

Network extraction by routing optimization

Diego Baptista^{1,+}, Daniela Leite^{1,+}, Enrico Facca², Mario Putti³, and Caterina De Bacco^{1,*}

¹Max Planck Institute for Intelligent Systems, Cyber Valley, 72076, Tübingen, Germany

²Centro di Ricerca Matematica Ennio De Giorgi, Scuola Normale Superiore, Piazza dei Cavalieri, 3, Pisa, Italy

³Department of Mathematics “Tullio Levi-Civita”, University of Padua, via Trieste 63, Padua, Italy

*caterina.debacco@tuebingen.mpg.de

+These authors contributed equally to this work

ABSTRACT

Routing optimization is a relevant problem in many contexts. Solving directly this type of optimization problem is often computationally unfeasible. Recent studies suggest that one can instead turn this problem into one of solving a dynamical system of equations, which can instead be solved efficiently using numerical methods. This results in enabling the acquisition of optimal network topologies from a variety of routing problems. However, the actual extraction of the solution in terms of a final network topology relies on numerical details which can prevent an accurate investigation of their topological properties. In fact, in this context, theoretical results are fully accessible only to an expert audience and ready-to-use implementations for non-experts are rarely available or insufficiently documented. In particular, in this framework, final graph acquisition is a challenging problem in-and-of-itself. Here we introduce a method to extract networks topologies from dynamical equations related to routing optimization under various parameters' settings. Our method is made of three steps: first, it extracts an optimal trajectory by solving a dynamical system, then it pre-extracts a network and finally, it filters out potential redundancies. Remarkably, we propose a principled model to address the filtering in the last step, and give a quantitative interpretation in terms of a transport-related cost function. This principled filtering can be applied to more general problems such as network extraction from images, thus going beyond the scenarios envisioned in the first step. Overall, this novel algorithm allows practitioners to easily extract optimal network topologies by combining basic tools from numerical methods, optimization and network theory. Thus, we provide an alternative to manual graph extraction which allows a grounded extraction from a large variety of optimal topologies. The analysis of these may open up the possibility to gain new insights into the structure and function of optimal networks. Our algorithm is open source and available at <https://github.com/Danielaleite/NextRout>.

Introduction

Investigating optimal network topologies is a relevant problem in several contexts, with applications varying from transportation networks¹⁻⁴, communication systems⁵⁻⁷, biology^{8,9} and ecology¹⁰⁻¹². Depending on the specified objective function and set of constraints of a routing optimization problem¹³, optimal network topologies can be determined by different processes ranging from energy-minimizing tree-like structures ensuring steeper descent through a landscape as in river basins¹⁰ to the opposite scenario of loopy structures that favor robustness to fluctuations and damage as in leaf venation^{12,14}, the retina vascular system^{15,16} or noise-cancelling networks⁷.

In many applications, optimal networks can arise from an underlying process defined on a continuous space rather than a discrete network as in standard combinatorial optimization routing problems¹⁷⁻²⁰. Optimal routing networks try to move resources by minimizing the transportation cost. This cost may be taken to be a function of the traveled distance, such as in Steiner trees, or proportional to the dissipated energy, such as optimal channel networks or resistance networks. The common denominator of these configurations is that they have a tree-like shape, i.e., optimal routing networks are loopless^{1,21}. Recent developments in the mathematical theory of optimal transport^{11,13} have proved that this is indeed the case and that complex fractal-like networks arise from branched optimal transport problems²². While the theory starts to consolidate, efficient numerical methods are still in a pre-development stage, in particular in the case of branched transport, where only a few results are present^{23,24}, reflecting the obstacle that all these problems have an NP-hard genesis. Recent promising results^{25,26} map a computationally hard optimization problem into finding the long-time behavior of a system of dynamic partial differential equations, the so-called Dynamic Monge-Kantorovich (DMK) approach, which is instead numerically accessible, computationally efficient, and leads to network shapes that resemble optimal structures²⁷. Working in discretized continuous space, and in many network-based discretizations such as lattice-like networks as well, requires the use of threshold values for the identification of active network edges. This has the main consequence that there might be no obvious final resulting network, an output that would be trivial when starting from an underlying search space formed by predefined selected network structures. For example, the output of a numerically discretized (by, e.g., the Finite Element method) routing optimization problem in a 2D space is a real-valued

function on a set of (x, y) points defined on a grid or triangulation, which already has a graph structure. Despite the underlying graph, this grid function contains numerous side features, such as small loops and dangling vertices, that prevent the recognition of a clear optimal network structure. Obtaining this requires a suitable identification of vertices and edges that should contain the optimal network properties embedded in the underlying continuous space. In other words, the output of a routing optimization problem in continuous space carries unstructured information about optimality that is hard to interpret in terms of network properties. Extracting a network topology from this unstructured information would allow, on one hand, better interpretability of the solution and enable the comparison with networks resulting from discrete space. On the other hand, the use of tools from network theory to investigate the properties related to optimality, for instance, to perform clustering or classification tasks based on a set of network features. One can frame this problem as that of properly compressing the information contained in the “raw” solution of a routing problem in continuous space into an interpretable network structure while preserving the important properties connected to optimality. This is a challenging task, as compression might result in losing important information. The problem is made even more complex because one may not know in advance what are the relevant properties for the problem at hand, a knowledge that could help drive the network extraction procedure. This is the case for any real network, where the intrinsic optimality principle is elusive and can only be speculated about by observing trajectories, an approach adopted for instance when processing images in biological networks^{28–31}.

Several works have been proposed to tackle domain-specific network extraction. These methods include using segmentation techniques on a set of image pixels to extract a skeleton^{28,29,32} that is then converted into a network; a pipeline combining different segmentation algorithms building from OpenCV,³³ which is made available with an intuitive graphical interface³⁴; graph-based techniques³⁵ that sample junction-points from input images; methods that use deep convolutional neural networks³⁶ or minimum cost path computations³⁷ to extract road networks from images. These are mainly using image processing techniques as the input is an image or photograph, which might not necessarily be related to a routing optimization problem.

In this work, we propose a new approach for the extraction of network topologies and build a protocol to address this problem. This can take in input the numerical solution of a routing optimization problem in continuous space as described in^{25–27} and then processes it to finally output the corresponding network topology in terms of a weighted adjacency matrix. However, it can also be applied to more general inputs, such as images, which may not necessarily come from the solution of an explicit routing transportation problem. Specifically, our work features a collection of numerical routines and graph algorithms designed to extract network structures that can then be properly analyzed in terms of their topological properties. The extraction pipeline consists in a sequence of three main algorithmic steps: i) compute the steady state solutions of the DMK equations (*DMK-Solver*); ii) extract the optimal network solution of the routing optimization problem (*graph pre-extraction*); iii) filter the network removing redundant structure (*graph filtering*). While for this work we test and demonstrate our algorithm on routing scenarios coming from DMK, which constitute our main motivation, we remark that only the first step is specific to these, whereas the last two steps are applicable beyond these settings. The graph pre-extraction step consists of a set of rules aiming at building a network from an input that is not explicitly a topological structure made of nodes and edges. The filtering step is based on a principled mathematical model inspired by that of the first step, which leads to an efficient algorithmic implementation. Our network filter has a nice interpretation in terms of a cost function that interpolates between an operating cost and an infrastructure one, contrarily to common approaches used in image processing for filtering, which often relies on heuristics.

A successful execution will return a representation of the network in terms of an edge-weighted undirected network. The resulting weights are related to the optimal flow, solution of the routing problem. Once the network is obtained, practitioners can deploy arbitrary available network analysis software^{38–41} or custom-written scripts to investigate properties of the optimal topologies. While our primary goal is to provide a framework and tool to solve the research question of how to extract network topologies resulting from routing optimization problems in continuous space or any other image containing a network structure, we also aim at encouraging non-expert practitioners to automatically extract networks from such problems or from more general settings beyond that. Thus we make available an open-source algorithmic implementation and executables of this work at <https://github.com/Danielaleite/Nextroul>.

1 The routing optimization problem

In this section, we describe the main ideas and establish notation. We start by introducing the dynamical system of equations corresponding to the DMK routing optimization problem as proposed by Facca et al.^{25–27} In these works, the authors first generalize the discrete dynamics of the slime mold *Physarum Polycephalum* (PP) to a continuous domain; then they conjecture that, like its discrete counterpart, its solution tends to an equilibrium point which is the solution of the Monge-Kantorovich optimal mass transport⁴² as time goes to infinity.

We denote the space where the routing optimization problem is set as $\Omega \in \mathbb{R}^n$, an open bounded domain that compactly contains $f(x) = f^+(x) - f^-(x) \in \mathbb{R}$, the forcing function, describing the flow generating sources $f^+(x)$ and sinks $f^-(x)$. It is assumed that the system is isolated, i.e., no fluxes are entering or exiting the domain from the boundary. This imposes the

constraint $\int_{\Omega} f(x)dx = 0$ to ensure mass balance. It is assumed that the flow is governed by a transient Fick-Poiseuille type flux $q = -\mu \nabla u$, where $\mu(t, x), u(t, x)$ are called *conductivity* or *transport density* and *transport potential*, respectively.

The continuous set dynamical Monge-Kantorovich (DMK) equations are given by:

$$-\nabla \cdot (\mu(t, x) \nabla u(t, x)) = f^+(x) - f^-(x), \quad (1)$$

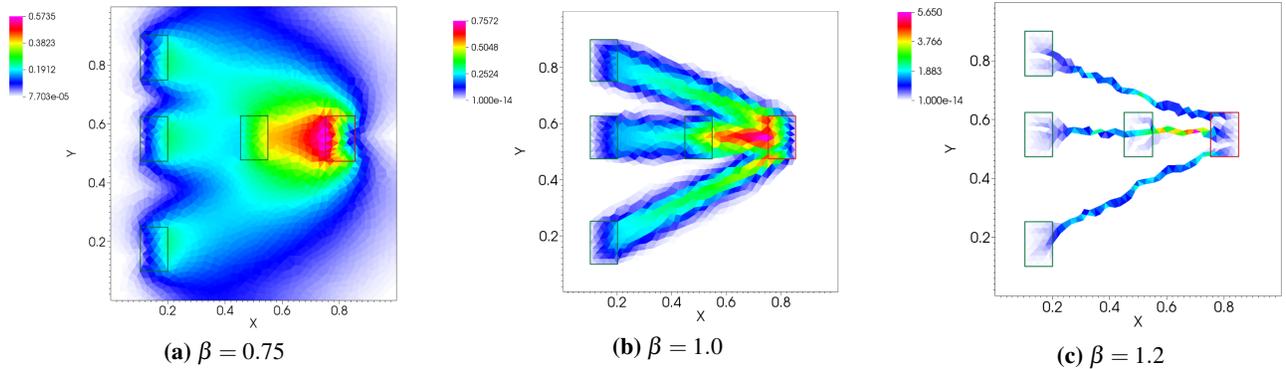
$$\frac{\partial \mu(t, x)}{\partial t} = [\mu(t, x) \nabla u(t, x)]^\beta - \mu(t, x), \quad (2)$$

$$\mu(0, x) = \mu_0(x) > 0, \quad (3)$$

where $\nabla = \nabla_x$. Equation (1) states the spatial balance of the Fick-Poiseuille flux and is complemented by no-flow Neumann boundary conditions; Eq. (2) enforces the system dynamics in analogy with the discrete PP model and Eq. (3) provides the initial configuration of the system. The parameter β captures different routing transportation mechanisms. A value of $\beta < 1$ enforces optimal solutions to avoid traffic congestion; $\beta = 1$ is shortest path-like; while $\beta > 1$ encourages consolidating the flow so to use a smaller amount of network-like infrastructure, and is related to branched transport^{11,43}. Within a network-like interpretation, qualitatively, $\mu(x, t)$ describes the capacity of the network edges⁴⁴. Thus, its initial distribution $\mu_0(x)$ describes how the initial capacities are distributed.

In this work, solving the routing optimization problem consists of finding the steady state solution $(\mu^*, u^*) : \Omega \rightarrow \mathbb{R}_{\geq 0} \times \mathbb{R}$ of Eq. (1), i.e. $(\mu^*(x), u^*(x)) = \lim_{t \rightarrow +\infty} (\mu(t, x), u(t, x))$. Numerical solution of the above model can be obtained by means of a double discretization in time and space²⁵⁻²⁷. The resulting solver (called from now on *DMK-Solver*) has been shown to be efficient, robust and capable of identifying the typically singular structures that arise from the original problem. In Fig. 1, some visual examples of the numerical μ^* obtained for different values of β are shown. The same authors showed that the *DMK-Solver* is able to emulate the results for the discrete formulation of the PP model proposed by Tero et al.⁴⁵ Under appropriate regularity

Figure 1. Different values of β in Eq. (2) lead to different settings of a routing optimization problem. Colors denote different intensities of conductivity μ as described by the color bar on the left. (a) $\beta < 1$ enforces avoiding mass congestion; (b) $\beta = 1$ is shortest path-like, the mass goes straight from source to sink; (c) $\beta > 1$ encourages traffic consolidation. Red rectangle denotes the sink, green ones the four sources.



assumptions, it can be shown^{26,27} that the equilibrium solution of the above problem $(\mu^*(x), u^*(x))$ is a minimizer of the following functional:

$$\mathcal{L}(\mu, u) = \frac{1}{2} \int_{\Omega} \mu |\nabla u|^2 dx + \int_{\Omega} \frac{\mu^{P(\beta)}}{P(\beta)} dx, \quad (4)$$

where $P(\beta) = (2 - \beta)/\beta$. In words, this functional is the sum of the total energy dissipated during transport (the first term is the Dirichlet energy corresponding to the solution of the first PDE) plus a nonlinear (sub-additive) function of the total capacity of the system at equilibrium. In terms of costs, this functional can be interpreted as the cost of transport, assumed to be proportional to the total dissipated energy, and the cost of building the transport infrastructure, assumed to be a nonlinear function (with power $2 - \beta$) of the total transport capacity of the system.

We exploit the robustness of this numerical solver to extract the solutions of DMK equations corresponding to various routing optimization problems. We here focus on the case $\beta \geq 1$, where the approximate support of μ^* displays a network-like structure. This is the first step of our extraction pipeline, which we denote as *DMK-Solver*. The numerical solution of these equations does not allow for a straightforward network representation. Indeed, depending on various numerical details related to the spatial discretization and other parameters, one usually obtains a visually well-defined network structure (see Fig. 1a)

whose rendering as a graph object is however uncertain and non-unique. This in turns can hinder a proper investigation of the topological properties associated to optimal routing optimization problems, motivating the main contribution of our work: the proposal of a graph extraction pipeline to automatically and robustly extract network topologies from the solutions' output of *DMK-Solver*. We reinforce that our contribution is not limited to this application, but is also able to extract network-like shapes from any kind of image where a color or greyscale thresholds can be used to identify the sought structure.

Our extraction pipeline then proceeds with two main steps: *pre-extraction* and *graph filtering*. The first one tackles the problem of translating a solution from the continuous scenario into a graph structure, while the second one addresses the problem of removing redundant graph structure resulting from the previous step. A pseudo-code of the overall pipeline is provided in Algorithm 1⁴⁶.

Algorithm 1 Generating optimal networks from solutions of dynamical systems: the pipeline

Input: parameters to set up network extraction problem:

- i) Set the space Ω : $\{T_i\}_i$ grid triangulation and mesh-related parameters.
- ii) Set up routing optimization problem: $\beta \geq 1$, μ_0 , f^+ , f^- and threshold δ .
- iii) Set up the *discrete* routing optimization (for *graph filtering*): β_d , δ_d and τ_{BC} .

Output: $G(V, E, W)$ final network

1. *Run DMK-Solver*: outputs (μ^*, u^*)
 2. *Graph pre-extraction*: outputs $G(V, E, W)$ with possible redundancies
 3. *Graph filtering*: removes redundancies from $G(V, E, W)$
-

Our final goal is to translate the solution pair (μ^*, u^*) into a proper network structure using several techniques from graph theory. With these networks at hand, a practitioner is then able to investigate topologies associated with this novel representation of routing optimization solutions.

2 Graph preliminary extraction

In this section, we expand on the *graph pre-extraction* step: extracting a network representation from the numerical solution output of the *DMK-Solver*. This involves a combination of numerical methods for discretizing the space and translating the values of μ^* , and u^* into edge weights of an auxiliary network, which we denote as $G = (V, E, W)$, where V is the set of nodes, E the set of edges and W the set of weights.

The DMK solver outputs the solution on a triangulation of the domain Ω (here also named *grid*) and denoted as $\Delta_\Omega = \{T_i\}_i$, with $\cup T_i = \Omega$. The numerical solution, piecewise constant on each triangle T_i , is considered assigned to the triangle barycenter (center of gravity) at position $\mathbf{b}_i = (x_i, y_i) \in \Omega$ ⁴⁷. This means that the result is a set of pairs $\{(\mu^*(\mathbf{b}_i), u^*(\mathbf{b}_i))\}_i$. We can track any function of these two quantities. For simplicity, we use μ^* (see Fig. 1 for various examples), but one could use u^* or a function of these two. This choice does not affect the procedure, although the resulting network might be different.

We neglect information on the triangles where the solution is smaller than a user-specified threshold $\delta \in \mathbb{R}_{\geq 0}$, in order to work only with the most relevant information. Formally, we only keep the information on T_i such that $\mu^*(\mathbf{b}_i) \geq \delta$. We observed empirically that in many cases, several triangles contain a value of μ^* that is orders of magnitude smaller than others, see for instance the scale of Fig. 1. Since we want to build a network that connects these barycenters, we remark that this procedure depends on the choice of the threshold δ : if $\delta_1 < \delta_2$, then $G(\delta_2) \subset G(\delta_1)$. On one hand, the smaller δ , the more likely G is to be connected, but at the cost of containing many possibly loop-forming edges and nodes (the extreme case $\delta = 0$ uses the whole grid to build the final network); on the other hand, the higher δ , the smaller the final network is (both in terms of the number of nodes and edges). Thus one needs to tune the parameter δ such that resulting paths from sources to sinks are connected while avoiding the inclusion of redundant information.

The set of relevant triangles does not correspond to a straightforward meaningful network structure, i.e. a set of nodes and edges connecting neighboring nodes. In fact, we want to remove as much as possible the biases introduced by the underlying triangulation and thus we start by connecting the triangle barycenters. For this, we need rules for defining nodes, edges and weights on the edges. Here, we propose three methods for defining the graph nodes and edges and two functions to assign the weights. The overall *graph pre-extraction* routine is given by choosing one of the former and one of the latter, and it can be applied also to more general inputs beyond solutions of the *DMK-Solver*.

2.1 Rules for selecting nodes and edges

Selecting V and E requires defining the neighborhood $\sigma(T_i)$ of a triangle in the original triangulation Δ_Ω (for i such that $\mu^*(\mathbf{b}_i) \geq \delta$). We consider three different procedures:

- (I) Edge-or-node sharing: $\sigma(T_i)$ is the set of triangles that either share a grid edge or a grid node with T_i .

- (II) Edge-only sharing: $\sigma(T_i)$ is the set of triangles that share a grid edge with T_i . Note that $|\sigma(T_i)| \leq 3, \forall i$.
- (III) Original triangulation: let v, w, s be the grid nodes of T_i ; then add v, w, s to V and $(v, w), (w, s), (s, v)$ to E .

It is worth mentioning that since the grid Δ_Ω is non uniform and μ^* is not constant, we cannot control *a priori* the degree d_i of a node i in the graph G generated for a particular threshold δ . We give examples of networks resulting from these three definitions in Fig. 2 and a pseudo-code for the first two in Algorithm 2.

Algorithm 2 Graph extraction

Input: (μ, u) solution of the *DMK-Solver*, δ threshold, $\{T_i\}_i$ grid triangulation
Output: a network $G(V, E, W)$
Initialize: $V, E = \emptyset$
for i *s.t.* $\mu(\mathbf{b}_i) \geq \delta$ **do**
 $V \leftarrow V \cup \{i\}$
 for $T_j \in \sigma(T_i)$ *s.t.* $\mu(\mathbf{b}_j) \geq \delta$ **do**
 $V \leftarrow V \cup \{j\}$
 $E \leftarrow E \cup e_{ij} := (i, j)$
 $w_{ij} = f(\mu(\mathbf{b}_i), \mu(\mathbf{b}_j))$
 end for
end for

2.2 Rules for selecting weights

The weights w_{ij} to be assigned to edges $e_{ij} := (i, j) \in E$ should be a function of $\mu(\mathbf{b}_i)$ and $\mu(\mathbf{b}_j)$, the density contained in the original triangles. We consider two possibilities for this function:

- (i) Average (AVG): $w_{ij} = \frac{\mu(\mathbf{b}_i) + \mu(\mathbf{b}_j)}{2}$.
- (ii) Effective reweighing (ER): $w_{ij} = \frac{\mu(\mathbf{b}_i)}{d_i} + \frac{\mu(\mathbf{b}_j)}{d_j}$.

While using the average as in (i) captures the intuition, it may overestimate the contribution of a triangle when this has more than one neighbor in G with the risk of calculating a total density larger than the original output of the *DMK-Solver*. To avoid this issue, we consider an *effective* reweighing as in (ii), where each triangle contribution by the degree $d_i = |\sigma_i|$ of a node $i \in V$ is reweighted, with σ_i the set of neighbors of i . This guarantees the recovery of the density obtained from *DMK-Solver*, since $\frac{1}{2} \sum_{i,j} w_{ij} = \frac{1}{2} \sum_i \left[\mu(\mathbf{b}_i) + \sum_{j \in \sigma_i} \frac{\mu(\mathbf{b}_j)}{d_j} \right] = \sum_i \mu(\mathbf{b}_i)$, where in the sum we neglected isolated nodes, i.e. i s.t. $d_i = 0$. Note that in the case of choosing the original triangulation for node and edge selection (case (III) above), the ER rule does not apply; in that case, we use AVG.

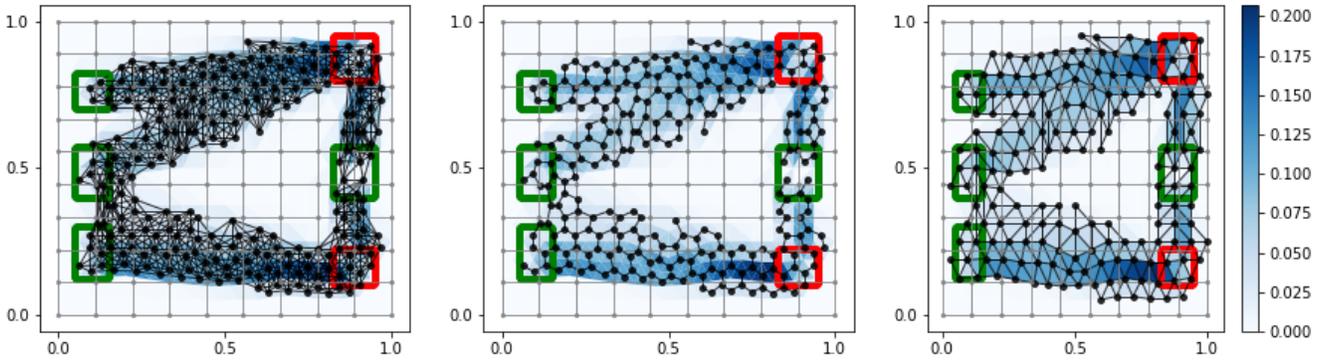


Figure 2. Graph pre-extraction rules. Left): edge-or-node sharing (I); center): edge-only sharing (II); right): original triangulation (III). We monitor the conductivity μ and use parameters $\mu_0 = 1, \beta = 1.02, \delta = 0.0001$. Weights w_{ij} are chosen with AVG (i), f is chosen such that sources and sinks are inside green and red rectangles respectively.

3 Graph filtering

The output of the graph extraction step is a network closer to our expectation of obtaining an optimal network topology resulting from a routing optimization problem. However, this network may contain redundant structures like dangling nodes or small irrelevant loops (see Fig. 2). These are not related to any intrinsic property of optimality, but rather are a feature of the discretization procedure resulting from the graph pre-extraction step. It is thus important to filter the network by removing these redundant parts. However, how to perform this removal in an automated and principled way is not an obvious task. One has to be careful in removing enough structure, while not compromising the core optimality properties of the network. This removal is then a problem in-and-of-itself, we name it *graph filtering* step. We now proceed by explaining how we tackle it in a principled way and discuss its quantitative interpretation in terms of minimizing a cost function interpolating between an operating and an infrastructural cost.

3.1 The *Discrete DMK-Solver*

Going beyond heuristics and inspired by the problem presented in Sec. 1, we consider as a solution for the graph filtering step, the implementation of a second routing optimization algorithm to the network G output of the pre-extraction step, i.e. in discrete space. Several choices for this could be drawn, for instance, from routing optimization literature⁴⁸, but we need to make sure that this second optimization step does not modify any of the intrinsic properties related to optimality resulting from the *DMK-Solver*. We thus propose to use a *discrete* version of the *DMK-Solver* (*discrete-DMK-Solver*). This was proven to be related to the *Basis Pursuit* (BP) optimization problem⁴⁹. In fact, BP is related⁵⁰ to the PP dynamical problem in discrete space and the *discrete-DMK-Solver* gives a solution to the PP in discrete space⁴⁹. The discretization results in a reduction of the computational costs for solutions of BP problems, compared to standard combinatorial optimization approaches⁴⁹. Being an adaptation to discrete settings of our original optimization problem, it is a natural candidate for a graph filtering step, preserving the solution's properties.

The problem is stated as follows. Consider the *signed incidence* matrix $\mathbf{B} \in \mathbb{M}_{N \times M}$ of a weighted graph $G = (V, E, W)$, with entries $B_{ie} = \pm 1$ if the edge e has node i as start/end point, 0 otherwise; $N = |V|$ and $M = |E|$. Denote $\ell = \{\ell_e\}_e$ the vector of edge lengths, \mathbf{f} a N -dim vector of source-sink values with entries satisfying $\sum_{i \in V} f_i = 0$; this is the discrete analogues of the source-sink function $f(x)$ introduced in Section 1; the functions $\mu(t) \in \mathbb{R}^M$ and $u(t) \in \mathbb{R}^N$ correspond to the *conductivity* and *potential* respectively, similarly to the continuous case, but this time they are vectors with entries $\mu_e(t)$ and $u_i(t)$ defined on edges and nodes respectively. The PP discrete dynamics corresponding to the original routing optimization problem can be written as:

$$f_i = \sum_e B_{ie} \frac{\mu_e(t)}{\ell_e} \sum_j B_{ej} u_j(t), \quad (5)$$

$$\mu_e'(t) = \left[\frac{\mu_e(t)}{\ell_e} \left| \sum_j B_{ej} u_j(t) \right| \right]^{\beta_d} - \mu_e(t), \quad (6)$$

$$\mu_e(0) > 0, \quad (7)$$

where $|\cdot|$ is the absolute value element-wise. Equation (5) corresponds to Kirchoff's law, Eq. (6) is the discrete dynamics with β_d a parameter controlling for different routing optimization mechanisms (analogously to β in Eq. 2); Eq. (7) is the initial condition. The importance of this systems stems in having an interesting theoretical correspondence: its equilibrium point corresponds to the minimizer of a cost function analogous to Eq. (4) that, similarly to the continuous case, can be interpreted as global energy functional. This is:

$$\mathcal{L}_\beta(\mu(t)) = \frac{1}{2} \sum_e \mu_e(t) \left(\frac{1}{\ell_e} \sum_j B_{ej} u_j(\mu(t)) \right)^2 \ell_e + \frac{1}{2} \sum_e \frac{\mu_e(t)^{P(\beta)}}{P(\beta)} \ell_e, \quad (8)$$

where $P(\beta) = 2 - \beta/\beta$ and $u(\mu(t))$ is a function implicitly defined as the solution of Eq. (5). The first term corresponds to the energy dissipated during transport, it can be interpreted as the operating costs, whereas the second is the infrastructural cost. The equilibrium point of $\mu_e(t)$ is stationary at the previous energy function, and for $\beta_d = 1$ it acts also as the global minimizer due to its convexity. For $\beta_d > 1$ the energy is not convex, thus in general the functional will present several local minima towards which the dynamics will be attracted. The case $\beta_d < 1$ does not act as a filter because it encourages trajectories to spread through the network, instead of removing edges, and so not interesting to our purposes. Discretization in time of Eq. (6) by the implicit Euler scheme combined with Newton method leads to an efficient numerical solver, see Facca et al.⁴⁹ for more details. The above scheme gives the solution to the BP problem and represents the *discrete-DMK-Solver*. Similarly to the graph pre-extraction step, the filtering is also valid beyond networks related to solutions of the *DMK-Solver*. It applies to more general

inputs if defined on a discrete space, for instance, images. Finally, notice that the filter generates a graph with a new set of nodes and edges, both subsets of the corresponding ones in G , result of the pre-extraction. The weights of the final graph can then be assigned with same rules as in 2.1; in addition, one can consider as weights the values of μ_e resulting from the BP problem (we named this weighing method “BPW”). Alternatively, one can ignore the weights of BP and keep (for the edges remaining after the filter) the weights as in the previous pre-extraction step (labeled as “IBP”). Figure 3 shows an example of three filtering settings on the same input.

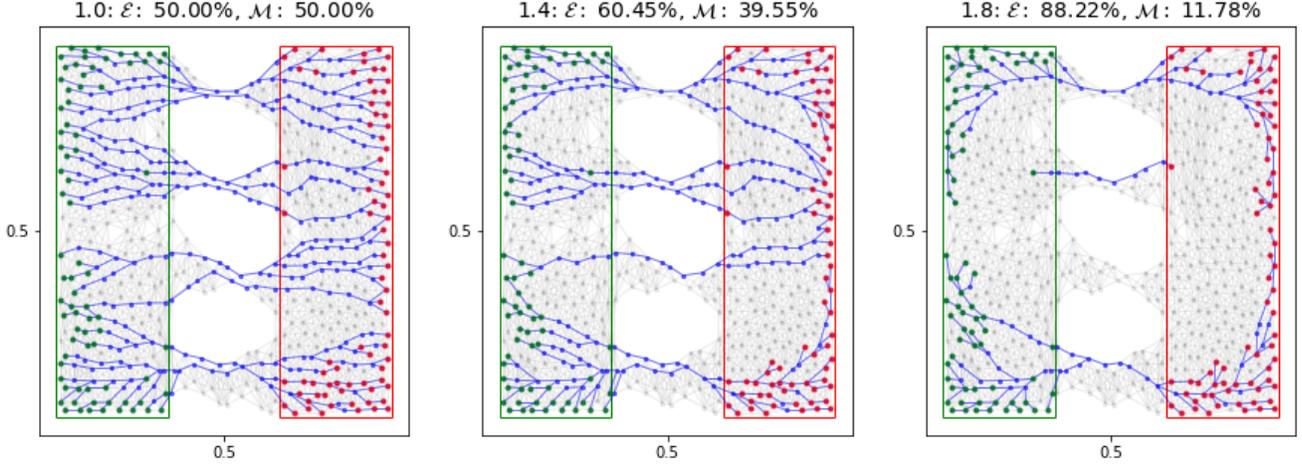


Figure 3. Graph filtering rules. Left): $\beta_d = 1.0$; center): $\beta_d = 1.4$; right): $\beta_d = 1.8$. The number on top denote the contributions of operating and infrastructural cost to the energy. Green and red dots represent sources and sinks respectively ($\tau_{BC} = 10^{-1}$); blue edges are those e with $\mu_e^* \geq 10^{-3}$. The filtering input is generated from *DMK-Solver* with $\beta = 1.05$. The apparent lack of symmetry of the network’s branches is due to numerical discretization of the domain, solver and threshold δ . As the relative size of the terminal set decreases compared to the size of remaining part of the domain, this lack of symmetry becomes negligible.

3.2 Selecting sources and sinks

The *discrete-DMK-Solver* requires in input a set of source and sink nodes (S^+ and S^-) that identify the support of the forcing vector \mathbf{f} introduced in 3.1. However, the graph pre-extraction output G might contain redundant nodes (or edges) as mentioned before. In principle, among the nodes $i \in V$, all of those contained in the support of $f(x = \mathbf{b}_i)$, i.e. contained in the supports of sources and sinks of the original routing optimization problem in Eq. (1), are *eligible* to be treated as sources or sink in the resulting network. However, several paths connecting source and sink nodes may be redundant and clearly not compatible with an optimal routing network (see Supplementary Fig. S2 for such an example). Therefore, it is important to select “representatives” for sources and sinks, such that the final network is heuristically closer to optimality. Here we propose a criterion to select source and sink nodes from the eligible ones in each of the connected components $\{C_m\}_m$ of G , using a combination of two network properties. Starting from the complete graph formed by all the nodes characterized by a significant (above the threshold) density, source and sink nodes and rates are defined as follows. A node $i \in S^+$, i.e. is a source $f_i > 0$, if either i) is in the convex hull of the set of eligible sources or ii) its betweenness centrality is smaller than a given threshold τ_{BC} . Similarly for sink nodes in S^- . This is because, on one side, nodes in the convex hull capture the outer shape structure of the source and sink sets defined in the continuous problem; on the other side, nodes with *small* values of the betweenness centrality capture the end-points of G inside the source and sink sets, analogously to leaves (i.e., degree-one nodes)⁵¹. We present these ideas in more detail in the Supplementary Fig. S2. Once we have identified the sets of source and sink vertices, we need to assign a proper value f_i such that Kirchhoff law is satisfied in each of the different connected components C_m . It is reasonable to assume that each connected component is “closed”, i.e. $\sum_{i \in C_m} f_i = 0, \forall C_m$. Denoting with $|S|$ the number of elements in a set S and $V(C_m)$ the set of nodes in C_m , we then distribute the mass-fluxes uniformly by setting $f_i = \frac{1}{|S^+ \cap V(C_m)|}$ for $i \in S^+$, and $f_i = -\frac{1}{|S^- \cap V(C_m)|}$ for $i \in S^-$ sinks ($f_i = 0$ otherwise) so that the total original source and sink flux is assigned to the overall source/sink nodes of all C_m . Note that this procedure maintains the overall system and each connected component “closed”, as stated above.

Model validation

Our extraction pipeline proceeds by compressing routing information in the raw output of the *DMK-Solver* (although what follows is not restricted to this case) on a lean network structure. This might lead us to lose relevant information in the process. Hence, we need to devise *a posteriori* estimates that provide quantitative guidance on the “leanness” and information loss of the final network. Here we propose metrics to evaluate the compression performance of the various graph pre-extraction and filtering protocols. The raw information is made of a set of weights $w(T_i)$ representing the values (μ^*, u^*) on each of the triangles $T_i \in \Omega$. We consider as the truth benchmark the distribution of w , or any other quantity of interest, supported on the subgrid $\Delta_\Omega^\delta \subset \Delta_\Omega$ formed by all triangles where w is larger than the threshold value δ , i.e., $\Delta_\Omega^\delta := \{T_i \in \Delta_\Omega : w(T_i) \geq \delta\}$. We expect that a good compression scheme should preserve both the total *amount* of the weights from the original solution in Δ_Ω^δ and the information of *where* these weights are located inside the domain Ω . Also, we want this compression to be parsimonious, i.e. to store the least amount of information as possible. We test against these two requirements by proposing two metrics that measure: i) an information difference between the raw output of the *DMK-Solver* and the network extracted using our procedure, capturing the information of where the weights are located in space; ii) the amount of information needed to store the network.

Our first proposed metric relies in partitioning Ω in several subsets and then calculating the difference in the extracted network weights and the uncompressed output, *locally* within each subset. More precisely, we partition Ω into P non intersecting subsets $C_\alpha \subset \Omega$, with $\alpha = 1, \dots, P$ and $\cup_1^P C_\alpha = \Omega$. For example, we define $C_\alpha = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$, for x_i, x_{i+1}, y_j and y_{j+1} , consecutive elements of N -regular partitions of $[0, 1]$, and $P = (N - 1)^2$. Denote with $w_\delta(T_i)$ the weight on the triangle $T_i \in \Delta_\Omega^\delta$, resulting from the *DMK-Solver* (usually a function of μ^* and u^*). If we denote the *local weight* of Δ_Ω^δ inside C_α as $w_\alpha = \sum_{i: \mathbf{b}_i \in C_\alpha} w_\delta(T_i)$, then we propose the following evaluation metric:

$$\hat{w}_q(G) := \frac{1}{P} \left[\sum_{\alpha=1}^P \left(\sum_{e \in E} \mathbb{I}_\alpha(e) |w_e - w_\alpha| \right)^q \right]^{\frac{1}{q}}, \quad (9)$$

where $\mathbb{I}_\alpha(e)$ is an indicator of whether an edge $e = (i, j) \in E$ is inside an element C_α of the partition, i.e. $\mathbb{I}_\alpha(e) = 1, 0, 1/2$ if both $\mathbf{b}_i, \mathbf{b}_j$ are in C_α , none of them are, or only one of them is, respectively. In words, $\hat{w}_q(G)$ is a distance between the weights of the network extracted by our procedure and the original weights output of the *DMK-Solver*, over each of the local subsets C_α . This metric penalizes networks that either place large-weight edges where they were not present in the original triangulation, or low-weight ones where they were instead present originally. In this work we consider the Euclidean distance, i.e. $q = 2$, but other choices are also possible. Note that $\hat{w}_q(G)$ does not say anything about how much information was required to store the processed network. If we want to encourage parsimonious networks, i.e. networks with few redundant structures, then we should include in the evaluation the monitoring of $L(G) = \sum_{e \in E} \ell_e$, the total path length of the compressed network, where the edge length ℓ_e can be specified based on the application. Standard choices are uniform $\ell_e = 1, \forall e$ or the Euclidean distance between \mathbf{b}_i and \mathbf{b}_j . Intuitively, networks with small values of both $\hat{w}_q(G)$ and $L(G)$ are both accurate and parsimonious representations of the original DMK solutions defined on the triangulation.

We evaluate numerous graph extraction pipelines in terms of these two metrics on various routing optimization problem settings and parameters. In Fig. 4 we show the main results for a distribution of 170 networks obtained with $\beta \in \{1.1, 1.2, 1.3\}$ and $\beta_d = 1.1$. Similar results were obtained for other parameter settings. Networks are generated as follows: first, we choose a set of 5 different initial transport densities μ_0 , grouped in parabola-like, delta-like and uniform distributions, and a set of 12 different configurations for sources/sinks (mainly rectangles placed in different positions along the domain, see Supplementary Information for more details). Then, for each of these setup, we run our procedure: i) first the *DMK-Solver* calculates the solution of the continuous problems; ii) then we apply the *graph pre-extraction* procedure according to the rules of Sec. 2.1 and weights as in Sec. 2.2; iii) finally, we run the *graph filtering* step and consider various weight functions, as described in Fig. 4.

We observe that not applying the final filtering step and considering rule I with ER to build the graph (I-ER-None), the values of $\hat{w}_2(G)$ are smaller than other cases. This is expected as by filtering we remove information and thus achieve better performance with this metric when compared to no filtering. However, we pay a price in terms of total relative length as $L(G)/L_{max}$ is larger for this case. When working with rule II, we notice the appearance of many non-optimal small disconnected components and this effect deteriorates if filtering is activated. Corresponding statistics show low values for both $\hat{w}_2(G)$ and $L(G)/L_{max}$. We argue that this is because rule II produces, by construction, fewer redundant objects than rule I in the initial phase. This might have a similar effect as a filter but is done *a priori* during the pre-extraction, because rule II produces in this phase a limited number of effective neighbors. However, this comes at a price of higher variability with the sampled networks, as the variance of $\hat{w}_2(G)$ is higher than for the other combinations. Among the possibilities with filtering applied, we observe that rule I performs better than rule III, while all the weighting rules give a similar performance in terms of both metrics. Any combination involving rule I plus filtering has a similar performance as rule II in terms of both metrics but with smaller variability. Finally, these combinations perform differently in terms of the number of disconnected components (not

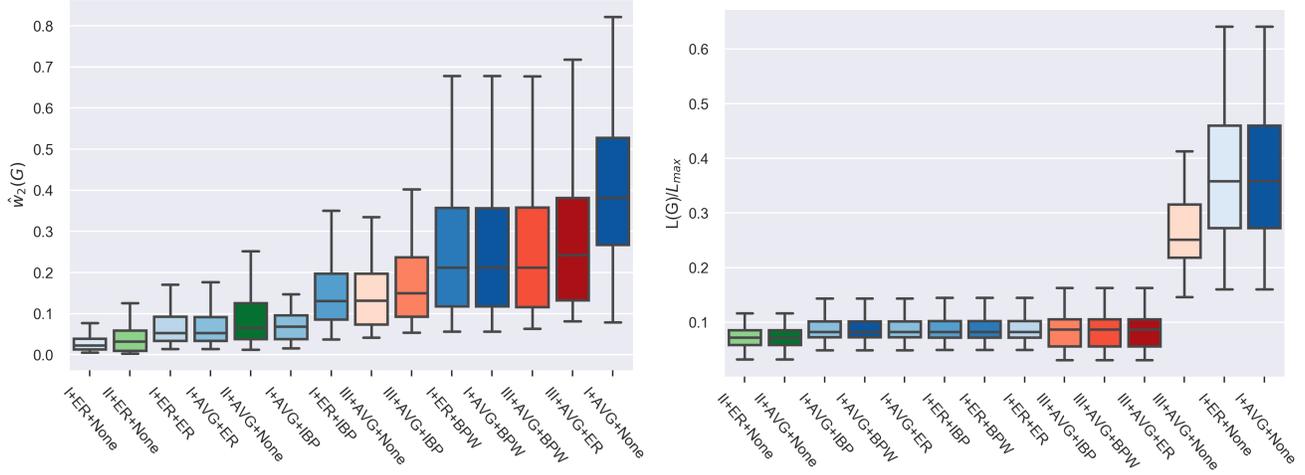


Figure 4. Graph extraction performance evaluation. We plot results for the different combinations of the graph extraction rule in terms of: (left) the metric $\hat{w}_2(G)$ of Eq. (9); (right) total network length $L(G)$ normalized by L_{max} , the max length over the 170 networks. Each bar denotes a possible combination as follows: roman numbers denote one of the three rules I-III (2.1); first label after the number denotes one of the rules to assign weights i-ii 2.2), which is applied to the output of the first step; the second (and last) label denotes the same rule but applied after the filtering step, “None” means that nothing is done, i.e. no filter applied, “IBP” means filter applied but with no reweighing, i.e. when an edge is removed by the filter we simply lose information without relocating its weight. Bars are color-coded so that rules I-III have three different primary colors and their corresponding routines have different shades of that main color. Here, we keep track of the conductivity μ and show medians and quartiles of a distribution over 170 networks generated with $\beta \in \{1.1, 1.2, 1.3\}$, $\beta_d = 1.1$ and $\delta = 0.01$.

shown here), with rule II producing more spurious splittings, as already mentioned. Depending on the application at hand, a practitioner should select one of these combinations based on their properties as discussed in this section. We give an example of a network generated with I-ER-ER in Fig. 5.

Application: network analysis of a vein network

We demonstrate our protocol on a biological network of fungi foraging for resources in space. The network structure corresponds to the fungi response to food cues while foraging⁵². Edges are veins or venules and connect adjacent nodes. This and those of other types of fungi are well known networks typically studied using image segmentation methods²⁸⁻³¹. It is thus interesting to compare results found by these techniques and by our approach, under the conjecture that the underlying dynamic driving the network structure could be the same as the optimality principles guiding our extraction pipeline. In particular, we are interested in analyzing the distribution $P(\ell)$ of the veins lengths, i.e. the network edges. The benchmark $P(\ell)$ distribution obtained by Baumgarten and Hauser²⁸ using image processing techniques is an exponential of the type $P(\ell) = P_0 e^{-\gamma\ell}$. Accordingly, as shown in Fig. 6, we find that an exponential fit (with values $P_0 = 234.00$, $\gamma = 36.32$) well captures the left part of the distribution, i.e. short edges. Differences between fit and observed data can be seen in the right-most tail of the graph, corresponding to longer path lengths, where the data decay faster than the fit. However, we find that the exponential fit is nevertheless better than other distributions, such as the gamma and log-normal proposed in Dirnberger and Mehlhorn⁵³ for the *P. polycephalum*. Drawing definite quantitative conclusions is beyond the scope of our work, as this example aims at a qualitative illustration of possible applications that can be addressed with our model. In general, however, it seems not possible to choose a single distribution that well fits both center and tails of the distribution for various datasets of this type⁵³.

To conclude, we demonstrate the flexibility of our graph extraction method on a more general input than the one extracted from *DMK-Solver*. Specifically, we consider as example an image of *P. polycephalum* taken from data publicly available in the Slime Mold Graph Repository (SMGR) repository⁵⁴. We first downsample an image of the SMGR’s *KIST Europe data set*, using *OpenCV* (left) and a color scale defined on the pixels as an artificial μ^* function. We build a graph using the *graph pre-extraction* and *graph filtering* steps as shown in Fig. 7. Notice that our protocol in its standard settings with filtering can only generate tree-like structures. Therefore, if we want to obtain a network with loops as we did in Fig. 7, we should consider a modification of our routine, which can be done in a fully automatized way, as explained in more details in the Supplementary S4. In short, after the *graph pre-extraction* step, where loops are still present, we extract a tree-like structure close to the original loopy graph and give this in input to the filtering. We can then add *a posteriori* edges that connect terminals that were

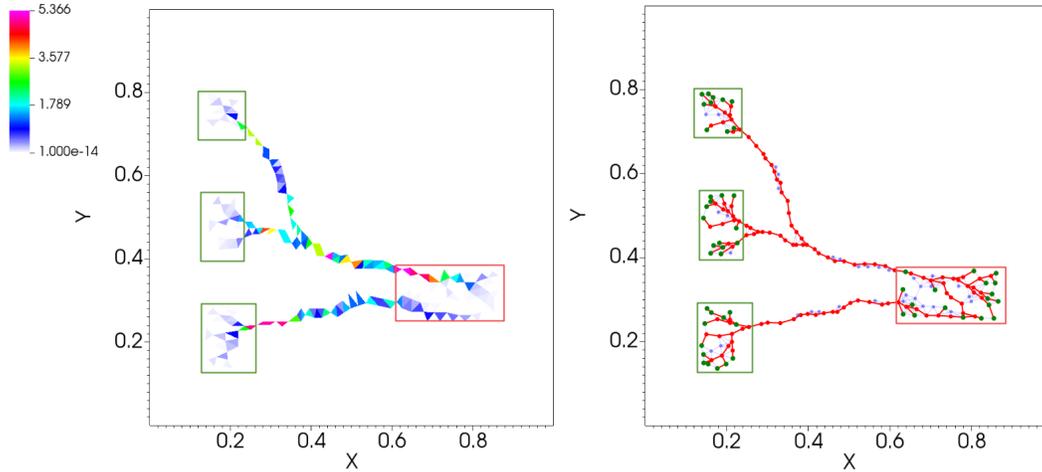


Figure 5. Network extraction example. We show a network generated from a routing optimization problem with parameters $\beta_c = 1.4$, $\beta_d = 1.3$ and $\delta = 0.001$; (left) raw output of the *DMK-Solver*; (right) final network extracted using the routine *I-ER-ER*.

close by in the graph obtained from the pre-extraction step but removed by the filter, thus recovering loops. In case obtaining loops is not required, our routine can be used with no modifications. Adapting our filtering model to allow for loopy structures in a principled way, analogously to what done in Sec. 3, will be subject of future work.

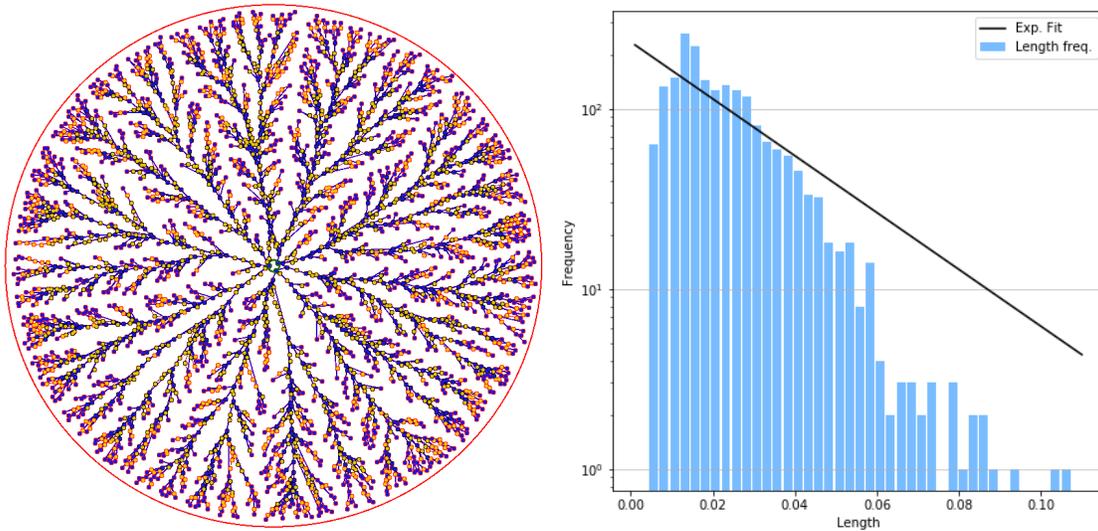


Figure 6. Application to fungi network. We generate a synthetic network similar to the image of Fig. 1a reported in Boddy et al.³¹ and Fig. 4a in Obara et al.²⁹ for the of *Phanerochaete velutina* fungus⁵² and Fig. 1 in their supplementary for the *Coprinus picaceus*. Fitted parameters are: $P_0 = 234.00$, $\gamma = 36.32$. Here $f^+(x, y) = 1$, if $(x - 0.5)^2 + (y - 0.5)^2 \leq 0.01$; $f^+(x, y) = 0$, otherwise; $f^-(x, y) = -1$, if $0.01 < (x - 0.5)^2 + (y - 0.5)^2 \leq 0.45$; $f^-(x, y) = 0$, otherwise. The network on the left corresponds to the filtered graph. Yellow nodes are degree-2 nodes that we omitted when computing the length distribution. Green and red outlines are used to denote nodes in S^+ and S^- , respectively.

Discussion

We propose a graph extraction method for processing raw solutions of routing optimization problems in continuous space into interpretable network topologies. The goal is to provide a valuable tool to help practitioners bridging the gap between abstract mathematical principles behind optimal transport theory and more interpretable and concrete principles of network theory. While the underlying routing optimization scheme behind the first step of our routine uses recent advances of optimal transport theory, our tool enables automatic graph extraction without requiring expert knowledge. We purposely provide a flexible routine

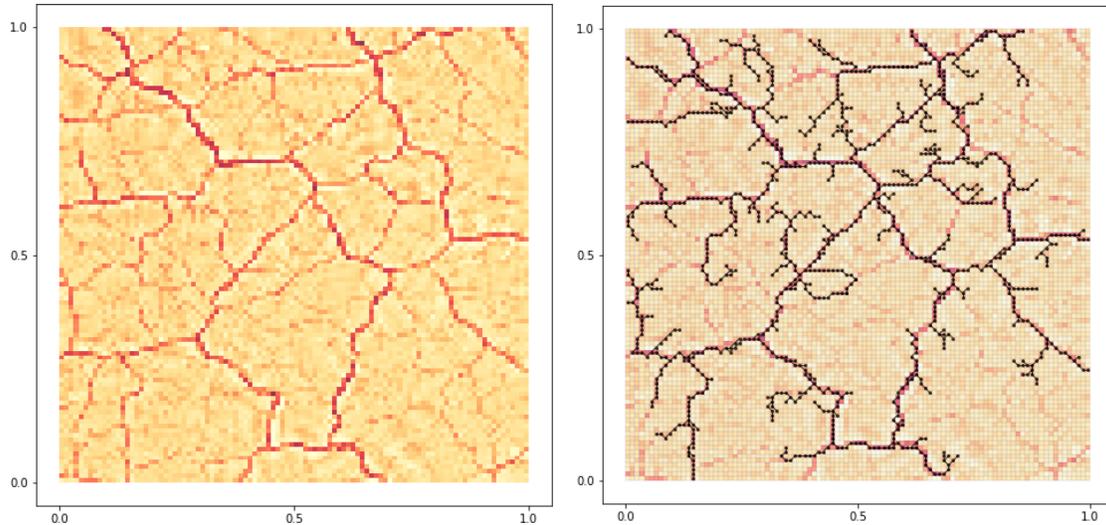


Figure 7. Application to images. We take an image of the *P. Polycephalum* from the SMGR repository⁵⁴. The picture used is a 1200x1200-pixel section of an original image of size 5184x3456 pixels (see Supplementary S4 for details) and extract a network with step 2 and 3 of our protocol. As a pre-processing step, we downsample the image using *OpenCV* (left) and use the color scale defined on the pixels as an artificial μ^* function. Using this information and the grid structure associated to the image's pixels, we first build a graph G with the *graph pre-extraction* step described in Sec. 2; then, we obtain a graph G_f (right) using the *graph filtering* step of Sec. 3, for an appropriate selection of sources and sinks, and adding a correction to retrieve loops. Notice that our protocol in its standard settings with filtering can only generate tree-like structures. Therefore, if we want to obtain a network with loops, we should consider a minor modification of our routine, which can be done in a fully automatized way, as explained in more details in the Supplementary S4.

for graph extraction so that it can be easily adapted to serve the specific needs of practitioners from a wider interdisciplinary audience. We thus encourage users to choose the parameters and details of the subroutines to suitably customize the protocol based on the application of interest. To help guiding this choice, we provide several examples here and in the Supplementary Information. We anticipate that this work will find applications beyond that of automating graph extraction from routing optimization problems. We remark that two of the three steps of our protocol apply to inputs that might not necessarily come from solutions of routing optimization. Indeed, the pipeline can be applied to any image setting where an underlying network needs to be extracted. The advantage of our setting with respect to more conventional machine learning methods is that the final structure extracted with our approach minimizes a clearly defined energy functional, that can be interpreted as the combination of the total dissipated energy during transport and the cost of building the transport infrastructure. We foresee that this minimizing interpretation together with the simplification of the pipeline from abstract modeling to final concrete network outputs will foster cross-breeding between fields as our tool will inform network science with optimal transport principles and vice-versa. In addition, we expect to advance the field of network science by promoting the creation of new network databases related to routing optimization problems.

References

1. Banavar, J. R., Colaiori, F., Flammini, A., Maritan, A. & Rinaldo, A. Topology of the fittest transportation network. *Phys. Rev. Lett.* **84**, 4745 (2000).
2. Corson, F. Fluctuations and redundancy in optimal transport networks. *Phys. Rev. Lett.* **104**, 048703 (2010).
3. Li, G. *et al.* Towards design principles for optimal transport networks. *Phys. review letters* **104**, 018701 (2010).
4. Yeung, C. H., Saad, D. & Wong, K. M. From the physics of interacting polymers to optimizing routes on the london underground. *Proc. Natl. Acad. Sci.* **110**, 13717–13722 (2013).
5. Guimerà, R., Díaz-Guilera, A., Vega-Redondo, F., Cabrales, A. & Arenas, A. Optimal network topologies for local search with congestion. *Phys. review letters* **89**, 248701 (2002).
6. Donetti, L., Hurtado, P. I. & Munoz, M. A. Entangled networks, synchronization, and optimal network topology. *Phys. Rev. Lett.* **95**, 188701 (2005).

7. Ronellenfitsch, H., Dunkel, J. & Wilczek, M. Optimal noise-canceling networks. *Phys. review letters* **121**, 208301 (2018).
8. Gazit, Y., Berk, D. A., Leunig, M., Baxter, L. T. & Jain, R. K. Scale-invariant behavior and vascular network formation in normal and tumor tissue. *Phys. review letters* **75**, 2428 (1995).
9. Garlaschelli, D., Caldarelli, G. & Pietronero, L. Universal scaling relations in food webs. *Nature* **423**, 165 (2003).
10. Balister, P. *et al.* River landscapes and optimal channel networks. *Proc. Natl. Acad. Sci.* **115**, 6548–6553 (2018).
11. Santambrogio, F. Optimal channel networks, landscape function and branched transport. *Interfaces Free. Boundaries* **9**, 149–169 (2007).
12. Katifori, E., Szöllösi, G. J. & Magnasco, M. O. Damage and fluctuations induce loops in optimal transport networks. *Phys. Rev. Lett.* **104**, 048704 (2010).
13. Santambrogio, F. Optimal transport for applied mathematicians. *Birkäuser, NY* **55**, 58–63 (2015).
14. Messinger, S. M., Mott, K. A. & Peak, D. Task-performing dynamics in irregular, biomimetic networks. *Complexity* **12**, 14–21 (2007).
15. Fruttiger, M. Development of the mouse retinal vasculature: angiogenesis versus vasculogenesis. *Investig. ophthalmology & visual science* **43**, 522–527 (2002).
16. Schaffer, C. B. *et al.* Two-photon imaging of cortical surface microvessels reveals a robust redistribution in blood flow after vascular occlusion. *PLoS biology* **4**, e22 (2006).
17. Altarelli, F., Braunstein, A., Dall’Asta, L., De Bacco, C. & Franz, S. The edge-disjoint path problem on random graphs by message-passing. *PloS one* **10**, e0145222 (2015).
18. Bayati, M. *et al.* Statistical mechanics of steiner trees. *Phys. review letters* **101**, 037208 (2008).
19. De Bacco, C., Franz, S., Saad, D. & Yeung, C. H. Shortest node-disjoint paths on random graphs. *J. Stat. Mech. Theory Exp.* **2014**, P07009 (2014).
20. Braunstein, A. & Muntoni, A. P. The cavity approach for steiner trees packing problems. *J. Stat. Mech. Theory Exp.* **2018**, 123401 (2018).
21. Ronellenfitsch, H. & Katifori, E. Global optimization, local adaptation, and the role of growth in distribution networks. *Phys Rev Lett* **117**, H364–5 (2016).
22. Brancolini, A. & Solimini, S. Fractal regularity results on optimal irrigation patterns. *J. de Mathématiques Pures et Appliquées* **102**, 854–890 (2014).
23. Xia, Q. Motivations, ideas and applications of ramified optimal transportation. *ESAIM Math. Model. Numer. Anal.* **49**, 1791–1832 (2015).
24. Pegon, P., Santambrogio, F. & Xia, Q. A fractal shape optimization problem in branched transport. *J. de Mathématiques Pures et Appliquées* **123**, 244–269 (2019).
25. Facca, E., Cardin, F. & Putti, M. Towards a stationary Monge–Kantorovich dynamics: The physarum polycephalum experience. *SIAM J. on Appl. Math.* **78**, 651–676 (2018).
26. Facca, E., Daneri, S., Cardin, F. & Putti, M. Numerical solution of Monge–Kantorovich equations via a dynamic formulation. *J Sci Comput.* **82**, 1–26 (2020).
27. Facca, E. & Cardin, F. Branching structures emerging from a continuous optimal transport model. *J Comput. Phys* **Submitted** (2020).
28. Baumgarten, W. & Hauser, M. Detection, extraction, and analysis of the vein network. *J. Comput. Interdiscip. Sci.* **1**, 241–249 (2010).
29. Obara, B., Grau, V. & Fricker, M. D. A bioimage informatics approach to automatically extract complex fungal networks. *Bioinformatics* **28**, 2374–2381 (2012).
30. Bebber, D. P., Hynes, J., Darrah, P. R., Boddy, L. & Fricker, M. D. Biological solutions to transport network design. *Proc. Royal Soc. B: Biol. Sci.* **274**, 2307–2315 (2007).
31. Boddy, L., Wood, J., Redman, E., Hynes, J. & Fricker, M. D. Fungal network responses to grazing. *Fungal Genet. Biol.* **47**, 522–530 (2010).
32. Dehkordi, M. T., Sadri, S. & Doosthoseini, A. A review of coronary vessel segmentation algorithms. *J. medical signals sensors* **1**, 49 (2011).

33. Bradski, G. & Kaehler, A. *Learning OpenCV: Computer vision with the OpenCV library* (" O'Reilly Media, Inc.", 2008).
34. Dirnberger, M., Kehl, T. & Neumann, A. Nefi: Network extraction from images. *Sci. reports* **5**, 15669 (2015).
35. Chai, D., Forstner, W. & Lafarge, F. Recovering line-networks in images by junction-point processes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1894–1901 (2013).
36. Wang, J., Song, J., Chen, M. & Yang, Z. Road network extraction: A neural-dynamic framework based on deep learning and a finite state machine. *Int. J. Remote. Sens.* **36**, 3144–3169 (2015).
37. Wegner, J. D., Montoya-Zegarra, J. A. & Schindler, K. Road networks as collections of minimum cost paths. *ISPRS J. Photogramm. Remote. Sens.* **108**, 128–137 (2015).
38. Hagberg, A., Swart, P. & S Chult, D. Exploring network structure, dynamics, and function using networkx. Tech. Rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2008).
39. Xu, K., Tang, C., Tang, R., Ali, G. & Zhu, J. A comparative study of six software packages for complex network research. In *2010 Second International Conference on Communication Software and Networks*, 350–354 (IEEE, 2010).
40. Batagelj, V. & Mrvar, A. Pajek-program for large network analysis. *Connections* **21**, 47–57 (1998).
41. Bastian, M., Heymann, S. & Jacomy, M. Gephi: an open source software for exploring and manipulating networks. In *Third international AAAI conference on weblogs and social media* (2009).
42. Evans, L. C. & Gangbo, W. *Differential equations methods for the Monge-Kantorovich mass transfer problem*. 653 (American Mathematical Soc., 1999).
43. Xia, Q. On landscape functions associated with transport paths. *Discret. Contin. Dyn. Syst* **34**, 1683–1700 (2014).
44. With hydraulic interpretation, we can think of the edges as pipes, small cylindrical channels where the mass is passing through, and the capacity is proportional to the size of the pipe diameter.
45. Tero, A., Kobayashi, R. & Nakagaki, T. A mathematical model for adaptive transport network in path finding by true slime mold. *J. theoretical biology* **244**, 553–564 (2007).
46. Mesh-related parameters specify how the mesh for the discretization of space is built. specifically, we could specify *ndiv*, the number of divisions in the *x* axis and *nref*, the number of refinements, i.e. the number of times each triangle on the grid generated by a specific *ndiv* is subdivided into 4 triangles.
47. In this work we focus on a 2d space, but the procedure can be generalized to 3d.
48. Oliveira, C. A. & Pardalos, P. M. *Mathematical aspects of network routing optimization* (Springer, 2011).
49. Facca, E., Cardin, F. & Putti, M. Physarum dynamics and optimal transport for basis pursuit. *arXiv preprint arXiv:1812.11782* (2018).
50. Straszak, D. & Vishnoi, N. K. Irls and slime mold: Equivalence and convergence. *arXiv preprint arXiv:1601.02712* (2016).
51. Due to the high graph connectivity, degree centrality is not appropriate for selecting these ending parts.
52. Fricker, M., Boddy, L. & Bebbber, D. Network organisation of mycelial fungi. In *Biology of the fungal cell*, 309–330 (Springer, 2007).
53. Dirnberger, M. & Mehlhorn, K. Characterizing networks formed by p. polycephalum. *J. Phys. D: Appl. Phys.* **50**, 224002 (2017).
54. Dirnberger, M., Mehlhorn, K. & Mehlhorn, T. Introducing the slime mold graph repository. *J. Phys. D: Appl. Phys.* **50**, 264001 (2017).

Acknowledgements

The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Diego Baptista and Daniela Leite.

Additional information

Accession codes: open source codes and executables are available at <https://github.com/Danielaleite/Nexttrout>.

Supporting Information (SI)

S1 Examples of *graph pre-extraction* routines

We provide here several examples of networks resulting from different routing optimization setup for each of the three graph definitions.

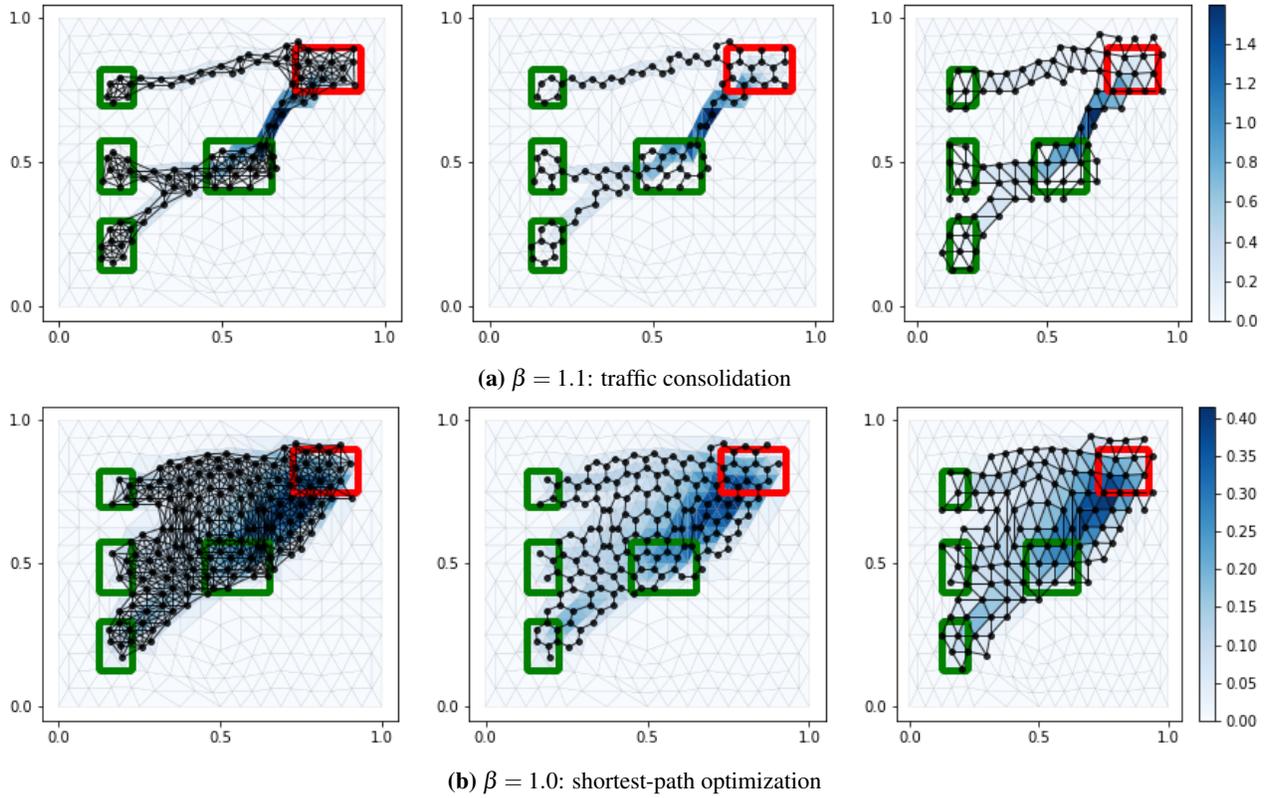
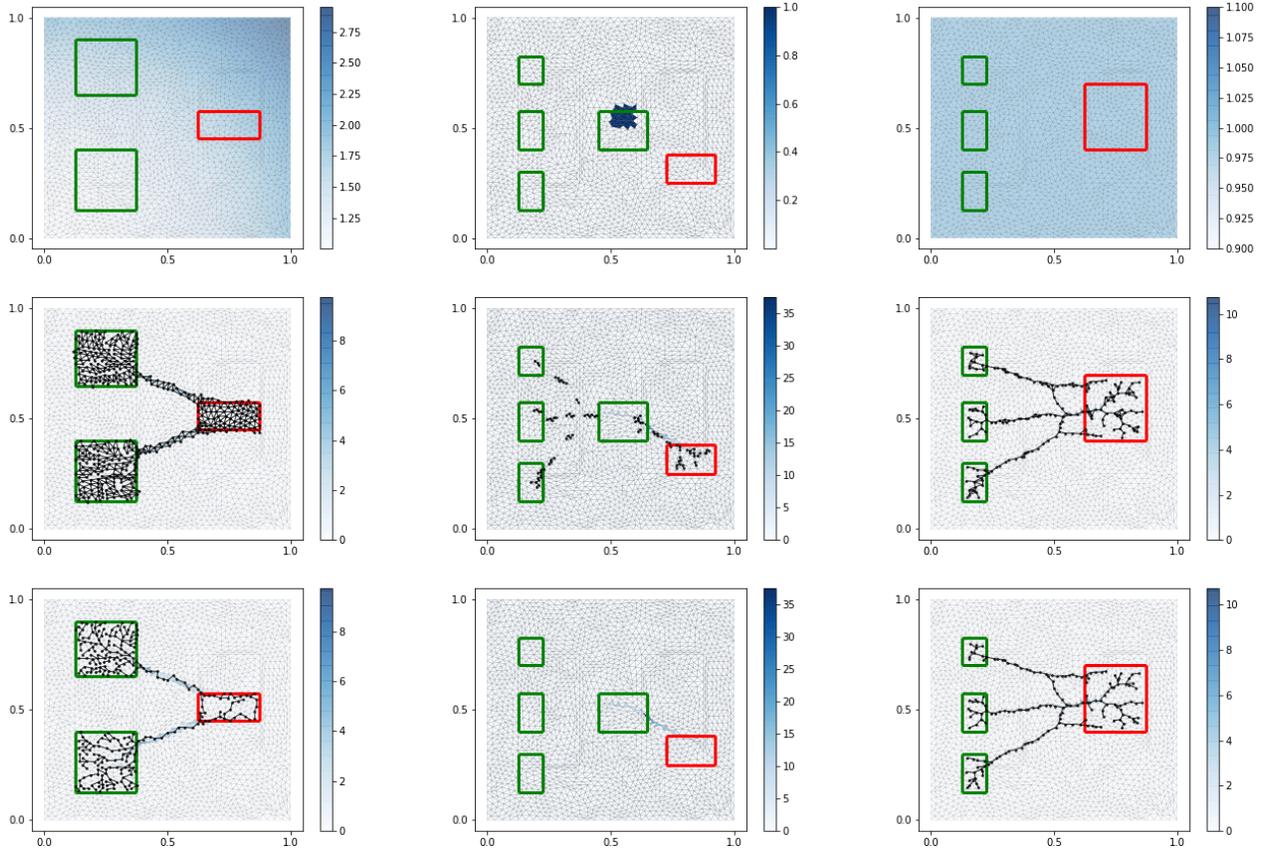


Figure S1. *Graph pre-extraction* rules for three transportation optimization problems. Left): edge-or-node sharing; center): edge-only sharing; right): original triangulation. We monitor the density μ and use parameters: $\mu_0 = 1$, $f = 5\text{rch}$, $\delta = 0.0001$, $w_{ij} = \text{ER}$.

S2 Examples networks and routing optimization scenarios

We provide here several examples of networks resulting from different routing optimization setups for several choices of our proposed routines. This should serve as an example guideline on what parameters to choose based on the application. More specifically, we show three different setups given in input as initial routing optimization problems to the *DMK-Solver*. We consider: i) different locations of sources and sinks to show how this impacts the formation of branches and their symmetry; ii) different values of $\beta \in \{1.2, 1.3, 1.5\}$ to show how traffic consolidates into fewer edges as β grows; iii) different initial $\mu_0(x, y)$ (parabolic, delta-like and uniform) to show variability in the initial transport density.



(a) Example 1: parabolic transport density, (b) Example 2: delta-like transport density, (c) Example 3: uniform transport density, $\beta = 1.2$ $\beta = 1.3$ $\beta = 1.5$

Figure S2. Network extraction examples for branched transportation optimization. Columns denote a particular setup based on source/sink locations and values of β and μ_0 as described in the sub-captions. Each row represents a step of our protocol: top) initial transport density $\mu_0(x, y)$ given in input to *DMK-Solver*; center) graph *pre-extraction* using III-AVG (left), II-AVG (center) and I-AVG (right); bottom) graph *filtering* with weight assigned with rule IBP (left and right) and no simplification (center). The color scheme refers to the value of μ . Sources and sinks are points inside the green and red rectangles, respectively. Note that pre-extraction rule II tends to break the network into disconnected components. This is also the reason why this rule scores well in terms of total path length (see Fig. 4 in the main manuscript). Since the path is already broken and sources cannot reach the sinks, it does not make sense to apply the filtering, hence we left the center-bottom plot empty.

S3 Source and sink selection

Selecting source and sink is an important task during the graph filtering step as explained in the main manuscript Sec. 3.2. Here we give examples showing why this is the case. Specifically, for a particular problem setup, we consider four cases: i) no selection of sources and sinks. This means that we simply consider sources and sinks *all* the points inside the support of the forcing function $f(x)$, input of the *DMK-Solver*. In this case, the final network structure is dominated by the many branches connecting the sources and sinks within the support of $f(x)$, thus hindering the contribution of the bulk of the network, the one connecting sources and sinks; ii) convex hull-only: we set as source (or sink) all the points inside the convex hull of the set of eligible sources (or sinks); iii) betweenness centrality-only: we set as source (or sink) all the eligible sources (or sinks) that have betweenness centrality smaller than a threshold $0 \leq \tau_{BC} \leq 1$. These two criteria miss possibly relevant sub-branches, as shown in Fig. S3 top-right and bottom-left; iv) selection using both convex-hull and betweenness centrality criteria. In this case, we capture the relevant sub-branches from both criteria, as shown in the bottom-right of Fig. S3. In the main manuscript we always consider this final criterion as it seems to capture all the relevant branches inside the support of $f(x)$. However, we encourage the final user to choose what criteria to choose based on the application. Figure S3 should help driving this decision.

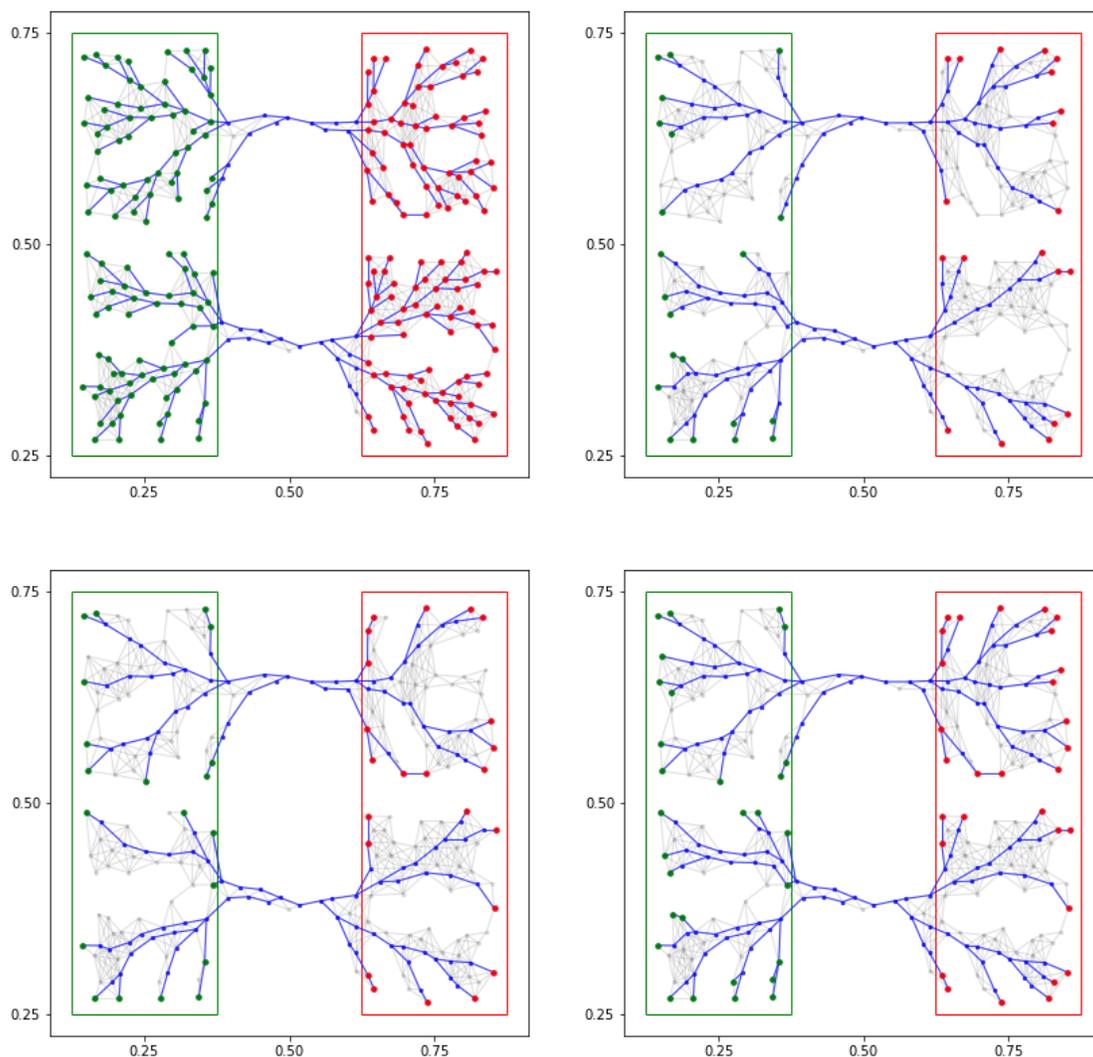


Figure S3. Choosing sources and sinks. Top-left) no selection of sources and sink ($\tau_{BC} = 1$); top-right) selection using betweenness centrality only ($\tau_{BC} = 0.01$); bottom-left) selection using convex hull-only ($\tau_{BC} = -1$); bottom-right) selection with convex-hull and betweenness centrality criteria ($\tau_{BC} = 0.01$).

S4 Network extraction from image

We describe here the steps followed to get the network shown in Fig. 7 in the main manuscript¹. The original figure contains loops, however our protocol in its standard settings with filtering can only generate tree-like structures. Therefore, if we want to obtain a network with loops, we should consider a minor modification of our routine, as explained in detail below. In short, we need to find first a tree-like network close to the original image with loops and then give this one in input to our routine. Then, after applying our protocol, we can obtain loops by adding edges that properly *close* the loops in the tree. This can all be done in an automatized way as explained below. Otherwise, if obtaining loops is not required, our routine can be used with no modifications. The whole procedure can be divided into 6 parts: image selection and approximation, graph pre-extraction, tree reduction, terminal identification, graph filtering and edge correction.

1. Image selection and approximation: this consists in choosing an image and, if desired, mapping it into a reduced version of it. This reduction is useful when dealing with high-resolution images. We use the *resize* function included in *OpenCV*. The pixel values for the reduction image can be computed using different interpolation methods. We use linear interpolation (*cv.INTER_NEAREST*). The original image's dimensions are 1200x1200. The reduced ones are 100x100.
2. Graph pre-extraction: the RGB values of the pixels are mapped into an integer in a one-to-one way, and then they are used to define the function μ^* on each pixel. The graph G is obtained using rule I-ER as defined in Sec. 2 of the main manuscript.
3. Tree reduction: the filtering could be applied directly on G after choosing some sources and sinks, but this will generate a filtered tree-like network, i.e., no loops from the image will be captured. Thus, we propose to change G by a tree graph whose structure could then be easily "corrected" (after the filtering step) to get the mentioned loops. This tree is taken to be the *Breadth First Search* graph (G_{BFS}) of G from a random root.
4. Terminal identification: terminals (the union of sources and sinks) are defined using *filters*, i.e., squared pieces of the image. We place terminals in the graph by counting how many intersections are between the image and the boundary of the filter: if just one intersection is found, then a terminal is placed in the node with the lowest closeness centrality; if three or more intersections are found, then a terminal is placed in the node with the highest closeness centrality. We cover the whole image by disjoint filters, thus we do not miss the relevant parts of the image. Notice that if some parts of the image do not have a representative in one of the two source or sink sets, then they will not be part of the filtered graph.
5. Graph filtering: G_{BFS} is filtered as in Sec. 3 of the main manuscript (to obtain a graph G_f) by defining the source set (S^+) to be a random element on the set of terminals and the sink set (S^-) to be the remaining ones; $\beta_d = 1$ in Fig 7.
6. Edge correction: to *close* the loops we use the following rule. For each pair of terminals $u, v \in G_f$, if the shortest path P_{BFS}^{uv} on G_{BFS} is *long* but the shortest path P^{uv} on G is *short*, then add P^{uv} to G_f . The notions of *short* and *long* paths are precisely defined by introducing two parameters L_{BFS} and L_G , respectively. We say that a path P is *long*, if $l(P) > L_{BFS}$; and it is *short*, if $l(P) < L_G$, where l is the number of nodes in the path P ; the values of L_{BFS} and L_G can be tuned based on the system size at hand.

¹We used image "IMG_0379.jpg" stored at KIST_Europe_data _set/raw_images/ motion16/ inside the SMGR repository⁵⁴. The dimension of the original one is 5184x3456 pixels. We used a 1200x1200-pixel section in Fig. 7. The coordinates of the bottom-left vertex of the box are (2000,2000) (assuming (1,1) to be the bottom-left pixel of the original image).