Fair sharing of resources in a supply network with constraints

Rui Carvalho,[1,*] Lubos Buzna,[2,†] Wolfram Just,[1] Dirk Helbing,[3,4,5] and David K. Arrowsmith[1]

[1] *School of Mathematical Sciences, Queen Mary University of London, Mile End Road, London E1 4NS, U.K.*
[2] *University of Zilina, Univerzitna 8215/5, 01026 Zilina, Slovakia*
[3] *ETH Zurich, CLU E1, Clausiusstr. 50, 8092 Zurich, Switzerland*
[4] *Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, USA*
[5] *Collegium Budapest - Institute for Advanced Study, Szentháromság u. 2, 1014 Budapest, Hungary*

This paper investigates the effect of network topology on the fair allocation of network resources among a set of agents, an all-important issue for the efficiency of transportation networks all around us. We analyse a generic mechanism that distributes network capacity fairly among existing flow demands. The problem can be solved by semi-analytical methods on a nearest neighbour graph with one source and sink pair, when transport occurs over shortest paths. For this setup, we uncover a broad range of patterns of intersecting shortest paths as a function of the distance between the source and the sink. When the number of intersections is the maximum and the distance between the source and the sink is large, we find that a fair allocation implies a decrease of at least 50% from the maximum throughput. We also find that the histogram of the flow allocations assigned to the agents decays as a power-law with exponent -1. Our semi-analytical framework suggests possible explanations for the well-known reduction of the throughput in fair allocations. It also suggests that the combination of network topology and routing rules can lead to highly uneven (but fair) distributions of resources, a remark of caution to network designers.

PACS numbers: 05.10.-a, 89.20.Hh, 89.40.-a, 89.75.-k

I. INTRODUCTION

Transportation networks carry the flow of some commodity or entity using routing rules that aim to be efficient and avoid congestion [1, 2]. Recently, researchers have focused on the design of optimal transportation networks [3, 4, 5, 6], the characterisation of robustness and damage in transport networks [7, 8, 9], and routing strategies that minimize congestion [10, 11, 12, 13, 14]. Despite these efforts, little is known about the effect of network topology and various routing protocols on the competing interests of network users. Indeed, transport frequently involves a very large number of agents, and the operation of the network has to be managed in a smooth and fair way. Moreover, a socially efficient solution, *i.e.*, one that maximizes the sum of the utilities of individual agents, can be difficult to implement because it may be perceived as unfair to some of the agents [15, 16, 17, 18].

The fair division of network capacity among agents is a problem related to the procedure of dividing a cake fairly. Cake cutting, or the fair division of a divisible good among a set of agents, can be analysed using several mathematical interpretations of 'fairness' [19, 20, 21, 22]. When all N agents agree on the value of each portion, the problem becomes trivial and each agent receives a piece of size $1/N$. If the transport routes intersect along one edge only, the problem of dividing the edge capacity is akin to cutting a cake. However, in most situations of interest, distinct agents share transport routes in sophisticated ways, and the problem is then how to allocate fairly the available capacity on the intricate web of congested edges. Two prominent strategies to address this conundrum are the *Max-Min Fairness* [23] and the *Proportional Fairness* [24] allocations, and this paper focuses on the former. A set of flows is Max-Min Fair if no flow can be increased without simultaneously decreasing another flow that is already less than or equal to the former. To oversimplify, an allocation is Max-Min Fair if the wealthy can only get wealthier by making the poor even poorer.

The related problem of network congestion control aims at devising mechanisms for the allocation of resources in transport systems [24, 25]. There is a long line of work on addressing the challenge of congestion control with the Max-Min Fair allocation, mainly by researchers working on communication networks. Indeed, the fair allocation of capacity has been extensively studied in the context of flow control in IP packet switching networks, such as the Internet and ATM networks [23, 24, 25, 26, 27, 28]. Moreover, researchers have taken a wide range of approaches to the allocation of communication rates, encompassing both non pre-allocated [29, 30] and fixed [23, 32] paths. Despite these efforts, to the best of our knowledge, previous approaches have focused on the design of algorithms [23, 24, 27, 28, 29, 30, 32, 33, 34], or the computation of upper bounds of network throughput [18]. In contrast, our results offer a proof of concept that flow control can be analysed semi-analytically in large networks, at least regular ones, and reveal insights into the interplay of network topology, capacity, route selection and distance between a source and a sink, and how these elements in turn affect congestion in fair networks.

*Electronic address: rui@maths.qmul.ac.uk
†Electronic address: buzna@frdsa.fri.uniza.sk

The choices of network topology and routing place obvious constraints on the fair allocation of network resources. If the routing is such that paths do not intersect, each path is assigned the minimum capacity among all its edges and the throughput is the max-flow. In general, however, the choice of routing has an impact on the number of path intersections and thus on the reduction of network throughput in fair allocations. One possible choice of routing is to allow transport over all possible s-t paths. In such case, the Max-Min Fairness algorithm is typically solved by an iterative LP model [30, 35], and the fair allocation to one source-sink pair gives the same throughput as max-flow[30]. However, the model is computationally heavy when the number of s-t pairs is large [36]. Another possibility, followed in the vast majority of work on Max-Min Fairness [23, 25, 27, 32, 37, 38] and in this paper, is to fix the paths off-line before the algorithm is run. Fixing the paths is justified, although network managers are confronted with dynamic real-time traffic conditions, because it is frequently impractical to update the routing schemes according to changes in the distribution of traffic [35, 38]. Hence, network operators often design the best possible static network conditions for a set of estimated average or worst-case demands [38]. Here we choose to analyse routing over shortest paths, because this is a highly schematic way to model several transport processes in the real world, which is frequently used in the literature [39, 40]. Some real-world examples motivating its usage are packet transfer on the Internet [41], spatial distribution networks such as sewer systems [42], gas pipeline networks [7], commuter rail [42] and subway networks [43].

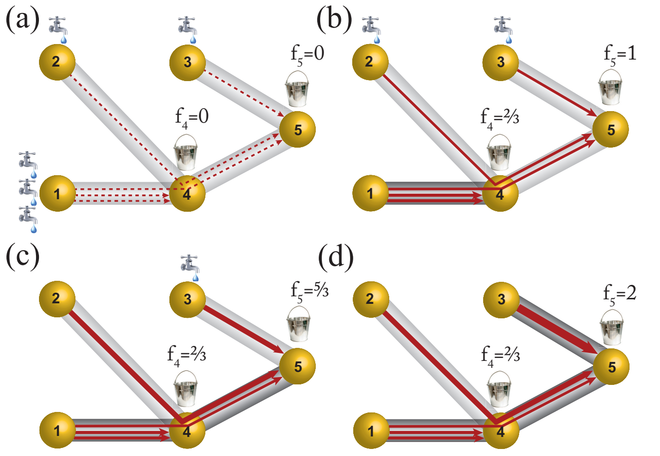

FIG. 1: (Colour online). Illustration of the Max-Min Fair (MMF) allocation on a small water pipeline network with three source nodes (identified by *water taps*), two sink nodes (identified by *buckets*), and pipelines with unit capacity. (a) Shows the initial configuration, before the application of the MMF algorithm. Five paths, connect the sources (taps) to the sinks (buckets). (b) At iteration one, the flow is increased uniformly on all taps, until the capacity is reached on pipeline $e_{1,4}$. The path flows that cross $e_{1,4}$ are frozen, the corresponding taps can no longer add to network flow and are thus removed from the figure, and we say that the pipeline is saturated (saturated pipelines are marked in dark gray colour). (c) and (d) The process is repeated for the unsaturated pipelines, until pipelines $e_{4,5}$ and $e_{3,5}$ are saturated at iterations two and three, respectively. The sink inflow f_4 and f_5 is displayed on each panel. The final allocation on panel (d) is Max-Min Fair, because no path flow can then be increased without decreasing another path flow already smaller.

II. FAIR DIVISION ON NETWORKS

A. How to share network capacity fairly in supply networks

The Max-Min Fair (*MMF*) allocation can be found with the *water-filling* algorithm [37]. The algorithm starts with all path flows from zero. Path flows are then increased equally for all paths, until one or more edges are saturated. We say that such edges are *bottlenecks*, and paths that pass through bottlenecks are also said to be saturated. The path flow of saturated paths is the Max-Min Fair flow allocation of the path. The algorithm is repeated until all paths are saturated, that is until each path passes through at least one bottleneck edge.

We give a visual representation of the water-filling algorithm in Fig. 1, where a water network connects *taps* (sources) to *buckets* (sinks) with individual paths on pipelines of unit capacity. Here, the algorithm proceeds by simultaneously opening the taps at the sources (see Fig. 1a). In the first iteration of the water-filling algorithm all paths get an equal flow of $1/3$ and the edge $e_{1,4}$ becomes saturated (see Fig. 1b). In the second iteration of the algorithm, the tap at node 1 cannot be opened any further, as the unit capacity of $e_{1,4}$ has been fully used. Nevertheless, the taps at nodes 2 and 3 can still be opened further, and the path flows of paths passing through $e_{2,4}$ and $e_{3,5}$ are increased to $2/3$ (see Fig. 1c). After the second iteration, the tap at node 2 cannot be opened further as the edge $e_{4,5}$ is saturated. On the third iteration, the tap at node 3 can be opened further, until the path flow on the path passing through $e_{3,5}$ is 1 (see Fig. 1d). The algorithm then stops and the allocation is Max-Min Fair because no path flow can be increased without decreasing another path flow less or equal to it. The analogy with cake-cutting can now be established: at each iteration of the algorithm, the available capacity on the bottleneck edge is shared equally among the paths that cross it and that have not been allocated a share of any edge capacity in previous iterations. In other words, allocating the network capacity can be seen as cutting a sequence of ‘cakes’ (the capacity of bottleneck edges), and the Max-Min Fair allocation establishes the sequence of ‘cakes’ to be shared equally among the paths that cross the bottleneck edges. We are now ready to formalise the iterative MMF algorithm [23, 25, 44]. To do this, we first need some mathematical definitions.

B. The Max-Min Fair algorithm explained formally

Let $G = (V, E, c, S, T)$ be a connected and undirected network with node-set V, edge-set E, an edge capacity function $c : E \to \mathbb{R}_0^+$, and a set of source and sink pairs $(s_i, t_i) \in S \times T$ with $i = 1, \cdots, M$. Each source and sink pair (s_i, t_i) is connected by a set of $s_i - t_i$ paths $P_i = \{p_{(i,1)}, \cdots, p_{(i,k_i)}\}$, where $P = \cup_{i=1}^{M} P_i$ is the set of all source to sink paths on the network. The undirected edge $e_{i,j} \in E$ connects nodes i and j, and we will omit the subscript indexes when we are not referring to a specific edge. All edges $e \in E(p_{(i,k)})$ of a $s_i - t_i$ path $p_{(i,k)}$ $(1 \leq i \leq M$ and $1 \leq k \leq k_i)$ transport the same *path flow* $f_{p_{(i,k)}} \in \mathbb{R}_0^+$ between s_i and t_i. This implies conservation of the flow at the nodes. Different paths can share an edge, even to perform transport in different directions (*e.g.*, during distinct time intervals) [50]. The set of paths passing through edge $e \in E$ is $P(e) = \{p_{(i,k)} \in P : e \in E(p_{(i,k)})\}$. The flow on edge $e \in E$ is the sum of the path flows on paths passing through the edge, and is given by $F(e) = \sum_{p_{(i,k)} \in P} \delta_{p_{(i,k)}}(e) f_{p_{(i,k)}}$, where $\delta_{p_{(i,k)}}(e)$ is 1 if $e \in E(p_{(i,k)})$ and is 0 otherwise. A flow is then *feasible* if $0 \leq F(e) \leq c(e)$ for all $e \in E$, where $c(e)$ is the capacity of edge e. Formally, a set $\{f\}$ of path flows on P is Max-Min Fair, if it is feasible and if for any other feasible set $\{f'\}$ of path flows on P, there exists a path $p_{(i,k)} \in P : f'_{p_{(i,k)}} > f_{p_{(i,k)}}$ implies that there exists another path $p_{(j,l)} \in P : f'_{p_{(j,l)}} < f_{p_{(j,l)}}$ and $f_{p_{(j,l)}} \leq f_{p_{(i,k)}}$.

We define $P^{(i)}$ to be the set of paths on the network at iteration i of the MMF algorithm, and $P^{(i)}(e)$ to be the subset of paths in $P^{(i)}$ that pass through edge e. Before we start the algorithm, we assign $P^{(1)} = P$ and $c^{(1)}(e) = c(e)$ for all $e \in E$, and a path flow $f^{(0)}_{p_{(j,k)}} = 0$ to each path $p_{(j,k)} \in P^{(1)}$. We then start the algorithm and initialize the iteration counter $i = 1$.

In the first step of the MMF algorithm, for each edge e with non-zero capacity that belongs to at least one path, we define the edge capacity divided equally among all paths that cross the edge at iteration i of the MMF algorithm as

$$\phi^{(i)}(e) = c^{(i)}(e) / \left| P^{(i)}(e) \right|, \tag{1}$$

for all $c^{(i)}(e) \neq 0$. We then find the minimum of $\phi^{(i)}(e)$, given by

$$\Delta f^{(i)} = \min_{e \in E, c^{(i)}(e) \neq 0} \phi^{(i)}(e). \tag{2}$$

In the second step of the MMF algorithm, we increase all path flows of paths in $P^{(i)}$ by $\Delta f^{(i)}$, such that

$$f^{(i)}_{p_{(j,k)}} = \begin{cases} f^{(i-1)}_{p_{(j,k)}} + \Delta f^{(i)} & \text{if } p_{(j,k)} \in P^{(i)} \\ f^{(i-1)}_{p_{(j,k)}} & \text{if } p_{(j,k)} \in P \backslash P^{(i)} \end{cases}. \tag{3}$$

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

$$c^{(i+1)}(e) = c^{(i)}(e) - \sum_{p_{(j,k)} \in P^{(i)}} \delta_{p_{(j,k)}}(e) \Delta f^{(i)}. \tag{4}$$

Note that all edges $e_B \in E_B^{(i)}$ will be saturated, that is each will have $c^{(i+1)}(e_B) = 0$ after this step. We also say that all paths that contain at least one edge $e_B \in E_B^{(i)}$ are saturated paths, to mean that their path flow will not be increased in subsequent iterations of the MMF algorithm. Next, we remove the set of saturated paths from P, that is

$$P^{(i+1)} = P^{(i)} \backslash \cup_{e_B \in E_B^{(i)}} P^{(i)}(e_B). \tag{5}$$

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

$$F(e) = \sum_{p_{(j,k)} \in P} \delta_{p_{(j,k)}}(e) f^{(i^*)}_{p_{(j,k)}}. \tag{6}$$

At each iteration, the MMF algorithm increases all path flows equally. Thus, the algorithm finds the bottleneck edges, which determine the maximum path flow possible for path flows that are the smallest. In other words, by finding the minimum of Eq. (1) at each iteration, we ensure that the smallest path flows are maximized.

III. MAX-MIN FAIR FLOWS IN NEAREST NEIGHBOUR NETWORKS

A. The network constraints

Recall from Eq. (1) that the capacity available on the bottleneck edges at each iteration of the MMF algorithm is shared by the paths passing through those edges. Thus, on a network with uniform capacity, the MMF flow allocation depends exclusively on the number of paths passing through each edge. To gain a better insight into the MMF allocation of flows, we choose to study K-nearest neighbour networks where each edge has constant capacity before the MMF algorithm is applied, that is,

$$c(e) = c^{(1)}(e) = c, \text{ for all } e \in E. \tag{7}$$

The advantage of studying these regular networks with constant capacity is that we can find simple closed form expressions for counting paths. In these regular graphs of

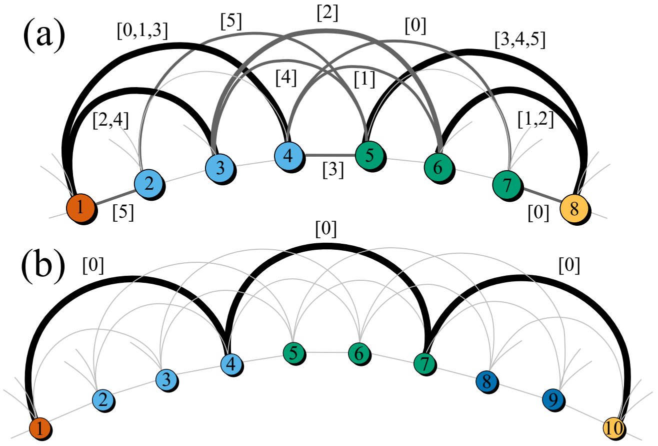

FIG. 2: (Colour online). Two layouts of 6-nearest neighbour networks. Nodes are numbered by an index from west to east, and the distance d that separates two nodes is the difference between their indexes. A source s and a sink t pair are placed (a) $d = 7$ and (b) $d = 9$ nodes apart, respectively. The 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.

even degree K, the nodes are placed on a one dimensional lattice with periodic boundary conditions (the lattice is a ring). Nodes are then connected to their nearest, next-nearest neighbours and so on, up to the constant range $K/2$. K-nearest neighbour graphs were recently the object of intense study as prototypes of networks where small amounts of rewiring can generate small-world networks [46, 47]. To simplify the analysis, we also introduce the constraint that transport only occurs along the set of shortest paths connecting the source and sink nodes.

B. Shortest paths in K-nearest neighbour networks with one (s,t) pair

We place a source and sink pair (s, t) on a K-nearest neighbour network with $|V|$ nodes. Nodes are numbered by an integer index, which indicates their relative position $d \leq \lfloor |V|/2 \rfloor$ from the source. Hence, there are two ways of measuring the distance between the source s and the sink t. We can either measure the s-t distance as the difference d between the indexes of s and t on the ring, or as the shortest path distance L between s and t. Since each node has K neighbours, there are $K/2$ nodes at the shortest path distance L from the source, each of them at a distinct distance d from the source. Fig. 2 is an illustration of the diversity of source to sink (s-t) shortest paths when $K = 6$ and $L = 3$, for $d = 7$ and $d = 9$.

We can now analyse the smallest distance d between s and t for a given shortest path length L. This distance

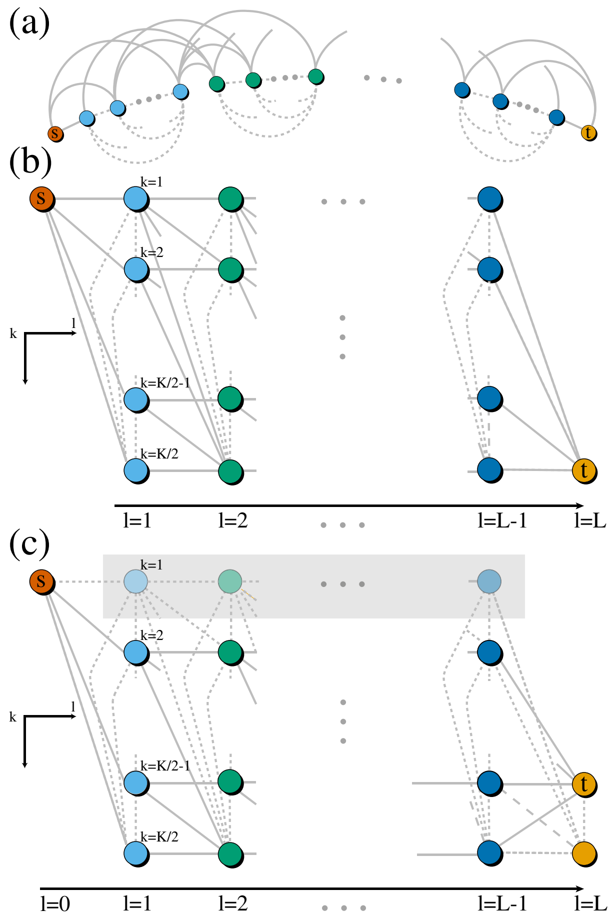

FIG. 3: (Colour online). (a) Layout of a K-nearest neighbour network where the s-t pair is separated by $d^{\min}(K, L)$ nodes on the ring for a given shortest path length L. Nodes are coloured according to their shortest path length from the source. Dashed edges connect two nodes at the same shortest path length from the source. (b) Rectangular layout of the network shown in (a), where all nodes except the source and sink are rearranged onto a rectangular lattice $K/2 \times (L-1)$ according to their distance from the source. Nodes on this rectangular lattice are identified by their (k, l) coordinates. (c) K-nearest neighbour network, where s-t are separated by $d = d^{\min}(K, L) + i$ nodes on the ring ($i = 1, \ldots, K/2-1$), has the same structure of shortest paths as a less dense $(K-2i)$-nearest neighbour network, where the two nodes are at the smaller distance $d^{\min}(K-2i, L)$. This panel illustrates the case $i = 1$, for which the gray area shows that s-t shortest paths no longer cross row $k = 1$.

is given by

$$d^{\min}(K, L) = (L-1) * K/2 + 1, \tag{8}$$

where

$$L = \lfloor (d-1)/(K/2) \rfloor + 1, \tag{9}$$

as is illustrated in Fig. 3a. To simplify the counting, we rearrange the network layout as in Fig. 3b, where each node (except for the source and sink) is identified by its row and column coordinates (k,l), $1 \leq k \leq K/2$ and $1 \leq l \leq L-1$. A s-t shortest path can be visualised on such *rectangular subgraphs* as a sequence of nodes $s, v_{(k_1,1)}, v_{(k_2,2)}, \cdots v_{(k_{L-1},L-1)}, t$, where the nodes $v_{(k_i,i)} \in V$ are such that there is an edge between $v_{(k_i,i)}$ and $v_{(k_{i+1},i+1)}$, and $k_1 \leq k_2 \leq \cdots \leq k_{L-1}$ is a non-decreasing sequence of rows in the rectangular subgraph. In other words, shortest paths contain only horizontal (east) and diagonal (south east) edges. The number of s-t shortest paths when the pair is at the distance $d^{\min}(K,L)$ is the number of ways we can distribute $L-1$ identical edges to $K/2$ distinguishable rows. Such allocation can be made in

$$N(K/2,L) = \left(\!\!\binom{K/2}{L-1}\!\!\right) = \binom{K/2+L-2}{L-1} \qquad (10)$$

different ways, where $\left(\!\binom{n}{m}\!\right) = \binom{n+m-1}{m}$ is the number of ways that m indistinguishable balls can be assigned to n distinguishable urns (see the twelvefold way of combinatorics in [48, pp. 31-38]). Equation (10) implies that the number of s-t shortest paths between a pair of nodes at distance $d^{\min}(K,L)$ is given by an entry in Pascal's triangle as visualised in Fig. 4.

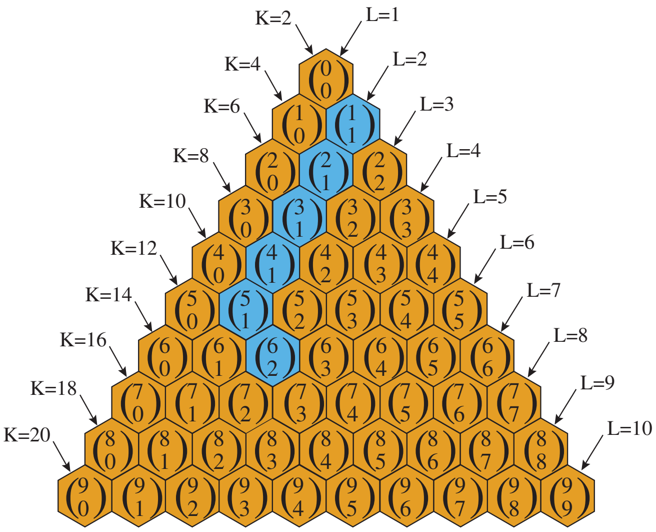

FIG. 4: (Colour online). The entries in Pascal's triangle are the number of s-t shortest paths between a source and sink nodes at distance $d^{\min}(K,L)$ from each other on a K-nearest neighbour network, for a chosen shortest path length L between the two nodes. Entries of Pascal's triangle coloured in blue illustrate that $\binom{6}{2}$ is the sum of the number of s-t shortest paths over the *hockey stick*.

We are now able to determine the number of s-t shortest paths when the pair is at distance $d^{\min}(K,L)$. Nevertheless, this is not the general case. To illustrate the point, the number of s-t shortest paths in nearest neighbour lattices is represented in Fig. 5 as a sequence $\{n(K,d)\}_{d=1,\dots,\lfloor|V|/2\rfloor}$, which is in itself a batch of monotonically decreasing sequences. Each integer at a position d of the sequence is the number of s-t shortest paths when

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

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

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

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

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

FIG. 5: (Colour online). Batch of sequences denoting the number of shortest paths for a single (s,t) pair positioned on a K-nearest neighbour network at distance d from each other. K is the node degree and L is the length of a shortest path between s and t. The position of the number of shortest paths for each K value, is the s-t distance d, *e.g.* there are 4 s-t shortest paths for $K=8$ at a s-t distance $d=5$ and shortest path distance $L=2$ (see node $\boxed{4}$). The boxes in blue and gray show that the structure of s-t shortest paths for given K and d can be recovered as a function of less dense K-nearest neighbour networks, where the s-t pair is closer (see also Fig. 4).

the source and sink nodes are at distance d on the ring. Each of the sub-sequences is characterised by a (K,L) pair, but so far we have only determined the first element of such sub-sequences, which is $N(K/2, d^{\min})$. These first elements of each sub-sequence follow a *hockey stick* pattern in Pascal's triangle, which is illustrated in Figs. 4 and 5 for $K=10$ and $L=3$. Such a pattern tells us that the first element of each subsequence can be found from the sum of previous first elements for a smaller shortest path length $L-1$, that is $N(K/2,L) = \sum_{i=1}^{K/2} N(i, L-1)$.

To understand the structure of the sub-sequences, observe that the set of shortest paths on a K-nearest neighbour network when s and t are placed $d^{\min}(K,L)+i$ nodes apart is equivalent to the set of $s-t$ shortest paths on a less dense $(K-2i)$-nearest neighbour network where s and t are placed closer to each other, at the distance $d^{\min}(K-2i,L)$. For example, if $K=8$ and $L=4$, and the s-t nodes are at distance $d^{\min}(8,4)=13$, increasing the distance to $d=14$ does not change the shortest path length L, but the set of shortest paths between the two nodes is the same as when the s-t pair are placed on a 6-nearest neighbour network at distance $d^{\min}(6,4)=10$. This effect is illustrated in Fig. 5 (see the nodes in the gray boxes) and in Figs. 3b and c for $i = 1$. As can be visualised from Fig. 3c, increasing the lattice position of the sink by one has the consequence that the connections between the new sink node and nodes $(k = 1, l)$ for $1 \le l \le L-1$ (the first row in the rectangular subgraph of Fig. 3c) no longer belong to any s-t shortest paths. The argument can be easily generalized to any i.

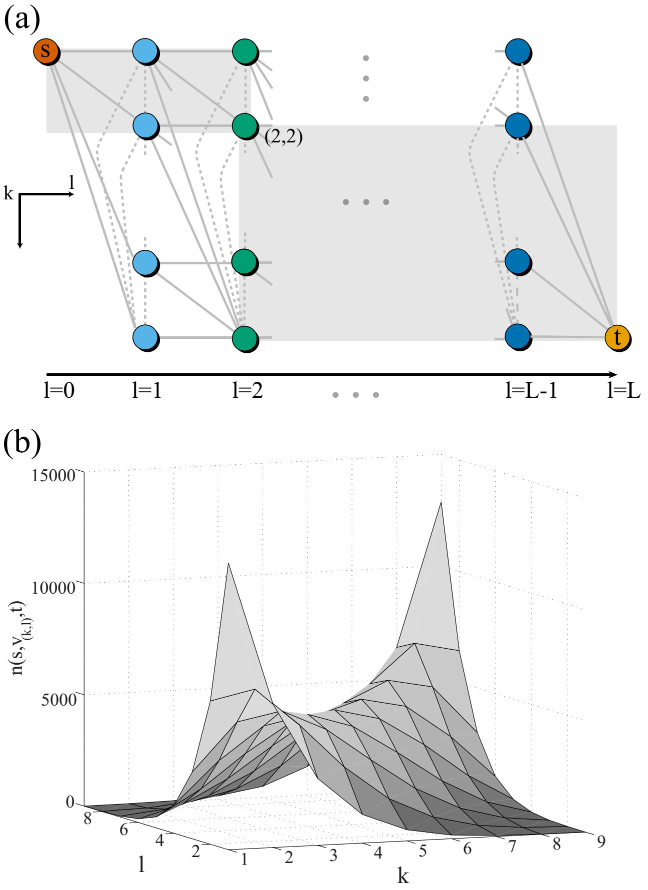

FIG. 6: (Colour online). (a) The number $n(s, v_{(2,2)}, t)$ of s-t shortest paths that cross node $v_{(2,2)}$ is determined from the product rule as the product of the number of s-t shortest paths in the two rectangular subgraphs highlighted in gray. (b) The number $n(s, v_{(k,l)}, t)$ of s-t shortest paths between the source s and the sink t through the node (k, l) (where $L = 10$ and $K = 12$) has a saddle shape, with maxima at $(k = 1, l = 1)$ and $(k = K/2, l = L-1)$.

The number of s-t shortest paths passing through a set of given nodes is given by the product rule. The number of shortest paths connecting nodes s and t and passing through the node $v_{(k,l)}$ in the rectangular subgraph is given by:

$$
\begin{aligned}
n(s, v_{(k,l)}, t) &= N(k, l) N(K/2 - k + 1, L - l) \\
&= \left(\!\!\binom{k}{l-1}\!\!\right) \left(\!\!\binom{K/2 - k + 1}{L - l - 1}\!\!\right),
\end{aligned} \tag{11}
$$

where $1 \le k \le K/2$ and $1 \le l \le L-1$. Figure 6a shows schematically how Eq. (11) is derived, and Fig. 6b is a plot of $n(s, v_{(k,l)}, t)$ that illustrates that the concentration of shortest paths is higher along the diagonal of the rectangular layout. The number of shortest paths connecting nodes $s = (k_0, l_0) = (1, 0)$ and $t = (k_{n+1}, l_{n+1}) = (K/2, L)$ and passing through nodes $v_{(k_1,l_1)}, \dots, v_{(k_n,l_n)}$ is given by:

$$
\begin{aligned}
&n(s, v_{(k_1,l_1)}, \dots, v_{(k_n,l_n)}, t) \\
&= \prod_{i=0}^{n} N(k_{i+1} - k_i + 1, l_{i+1} - l_i) \\
&= \prod_{i=0}^{n} \left(\!\!\binom{k_{i+1} - k_i + 1}{l_{i+1} - l_i - 1}\!\!\right),
\end{aligned} \tag{12}
$$

where $1 \le k_1 \le \cdots \le k_n \le K/2$ and $1 \le l_1 < l_2 < \cdots < l_n \le L-1$. From Eq. (11), the number of shortest paths containing edge $e_{s,(k,1)}$ is defined by $n(e_{s,v_{(k,1)}}) = n(s, v_{(k,1)}, t)$ and is given by

$$
\begin{aligned}
n(e_{s,v_{(k,1)}}) &= N(K/2 - k + 1, L - 1) \\
&= \left(\!\!\binom{K/2 - k + 1}{L - 2}\!\!\right).
\end{aligned} \tag{13}
$$

Similarly, from Eq. (12) the number of s-t shortest paths passing through a set of nodes, we derive the number of s-t shortest paths containing edges $e_{s,(k_1,1)}$ and $e_{(k_2,L-1),t}$ that is defined as $n(e_{s,v_{(k_1,1)}}, e_{v_{(k_2,L-1)},t}) = n(s, v_{(k,1)}, v_{(k_2,L-1)}, t)$, to be

$$
\begin{aligned}
n(e_{s,v_{(k_1,1)}}, e_{v_{(k_2,L-1)},t}) &= N(k_2 - k_1 + 1, L - 2) \\
&= \left(\!\!\binom{k_2 - k_1 + 1}{L - 3}\!\!\right).
\end{aligned} \tag{14}
$$

Equations (11) to (14) show that the number of s-t shortest paths passing through a node or a set of nodes can be written as a product of binomials. Each of these binomials is the number of s-t shortest paths when the s-t pair is located at the distance $d^{\min}$ on networks with specific K and L values and is given by Eq. (10).

To summarise, the number of s-t shortest paths depends on d and is given by a batch of sequences (see Fig. 5), each having $K/2$ elements. If the s-t pair is at the smallest distance $d^{\min}(K, L)$ given by Eq. (8) for a given shortest path length L, then the number of s-t shortest paths is maximal for that L. Increasing the distance between the s-t pair to $d^{\min}(K, L) + i$ (for $1 \le i \le K/2 - 1$) creates the same structure of s-t shortest paths as placing the s-t pair on a $(K - 2i)$-nearest neighbour network at the distance $d^{\min}(K - 2i, L)$, which is the smallest distance at which the two nodes can be located on a $(K - 2i)$-nearest neighbour network so that the shortest path length between them is L. This implies that the problem of determining sink inflow on a K-nearest neighbour network, as the distance between the s-t pair is varied, is reduced to the problem of calculating sink inflow at specific distances $d^{\min}(K,L)$ given by Eq. (8) as K and L are varied. Hence, we concentrate on determining the sink inflow rigorously for s-t pairs placed at distances $\{d^{\min}(K,L)\}|_{L=1,\dots,\infty}$ on K-nearest neighbour networks. From now on we will refer to a s-t pair at the shortest path length L to mean that the s-t pair is at the distance $d^{\min}(K,L)$, where $d^{\min}(K,L)$ is related to L in Eq. (8). As we will see in Sec. III C, the calculation of the sink inflow and of the path flows requires the number of s-t shortest paths in various situations, as given by Eqs. (11) to (14).

C. Exact calculation of flows with the path counting methods in K-nearest neighbour networks with one (s,t) pair

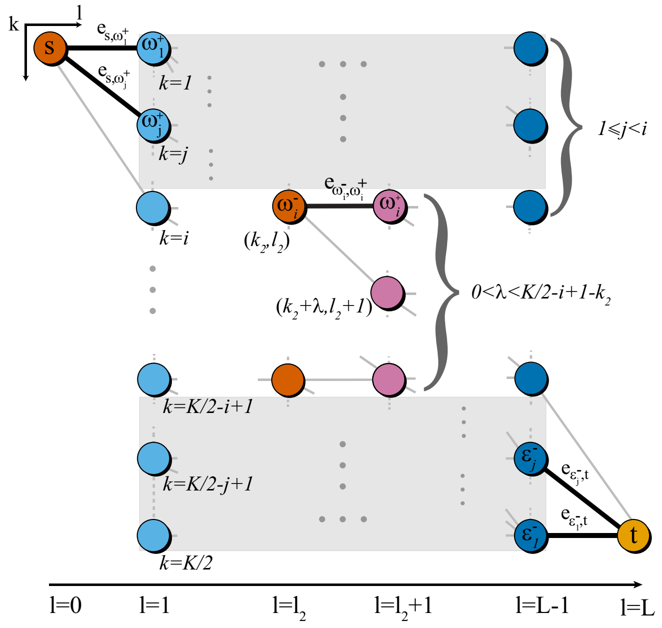

FIG. 7: (Colour online). Rectangular layout of the subgraph between the source (s) and the sink (t) nodes at iteration i of the MMF algorithm. The bottlenecks found at iterations $1 \le j < i$ are edges of s or t and are represented as thick black edges. The pair of bottlenecks found at iteration i of the MMF algorithm, are identified by the notation $e_{\omega_i^-,\omega_i^+}$ (the western bottleneck) and $e_{\varepsilon_i^-,\varepsilon_i^+}$ (the eastern bottleneck, not represented). When the bottlenecks are edges of the source or sink, $\omega_i^- = s$ and $\varepsilon_i^+ = t$, respectively.

Our goal is now to get analytical insights into the MMF allocation on K-nearest neighbour networks with one s-t pair. In particular, we wish to determine all path flows as a function of the distance L between the source and sink nodes. The knowledge of all path flows also allows us to determine the incoming flow at the sink node as a function of L. The sink inflow can be determined as the sum of the path flows over the $K/2$ edges of the sink that contain shortest paths,

$$F(K,L) = \sum_{i=1}^{K/2} \sum_{p(j,k)\in P} \delta_{p(j,k)}(e_{v_i,t}) f^{(i^*)}_{p(j,k)}, \tag{15}$$

where $v_i = (K/2 - i + 1, L-1)$, $i = 1,\dots,K/2$ are the $K/2$ neighbours of the sink that belong to s-t shortest paths.

Such a problem becomes more interesting as we realise that the contribution to the sink inflow from each iteration of the MMF algorithm can be found by path counting methods alone. Indeed, the ratio of capacity to the number of paths for an edge on Eq. (1), is a function of both the edge capacity on the residual network and the number of unsaturated paths passing through the edge. In turn, the residual edge capacity depends exclusively on the number of paths that pass through the edge and are saturated at the previous iterations of the MMF algorithm. This implies that the calculation of sink inflow can be done rigorously from path counting alone, once we have found the pattern of the location of bottleneck edges at each iteration of the MMF algorithm.

We start by assuming that all bottlenecks are found on edges of the source and sink nodes (see Fig. 7). When $K/2$ is odd, this means that the row $\lceil K/4 \rceil$ in the rectangular layout contains two bottleneck edges instead of one, and we observe a total of $2\lceil K/4\rceil$ bottlenecks. The network is symmetric (*i.e.*, the source and sink can be interchanged) and thus bottlenecks are positioned symmetrically in pairs. We introduce the notation $e_{\omega_i^-,\omega_i^+}$ and $e_{\varepsilon_i^-,\varepsilon_i^+}$, to refer to a symmetric pair of bottleneck edges found at iteration i. In general, we will refer to a bottleneck closer to the source $e_{\omega_i^-,\omega_i^+}$ as a *western* bottleneck and a bottleneck closer to the sink $e_{\varepsilon_i^-,\varepsilon_i^+}$ as an *eastern* bottleneck. Thus, at the i-th iteration of the MMF algorithm, the western bottleneck e_{s,ω_i^+} connects the source and node

$$\omega_i^+ = v_{(i,1)}, \tag{16}$$

and a symmetric eastern bottleneck $e_{\varepsilon_i^-,t}$ will be located between the sink and node

$$\varepsilon_i^- = v_{(K/2-i+1,L-1)}. \tag{17}$$

Since the bottleneck edges found at iteration i are located on e_{s,ω_i^+} and the symmetric edge $e_{\varepsilon_i^-,t}$, the path flow increment (2) at iteration i of the MMF algorithm is given by

$$\begin{aligned}\Delta f^{(i)} &= \min_{e\in E} \phi^{(i)}(e) = \phi^{(i)}(e_{s,\omega_i^+}) \\ &= c^{(i)}(e_{s,\omega_i^+}) / \left|P^{(i)}(e_{s,\omega_i^+})\right|,\end{aligned} \tag{18}$$

for all $c^{(i)}(e_{s,\omega_i^+}) \neq 0$. While the denominator of (18) depends only on the number of unsaturated s-t shortest paths passing through e_{s,ω_i^+} at iteration i, the residual capacity $c^{(i)}(e_{s,\omega_i^+})$ on the numerator is found recursively as a function of the path flow increment added to unsaturated paths at iterations $j < i$ of the MMF algorithm. This recursive calculation depends on the number $\left|P^{(j)}(e_{s,\omega_i^+})\right|$ of s-t shortest paths passing through edge e_{s,ω_i^+} that have not been saturated until iteration $j < i$. The path flow of each of these paths is increased by $\Delta f^{(j)}$ at iteration j, and the capacity of all edges that these paths cross is reduced by $\Delta f^{(j)}$.

Paths passing through e_{s,ω_i^+} that have been saturated in the previous iteration $j < i$ of the MMF algorithm, have the property that each cross e_{s,ω_i^+} as well as one of the bottleneck edges $e_{\varepsilon_q^-,t}$ found at a previous iteration $q = 1, \dots, j-1$. For each $q = 1, \dots, j-1$, there are $n(e_{s,\omega_i^+}, e_{\varepsilon_q^-,t})$ such paths given by Eq. (14). Hence, $\left|P^{(j)}(e_{s,\omega_i^+})\right|$, which is the number of unsaturated paths at iteration j crossing e_{s,ω_i^+}, can be determined from Eqs. (13) and (14) as

$$\begin{aligned}\left|P^{(j)}(e_{s,\omega_i^+})\right| &= n(e_{s,\omega_i^+}) - \sum_{q=1}^{j-1} n(e_{s,\omega_i^+}, e_{\varepsilon_q^-,t}) \\ &= n(e_{s,\omega^+_{(i+j-1)}}) \\ &= N(K/2 - (i+j) + 2, L-1), \end{aligned} \tag{19}$$

where $1 \leq i + j - 1 \leq K/2 - 1$, and from the paths counting formulas (10), (13) and (14), and the location (16) and (17) of the western and eastern bottlenecks,

$$n(e_{s,\omega_i^+}) = N(K/2 - i + 1, L - 1) \tag{20}$$

and

$$n(e_{s,\omega_i^+}, e_{\varepsilon_q^-,t}) = N(q - i + 1, L - 2). \tag{21}$$

To understand Eq. (19), observe that the s-t shortest paths crossing both e_{s,ω_i^+} and edges in row $K/2 - j + 1$ of the rectangular subgraph (see Fig. 3) also cross bottleneck $e_{\varepsilon_i^-,t}$, and these paths are saturated at iteration $j < i$ of the MMF algorithm. By symmetry, the s-t shortest paths crossing $e_{\varepsilon_i^-,t}$ and edges in row j also cross the bottleneck e_{s,ω_j^+}, *i.e.* they are also saturated at iteration j. The effect of this pattern of saturated s-t shortest paths is that, for each iteration $j < i$, the MMF algorithm saturates two bottleneck edges, e_{s,ω_j^+} and $e_{\varepsilon_j^-,t}$, as well as all the paths that cross rows j and $K/2 - j + 1$ of the rectangular subgraph. Thus, the effect of j iterations of the MMF algorithm is to remove $2j$ rows (j rows from the top and j rows from the bottom) of the original $K/2$-by-L rectangular subgraph, creating an equivalent $(K/2 - 2j)$-by-L rectangular subgraph. Note that the equivalent rectangular subgraph is only useful for path counting purposes, since the capacity of its edges will have been reduced after j steps of the MMF algorithm.

Each of the $\left|P^{(j)}(e_{s,\omega_i^+})\right|$ unsaturated paths has its path flow incremented by $\Delta f^{(j)}$ at iteration $j < i$, according to Eq. (18). So, each path in $P^{(j)}(e_{s,\omega_i^+})$ reduces the available capacity $c^{(j)}(e_{s,\omega_i^+})$ by $\Delta f^{(j)}$ according to Eq. (4). Since all edges have initial capacity c (see Eq. (7)), the path flow increment at iteration i of the MMF algorithm, given by Eq. (18), can be written as a function of Eq. (19):

$$\begin{aligned}\Delta f^{(i)} &= \frac{c - \sum_{q=0}^{i-1} \left|P^{(q)}(e_{s,\omega_i^+})\right| \Delta f^{(q)}}{\left|P^{(i)}(e_{s,\omega_i^+})\right|} \\ &= \frac{c - \sum_{q=0}^{i-1} n(e_{s,\omega^+_{(i+q-1)}}) \Delta f^{(q)}}{n(e_{s,\omega^+_{(2i-1)}})} \\ &= \frac{c - \sum_{q=0}^{i-1} N(K/2 - (i+q) + 2, L-1) \Delta f^{(q)}}{N(K/2 - 2(i-1), L-1)}, \end{aligned} \tag{22}$$

where $\Delta f^{(0)} = 0$. Equation (22) says that the increment to path flows $\Delta f^{(i)}$ at iteration i on a $K/2$-by-L rectangular subgraph can be written as a recursive function of the number of s-t shortest paths on a sequence of nearest-neighbour induced subgraphs with smaller node degree than the original rectangular subgraph.

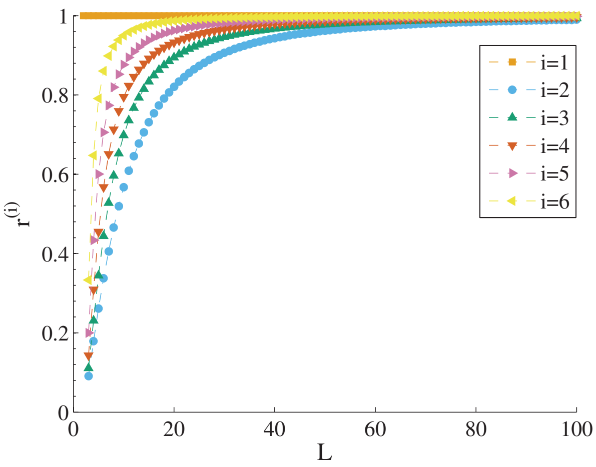

FIG. 8: (Colour online). Plot of the ratio $r^{(i)}$ between the path flow increment at iteration i and the path flow of paths that are saturated at iteration i, as a function of L for $K = 24$ as the iterations i increase. The path flow is dominated by the path flow increment at the last iteration when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

To gain insight into the Max-Min Fair path flows $f^{(i^*)}$ when all bottlenecks are edges of s or t, we plot in Fig. 8 the ratio

$$r^{(i)} = \frac{\Delta f^{(i)}}{f^{(i^*)}} = \frac{\Delta f^{(i)}}{\sum_{j=1}^{i} \Delta f^{(j)}}, \tag{23}$$

for all paths that are saturated at iteration i. These paths that are saturated at iteration i are all the paths that pass through a bottleneck found at iteration i, or equivalently, all paths in $P^{(i)}(e_B)$. Thus, the ratio $r^{(i)}$ represents the relative contribution of the path flow increment at iteration i to the path flow assigned to paths that are saturated at iteration i. The observed $r^{(i)} \to 1$ tells us that the path flows allocated by the MMF algorithm are dominated by the flow increment $\Delta f^{(i)}$ at the last iteration. A histogram of the path flows is thus well approximated by a plot of the number $\eta^{(i)} = N(K/2-2(i-1), L) - N(K/2-2i, L)$ of paths saturated at iteration i versus $\Delta f^{(i)} \propto 1/N(K/2-2(i-1), L-1)$, and for L large $\Delta f^{(i)} \sim 1/\eta^{(i)}$. In other words, when the bottlenecks are edges of s or t and L is large, the histogram of path flows is a discrete set of points on a line with slope -1.

Combining the number of s-t shortest paths and the path flow increment according to Eqs. (10) and (22), the MMF flow at the sink is then

$$F(K,L) = \sum_{j=1}^{\lceil K/4 \rceil} N(K/2-2(j-1), L)\Delta f^{(j)} \qquad (24)$$

Combining the number of s-t shortest paths Eqs. (10) and the number (20) of paths passing through the western bottleneck, Eq. (24) can be simplified for large L to

$$F(K,L) \sim \sum_{j=1}^{\lceil K/4 \rceil} \frac{\binom{K/2-2j+L}{L-1}}{\binom{K/2-2j+L-1}{L-2}} c. \qquad (25)$$

Using the asymptotic expansion of the binomial coefficient, Eq. (25) can be simplified further to

$$F(K,L) \sim c \sum_{j=1}^{\lceil K/4 \rceil} \frac{(L-1)^{K/2-2j+1}}{(L-2)^{K/2-2j+1}}, \qquad (26)$$

and thus

$$\lim_{L\to\infty} F(K,L) = \lceil K/4 \rceil c. \qquad (27)$$

This asymptotic behaviour can be easily observed in Table I for small K.

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

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

So far, we have made the assumption that, at each iteration of the MMF algorithm, the algorithm saturates one edge of the source and another of the sink, up to a total of $2\lceil K/4 \rceil$ bottleneck edges. This assumption allows us to compute exactly the path flow increments (see Eq. (22)) and the sink inflow (see Eq. (24)). Knowing the pattern of the location of bottleneck edges gives us a clear advantage over solving the MMF allocation with the generic implementation of the algorithm described in Sec. II B. Indeed, the generic implementation of the MMF algorithm has to keep track of all paths and path flows at each iteration. This process consumes increasing resources as K and L increase, owing to the number of s-t shortest paths growing polynomially or faster with either K or L. Thus, knowledge of the location of the bottleneck edges is particularly important for some regions of the parameter space, where the number of s-t shortest paths is large, and it is impractical to implement the MMF algorithm as described in Sec. II B.

Since our assumption has been that the bottlenecks are edges of s or t, it is then natural to investigate the regions of the parameter space (K, L) where this assumption holds. To answer this question for each (K, L) pair, we compute the smallest iteration of the MMF algorithm such that the bottleneck edges are not edges of s or t. Since the bottleneck edges found at iteration i are located on rows i and $K/2-i+1$, to find these bottlenecks we need to search for the minimum of Eq. (1) over all edges that link to a node in row i. This search can be greatly simplified when all $2(i-1)$ bottlenecks found up to iteration $i-1$ are edges of s or t. Let us assume that the western bottleneck $e_{\omega_i^-,\omega_i^+}$, which is not an edge of s or t, connects the two nodes

$$\begin{cases} \omega_i^- = v_{(k_2,l_2)} \\ \omega_i^+ = v_{(k_2+\lambda,l_2+1)}, \end{cases} \qquad (28)$$

where $i \le k_2 \le \lceil K/4 \rceil$, $0 \le \lambda \le K/2-i+1-k_2$ is an integer, and the eastern bottleneck is defined in a similar way. When $\lambda = 0$, we say that the bottleneck is *horizontal* because it links two nodes that share the same row and are placed on adjacent columns on the rectangular subgraph in Fig. 3. When $\lambda \neq 0$, we say that the bottleneck is *diagonal*, because it connects two nodes that belong to adjacent columns, but different rows of the rectangular subgraph (see Fig. 7). The search for bottleneck edges is simplified because we only need to search over horizontal edges. The proof consists in showing that the numerator (denominator) of Eq. (1) has a minimum (maximum) for horizontal edges, i.e. when $\lambda = 0$ and thus the search can be restricted to horizontal edges.

We start by observing that the numerator of Eq. (1) is the edge capacity on the residual network at iteration i. To show that this edge capacity is the smallest when $e_{\omega_i^-,\omega_i^+}$ is a horizontal edge, we demonstrate that it decreases at iteration i by the largest value when $e_{\omega_i^-,\omega_i^+}$ is horizontal, according to Eq. (4). Such decrease is given, at each iteration $j < i$ of the MMF algorithm, by multiplying the path flow increment Eq. (2) by the number of paths that pass through $e_{\omega_i^-,\omega_i^+}$ and are saturated at that iteration j. The path flow increment is constant for all paths at each iteration. So we just need to show that, for all iterations $j < i$, the number of s-t shortest paths saturated at each iteration j, passing through $e_{\omega_i^-,\omega_i^+}$ is larger when $e_{\omega_i^-,\omega_i^+}$ is a horizontal, than when it is a diagonal edge. Recall that here we are assuming that all bottlenecks found until iteration $i-1$ are edges of s or t.

Let us start with a $K/2$-by-$(L-1)$ network. At iteration $j < i$, we have an equivalent $(K/2-2(j-1))$-by-$(L-1)$ network. The paths that are saturated at iteration j and cross $e_{\omega_i^-,\omega_i^+}$ also cross *i)* the j-th western bottleneck $e_{s,v_{(j,1)}}$ and one of the edges $e_{v_{(K/2-k+1,L-1)},t}$ for $j \le k < i$; *ii)* the j-th eastern bottleneck $e_{v_{(K/2-j+1,L-1)},t}$ and one of the edges $e_{s,v_{(k,1)}}$ for $j < k < i$. Therefore, the total number of paths saturated at iteration j crossing $e_{\omega_i^-,\omega_i^+}$ is

$$\begin{aligned} n_{sat}^{(j)}(e_{\omega_i^-,\omega_i^+}) &= n(e_{s,v_{(j,1)}}, e_{\omega_i^-,\omega_i^+}, e_{v_{(K/2-j+1,L-1)},t}) \\ &+ \sum_{k=j+1}^{k_2} n(e_{s,v_{(k,1)}}, e_{\omega_i^-,\omega_i^+}, e_{v_{(K/2-j+1,L-1)},t}) \\ &+ \sum_{k=k_2+\lambda}^{K/2-j} n(e_{s,v_{(j,1)}}, e_{\omega_i^-,\omega_i^+}, e_{v_{(k,L-1)},t}). \end{aligned} \tag{29}$$

The symmetry of paths from the source to the sink and vice-versa, together with Pascal's rule allow us to simplify Eq. (29) to

$$\begin{aligned} n_{sat}^{(j)}(e_{\omega_i^-,\omega_i^+})) = &\binom{-j+\frac{K}{2}+L-l_2-\lambda-k_2-2}{L-l_2-3} \\ &\left[\binom{-j-l_2+k_2-2}{l_2-2} + \binom{-j-l_2+k_2-2}{l_2-1}\right] \\ &+ \binom{-j-l_2+k_2-2}{l_2-2} \\ &\left[\binom{-j+\frac{K}{2}+L-l_2-\lambda-k_2-2}{L-l_2-2} - (L-l_2-3)\right]. \end{aligned} \tag{30}$$

Observe that $K/2-j-k_2+2 \ge 0$, and thus the total number of paths saturated at iteration $j < i$ that cross $e_{\omega_i^-,\omega_i^+}$, which is given by Eq. (30), is maximum for $\lambda = 0$, that is when $e_{\omega_i^-,\omega_i^+}$ is a horizontal edge. This implies that the numerator of Eq. (1) has a minimum when the edge is horizontal. Next, we show that the denominator of Eq. (1) has a maximum for horizontal edges. To do this, we need to demonstrate that the number of unsaturated paths crossing $e_{\omega_i^-,\omega_i^+}$ at iteration $j < i$ is larger for horizontal edges than for diagonal ones. Note that all paths passing through the $(K/2-2(j-1))$-by-$(L-1)$ residual network are unsaturated at iteration j, so the number of unsaturated s-t shortest paths that cross $e_{\omega_i^-,\omega_i^+}$ at iteration j is found from Eqs. (12) and (28) as

$$\begin{aligned} &n_{unsat}^{(j)}(e_{\omega_i^-,\omega_i^+}) = \\ &\left(\binom{k_2-j+1}{l_2-1}\right)\left(\binom{K/2-j-k_2-\lambda+2}{L-l_2-2}\right). \end{aligned} \tag{31}$$

Equation (31) yields the maximum value for $\lambda = 0$, *i.e.* for horizontal edges. Thus, the minimum of Eq. (1) is on a horizontal edge. This means that when the first $2(i-1)$ bottlenecks are edges of the source or sink, the $2i$-th bottleneck is either again an edge of s or t, or it is a horizontal bottleneck.

Taken together, these results yield a simple procedure to locate the bottlenecks edges. Starting from the first row $i = 1$ on the rectangular subgraph, we search for the pair of bottleneck edges, i.e. the minima of Eq. (1) over the set of horizontal edges on that row. When $i = 1$, the bottlenecks are edges of s or t. Nevertheless, this is not necessarily the case when $i > 1$. In other words, there may exist an iteration i such that the bottlenecks previously found are edges of s or t, but the new pair of bottlenecks is not. We then know the exact location of the $2i$ bottleneck edges found up to iteration i. When i is the last iteration of the MMF algorithm, we can derive the Max-Min Fair flows analytically from the path counting methods developed above, otherwise we still have an approximation of the MMF flow that is exact up to iteration i of the algorithm. Applying a similar reasoning to the one presented for the first inner bottleneck, it is possible to show analytically that all other bottlenecks, except the bottlenecks of the source (or sink), are horizontal edges as well when *i)* each row $k_j < k_i$ has at least one bottleneck edge found in a previous iteration; *ii)* the last bottleneck to be found at each row $k_j < k_i$ is an edge of the source (or the sink). The first condition guarantees that, up to iteration i, all rows up to row k_i have at least one bottleneck. The second condition guarantees that all s-t shortest paths crossing any edge in all rows $k_j < k_i$ have been saturated, and thus the only unsaturated paths are in the $(K/2-2(i-1))$-by-$(L-1)$ rectangular subgraph. Since these two conditions simplify the path counting, we are able to compute the flow distribution efficiently by a semi-analytical method. This method relies on an analytical analysis to restrict the search to horizontal links, combined with a numerical search over the same horizontal links.

IV. RESULTS

Taken together, the methods developed in Sec. III can now direct our investigations into Max-Min Fair (MMF) flows on nearest neighbour networks with one source and sink pair (s, t). We start by asking how sink inflow varies with node degree K, and the distance between the s and t nodes. As discussed in Sec. III, we can measure the s-t distance in two different ways. On one hand, L is the

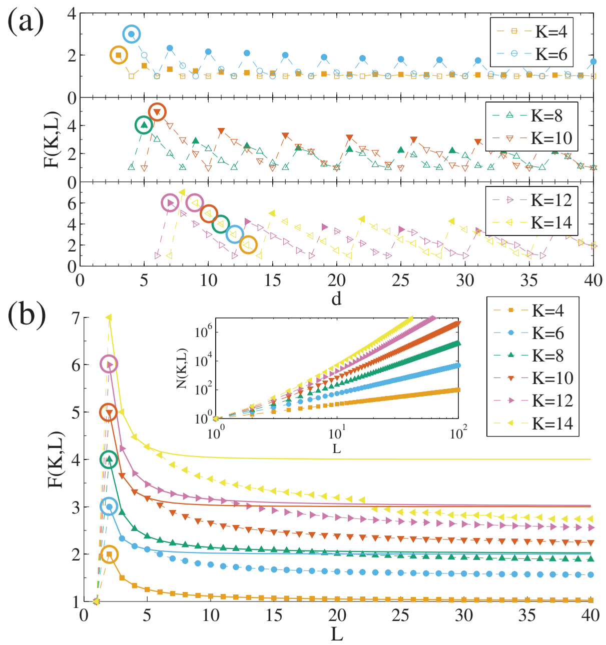

FIG. 9: (Colour online). Plot of the sink inflow as a function of (a) the s-t distance d and (b) the s-t shortest path length L for even K and $4 \leq K \leq 14$. The relation between L and d is given by Eq. (9). The solid symbols denote the sink inflow values common to both panels. The solid curves in (b) are computed assuming that all bottlenecks are edges of the source s or the sink t. The inset in panel (b) shows the approximate polynomial growth of the number of s-t shortest paths as L is varied for several K values. The sink inflow on a $s-t$ pair at the distance $d^{min}(K=14, L=2)+i$ for $i = 1, \cdots, 5$ is the same as the sink inflow on a $s-t$ pair at a smaller distance on a less dense network. The two cases are identified by circles of the same colour on panels (a) and (b).

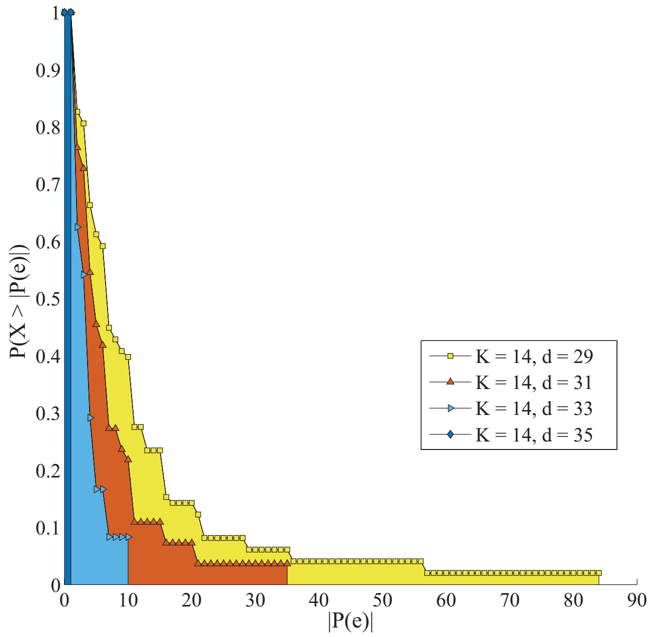

FIG. 10: (Colour online). Complementary cumulative distribution of the number of s-t shortest paths passing through each edge for $L = 5$, as d is varied. We considered only edges that are crossed by at least one path. The diversity of the number of paths crossing each edge is illustrated by the different distributions. When the s-t pair is at the distance $d^{min}(K, L)$, 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 the s-t pair is at the distance $d^{min}(K, L) + K/2 - 1$ 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.

shortest path length between s and t. On the other hand, $d(K, L)$ is the difference between node index labels for t and s, as illustrated in Fig. 2, when these labels are increasing from the source to the sink. The two distances are closely related, because there are $K/2$ s-t pairs at distances $d^{\min}(K, L), \cdots, d^{\min}(K, L) + K/2 - 1$, but all of these s-t pairs are at the shortest path length L from the source s. Our first main result from Sec. III B is that we do not need to investigate the MMF algorithm for all s-t distances. In fact, when the s-t pair is at a distance d on a K-nearest neighbour network, the set of s-t shortest paths remains unchanged if the s-t pair is placed at a given smaller distance on a sparser network. To be more specific, displacing the s-t pair from the distance $d^{\min}(K, L)$ to the distance $d^{\min}(K, L)+i$, for $i = 1, \cdots, K/2-1$, is equivalent to placing the two nodes at the new distance $d^{\min}(K', L)$, on a K'-nearest neighbour network where $K'/2 = K/2-i$. In other words, the s-t shortest paths are equivalent for the values of d on the (K, L) network and the values of K' on the sparser (K', L) network, such that $d + K'/2$ is a constant for a given L. This result implies that the MMF path flows on K-nearest neighbour networks are uniquely determined by their values at the distances $d^{\min}(K, L)$, as K and L are varied. This effect can be observed in Fig. 9a, where we plot the sink inflow as a function of the distance d between the source s and the sink t for several values of K. Such transformation from $d^{\min}(K, L) + i$ to $d^{\min}(K', L)$ explains the oscillatory pattern in the data points. To illustrate this oscillatory effect, we highlight with circles in Fig. 9a the sink inflow values for $d^{\min}(K = 14, L = 2)+i$, $i = 1, \cdots, 5$, as well as the flow values for the corresponding $d^{\min}(K', L = 2)$, demonstrating that the sink inflow is fully determined by its values at the distances $d^{\min}(K, L)$.

Figure 9b is a plot of the sink inflow at distances $d^{\min}(K, L)$ as L is varied for several K values, where points common to Figs. 9a and 9b are represented by solid symbols. When all bottlenecks are edges of the source or the sink, the sink inflow can be computed with the procedure described in Sec. III C, and the result of such computation is given by the solid curves in Fig.9b. Indeed, the computation of the path flow increments in

Eq. (2), and thus the computation of the sink inflow, can be done by counting paths that pass through the bottleneck edges, if the location of these bottlenecks is known, and this is our second main result (see Eqs. (22) and (24)). Furthermore, the path flow increments Eq. (22) can be determined in closed-form for each K on the region of the (K, L) parameter space where the bottlenecks are edges of s or t.

Our choice of shortest path routing allows us to tune the level of interaction among the paths by varying the s-t distance from $d^{\min}$ (maximum number of shortest path intersections) to $d^{\min} - K/2 + 1$ (no intersections). This can be observed in Figs. 2 and 5, which illustrate how the number of s-t shortest paths depends on d. It can also be observed in Fig. 10, which shows the variation with d of the distribution of the number of s-t shortest paths passing through each edge.

Whereas the path flows can be computed in close form if all bottlenecks are edges of s or t, it is still possible to find the path flows from path counting methods alone, even when some bottlenecks are in less trivial locations. Indeed, our third main result is a semi-analytical procedure that reduces the computational complexity of the search for bottleneck edges. Brute force approaches to solving numerically the MMF flow assignment for large networks are beyond the capacity of the current generation of computers. Indeed, for L large, the input $N(K, L)$ to the MMF algorithm grows exponentially in size with K (for fixed L) and polynomially with L (for fixed K), as shown in the inset of Fig. 9b. The search for bottleneck edges on a generic network is done, at each iteration of the MMF algorithm, over all links of all nodes in the residual network, and that is a procedure of the order $O(|V|\,K)$. Our result simplifies the search procedure for nearest neighbour networks, so that its complexity becomes $O(|V|)$, and therefore is independent of node degree K. The simplified search procedure consists in finding the minimum in Eq. (2) over the edges of the residual network that connect two nodes at distance $K/2$ (we say that these edges are horizontal). Once the location of horizontal bottleneck edges is known, the path counting methods developed in Sec. III are the key to finding the path flows and the sink inflow. We find the location of the bottleneck edges, as well as the saturated paths, the path flows and the sink inflow, by repeating this procedure for each iteration of the MMF algorithm. Nevertheless, the bottlenecks found are not necessarily edges of s or t, and we refer to such edges that do not link the source or sink nodes as *inner bottleneck* edges. The sink inflow data points in Fig. 9b were computed with this procedure (also discussed in Sec. III C), and are thus exact results. The inset in Fig. 9b helps us to appreciate the power of the methodology: our method deals well with over 10^6 s-t shortest paths for $K = 14$ and $L = 40$ easily, whereas a brute force version of the MMF algorithm as discussed in Sec. III C would make heavy use of computational resources because all paths and path flows must be stored during the execution of the algorithm.

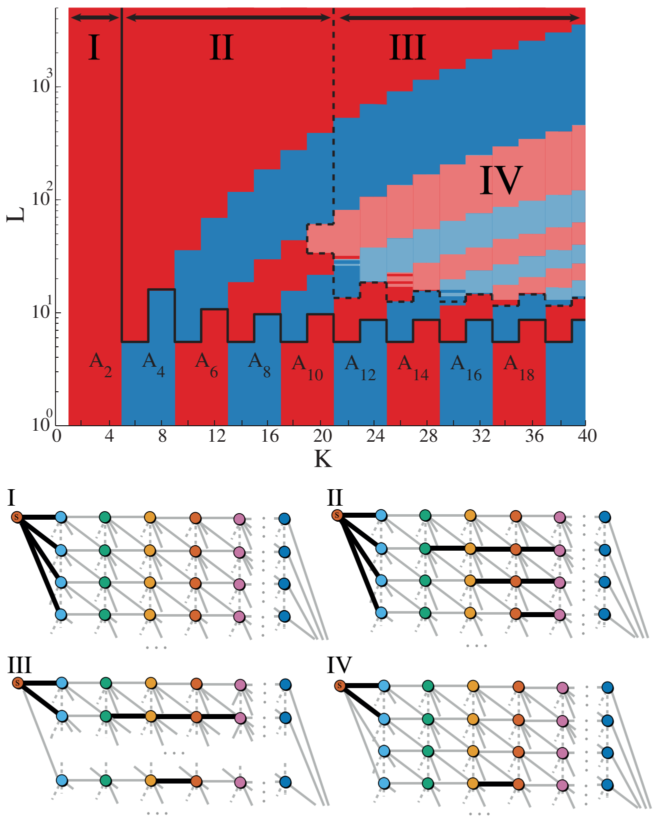

FIG. 11: (Colour online). Each cell corresponds to a (K, L) point in the parameter space, where K is even, and the source and sink pair are at a distance $d^{min}(K, L)$ as given by Eq. (8). We partition the parameter space (K, L) in areas $A_{(2i)}$, such that after the first i iterations the MMF algorithm 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. In region ⓐ, delimited by the solid line, all bottlenecks are edges of the source s or sink t. Region ⓑ, between the solid and dashed lines, is characterised by chains of inner bottlenecks followed by a bottleneck at s (or t) which still allows us to derive exact results. Region ⓒ, in lighter colours, is defined by the presence of gaps in the row number of consecutive bottleneck edges. Region ⓓ is a subset of ⓒ where the gap occurs between a bottleneck of s or t and the first inner bottleneck.

The case for our procedure becomes even more compelling when we investigate the structure of the parameter space (K, L). To show this, we construct a parameter space diagram, where we associate with each (K, L) pair the number of bottlenecks that are edges of s or t up to the iteration when the first inner bottleneck is found. The result, which we plot in Fig. 11, is a partition of the parameter space in areas A_{2i}, such that for K and L values in A_{2i} there are $2i$ bottlenecks which are edges of s or t. For L sufficiently large, the behaviour of the MMF algorithm is described by the area A_2. In other words, the bottlenecks are edges of s or t only for the first iteration of the MMF algorithm. This means that the two bottlenecks found at the first iteration are edges of s or t, but the bottlenecks found at the second iteration are already inner bottlenecks. In fact, when L is large, we only understand well the first iteration of the MMF algorithm, and this opens the possibility that the pattern of the location of bottlenecks in the area A_2 of the parameter space may be much more complicated than we have been able to describe. Four regions of the parameter space illustrate what we know about the MMF algorithm in nearest neighbour networks. The first region ⓐ, delimited in Fig. 11 by a black line, is the set of points in the parameter space (K, L) such that all bottlenecks are edges of s or t, and this is where we are able to derive analytical results. The second region ⓑ, located between the solid and dashed lines, limits the (K, L) values where the semi-analytical methods developed in Sec. III C can solve all iterations of the MMF algorithm. These semi-analytical methods enable the analysis of the path flows and the sink inflow as L grows. Our main finding is that the calculation of sink inflow for L large in region ⓐ (see Eq. (27)) is an upper bound to the sink inflow in region ⓑ, as can be observed in Fig. 9b. This can be explained by the pattern of the location of bottleneck edges in ⓑ (see Fig. 11). More precisely, the capacity constraints are larger in ⓑ than in ⓐ, because the pattern of bottlenecks in ⓑ contains the pattern of bottlenecks in ⓐ. The third region ⓒ is characterised by the presence of a gap in the row number of two consecutive bottlenecks on the rectangular subgraph. Region ⓓ is a subset of region ⓒ where the row gap occurs between a bottleneck adjacent to s or t. This 'gap' in the row number of bottleneck edges implies that a subsequent bottleneck has to appear in the rows that have been skipped. Such detailed knowledge of the location of bottleneck edges in the parameter space of nearest neighbour networks is one of the main contributions of this paper, but it also highlights the limits to our current knowledge of the MMF algorithm, as our methodology is not able to continue computing path flows efficiently for iterations where row gaps occur. The main limitation is that the implementation of the path counting method becomes extensively complicated with the appearance of row gaps.

A natural question to ask is then: can network topology influence the Max-Min Fair allocation of path flows? To gain insight into this problem, we plot the histograms of path flows in Fig. 12 for several (K, L) points inside the regions of the parameter space where we can solve the MMF algorithm exactly (regions ⓐ and ⓑ in Fig. 11). In the region ⓐ of parameter space where all bottlenecks are edges of s or t, the histograms show that the frequency of path flows decays approximately as a power-law with exponent -1 (see Fig. 12a). Two mechanisms contribute to the way that fair flows are allocated in this area. First, the path flows are dominated by the path flow increment at the last iteration, when the paths are saturated. From Eq. (24), we can write

$$f^{(i^*)} \simeq \Delta f^{(i^*)} = 1/N(K/2 - 2(i^* - 1), L). \qquad (32)$$

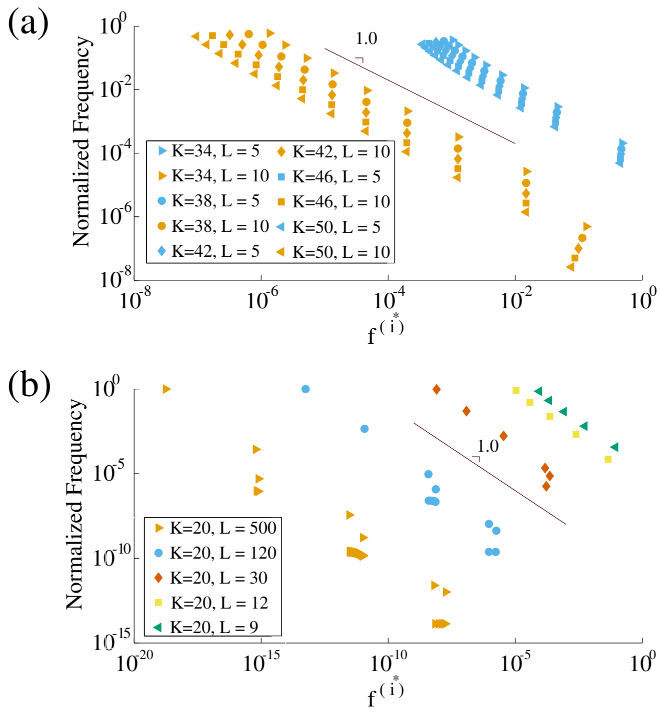

FIG. 12: (Colour online). Histograms of the path flow after the MMF algorithm terminates. (a) When the bottlenecks are edges of either the source or sink nodes (or we observe only a small number of inner bottlenecks) the histogram decays approximately as a power law. (b) As the number of inner bottlenecks grows, we observe a scattered distribution that is caused by different path flows, specific to each inner bottleneck. Two bottlenecks that are on the same row of the rectangular subgraph have similar path flows. As a consequence, path flows of bottlenecks on the same row are clustered on the histogram.

Second, the number $\eta(i)$ of s-t shortest paths that are saturated at each iteration is of the order of magnitude of the number of all paths in the residual network, that is,

$$\eta^{(i^*)} \simeq N(K/2 - 2(i^* - 1), L). \qquad (33)$$

Hence, at each iteration i^*, the path flows, given by Eq. (32), are approximately inversely proportional to the number of saturated paths at iteration i^*, determined by Eq. (33). In other words, $f^{(i^*)} \simeq 1/\eta^{(i^*)}$, and the histogram of path flows decays as a power-law with exponent -1, as seen in Fig. 12a. On the other hand, the presence of inner bottlenecks in the region ⓑ of the parameter space introduces a deviation from the power-law decay (see Fig. 12b). Indeed, paths crossing the bottlenecks that share the same row of the rectangular subgraph have similar path flows, and thus the frequencies of paths flows are clustered on the histogram. Taken together, these findings show that power-law allocations can be fair, even when network capacity is uniform, which is a counter-intuitive and unexpected result.

V. CONCLUSIONS

We analysed the Max-Min Fair flow allocation algorithm on nearest neighbour networks with uniform capacity and node degree K. Our study focused on one single source and sink pair, placed at shortest path distance L. Transport between the source and the sink can only occur along the shortest paths, each of which is assigned a path flow by the Max-Min Fair algorithm. We note that we have analysed the (K, L) parameter space under the assumption that, for each L value, the s-t pair is at the distance $d^{\min}(K, L)$ as given by Eq. (8). Other distances $d^{\min}(K, L) + i$ (for $1 \leq i \leq K/2 - 1$) can be mapped to an s-t pair located at a shorter distance on a sparser network (as discussed in Sec. III B).

Such configuration led us to analyse how the tuning of the parameters, K and L, affects the network throughput between the source and sink pair, as well as the fair allocation of individual path flows. Figure 9 shows the behaviour of the network throughput as a function of the distance between the source and the sink.

We found that the parameter space (K, L) can be partitioned into areas that correspond to distinct patterns in the location of congested edges, *i.e.* bottlenecks. When the bottlenecks are edges of the source s or sink t, we derived a recursive relation that yields the sink inflow and the path flows analytically. Furthermore, in the limit $L \to \infty$, the sink inflow grows with node degree as the piecewise constant function $\lim_{L\to\infty} F(K, L) = \lceil K/4 \rceil c$. This result is exact for node degree $K \leq 4$, when all bottlenecks are edges of the source or the sink, and we showed that $\lceil K/4 \rceil c$ is an upper bound to the sink inflow when $K > 4$ and L is large. In contrast, the maximum flow that can be allocated between the source and the sink node pair is $Kc/2$ [49]. Hence, in the limit $L \to \infty$ there is a decrease in throughput larger than 50% of max-flow, and this limitation of the Max-Min Fair algorithm is well-known in the literature (see *e.g.* [18, 27]). Indeed, Betsimas *et al.* [18] established an upper bound for the loss of throughput in a very general setup, and when applied to our case with $L \to \infty$ these results imply 100% reduction of the throughput. Moreover, numerical studies show that when the number of s-t pairs is large and all paths are allowed, the result is a reduction from max-flow comparable to the case of fixed shortest paths [35]. However, it should be noted that both our results and those of Nace *et al.* [35] are found when paths have many intersections and, hence, are not valid if the network operator would choose the paths not to intersect. Our results can thus be seen as going further, by providing an exact description of the flow reduction in specific areas of the parameter space. Taken together, these findings suggest that implementations of the Max-Min Fair algorithm in very large real world networks should be restricted to situations where throughput can be sacrificed to achieve fair allocations.

For small values of L, all the bottlenecks are edges of s or t and path flows can be computed analytically. As the distance L between the source and the sink increases for a fixed K, the pattern of intersections among shortest paths changes and the location of the bottlenecks becomes less regular. To address the new patterns of the location of the bottlenecks, we then derived a semi-analytical method that combines analytical results that reduce the complexity of the search for bottlenecks, with numerical computations for the search. As a result, we were able to solve the Max-Min Fair allocation for large nearest neighbour networks when $K \leq 20$, despite the very large number of shortest paths involved in the computations.

When all bottlenecks are edges of the source or the sink, we find that the histogram of path flows decays approximately as a power-law with exponent -1. This uneven distribution is unforeseen, because the Max-Min Fair algorithm assigns path flows fairly, *i.e.* the least well off get as much as possible under the capacity constraints. This counter-intuitive result that power-law allocations can be fair is a consequence of the constraints placed on the Max-Min Fair allocation by the combination of the network topology and the routing over shortest paths. Moreover, we observed deviations from this power-law decay in regions of the parameter space where some bottlenecks are not edges of the source or sink. These unexpected results suggest that network designers should be aware of the interplay between the structure of transport routes and network topology, which is a crucial factor in the fair allocation of network flows.

Finally, the analysis developed in this paper may be extended in several directions. A possible line of enquiry is to study the network throughput as a function of the number of distinct sink-source pairs. Still another possible step is to investigate the effect of network topology on Max-Min Fair flows. Further studies are required to understand whether the reduction in throughput that we have found is specific to the choice of fixed shortest paths, or whether it is a more general result. Our study is just a first step towards a rigorous understanding of fair allocations in complex networks. Due to the importance of the topic, further insights could be reached by analysing random graphs or real world network topologies. It would also be important to gain a better understanding of the relation between different routing strategies and the network throughput. Last, but not least, it would be worthwhile to consider other fair allocation schemes, such as the Proportional Fairness [24].

Acknowledgments

We gratefully acknowledge the support of the RAVEN EPSRC project (EP/H04812X/1). L.B. gratefully acknowledges partial financial support by the Ministry of Education of the Slovak Republic (project VEGA 1/0296/12).

[1] P. Whittle, *Networks Optimisation and Evolution*, Cambridge Series in Statistical and Probabilistic Mathematics (Cambridge University Press, 2007).
[2] A. Barrat, M. Barthélemy, and A. Vespignani, *Dynamical Processes on Complex Networks* (Cambridge University Press, 2008).
[3] S. Bohn and M. O. Magnasco, Phys. Rev. Lett. **98**, 088702 (2007).
[4] F. Corson, Phys. Rev. Lett. **104**, 048703 (2010).
[5] G. Li, S. D. S. Reis, A. A. Moreira, S. Havlin, H. E. Stanley, and J. S. Andrade, Phys. Rev. Lett. **104**, 018701 (2010).
[6] Y. X. Xia and D. Hill, EPL **89**, 58004 (2010).
[7] R. Carvalho, L. Buzna, F. Bono, E. Gutierrez, W. Just, and D. Arrowsmith, Phys. Rev. E **80**, 016106 (2009).
[8] E. Katifori, G. J. Szollosi, and M. O. Magnasco, Phys. Rev. Lett. **104**, 048704 (2010).
[9] A. Tero, S. Takagi, T. Saigusa, K. Ito, D. P. Bebber, M. D. Fricker, K. Yumiki, R. Kobayashi, and T. Nakagaki, Science **327**, 439 (2010).
[10] T. Ohira and R. Sawatari, Phys. Rev. E **58**, 193 (1998).
[11] B. Danila, Y. Yu, J. A. Marsh, and K. E. Bassler, Phys. Rev. E **74**, 046106 (2006).
[12] S. Sreenivasan, R. Cohen, E. Lopez, Z. Toroczkai, and H. E. Stanley, Phys. Rev. E **75**, 036105 (2007).
[13] D. De Martino, L. Dall'Asta, G. Bianconi, and M. Marsili, Phys. Rev. E **79**, 015101(R) (2009).
[14] B. Danila, Y. D. Sun, and K. E. Bassler, Phys. Rev. E **80**, 066116 (2009).
[15] D. Helbing, M. Schonhof, and D. Kern, New J. Phys. **4**, 33 (2002).
[16] D. Helbing, M. Schonhof, H. U. Stark, and J. A. Holyst, Adv. Complex Syst. **8**, 87 (2005).
[17] H. Youn, M. T. Gastner, and H. Jeong, Phys. Rev. Lett. **101**, 128701 (2008).
[18] D. Bertsimas, V. F. Farias, and N. Trichakis, Oper. Res. **59**, 17 (2011).
[19] S. J. Brams and A. D. Taylor, *Fair Division: From cake-cutting to dispute resolution* (Cambridge University Press, 1996).
[20] J. Robertson and W. Webb, *Cake-Cutting Algorithms: Be Fair If You Can* (A K Peters, 1998).
[21] J. Barbanel, *The Geometry of Efficient Fair Division* (Cambridge University Press, 2005).
[22] A. D. Taylor and A. M. Pacelli, *Mathematics and Politics: Strategy, Voting, Power and Proof* (Springer, 2008).
[23] D. P. Bertsekas and R. Gallager, *Data Networks* (Prentice Hall, 1992), 2nd ed.
[24] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, J. Oper. Res. Soc. **49**, 237 (1998).
[25] R. Srikant, *The Mathematics of Internet Congestion Control* (Birkhäuser, 2003).
[26] F. P. Kelly, Philos. Trans. R. Soc. Lond. Ser. A-Math. Phys. Eng. Sci. **358**, 2335 (2000).
[27] J. Kleinberg, Y. Rabani, and E. Tardos, J. Comput. Syst. Sci. **63**, 2 (2001).
[28] L. Massoulie and J. Roberts, IEEE-ACM Trans. Netw. **10**, 320 (2002).
[29] N. Megiddo, Mathematical Programming **7**, 97 (1974).
[30] N. Megiddo, Bull. Amer. Math. Soc. **83**, 407 (1977).
[31] K. Y. M. Wong and D. Saad, Phys. Rev. E **76**, 011115 (2007).
[32] J. M. Jaffe, IEEE Trans. Commun. **29**, 954 (1981).
[33] A. Kumar and J. Kleinberg, SIAM J. Comput. **36**, 657 (2006).
[34] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden, SIAM J. Comput. **38**, 1602 (2008).
[35] D. Nace, N. L. Doan, E. Gourdin, and B. Liau, IEEE-ACM Trans. Netw. **14**, 1272 (2006).
[36] D. Nace, L. N. Doan, O. Klopfenstein, and A. Bashllari, Comput. Oper. Res. **35**, 557 (2008).
[37] B. Radunovic and J. Y. Le Boudec, IEEE-ACM Trans. Netw. **15**, 1073 (2007).
[38] D. Nace and M. Pioro, IEEE Commun. Surv. Tutor. **10**, 5 (2008).
[39] L. D. Costa, O. N. Oliveira, G. Travieso, F. A. Rodrigues, P. R. V. Boas, L. Antiqueira, M. P. Viana, and L. E. C. Rocha, Adv. Phys. **60**, 329 (2011).
[40] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. U. Hwang, Phys. Rep. **424**, 175 (2006).
[41] M. A. de Menezes and A. L. Barabasi, Phys. Rev. Lett. **92**, 028701 (2004).
[42] M. T. Gastner and M. E. J. Newman, J. Stat. Mech. p. P01015 (2006).
[43] V. Latora and M. Marchiori, Phys. Rev. Lett. **87**, 198701 (2001).
[44] M. Pioro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks* (Morgan Kaufmann, 2004).
[45] An alternative would be to consider a directed network, and path flows as a snapshot of network usage in time, but we will not explore this possibility here because undirected networks are traditionally studied before directed ones.
[46] D. J. Watts and S. H. Strogatz, Nature **393**, 440 (1998).
[47] M. Newman, *Networks: An Introduction* (Oxford University Press, 2010).
[48] R. P. Stanley, *Enumerative Combinatorics*, vol. 1 (Cambridge University Press, 2000), 2nd ed.
[49] S. Carmi, Z. Wu, E. Lopez, S. Havlin, and H. E. Stanley, Eur. Phys. J. B **57**, 165 (2007).
[50] An alternative would be to consider a directed network, and path flows as a snapshot of network usage in time, but we will not explore this possibility here because undirected networks are traditionally studied before directed ones.