London

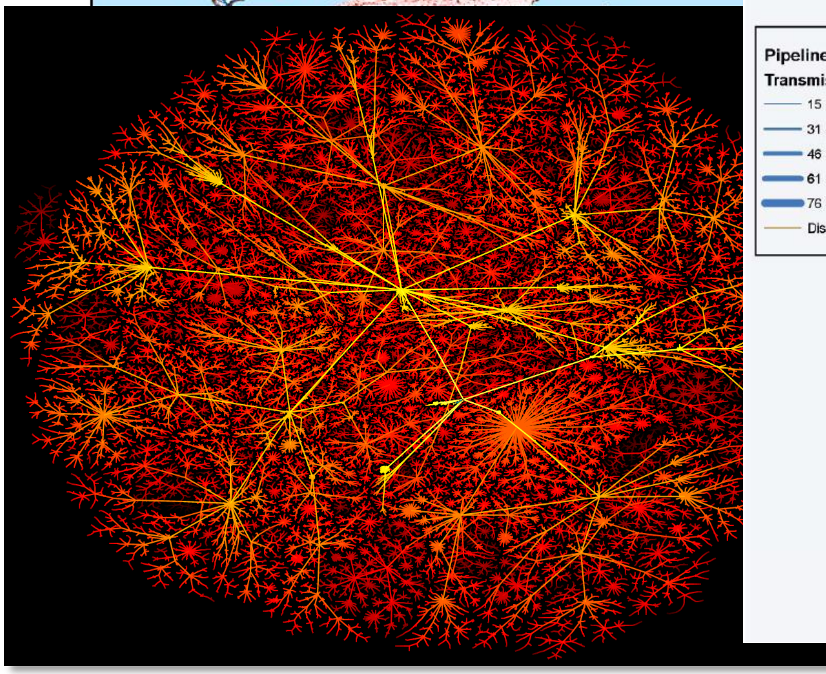
JRC EUROPEAN COMMISSION

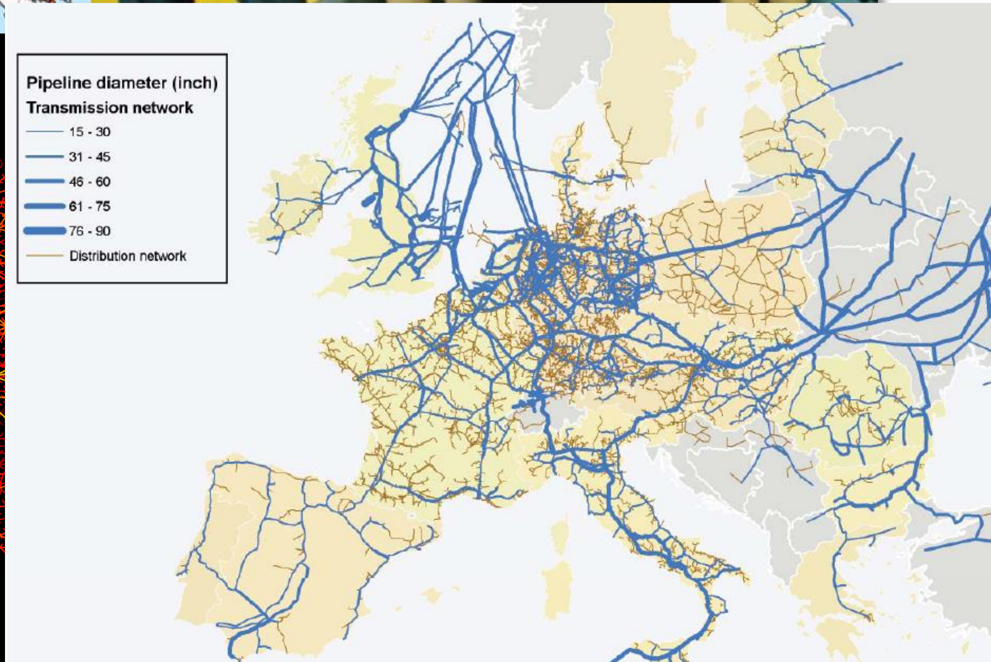
ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ŽILINSKÁ UNIVERZITA

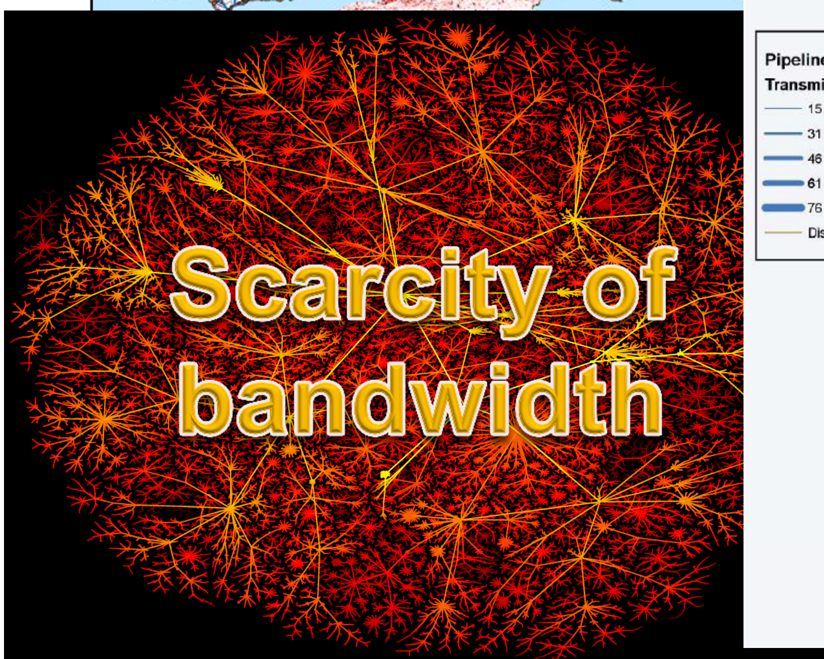
We need to find creative uses of available infrastructure networks





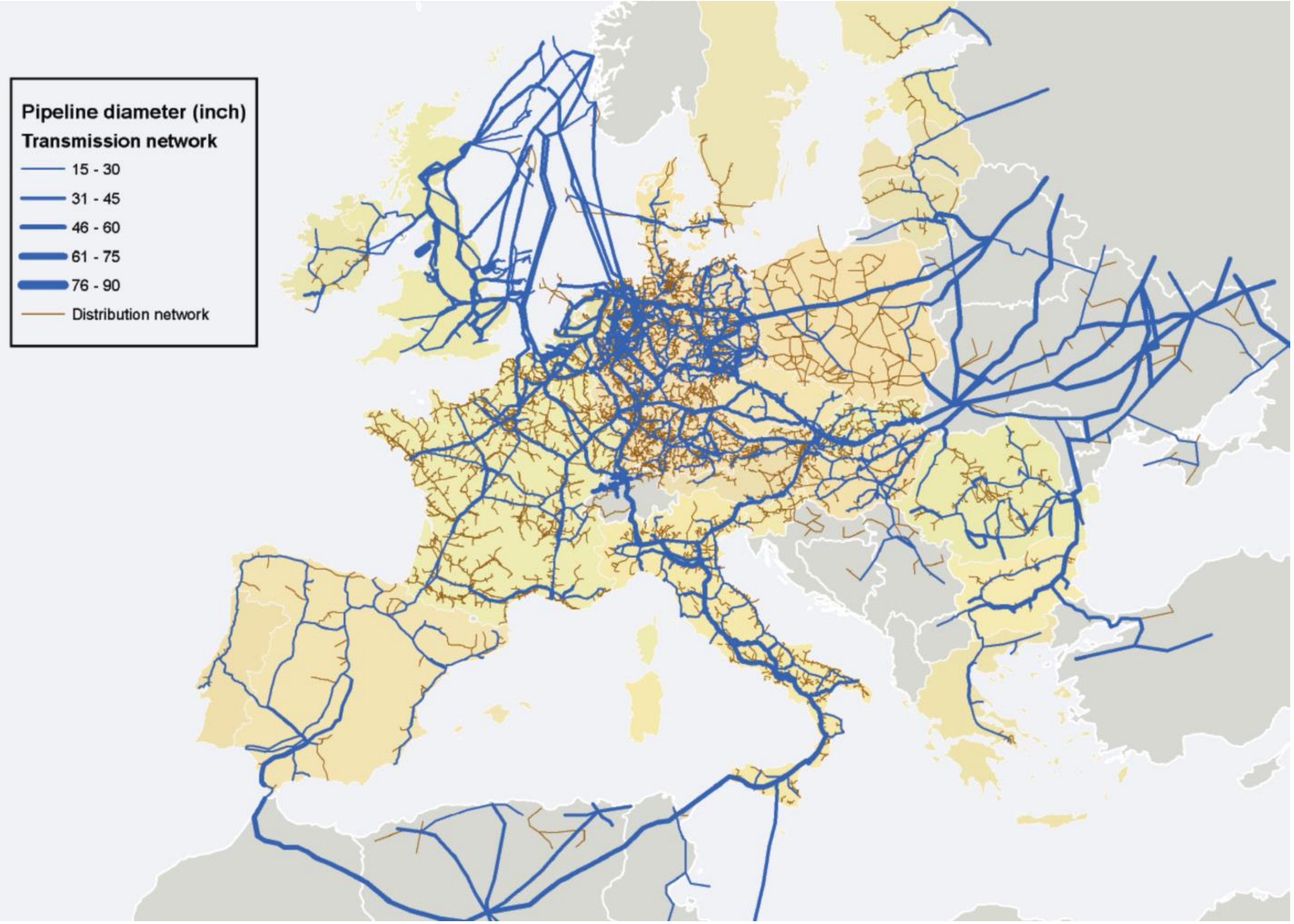


Especially, when things go wrong!



Robustness

Datasets: European gas pipeline network



Transmission network

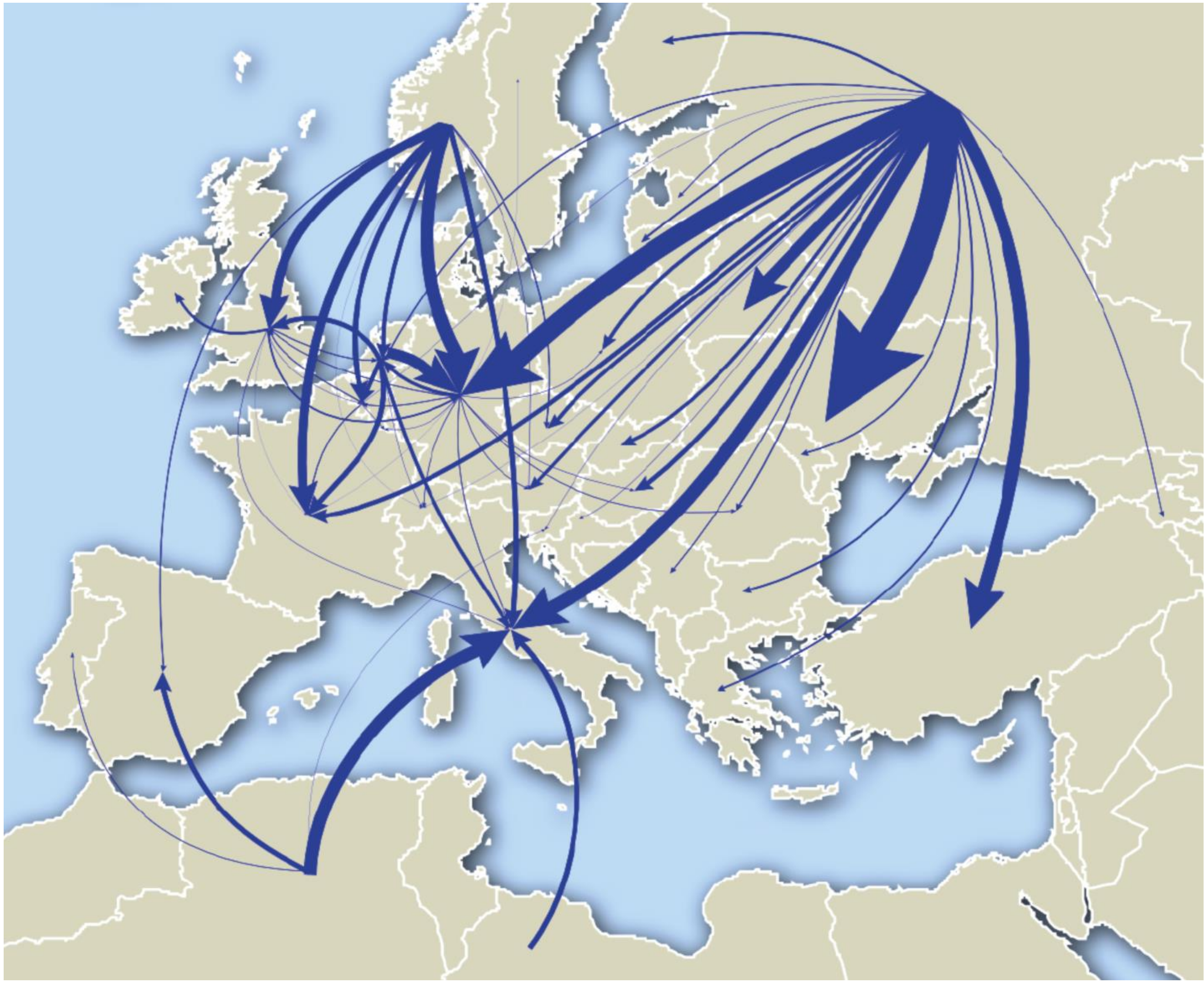
(d>= 15, + interconnections)

2207 nodes, 2696 links

Complete network

24010 nodes, 25554 links

Datasets: Gas trade movements by pipeline (2007)



Data collected from: www.bp.com

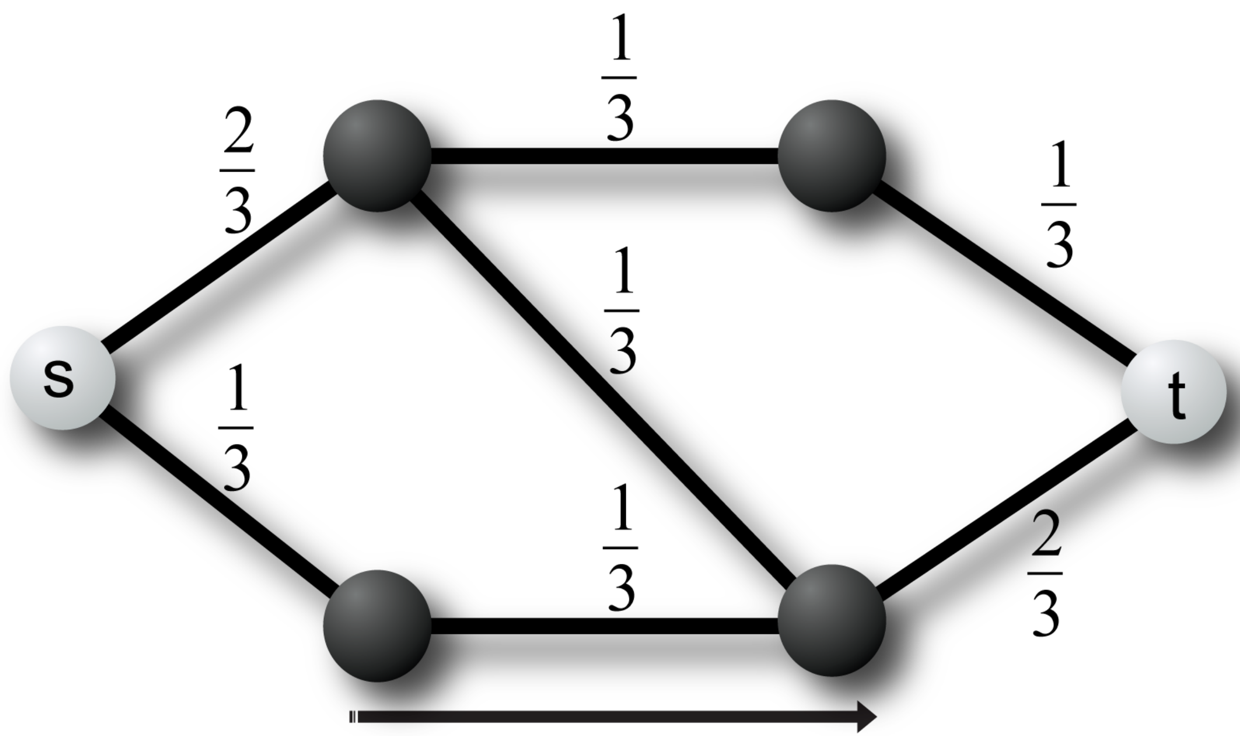
www.iea.org

Betweenness centrality

Consider a substrate network $G_S = (V_S, E_S)$ with node-set V_S and link-set E_S. The betweenness centrality of link $e_{ij} \in E_S$ is defined as the relative number of shortest paths between all pairs of nodes which pass through e_{ij},

$$g(e_{ij}) = \sum_{\substack{s,t \in V_S \\ s \neq t}} \frac{\sigma_{s,t}(e_{ij})}{\sigma_{s,t}} \qquad (1)$$

where $\sigma_{s,t}$ is the number of shortest paths from node s to node t and $\sigma_{s,t}(e_{ij})$ is the number of these paths passing through link e_{ij}.

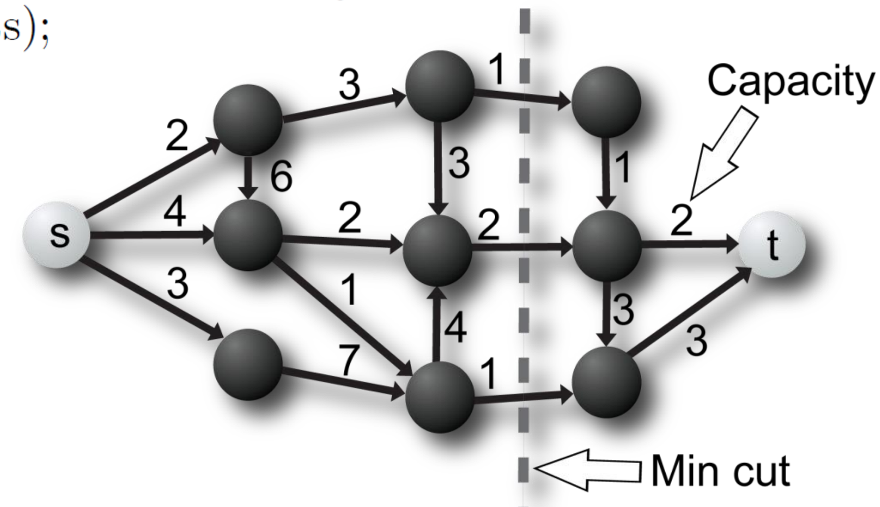


The max-flow problem

The maximum flow problem can be stated as follows: In a network with link capacities, we wish to send as much flow as possible between two particular nodes, a source and a sink, without exceeding the capacity of any link.

Formally, an *s-t flow network* $G_F = (V_F, E_F, s, t, c)$ is a digraph with node-set V_F, link-set E_F, two distinguished nodes, a *source* s and a *sink* t, and a capacity function $c : E_F \to \mathbb{R}_0^+$. A *feasible flow* is a function $f : E_F \to \mathbb{R}_0^+$ satisfying the following two conditions:

- $0 \leq f(e_{ij}) \leq c(e_{ij}), \forall e_{ij} \in E_F$ (capacity constraints);
- $\sum_{j:e_{ji} \in E_F} f(e_{ji}) = \sum_{j:e_{ij} \in E_F} f(e_{ij}), \forall i \in V \setminus \{s, t\}$ (flow conservation constraints);



The *maximum s-t flow* is defined as the maximum flow into the sink, $f_{st}(G_F) = \max(\sum_{i:e_{it} \in E_F} f(e_{it}))$ subject to the conditions that the flow is feasible.

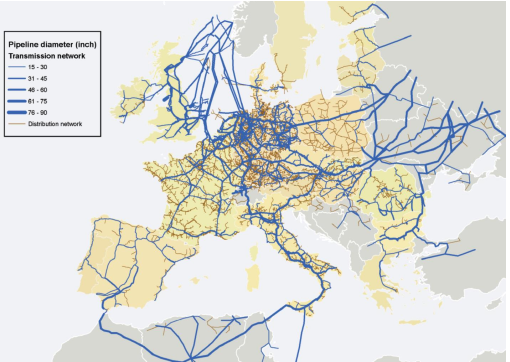
Generalized betweenness centrality

Create a flow network by partitioning the substrate network, $G_S = (V_S, E_S)$, into a set of disjoint subgraphs $V_F = \{(V_{S_1}, E_{S_1}), \cdots, (V_{S_M}, E_M)\}$. The flow network $G_F = (V_F, E_F)$ is then defined as the directed network of flows among the subgraphs in V_F, where the links E_F are weighted by the value of aggregate flow among the V_F.

The betweenness centrality of link $e_{ij} \in E_S$ is defined as

$$g(e_{ij}) = \sum_{\substack{s,t \in V_S \\ s \neq t}} \frac{\sigma_{s,t}(e_{ij})}{\sigma_{s,t}} \tag{1}$$

The *generalized betweenness centrality* (generalized betweenness) of link $e_{ij} \in E_S$ is defined as follows. Let $T_{K,L}$ be the flow from source subgraph $K = (V_K, E_K) \in V_F$ to sink subgraph $L = (V_L, E_L) \in V_F$. Take each link $e_{KL} \in E_F$ and compute the betweenness centrality from Eq. (1) of $e_{ij} \in E_S$ restricted to source nodes $s \in V_K$ and sink nodes $t \in V_L$. The contribution of that flow network link is then weighted by $T_{K,L}$ and normalized by the number of links in a complete bipartite graph between nodes in V_K and V_L



$$G_{ij} = \sum_{e_{KL} \in E_F} \sum_{s \in V_K, t \in V_L} \frac{T_{K,L}}{|V_K||V_L|} \frac{\sigma_{st}(e_{ij})}{\sigma_{st}}.$$

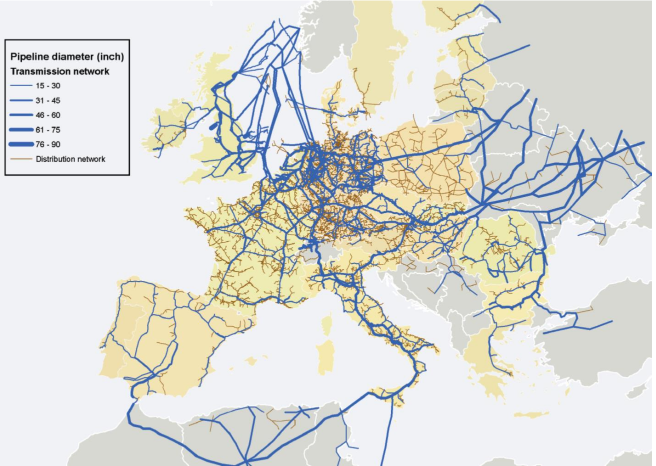
Generalized max-flow betweenness vitality

Question: How does the maximum flow between all sources and sinks change, if we remove a link e_{ij} from the network?

In the absence of a detailed flow model, we calculated the flow that is lost when a link e_{ij} becomes non-operational assuming that the network is working at maximum capacity. In agreement with Eq. (2), we define the *generalized max-flow betweenness vitality* (generalized vitality):

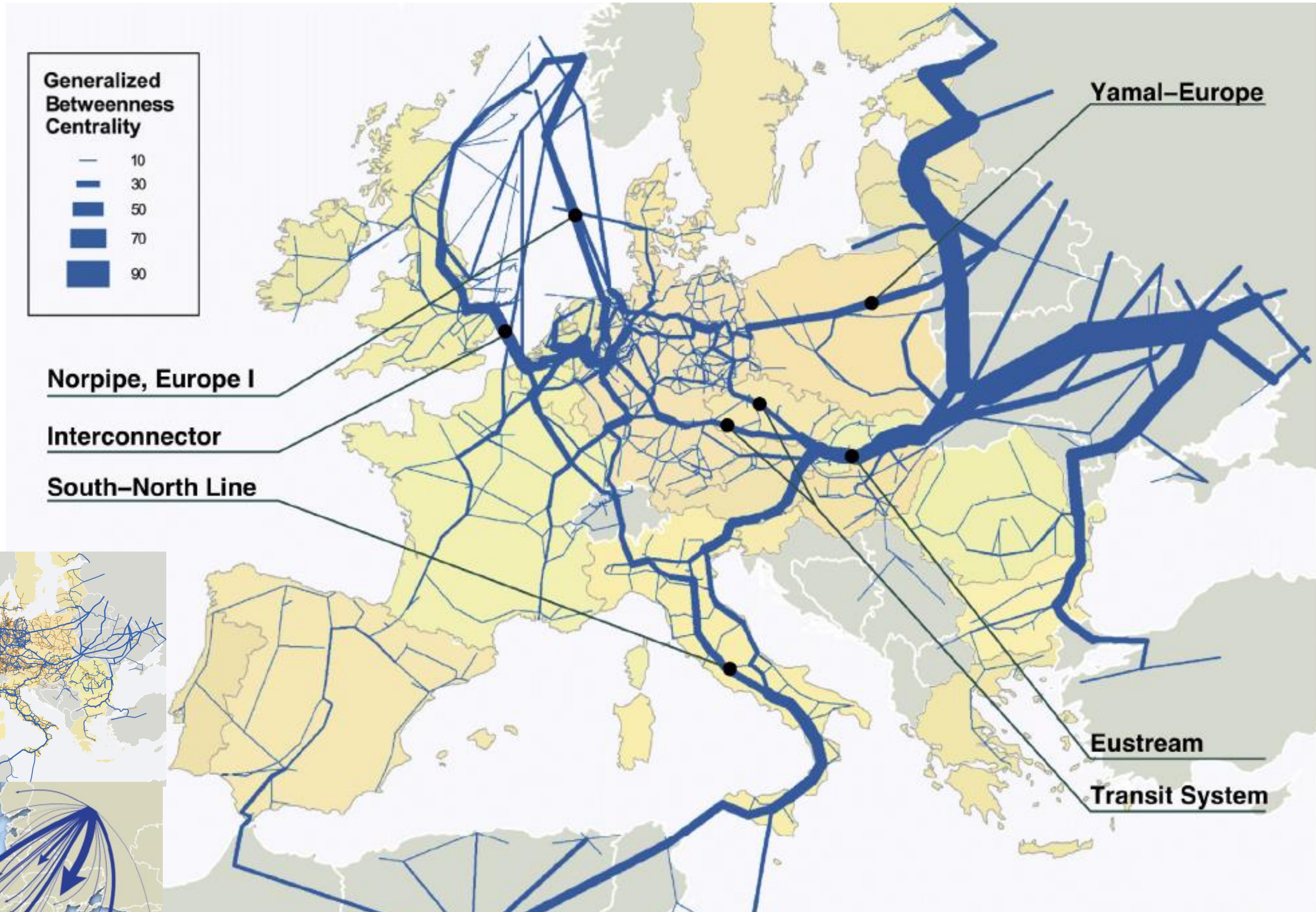
$$V_{ij} = \sum_{e_{KL} \in E_F} \sum_{s \in V_K, t \in V_L} \frac{T_{K,L}}{|V_K||V_L|} \frac{\Delta_{st}^{G_F}(e_{ij})}{f_{st}(G_F)}, \quad (3)$$

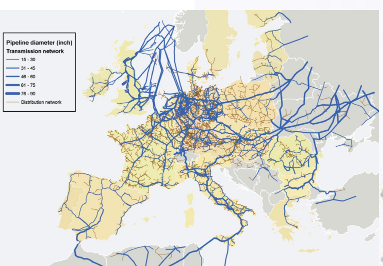
where the amount of flow which must go through link e_{ij} when the network is operating at maximum capacity is given by the vitality of the link: $\Delta_{st}^{G_F}(e_{ij}) = f_{st}(G_F) - f_{st}(G_F \backslash e_{ij})$, and $f_{st}(G_F)$ is the maximum s-t flow in G_F.



Generalized betweenness applied to gas networks

$$G_{ij} = \sum_{e_{KL}\in E_F} \sum_{s\in V_K, t\in V_L} \frac{T_{K,L}}{|V_K||V_L|} \frac{\sigma_{st}(e_{ij})}{\sigma_{st}}.$$

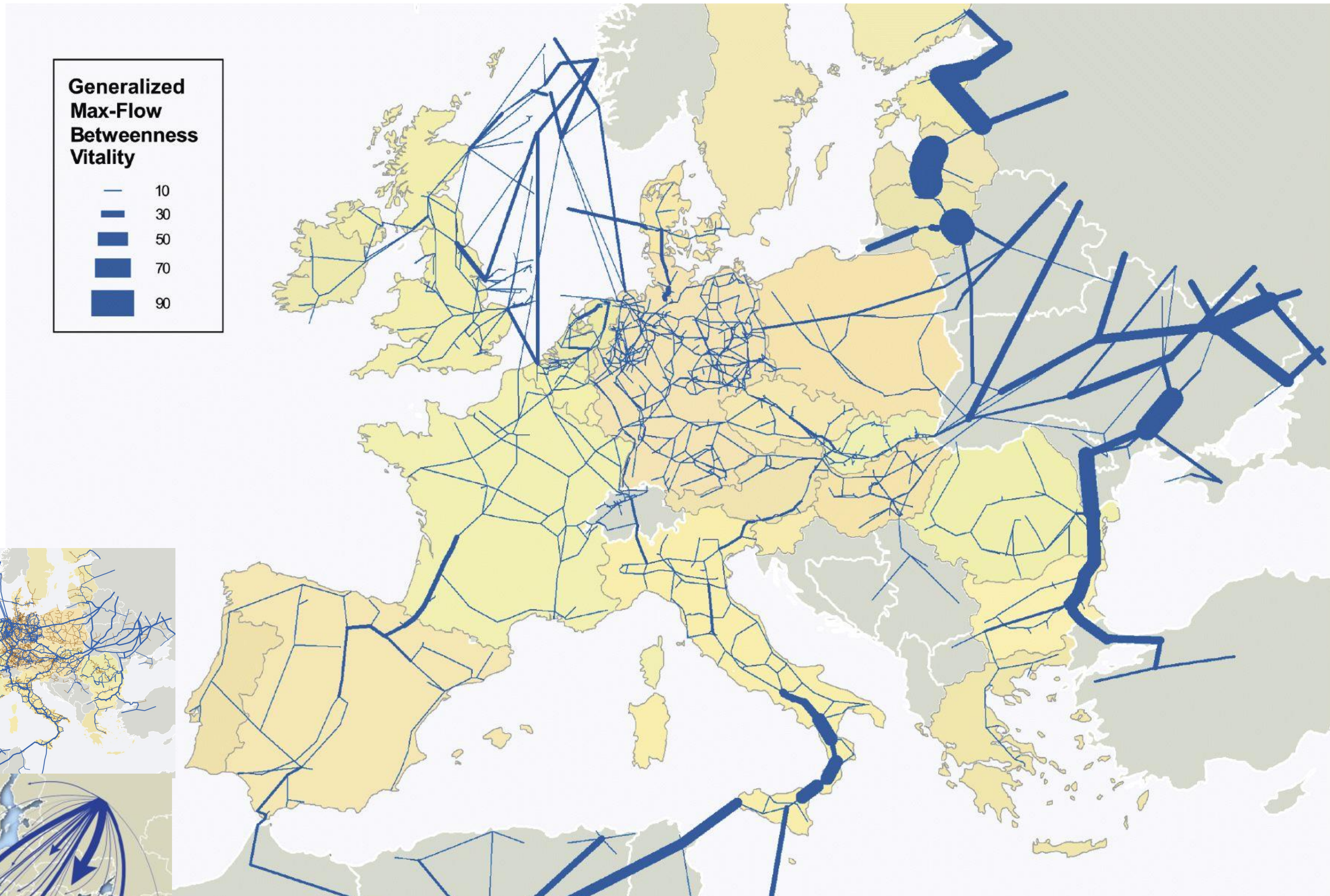


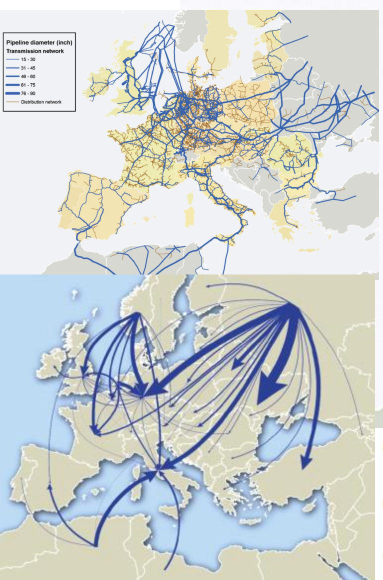


Trans-European natural gas network. Link thickness is proportional to the generalized betweenness centrality, where the sets K and L are countries and the values of $T_{K,L}$ are taken from the data in the flow network.

Generalized vitality applied to gas networks

$$V_{ij} = \sum_{e_{KL} \in E_F} \sum_{s \in V_K, t \in V_L} \frac{T_{K,L}}{|V_K||V_L|} \frac{\Delta_{st}^{G_F}(e_{ij})}{f_{st}(G_F)},$$



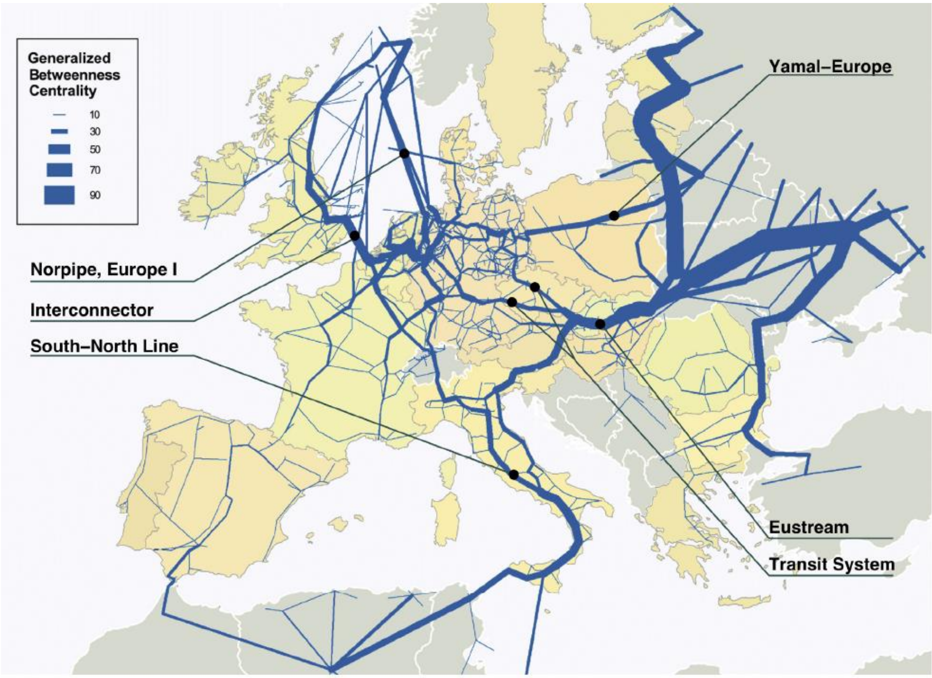


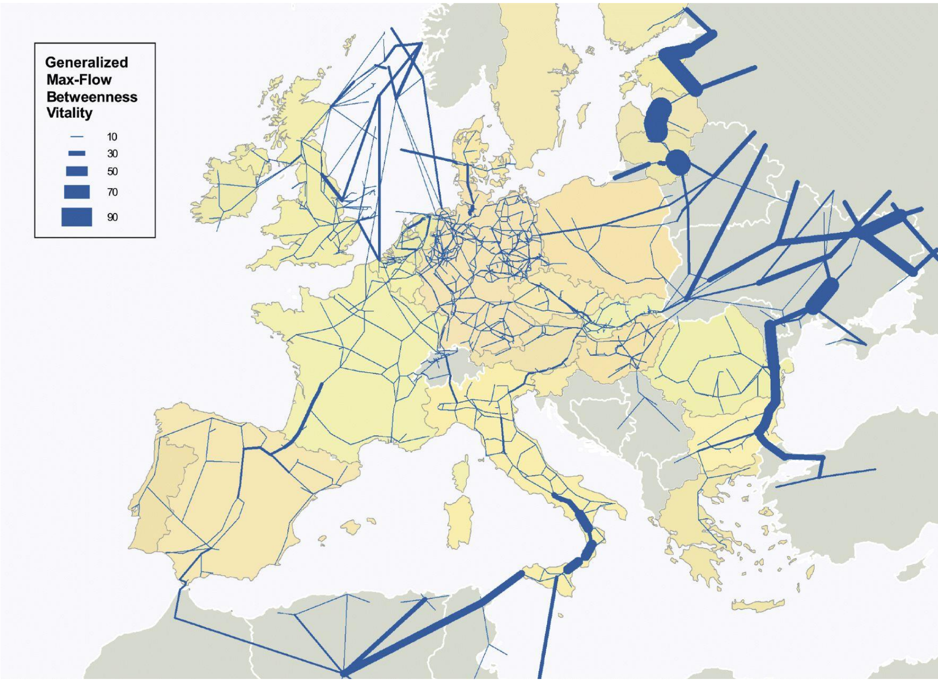
Trans-European natural gas network. Link thickness is proportional to the generalized max-flow betweenness vitality, where the sets K and L are countries and the values of $T_{K,L}$ are taken from the data in the flow network.

Robust infrastructure network: error tolerant to failures of high load links

High Traffic Backbone + Error Tolerance = Robustness

(*i.e.* **Good Engineering**)





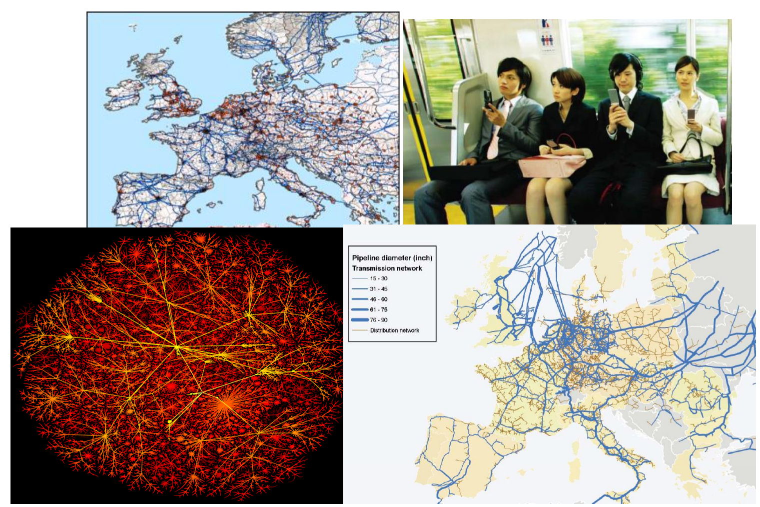
Rui Carvalho, Lubos Buzna, Flavio Bono, Eugenio Gutiérrez, Wolfram Just, and David Arrowsmith, *Phys. Rev. E* **80**, 016106 (2009)

Fair Flows

The Max-Min Fairness Algorithm

From cake –cutting to fair allocation of network resources

- Mathematicians have been occupied with fairness in cake-cutting since the 1940s (Steinhaus, Knaster and Banach);
- But what about the similar network problem: how to allocate network capacity among users in a fair way?
- Challenge: how to gain analytical insights into the fair allocation of network capacity on very large networks?



The Max-Min Fairness Algorithm

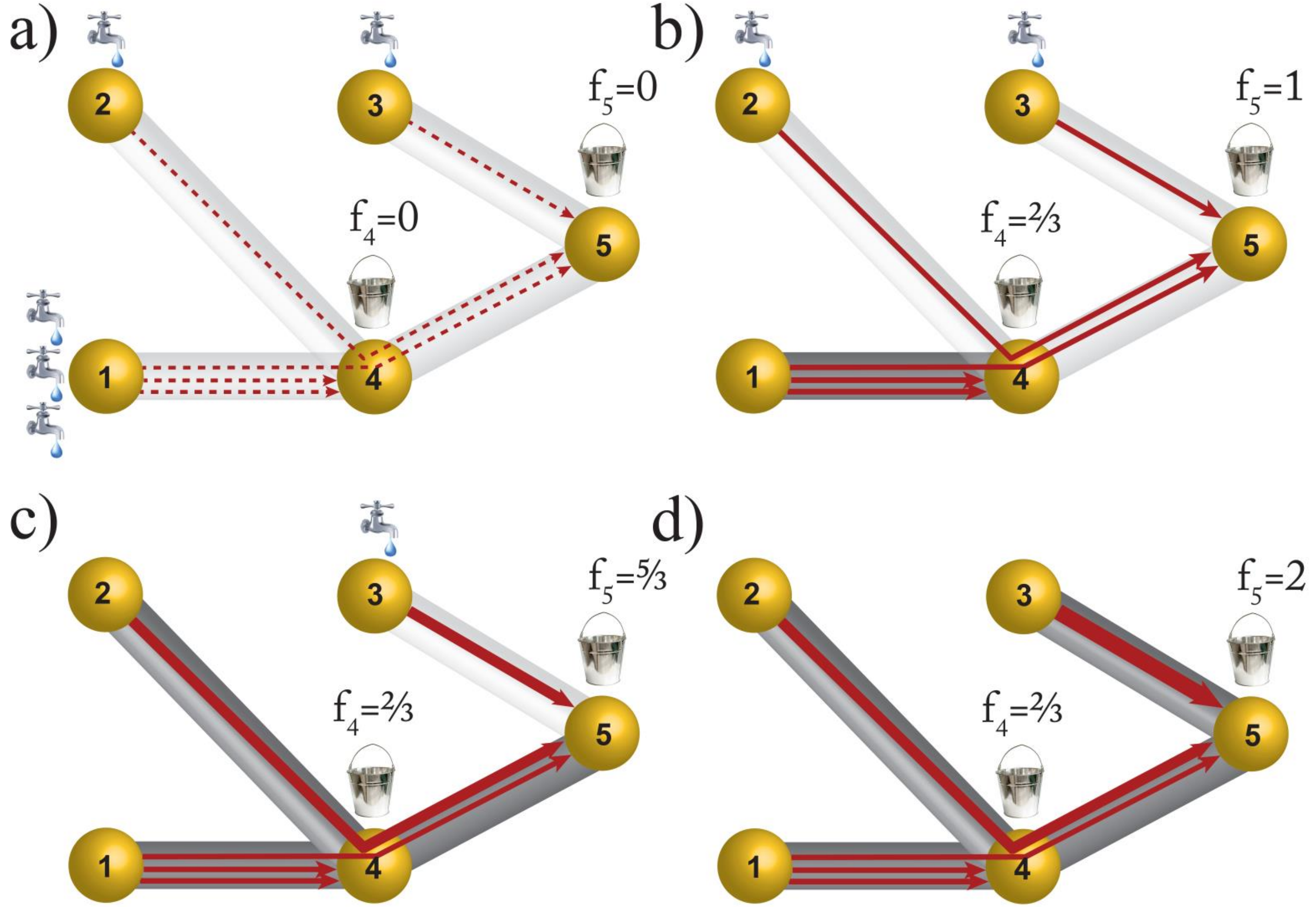
The max-min fairness problem as the lexicographic maximin problem

- The relationship between links and paths can be described by the *link-path incidence matrix* B. Set $B_{ij} = \delta_{p_j}(e_i) = 1$ if the link e_i belongs to the path p_j, and set $B_{ij} = \delta_{p_j}(e_i) = 0$ otherwise.
- The problem:

$$
\begin{aligned}
\underset{\mathbf{f}}{\mathit{lexmax}} \quad & \{(f_{p_1}, f_{p_2}, \ldots, f_{p_N})\} \\
\text{subject to} \quad & f_{p_1} \leq f_{p_2} \leq \ldots \leq f_{p_N} \\
& Bf \leq c \\
& f_{p_i} \geq 0,
\end{aligned}
$$

Max-Min Fair Flows

- Consider a set of *s*-*t* pairs, each connected by a set of paths;
- Each edge of a path transports the same path flow;



Max-Min Fairness

- Typically, connections are specified by a fixed set of paths, and one wants to allocate path flows to each of these paths.

- A set of path flows is max-min fair if no path flow can be increased without simultaneously decreasing another path flow that is already less or equal to the former.

$$\exists p_{(i,k)} \in P : f'_{p_{(i,k)}} > f_{p_{(i,k)}} \implies \exists p_{(j,l)} \in P : f'_{p_{(j,l)}} < f_{p_{(j,l)}} \wedge f_{p_{(j,l)}} \leq f_{p_{(i,k)}}$$

to increase a path flow

you need to decrease another path flow

that is already smaller

- D. Bertsekas and R. Gallager, Data Networks, Prentice-Hall, 1987
- J. Kleinberg, Y. Rabani and E. Tardos, Journal of Computer and Systems Sciences **63**, 2 (2001)

Max-min Fairness Algorithm: First Step

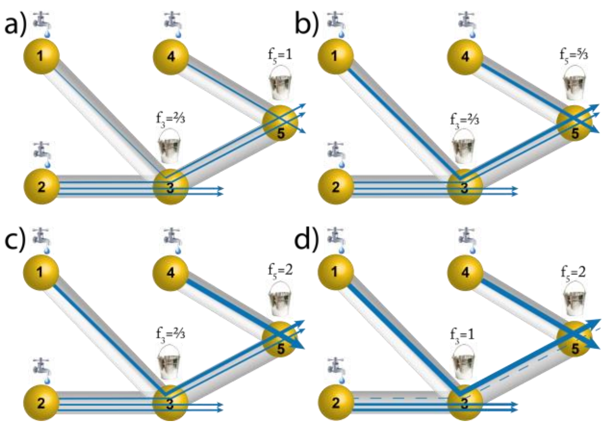
[2]

- $P^{(j)}$: set of paths on the network at iteration j of MMF, $P^{(j)}(e)$: subset of paths in $P^{(j)}$ that contain edge e;
- Assign $P^{(1)} = P$ and $c^{(1)}(e) = c(e)$ for all $e \in E$, and a path flow $f^{(0)}_{p_{(i,k)}} = 0$ to each path $p_{(i,k)} \in P^{(1)}$;
- In the first step of the MMF algorithm, for each edge e with non-zero capacity that belongs to at least one path, define the edge capacity divided equally among all paths that cross the edge at iteration j of the MMF algorithm as

$$\phi^{(j)}(e) = c^{(j)}(e) / \left| P^{(j)}(e) \right| .$$

- We then find the minimum of $\phi^{(j)}(e)$, given by

$$\Delta f^{(j)} = \min_{e \in E} \phi^{(j)}(e).$$



Max-min Fairness Algorithm: First Step

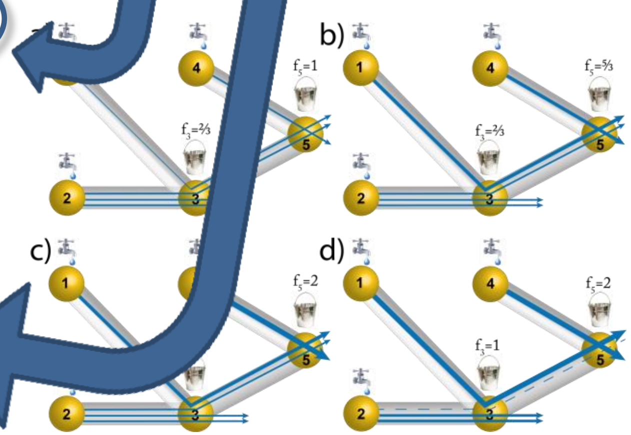
- $P^{(j)}$: set of paths on the network at iteration j of MMF, $P^{(j)}(e)$: subset of paths in $P^{(j)}$ that contain edge e;
- Assign $P^{(1)} = P$ and $c^{(1)}(e) = c(e)$ for all $e \in E$, and a path flow $f^{(0)}_{p_{(i,k)}} = 0$ to each path $p_{(i,k)} \in P^{(1)}$;
- In the first step of the MMF algorithm, for each e
non-zero capacity that belongs to at least one pat
edge capacity divided equally among all paths tha
edge at iteration j of the MMF algorithm as

$$\phi^{(j)}(e) = c^{(j)}(e) / \left| P^{(j)}(e) \right| .$$

- We then find the minimum of $\phi^{(j)}(e)$, given by

$$\Delta f^{(j)} = \min_{e \in E} \phi^{(j)}(e).$$

The crucial elements of the algorithm



Max-min Fairness Algorithm: Second Step

- Increase all path flows of paths in $P^{(j)}$ by $\Delta f^{(j)}$, such that

$$f_{p_{(i,k)}}^{(j)} = \begin{cases} f_{p_{(i,k)}}^{(j-1)} + \Delta f^{(j)} & \text{if } p_{(i,k)} \in P^{(j)} \\ f_{p_{(i,k)}}^{(j-1)} & \text{if } p_{(i,k)} \in P \setminus P^{(j)} \end{cases}$$

  The effect is to saturate the set of bottleneck edges $E_B^{(j)} = \{e_B \in E : \sum_{p_{(i,k)} \in P^{(j)}} \delta_{p_{(i,k)}}(e_B)\Delta f^{(j)} = c^{(j)}(e_B)\}$, and consequently also to saturate the set of paths that contain at least one bottleneck edge.
- Create a residual network

$$c^{(j+1)}(e) = c^{(j)}(e) - \sum_{p_{(i,k)} \in P^{(j)}} \delta_{p_{(i,k)}}(e)\Delta f^{(j)}.$$

  Note that $c^{(j+1)}(e_B) = 0$ for all $e_B \in E_B$, that is all edges in E_B will be *saturated* after this step.
- Paths that contain at least one edge $e_B \in E_B$ are *saturated paths*, i.e. their path flow will not be increased in subsequent iterations of MMF.

Max-min Fairness Algorithm: Second Step (cont.)

- Remove the set of saturated paths from P, that is

$$P^{(j+1)} = P^{(j)} \setminus \cup_{e_B \in E_B^{(j)}} P^{(j)}(e_B).$$

  We say that $P^{(j+1)}$ is the set of augmenting paths because the path flows of paths in $P^{(j+1)}$ can still be increased in subsequent iterations of the algorithm.
- If $P^{(j+1)}$ is not empty, increase the iteration, $j \to j+1$, and go back to the first step, otherwise stop and store the value of j as j^*.
- The max-min fair (MMF) flow on edge e is then the sum of path flows over all paths that cross the edge after the algorithm terminates:

$$F(e) = \sum_{p_{(i,k)} \in P} \delta_{p_{(i,k)}}(e) f_{p_{(i,k)}}^{(j^*)}.$$

Max-Min Fairness in Nearest Neighbour Networks

[3]

Assumptions

- Consider a set of *s*-*t* pairs, each connected by a set of paths;
- Each edge of a path transports the same path flow;

Furthermore, we consider transport:

- over shortest paths (path counting)
- on networks with uniform edge capacity (path counting)
- on regular grids (analytical results)
- on 1 s-t pair (analytical results);

Max-Min Fairness: Assumptions [3]

- Consider a set of *s*-*t* pairs, each connected by a set of paths;
- Each edge of a path transports the same path flow;

Furthermore, we consider transport:

- over shortest paths (path counting)
- on networks with uniform edge capacity (path counting)
- on regular grids (analytical results)
- on 1 s-t pair (analytical results);

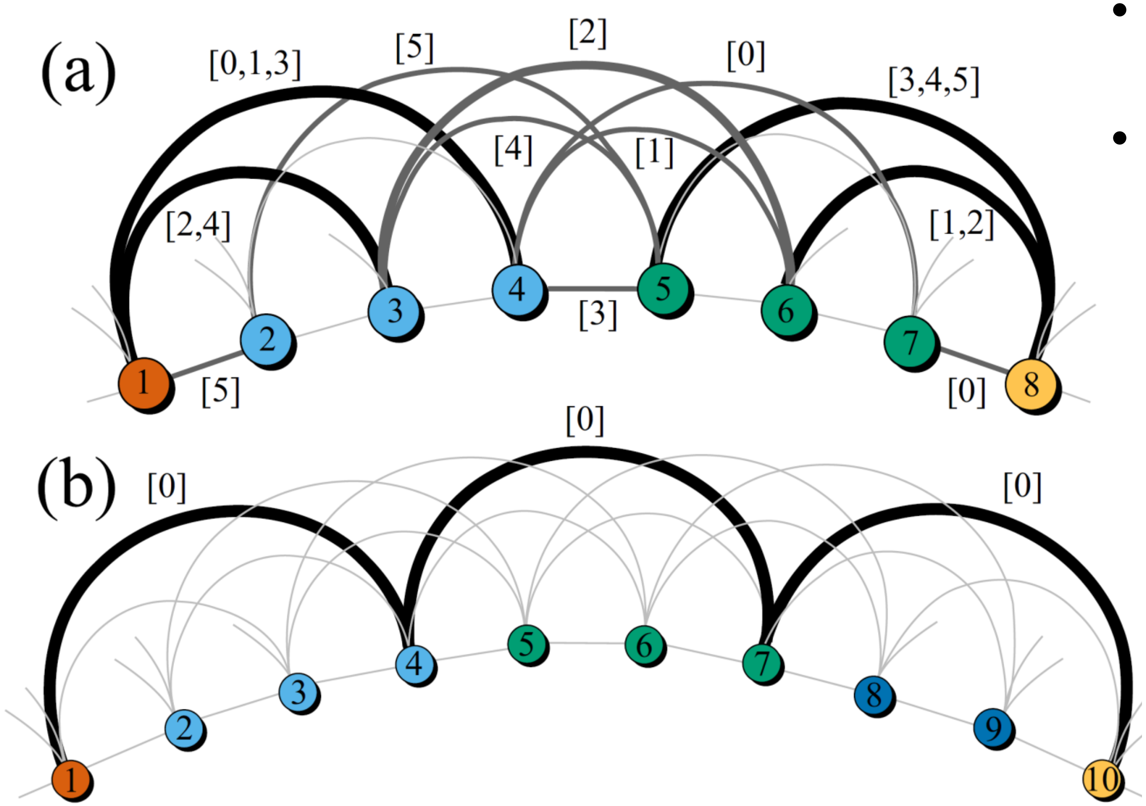
The MMF flow allocation depends exclusively on the **number of paths** passing through each edge

[4]

K-nearest neighbour networks

Two ways of measuring distance:

- difference *d* between the indexes of *s* and *t*
- Shortest path distance *L* between *s* and *t*

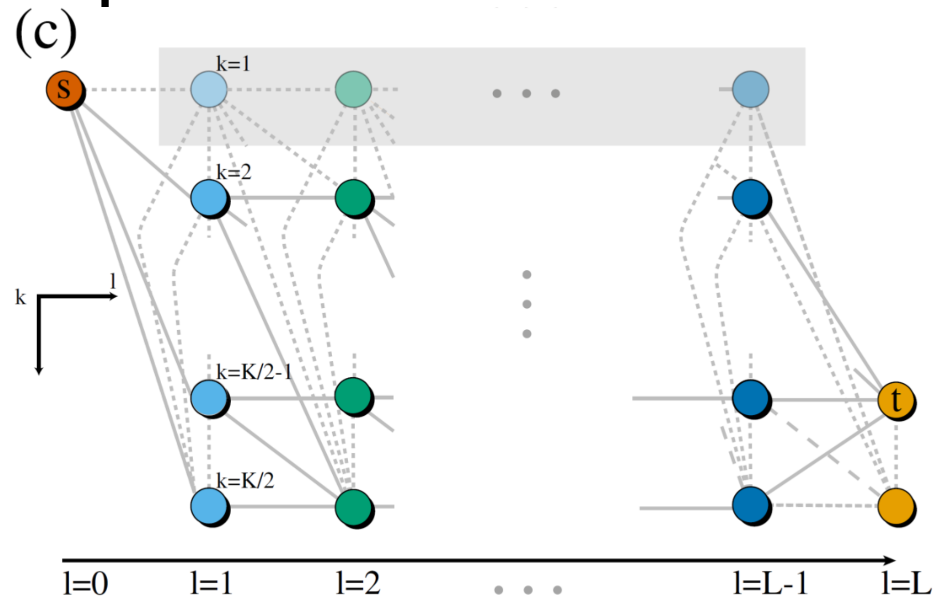


$$d^{\min}(K, L) = [d - ((d-1) \pmod{K/2})]$$
$$= (L-1) * K/2 + 1,$$

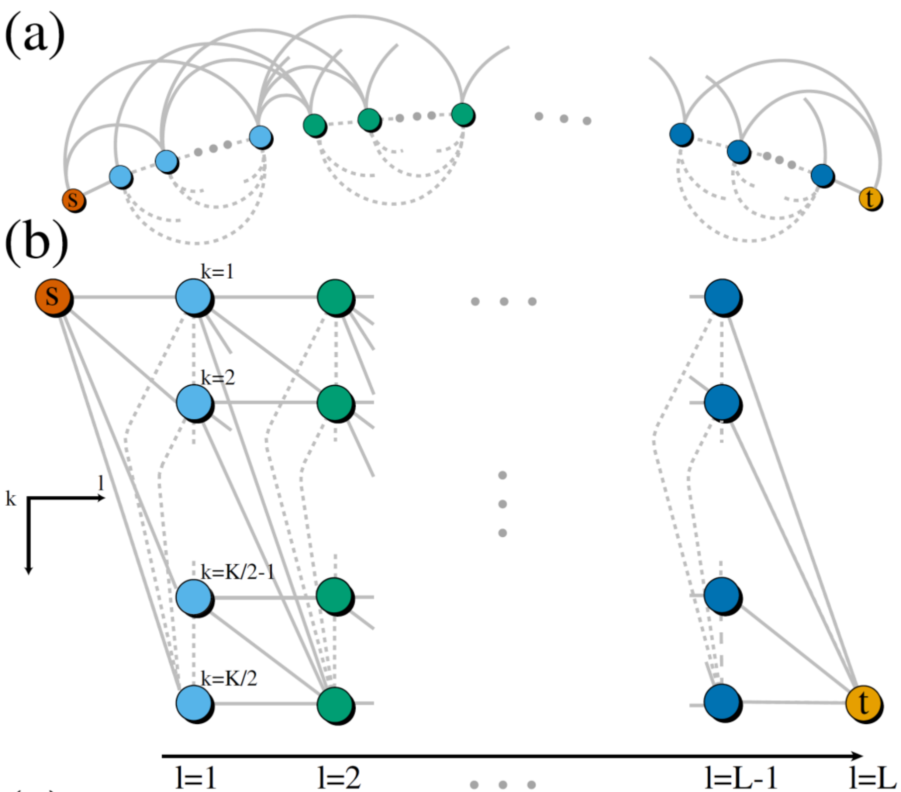
FIG. 2. (Colour online). Two layouts of 6-nearest neighbour networks. Nodes are numbered by an index from west to east, and the distance d that separates two nodes is the difference between their indexes. A source s and a sink t pair are placed (a) $d = 7$ and (b) $d = 9$ nodes apart, respectively. The shortest paths between s and t are identified by numeric labels on the edges. The shortest path length between the two nodes is $L = 3$ in both panels. There are six s-t shortest paths in (a), but only one in (b). Edges have unit capacity, edge thickness is proportional to the MMF flow allocation, and saturated edges (or bottlenecks) are drawn in black. Nodes are coloured according to their shortest path length from the source.

Number of shortest paths from *s* to *t*

[5]



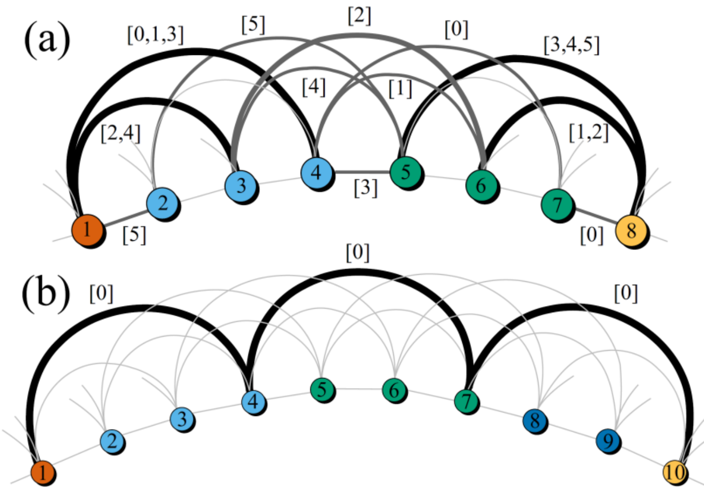
- A shortest path is an arrangement of the *K/2* rows into *L-1* 'stars' and *K/2-1* 'bars';
- Each ***** marks a row and each **|** marks a change of consecutive rows in the path, e.g. *|*||*;



- So the number of shortest paths between *s* and *t* is given by the number of ways to distribute *L-1* identical balls (*****) the into *K/2* distinct bags (rows), where each row can get any number of balls:

$$N(K/2, L) = \left(\!\!\binom{K/2}{L-1}\!\!\right) = \binom{K/2 + L - 2}{L - 1}$$

Number of s-t shortest paths as a function of s-t distance on K-nearest neighbour graphs



$$\{n(K=2,d)\}_{d=1,\ldots,\lfloor |V|/2 \rfloor} = \underbrace{1}_{L=1}, \underbrace{\boxed{1}}_{2}, \underbrace{1}_{3}, \underbrace{1}_{4}, \underbrace{1}_{5} \ldots$$

$$\{n(K=4,d)\}_{d=1,\ldots,\lfloor |V|/2 \rfloor} = \underbrace{1, 1}_{L=1}, \underbrace{\boxed{2}, 1}_{2}, \underbrace{3, 1}_{3}, \underbrace{4, 1}_{4}, \underbrace{5, 1}_{5} \ldots$$

$$\{n(K=6,d)\}_{d=1,\ldots,\lfloor |V|/2 \rfloor} = \underbrace{1, 1, 1}_{L=1}, \underbrace{\boxed{3}, 2, 1}_{2}, \underbrace{6, 3, 1}_{3}, \underbrace{\boxed{10}, 4, 1}_{4}, \underbrace{15, 5, 1}_{5}, \ldots$$

$$\{n(K=8,d)\}_{d=1,\ldots,\lfloor |V|/2 \rfloor} = \underbrace{1, 1, 1, 1}_{L=1}, \underbrace{\boxed{4}, 3, 2, 1}_{2}, \underbrace{10, 6, 3, 1}_{3}, \underbrace{20, \boxed{10}, 4, 1}_{4}, \underbrace{35, 15, 5, 1}_{5}, \ldots$$

$$\{n(K=10,d)\}_{d=1,\ldots,\lfloor |V|/2 \rfloor} = \underbrace{1, 1, 1, 1, 1}_{L=1}, \underbrace{\boxed{5}, 4, 3, 2, 1}_{2}, \underbrace{\boxed{15}, 10, 6, 3, 1}_{3}, \underbrace{35, 20, 10, 4, 1}_{4}, \underbrace{70, 35, 15, 5, 1}_{5}, \ldots$$

Determining the sink inflow on a *K*-nearest neighbour network as the *s*-*t* distance d is varied is reduced to the problem of calculating sink inflows for $d^{min}(K,L)$.

Path Counting Methods

- $N(K/2, L) = \left(\binom{K/2}{L-1}\right) = \binom{K/2+L-2}{L-1}$.
- Using the asymptotic expansion of the binomial coefficient for L large,

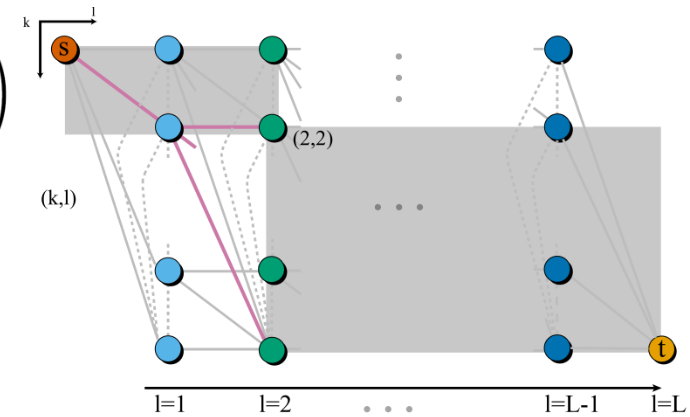
$$N(K, L) = \frac{(L-1)^{K/2-1}}{(K/2-1)!}\left(1 + \frac{(K/2-1)K/2}{2(L-1)} + O(L)\right)$$

- Problem: for L large, $N(K, L)$ –the input to the algorithm– grows polynomially with L (for fixed K) and exponentially with K (for fixed L).
- Solution: device a simplified MMF algorithm for K-nearest neighbour graphs that depends only on path counting methods.

Path Counting Methods

- $$n(s, v_{(k,l)}, t) = N(k,l)N(K/2-k+1, L-l) = \left(\!\!\binom{k}{l-1}\!\!\right)\left(\!\!\binom{K/2-k+1}{L-l-1}\!\!\right)$$

  where $1 \le k \le K/2$ and $1 \le l \le L-1$.



- $$n(s, v_{(k_1,l_1)}, \dots, v_{(k_n,l_n)}, t) =$$
  $$= N(k_1,l_1)\prod_{i=1}^{n-1} N(k_{i+1}-k_i+1, l_{i+1}-l_i)N(K/2-k_n+1, L-l_n)$$
  $$= \left(\!\!\binom{k_1}{l_1-1}\!\!\right)\prod_{i=1}^{n-1}\left(\!\!\binom{k_{i+1}-k_i+1}{l_{i+1}-l_i-1}\!\!\right)\left(\!\!\binom{K/2-k_n+1}{L-l_n-1}\!\!\right),$$

  where $1 \le k_1 \le \cdots \le k_n \le K/2$,
  $1 \le l_1 < l_2 < \cdots < l_n \le L-1$.

- $n(e_{s,v_{(k,1)}}) = N(K/2-k+1, L-1) = \left(\!\binom{K/2-k+1}{L-2}\!\right)$.

- $n(e_{s,v_{(k_1,1)}}, e_{v_{(k_2,L-1)},t}) = N(k_2-k_1+1, L-2) = \left(\!\binom{k_2-k_1+1}{L-3}\!\right)$.

Relation between path counting and path flows [7]

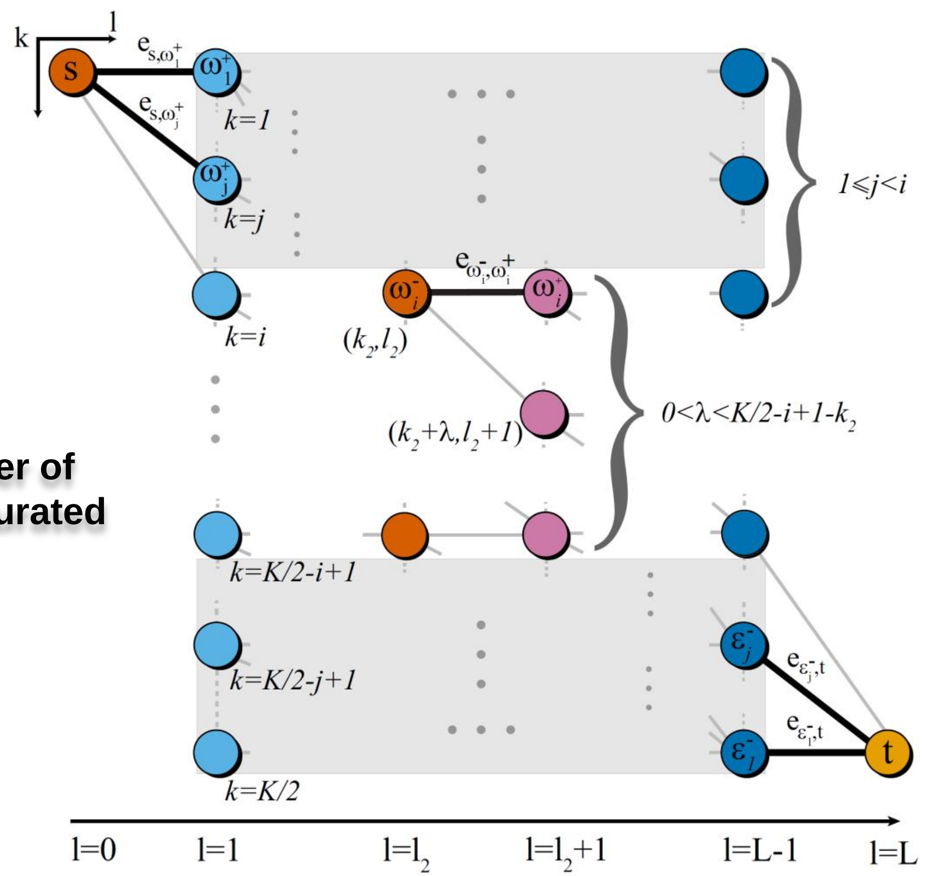
$$\left|P^{(j)}(e_{s,\omega_i^+})\right| = n(e_{s,\omega_i^+}) - \sum_{q=1}^{j-1} n(e_{s,\omega_i^+}, e_{\varepsilon_q^-,t})$$
$$= n(e_{s,\omega^+_{(i+j-1)}})$$
$$= N(K/2 - (i+j) + 2, L-1),$$

Assuming that the bottlenecks are edges of the source or sink:

$$\Delta f^{(i)} = \min_{e \in E} \phi^{(i)}(e) = \phi^{(i)}(e_{s,\omega_i^+})$$
$$= c^{(i)}(e_{s,\omega_i^+}) / \left|P^{(i)}(e_{s,\omega_i^+})\right|$$
$$= \frac{c - \sum_{q=0}^{i-1} \left|P^{(q)}(e_{s,\omega_i^+})\right| \Delta f^{(q)}}{\left|P^{(i)}(e_{s,\omega_i^+})\right|}$$
$$= \frac{c - \sum_{q=0}^{i-1} n(e_{s,\omega^+_{(i+q-1)}}) \Delta f^{(q)}}{n(e_{s,\omega^+_{(2i-1)}})}$$
$$= \frac{c - \sum_{q=0}^{i-1} N(K/2 - (i+q) + 2, L-1) \Delta f^{(q)}}{N(K/2 - 2(i-1), L-1)}$$

number of unsaturated paths

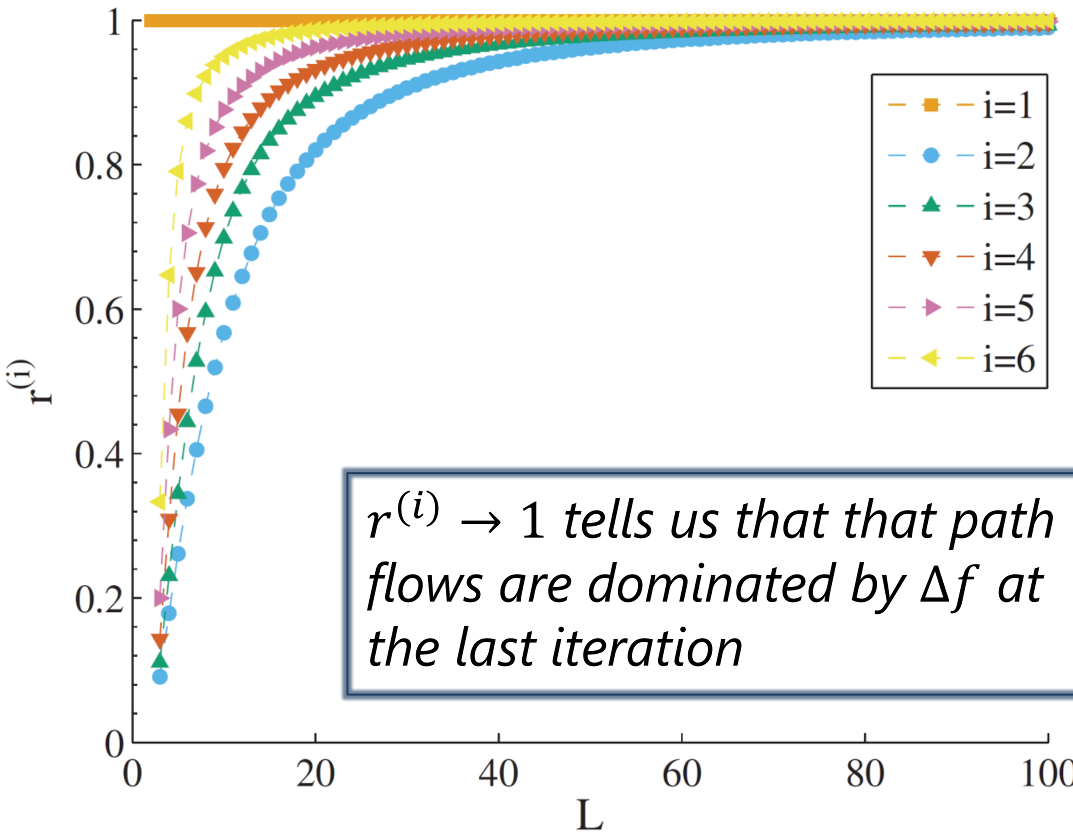
found recursively



The path flow increment at the last iteration

[8]

$$r^{(i)} = \frac{\Delta f^{(i)}}{\sum_{j=1}^{i} \Delta f^{(i)}}$$

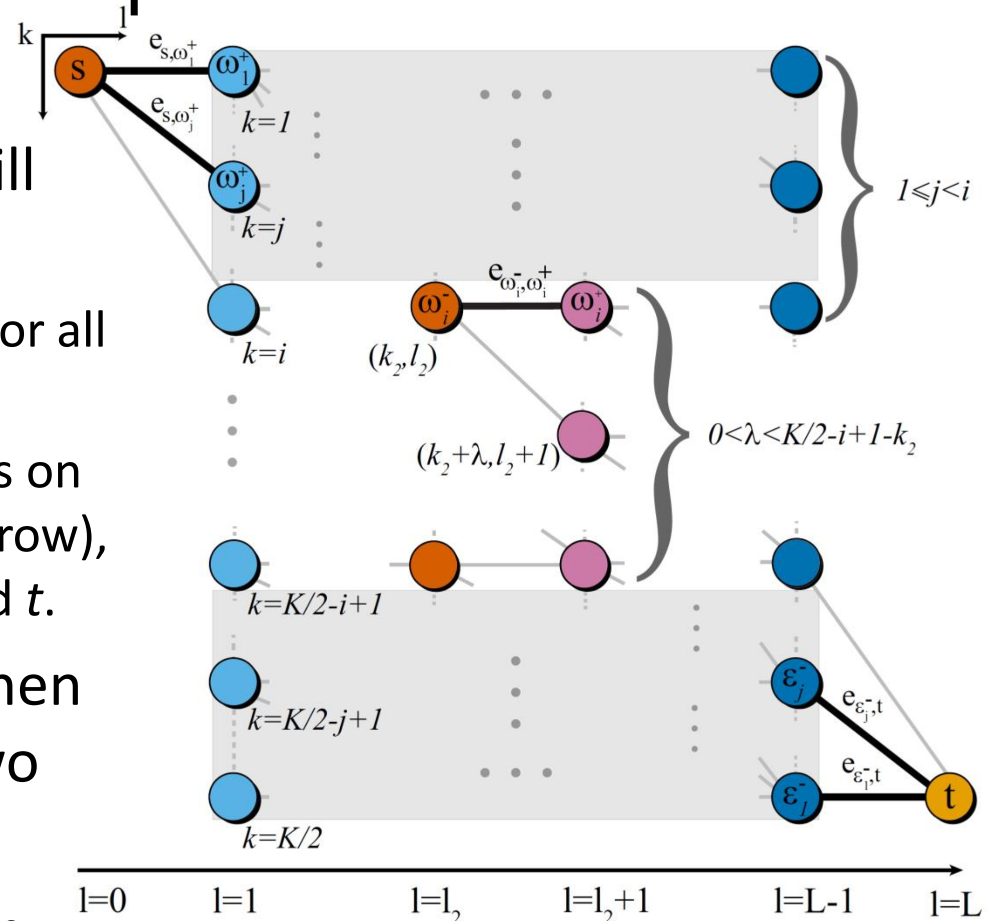


$r^{(i)} \rightarrow 1$ tells us that that path flows are dominated by Δf at the last iteration

FIG. 8. (Colour online). Plot of the ratio $r^{(i)}$ between the path flow increment at iteration i and the path flow of paths that are saturated at iteration i, as a function of L for $K = 24$ as the iterations i increase. The path flow is dominated by the path flow increment at the last iteration when $r \simeq 1$.

From path counting to path flows and back

- Path counting methods are still valid when :
  - we have bottlenecks at *s* and *t* for all rows above the current;
  - We have a 'chain' of bottlenecks on one row (and at the symmetric row), followed by bottlenecks at *s* and *t*.
- Our methods stop working when there is a gap in the row of two consecutive bottlenecks;
- So what can we conclude from this?



Sink Inflow

[11]

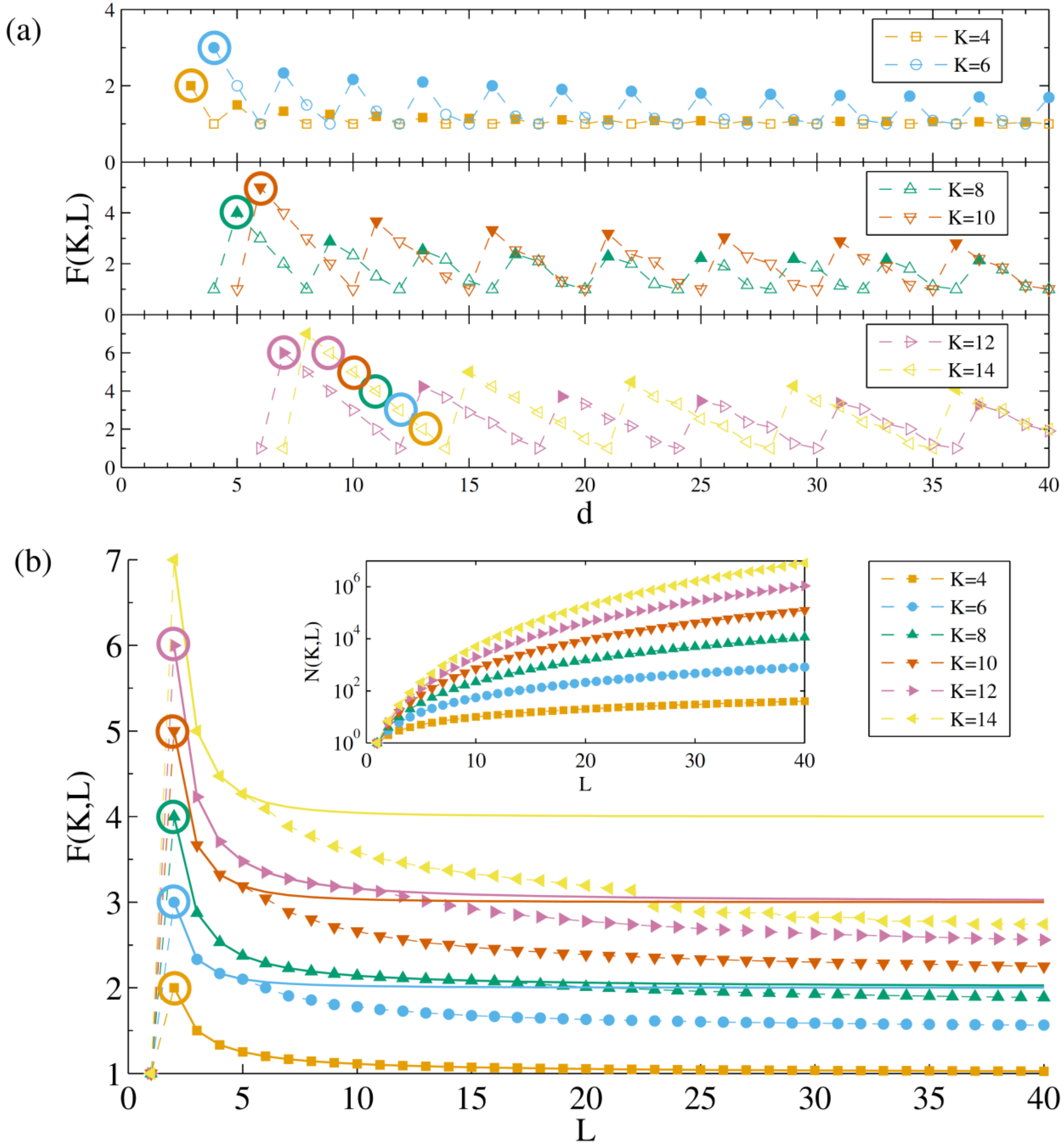


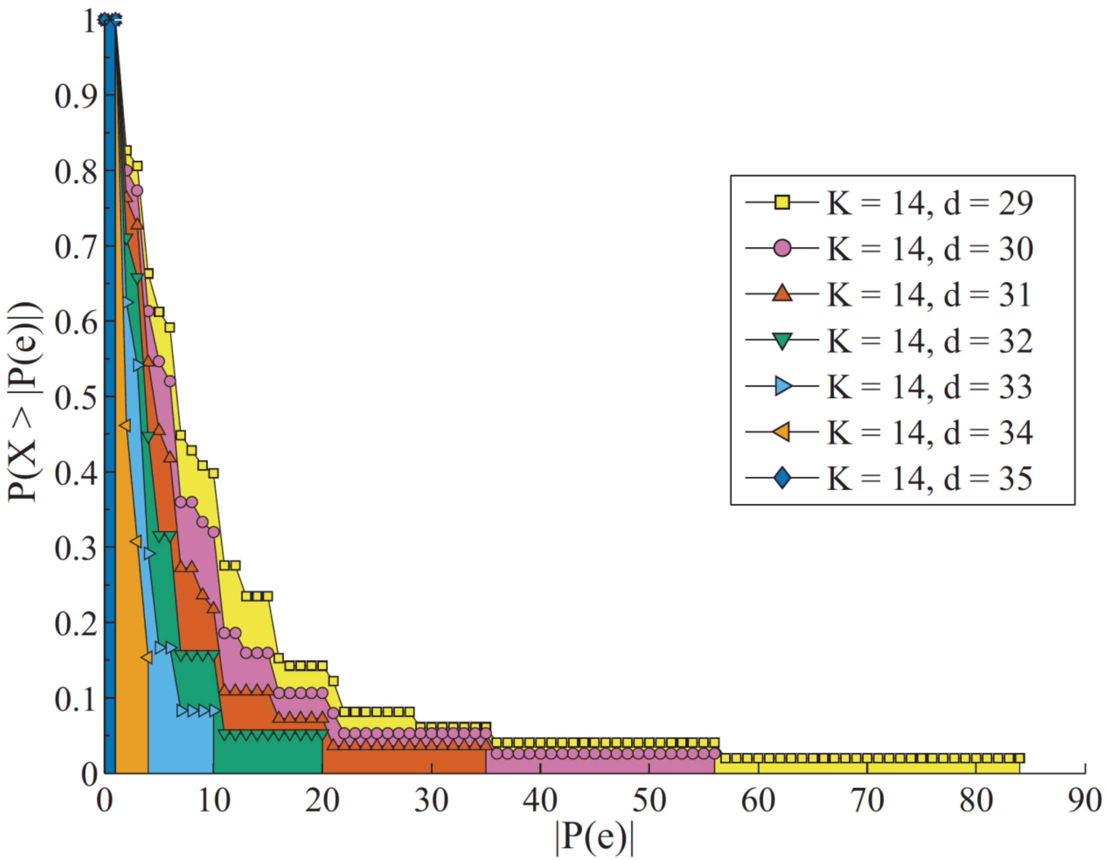
TABLE I. Sink inflow $F(K,L)$ as a function of s-t shortest path length L for node degree $4 \leq K \leq 14$, when all bottlenecks are edges of s or t. The assymptotic behaviour of the sink inflow is $\lim_{L\to\infty} F(K,L) = \lceil K/4 \rceil c$.

K	F(t)
4	$\frac{Lc}{L-1}$
6	$2c + \frac{2}{(L-1)L}c$
8	$2c + \frac{L+4}{L^2-1}c$
10	$3c + \frac{2(L^2+11L-2)}{L^4+2L^3-L^2-2L}c$
12	$3c + \frac{L^3+9L^2+51L+34}{L^4+5L^3+5L^2-5L-6}c$
14	$4c + \frac{2(L^4+18L^3+189L^2+88L-12)}{L^6+9L^5+25L^4+15L^3-26L^2-24L}c$

Fairness implies at least 50% loss of sink inflow compared to max-flow

FIG. 9. (Colour online). Plot of the sink inflow as a function of (a) the s-t distance d and (b) the s-t shortest path length L for even K and $4 \leq K \leq 14$. The solid curves in (b) are computed assuming that all bottlenecks are edges of the source s or the sink t.

Diversity of the number of paths crossing edges as d is varied



L=5

The pattern of intersections among these paths constraints the solution of the MMF flow, because the paths share the capacity of the edges they cross

FIG. 10. (Colour online). Complementary cumulative distribution of the number of s-t shortest paths passing through each edge for $L = 5$, as d is varied. We considered only edges that are crossed by at least one path. The diversity of the number of paths crossing each edge is illustrated by the different distributions. When $d = d^{min}(K, L)$, the distribution is broad and some edges are crossed by a large number of paths. The pattern of intersections among these paths constrains the solution of the MMF flow, because the paths share the capacity of the edges they cross. However, when $d = d^{min}(K, L) + K/2 - 1$, there is only one s-t shortest path. In this case, the MMF algorithm allocates the edge capacity to the path flow, because that path does not interact with any other.

Parameter Space Diagram

[12]

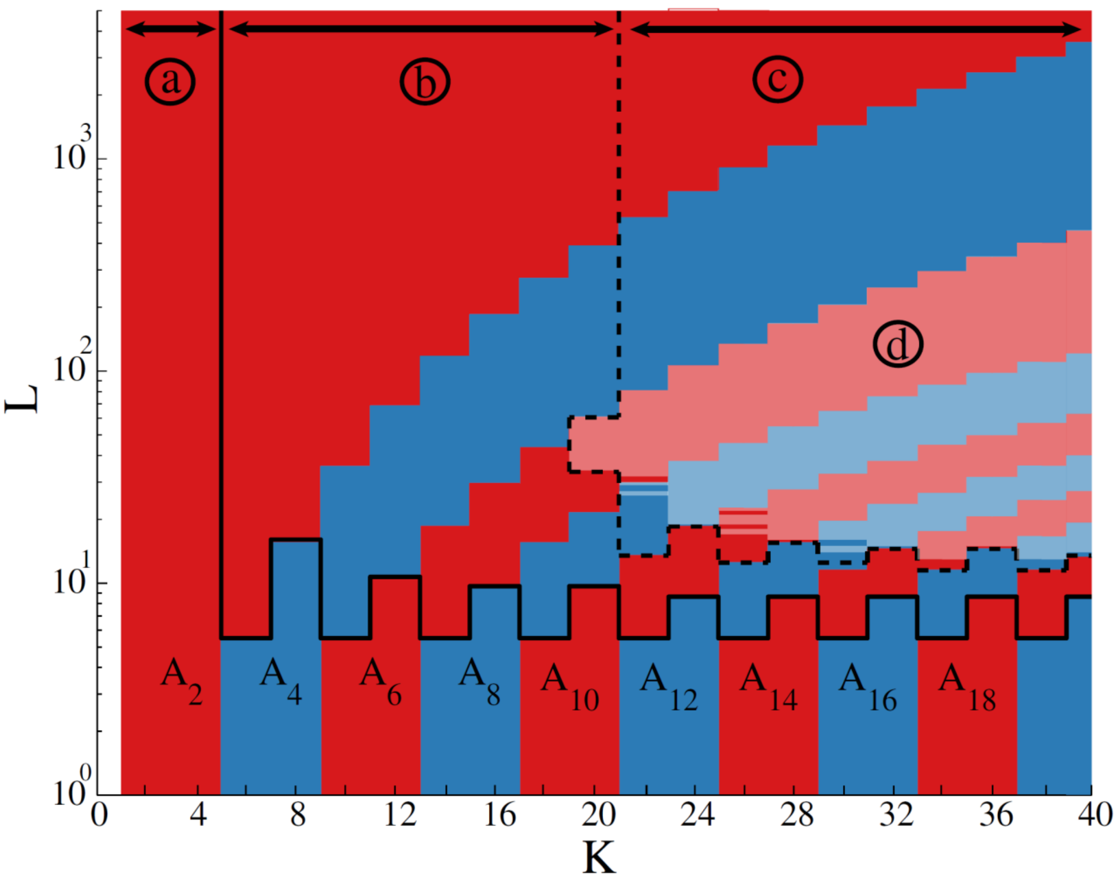
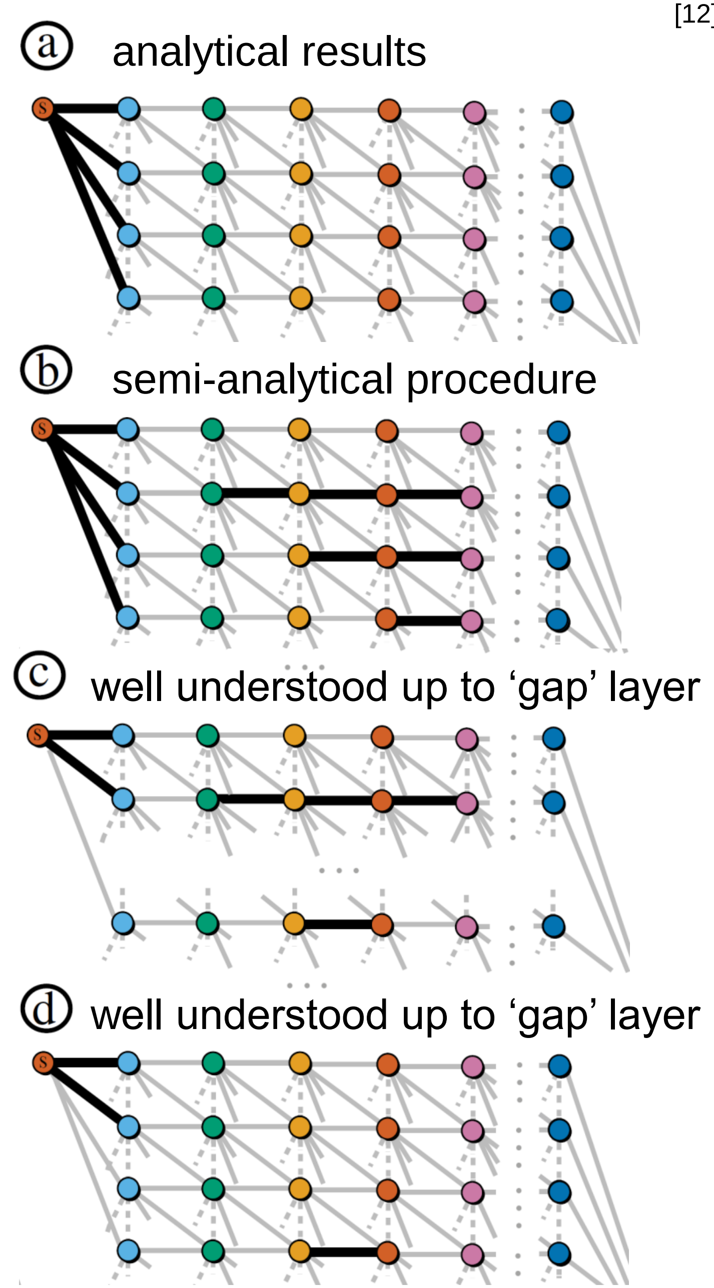


FIG. 11. We partition the parameter space (K, L) in areas $A_{(2i)}$, such that after the first i iterations the MMF algorithm finds $2i$ bottlenecks that are edges of the source s or sink t for all cells inside an $A_{(2i)}$ area. We use alternating red and blue coloured cells to distinguish neighbouring $A_{(2i)}$ areas. The parameter space is partitioned into four regions.



How do we generalise the path counting from the solid border to the dashed border?

- Instead of knowing the position of bottlenecks, we search for them;
- Once we find their location, we use path counting results as before;
- This works as long as there are no *gaps* in the row of bottlenecks.

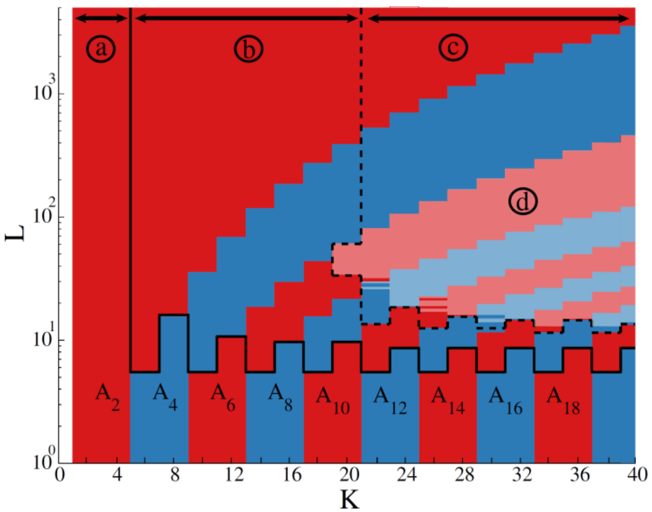
TWO CASES:

- **i)** all bottlenecks are edges of *s* or *t* until iteration *i-1*, but bottleneck at iteration *i* is not an edge of *s* or *t*;
- We show theoretically that

$$\phi^{(j)}(e) = c^{(j)}(e) / \left| P^{(j)}(e) \right|$$

is minimum on a horizontal edge. This simplifies the search.

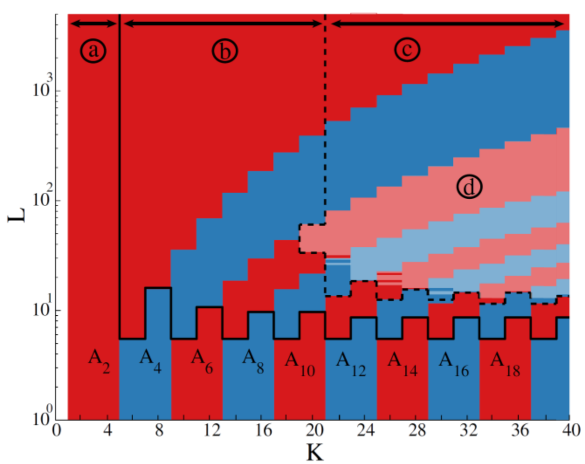
- **ii)** Case i) was valid up to iteration $j<i$.
  - Search over horizontal edges still valid, as long as there are no gaps in the rows of consecutive bottlenecks.
  - We get a chain of horizontal bottlenecks followed by a bottleneck at *s* or *t*.

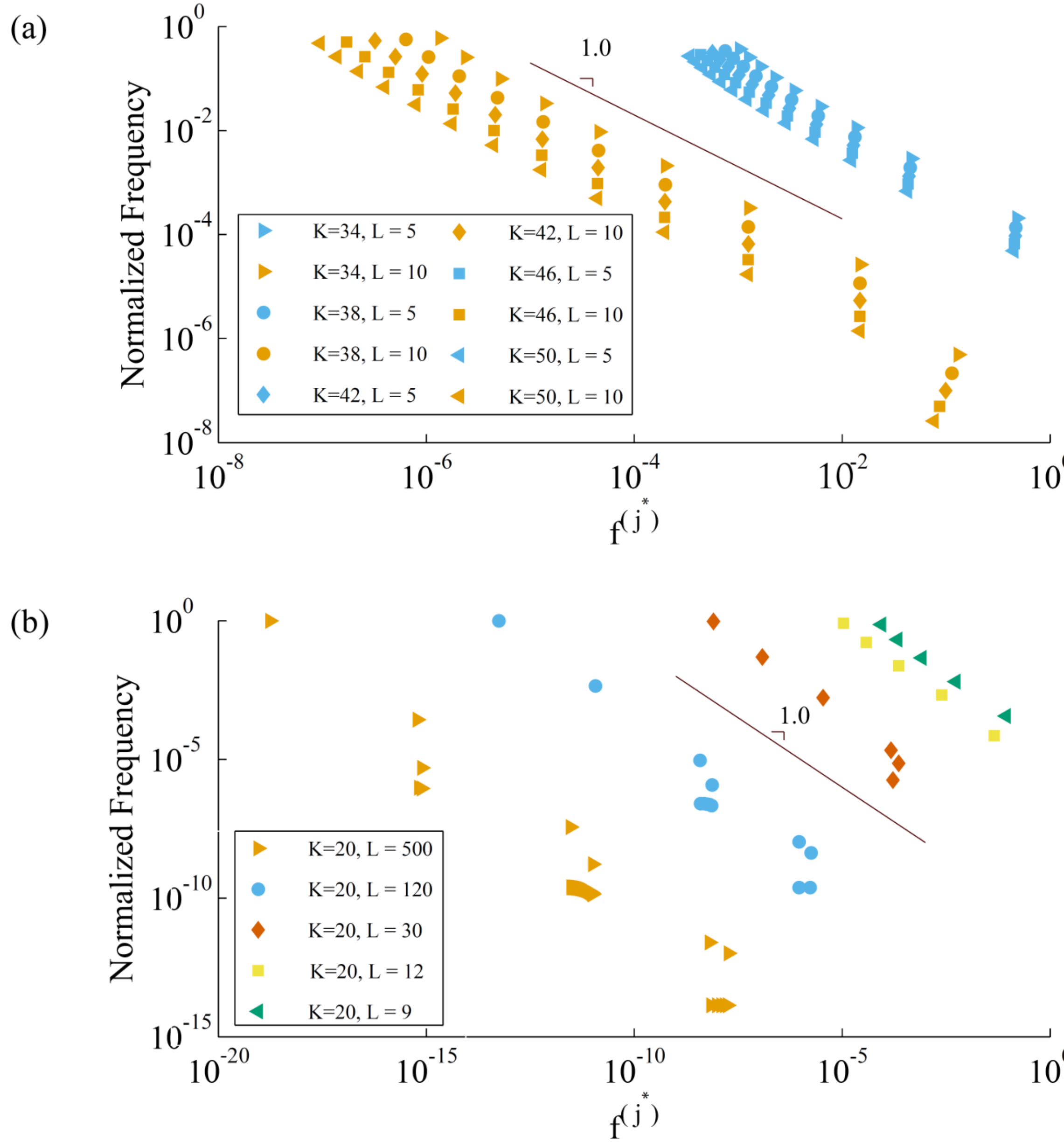


Can power-law flows be fair?

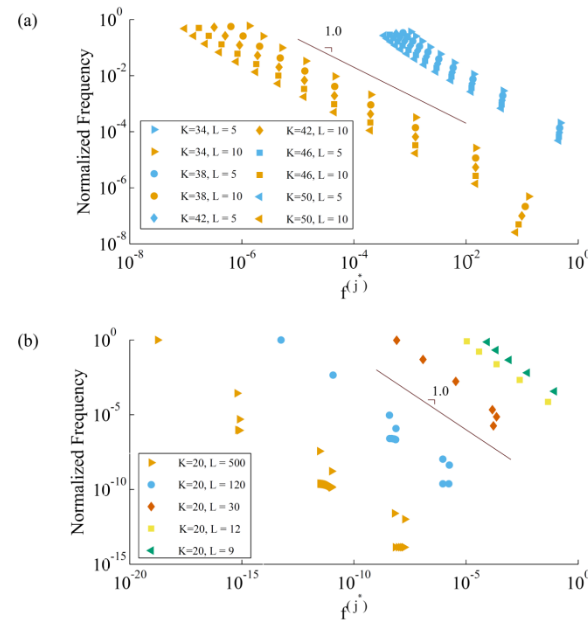
[13]

- Region defined by the solid line in the parameter diagram: histogram of path flows well described by power law with slope -1;
- Region defined by the dashed line: slight deviation from power law caused by 'chains' of bottlenecks.





Why do we get power-laws? Two factors



- a) the path flows are dominated by the path flow increments at the last iteration (when the paths are saturated):

$$f^{(i^*)} \simeq \Delta f^{(i^*)} = 1/N(K/2 - 2(i^* - 1), L).$$

- b) when *L* is large, the number of *s*-*t* shortest paths that are saturated at iteration *i* is of the order of magnitude of the number of *s*-*t* shortest paths in the residual network:

$$\eta^{(i^*)} \simeq N(K/2 - 2(i^* - 1), L).$$

$$f^{(i^*)} \simeq 1/\eta^{(i^*)}$$

Conclusions

- **Max-min fairness requires a big sacrifice in network throughput** (at least 50% in nearest-neighbour networks);
- Unexpected result: **power law allocations can be fair**!
- The location of bottlenecks is trivial for L small, but the pattern seem more and more elaborate as L increases for K large –how elaborate can it get as L is increased?
- We are currently finishing a paper on **proportionally fair allocations on the European gas pipeline network**, and I will be showing the results within the next few months.

Rui Carvalho, Lubos Buzna, Wolfram Just, Dirk Helbing, David Arrowsmith, *Phys. Rev. E* **85**, 046101 (2012)