Adaptive Node Positioning in Biological Transport Networks

Albert Alonso,[1, *] Lars Erik J. Skjegstad,[1, *] and Julius B. Kirkegaard[1, 2, †]

[1]*Niels Bohr Institute, University of Copenhagen, Denmark*
[2]*Department of Computer Science, University of Copenhagen, Denmark*
(Dated: November 4, 2024)

Biological transport networks are highly optimized structures that ensure power-efficient distribution of fluids across various domains, including animal vasculature and plant venation. Theoretically, these networks can be described as space-embedded graphs, and rich structures that align well with observations emerge from optimizing their hydrodynamic energy dissipation. Studies on these models typically use regular grids and focus solely on edge width optimization. Here, we present a generalization of the hydrodynamic graph model which permits additional optimization of node positioning. We achieve this by defining sink regions, accounting for the energy dissipation of delivery within these areas, and optimizing by means of differentiable physics. In the context of leaf venation patterns, our method results in organic networks that adapt to irregularities of boundaries and node misalignment, as well as overall improved efficiency. We study the dependency of the emergent network structures on the capillary delivery conductivity and identify a phase transition in which the network collapses below a critical threshold. Our findings provide insights into the early formation of biological systems and the efficient construction of transport networks.

Transport networks are ubiquitous in nature and in living systems. The efficiency of such networks is crucial for the evolutionary fitness of organisms such as those observed in leaf venation [1–3] and blood vasculature systems [3–5], and emerges in complex systems such as river networks [6] and human transport systems [7–9]. Understanding the structure and morphogenesis of such network structures has been facilitated by studying energetically optimal solutions to static, hydrodynamic networks [10–13].

The optimization of hydrodynamic transport networks is traditionally approached as an edge-optimization problem [2, 5, 10, 13], assuming systems where network nodes serve as sinks for their local area. Thus, optimizing the energy dissipation leads to optimal edge conductivities [10], while the positions of the nodes themselves are considered fixed. In principle, this approach can be used to model any bounded system if enough nodes are used. However, for a finite number of nodes confined within a bounded system, it is evident that the node placement itself influences the optimality of the fluid delivery system. In many systems, the finite number of nodes is a physical fact and must be imposed, e.g., due to a lower bound on the vein thicknesses determined by the capillary size.

Here, we consider bounded systems of finite nodes and generalize the hydrodynamic model to have well-defined optima both in edge conductivities and node positioning. Specifically, we model a bounded leaf venation network and let the boundary represent the leaf margin. Each node represents a source or sink (we set a single node at the leaf base as the source, and let all other nodes be sinks) and each edge represents a vein between two nodes. Fig. 1a shows the optimal solution obtained for a fixed, hexagonal grid in a bounded system. The imposition of the domain boundary and the inability of the nodes to adapt positionally results in non-uniform areas associ-

ated with each node. As fluid dissipation will typically be proportional to area, this thus implies that the sinks become non-uniform. By removing the spatial constraint on the node positions, we find more energy-efficient solutions, which yield networks that appear organic and more consistent with networks observed in nature (Fig. 1b).

The power of the transport network is given by [10]

$$P_{\mathrm{t}} = \sum_{e \in \mathrm{edges}} \left(\frac{F_e^2}{C_e} + c_t\, C_e^\gamma \right) L_e, \qquad (1)$$

where F_e is the flux associated with edge e, which has length L_e and conductivity C_e. The constant c_t defines the metabolic cost of maintaining the edge, which is equivalent to considering a system with a finite amount of resources. γ is a parameter that determines how the material cost scales with the conductivities, which we set to $\gamma = 1/2$ in this Letter. The flow is pressure-driven,
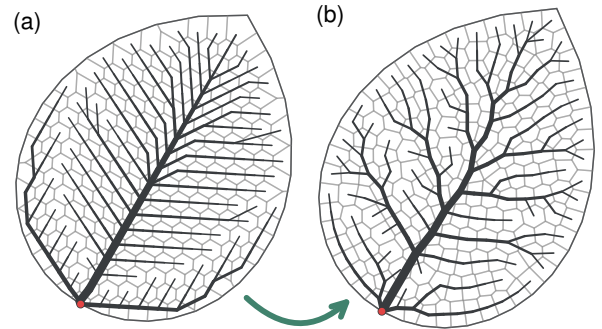

FIG. 1. Optimization and relaxation on an inclined leaf ($\varphi_0 = 60°$, $\beta = 0.4$, $N = 175$). Non-suppressed edges and delivery regions of each sink node are shown. The source is marked with a red dot. (a) Result after transport network optimization solving Eq. (3) on a hexagonal grid. (b) Result after relaxation, where the network optimizes $P = P_{\mathrm{t}} + P_{\mathrm{d}}$.

such that the flux over a (directed) edge $e = i \to j$ is given by $F_e = (p_i - p_j)C_e/L_e$. The pressures are indirectly determined by the need to satisfy Kirchhoff's law $\sum_{e \in i} \pm_e F_e = s_i$, where s_i is the source/sink at node i and the sign is set by the direction of the edge. This can be solved efficiently with hardware acceleration [14]. Typically, the sink magnitudes are assumed to be equal. However, once the nodes are permitted to move, these values must change as well. Thus, taking the source $s_0 = 1$, we define $s_i = -A_i/\sum A_i$ ($i \geq 1$), where A_i is the area surrounding a node, implicitly defined by the associated Voronoi cell.

In order for a transport system to have stable and well-defined optima in node positioning, the above model must be expanded to account for the power dissipation within the Voronoi cells. To achieve this, we add a simple power-delivery term, P_{d}, to the conventional formulation for transport power [10], and thus assume that the total power can be described by the sum of two contributions $P = P_{\mathrm{t}} + P_{\mathrm{d}}$. With the goal of defining a self-consistent model that equally considers both transport and delivery costs, we take

$$P_{\mathrm{d}} = \sum_{i \in \mathrm{sinks}} \frac{s_i^2}{\hat{C}_i} \langle \ell \rangle_i. \tag{2}$$

This is analogous to the first term in the transport formulation (Eq. (1)), but considers the power dissipation due to the delivery of the sink fluid s_i over an average Voronoi distance $\langle \ell \rangle_i$. $\hat{C}_i$ is the delivery conductivity, which effectively models a capillary system. Physically, this term favors equally sized and isotropic sink areas. We assume that the delivery conductivity is an intrinsic property of the material, such that $\hat{C}_i = \hat{C}$ is fixed and equal for all nodes.

Optimization — Our scheme to minimize P consists of the combination of an edge conductivity optimization and a node relaxation process. We employ a fully connected graph and initialize all edge conductivities to the same value. For a given node positioning $\{\boldsymbol{x}_i\}$, we optimize the conductivities by using an adaptation model [2, 13]

$$\frac{\mathrm{d}C_e}{\mathrm{d}t} = \left(\frac{F_e^2}{C_e^{\gamma+1}} - \gamma c_t \right) C_e + c_0 e^{-\lambda t}, \tag{3}$$

where the term $F_e^2/C_e^{\gamma+1}$ is the squared wall shear stress, and γc_t represents the optimal squared shear stress. Furthermore, we include a growth term characterized by the area growth rate λ, which has proven to be a robust strategy to achieve better optima [2]. To facilitate smooth convergence, we also decrease exponentially the magnitude of the growth term $c_0(\tau) \propto e^{-\nu\tau}$ during the node positioning *relaxation process*. Here, we use τ to denote the timescale of node relaxation and t for the conductivity optimization.
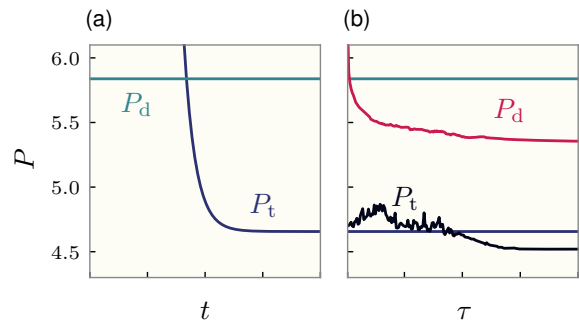

FIG. 2. Transport and delivery power terms (Eqs. (1) and (2)) during optimization. (a) Optimization of P_{t} by using Eq (3). (b) Node relaxation process. At each timestep one complete optimization of P_{t} is performed, resulting in an improvement of both power terms.

Crucially, even though our transport network is fully connected, optimization results in a sparse, planar graph (Fig. 1). While networks with static nodes can simplify the incidence matrix by, for instance, employing Delaunay triangulation to account for sparse connections, we rely on the derivative information of the entire graph during the optimization to accurately determine the correct gradients for node displacement and to allow for the spontaneous creation and suppression of connections.

To optimize over $\{\boldsymbol{x}_i\}$, we use gradient descent computed by automatic differentiation techniques [15, 16]. To propagate the gradients across the solution of the adaptation model in Eq. (3), we employ a custom back-propagation method based on the implicit function theorem [17], detailed in [18]. Importantly, the fact that we use Voronoi cells to define the sink magnitudes s_i and the delivery distances $\langle \ell \rangle_i$ means that all terms vary continuously with node positions, yet it requires the ability to differentiate through Voronoi calculations. We achieve this with a custom differentiable implementation of Voronoi tessellation [19–21], clipped by the domain boundary [22] (see [18] for details). Crucially, this procedure is parallelizable and amenable to hardware acceleration. As Voronoi cells are guaranteed to be convex, we can split the integrals over these cells into triangles T, e.g.,

$$\langle \ell \rangle_i = \frac{1}{A_i} \sum_{T \in \mathrm{vor}_i} \iint_T \|\boldsymbol{x}_i - \boldsymbol{x}\| \, \mathrm{d}A, \tag{4}$$

over which the integrals can be evaluated analytically (see [18]). We note that our optimization schemes are local optimizers and will generally not find global optima.

During the optimization of conductivities, delivery cost remains fixed (Fig. 2a). On the other hand, during node placement relaxation, we see that while delivery costs are monotonically decreasing, transport costs experience a phase of stochasticity where the efficiency is

reduced (Fig. 2b). Nevertheless, the total power continuously decreases, and after some iterations both the transport power P_t and the delivery power P_d reach lower values than those achievable in the regular grid, showing that the terms are not opposing criteria and that the simultaneous optimization yields networks that show increased power efficiency in both.

Network stability — The delivery term P_d depends on the capillary conductivity $\hat{C}$. To understand its influence on the network dynamics, we consider a one-dimensional system, where the nodes are solely connected to their adjacent nodes. This removes the need for Voronoi calculations, where instead the average delivery distance is

$$\langle \ell \rangle_i = \frac{1}{2(x_R - x_L)} \left[(x_R - x_i)^2 + (x_i - x_L)^2 \right], \quad (5)$$

where x_L and x_R are the midpoints between the left and right neighbor, respectively (see [18]). Figs. 3a–b show the mean node separation $\langle d \rangle$ in this 1D system confined to a fixed domain for increasing values of $\hat{C}$. We observe a smooth phase transition: When $\hat{C}$ is small, the energy is largely dominated by the delivery energy, which is minimal when the nodes are uniformly spread. As we increase the delivery conductivity of the system, we observe a continuous change in the network until the network collapses (Fig. 3b).
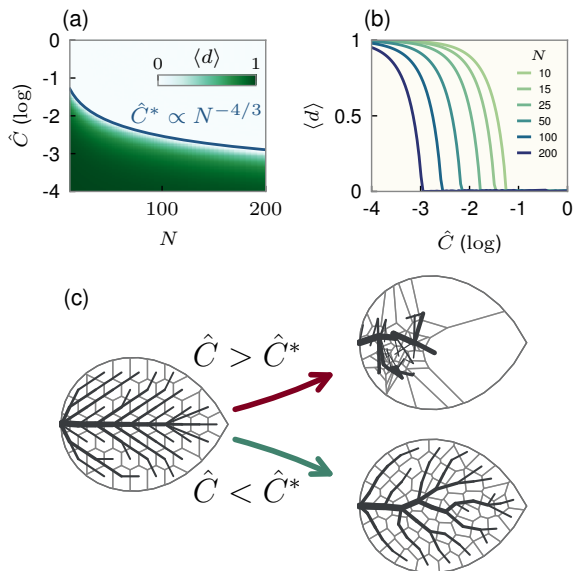

(a) (b) (c)

FIG. 3.   Network collapse as a function of effective delivery conductivity. (a) Phase diagram of (normalized) average nearest-neighbor distance for different $\hat{C}$ and network sizes in the one-dimensional system. The line shows the scaling of $\hat{C}$ at constant Ω, (analytically obtained from the scaling analysis). (b) Individual network sizes from (a) showing the sudden but continuous collapse. (c) Resulting networks with $\hat{C}$ above and below the critical value in two dimensions.

To understand the origin of this transition, we consider the contributions of each power term. If the energy needed to transfer fluid from a sink node to the surrounding region is significantly greater than the energy required for transportation (small $\hat{C}$), the cost incurred by reducing the connections between nodes – and thereby minimizing transport energy – is too high to justify compromising the size discrepancies of the regions. In contrast, when the delivery conductivity is sufficiently high to allow effectively energy-free fluid transfer from the sink to the region, the optimal solution is achieved by reducing the transport cost, i.e., moving all the nodes to the source, as a single node can then deliver fluid to the entire domain cheaply. This leads us to study the transition using the dimensionless power ratio $\Omega = P_\mathrm{t}/P_\mathrm{d}$.

Scaling analysis of the one-dimensional system reveals that $\mathcal{O}(\Omega) = N^{4/3}$ (see [18]). Since $\Omega \propto \hat{C}$, we observe that the abrupt collapse of the system in Fig. 3a aligns with the outcomes of the scaling analysis. This analysis can be extended to the two-dimensional formulation, resulting in $\mathcal{O}(\Omega) = N^{5/3}$, which predicts that *valid* relaxed networks only emerge below the critical $\hat{C}^*$, as shown in Fig. 3c. Although more complex in two dimensions, the phase transition still occurs.

We note that the above scaling laws are derived for a fixed domain space. If instead, we consider a 2D domain that increases with network size, e.g., $\mathcal{O}(L) = \sqrt{N}$ (i.e., a growing leaf), we find the resulting scaling to be $\mathcal{O}(\Omega) = N^{-1/3}$. Thus, for fixed domains, the more densely packed the network is, the lower the capillary conductivity must be in order for the formulation to hold. If, however, the domain expands, the system can maintain the same effective conductivity during growth – as expected of an intrinsic property. For systems with a scaling domain, delivery costs become negligible at large network sizes, where instead the transport costs dominate completely.

Adaptability — One constraint of regular lattice graphs is that their optimal solutions are dependent on the alignment of the main axis of the leaf with the grid. Typically, this alignment is chosen so that the main axis follows the shortest path between neighboring nodes, resulting in network shapes where the leaf main branch is perfectly parallel with the main axis (Fig. 1a). Another issue is the aforementioned non-uniformity of the sink areas along the leaf boundary. Thus, for fixed grids, careful construction and alignment of the network boundary is required. To illustrate this, we construct a leaf shape that can be rotated relative to the grid by using a simplified version of Gielis' superellipse equation [23], as it describes a broad range of leaf shapes accurately [24]:

$$r(\varphi; \varphi_0, \beta) = \left(\left| \cos\left(\tfrac{\varphi - \varphi_0}{4} \right) \right| + \left| \sin\left(\tfrac{\varphi - \varphi_0}{4} \right) \right| \right)^{-1/\beta}. \quad (6)$$
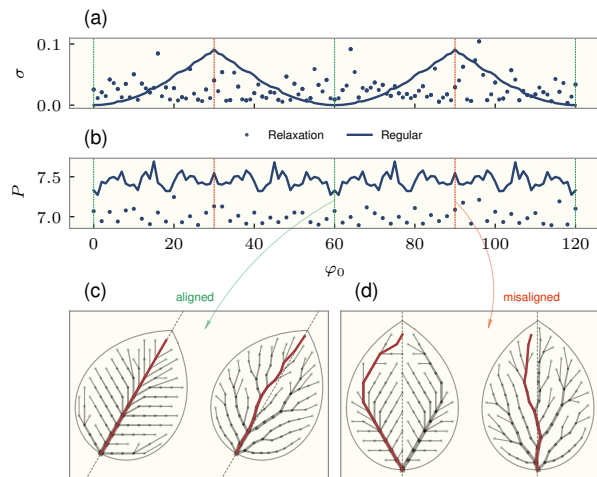
FIG. 4. Model adaptability on differently aligned leaves. (a-b) The deviation σ (a), as well as power P (b) are $60°$-periodic for the regular grid, and non-periodic for the relaxation model. (c-d) Resulting leaves for maximum alignment (c) and misalignment (d) are fundamentally different for the regular grid (c-d, left), but similar for the relaxation model (c-d, right).

Here φ_0 is the inclination of the major axis of the leaf, and β is a shape parameter.

We quantify the discrepancies by rotating the leaf and measuring the power P and the average deviation of the main branch with the grid, $\sigma = 1 - \langle c_i \cos(\theta_i - \varphi_0)\rangle/\langle c_i \rangle$. Here, θ_i and c_i are the angle and conductivity of the i'th edge of the main branch, respectively. The main branch is found by greedily following the edges with the highest conductivity, from the source to a leaf node. On the hexagonal grid, angles $\varphi_0 \equiv 0 \,(\mathrm{mod}\,60°)$ correspond to maximum alignment of the leaf with the grid, whereas $\varphi_0 \equiv 30° \,(\mathrm{mod}\,60°)$ correspond to maximum misalignment.

Fig. 4a shows the average main branch deviations for the outputs of the regular grid and the relaxation model. For the regular grid, we see that the deviation is completely dependent on angle, with a minimum of $\sigma = 0$ at the maximum values of leaf alignment, as well as peaks at maximum misalignment. The slightly non-smooth changes in deviation are caused by the discrete differences in the initializations of the node positions due to boundary clipping. For the deviations in the relaxation model, there is no apparent dependency on angle, which indicates that our model is independent of initial boundary alignment, with the noisiness explained by the fact that the relaxation model output is the result of local optimization. While zero deviation is never reached for the relaxation model (due to the stochastic nature of the optimization), the average deviation $\sigma_{\mathrm{avg}} = 0.026$ is nonetheless lower than $\sigma_{\mathrm{avg}} = 0.033$ for the regular grid.

The resulting power from the same data can be seen in Fig. 4b. The mean power is significantly lower for the relaxation model, which indicates that the solutions in general are more optimal. We note that at maximum alignment we still see a lower power in the relaxation model, even though the regular grid output has lower deviation, which can be attributed to the fact that power is not solely determined by the structure of the main branch, but by the network in its entirety.

Figs. 4c-d show illustrations of the output of the two models for maximum alignment and misalignment, respectively. The regular-grid model yields two extremes where the main branch either completely follows the main axis (Fig. 4c, left), or splits into two similarly sized branches (Fig. 4d, right). In contrast, all outputs from the relaxation model preserve the central main branch (Figs. 4c-d, right).

Vein curvature — A characteristic feature of our resulting networks is the emergence of smoothly curving veins. This is particularly noticeable in the branches extending from the main branch and leads to venation patterns that appear more organic compared to those produced by a regular grid. Previous work relied on complementary initialization techniques in order to effectively mimic biological stochasticity, for example by using disordered tessellation grids [2]. Such a method works by enforcing a repulsive potential between sink nodes that leads to evenly spaced positions in the domain. This is approximately equivalent to independently optimizing P_d in our formulation.

In Fig. 5 we compare the resulting morphologies of a leaf that mimics the domain of *L. xylosteum* (Fig. 5a), and observe how venation patterns agree with biological observations in our formulation (Fig. 5b), and a regular grid (Fig. 5c), respectively. In Fig. 5d we show the result of sequential optimization, i.e., one optimization of
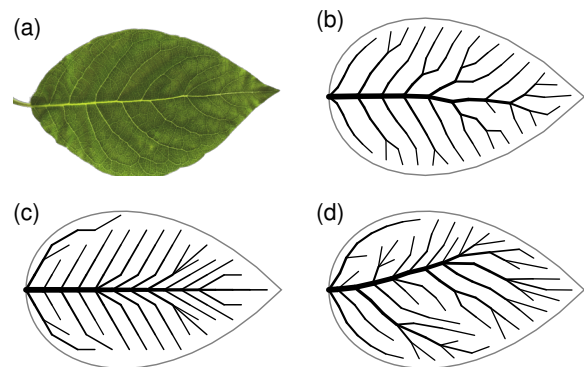

FIG. 5. Comparison between a real leaf and model venation patterns ($\beta = 0.25, N = 100$). (a) Sample of *L. xylosteum* [14]. (b-d) Model output from the (b) relaxation model, (c) regular grid, and (d) sequential optimization.

P_{d} without accounting for transport cost, and one subsequent optimization of P_{t}. We observe sub-optimal solutions where the venation patterns do not match those of the real leaf. Instead, we find that the optimization of P_{t} identifies approximately straight lines in the optimal node positions, and uses these to form the main branch.

Our observations indicate that incorporating coupled energy costs in the power formulation leads to solutions influenced by domain boundaries, closely resembling those found *in vivo* (Fig. 5b). This is reproduced to a lesser extent when performing sequential optimization (Fig. 5d) and is completely absent when relying on regular grids (Fig. 5c).

In this Letter, we have addressed the limitations of traditional transport network optimization models by allowing the optimization over the node positions themselves. We have shown this to be a well-defined problem when incorporating the cost of resource delivery into the energy formulation, with resulting configurations that show adaptability to domain boundaries and misalignment. The study has been enabled by exploiting a fully differentiable process including the Voronoi tessellation of the domain and the steady state of the conductivity adaptation model. Our results demonstrate the advantages of node localization both internally and in adapting to external domain boundaries and allow the emergence of natural organic networks. Our approach could find applications in the efficient design of human-engineered networks [25]. We identified a phase transition as a function of the conductivity $\hat{C}$ of the delivery system, showing network collapse above a critical value, reminiscent of the well-known phase transition in γ for static transport networks [10]. Scaling analysis further reveals that physical networks remain stable during network growth only if the domain expands along with the network. While our goal has been a minimal extension of the hydrodynamic network model, we note that a main insight is the addition of *some* delivery power term and not necessarily precisely the one of Eq. (2). For instance, similar phase transition behavior emerges from considering, e.g., $\langle \ell^2 \rangle$ (diffusion-limited costs). Our approach is limited by the local behavior of gradient descent and the presence of many local optima. Resulting patterns are thus sensitive to initial conditions even though the energy dissipation is similar between them. This problem similarly prevails in edge optimization alone [10]. Our findings pose an interesting question for further research: how can appropriate *local* feedback models [2, 13] be formulated that optimize $P_{\mathrm{t}} + P_{\mathrm{d}}$? Finally, we note that in this Letter we do not consider fluctuations in the sink magnitudes or similar phenomena that can result in reticulate networks [11, 12, 26], the relevance of which becomes evident for larger network sizes. This can be incorporated by choosing a suitable parameterization that allows for area-weighted sinks [14].

———————

* Authors contributed equally.
† Correspondence email address:juki@di.ku.dk

[1] K. A. McCulloh, J. S. Sperry, and F. R. Adler, Nature **421**, 939 (2003), publisher: Nature Publishing Group.

[2] H. Ronellenfitsch and E. Katifori, Physical Review Letters **117**, 138301 (2016), publisher: American Physical Society.

[3] E. Katifori and M. O. Magnasco, PloS one **7**, e37994 (2012), publisher: Public Library of Science San Francisco, USA.

[4] C. D. Murray, Proceedings of the National Academy of Sciences **12**, 207 (1926), publisher: National Acad Sciences.

[5] J. B. Kirkegaard and K. Sneppen, Physical Review Letters **124**, 208101 (2020), publisher: American Physical Society.

[6] A. Konkol, J. Schwenk, E. Katifori, and J. B. Shaw, Geophysical Research Letters **49**, e2022GL098284 (2022), publisher: Wiley Online Library.

[7] A. Lonardi, E. Facca, M. Putti, and C. De Bacco, Physical Review Research **3**, 043010 (2021), publisher: APS.

[8] A. Lonardi and C. De Bacco, Physical Review Letters **131**, 267401 (2023), publisher: APS.

[9] D. Leite and C. De Bacco, Nature Communications **15**, 7981 (2024), publisher: Nature Publishing Group.

[10] S. Bohn and M. O. Magnasco, Physical Review Letters **98**, 088702 (2007), publisher: American Physical Society.

[11] E. Katifori, G. J. Szöllősi, and M. O. Magnasco, Physical Review Letters **104**, 048704 (2010), publisher: American Physical Society.

[12] F. Corson, Physical Review Letters **104**, 048703 (2010).

[13] D. Hu and D. Cai, Physical Review Letters **111**, 138701 (2013), publisher: American Physical Society.

[14] L. E. J. Skjegstad and J. B. Kirkegaard, Modeling Full-Scale Leaf Venation Networks (2024), arXiv:2410.24009 version: 1.

[15] L. B. Rall, Mathematics Magazine **59**, 275 (1986), publisher: Taylor & Francis _eprint: https://doi.org/10.1080/0025570X.1986.11977261.

[16] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, JAX: composable transformations of Python+NumPy programs (2018).

[17] B. M. Bell and J. V. Burke, in *Advances in Automatic Differentiation*, edited by C. H. Bischof, H. M. Bücker, P. Hovland, U. Naumann, and J. Utke (Springer, Berlin, Heidelberg, 2008) pp. 67–77.

[18] See Supplementary Material online.

[19] S. Shumilin, A. Ryabov, S. Barannikov, E. Burnaev, and V. Vanovskii, A Method for Auto-Differentiation of the Voronoi Tessellation (2024), arXiv:2312.16192 [cs].

[20] L. Numerow, Y. Li, S. Coros, and B. Thomaszewski, Differentiable Voronoi Diagrams for Simulation of Cell-Based Mechanical Systems (2024), arXiv:2404.18629 [cs].

[21] F. Feng, S. Xiong, Z. Liu, Z. Xian, Y. Zhou, H. Kobayashi, A. Kawamoto, T. Nomura, and

B. Zhu, International Journal for Numerical Methods in Engineering **124**, 282 (2023), _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.7121.

[22] Code implementation is available at `https://github.com/kirkegaardlab/gradnodes`.

[23] J. Gielis, American Journal of Botany **90**, 333 (2003), _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.3732/ajb.90.3.333.

[24] P. Shi, D. A. Ratkowsky, Y. Li, L. Zhang, S. Lin, and J. Gielis, Forests **9**, 714 (2018), number: 11 Publisher: Multidisciplinary Digital Publishing Institute.

[25] R. W. Barber and D. R. Emerson, Microfluidics and Nanofluidics **4**, 179 (2008).

[26] R. Waszkiewicz, J. B. Shaw, M. Lisicki, and P. Szymczak, Physical Review Letters **132**, 137401 (2024), publisher: APS.

[27] D. Bertsekas, *Nonlinear Programming* (Athena Scientific, 2016).

Spatially continuous sinks: integrals

In this section, we provide a detailed analysis of the area computation of the region where the sink nodes deliver to and explore the integration of distance metrics in a triangle within the polygon. Take a *star convex* polygon (such as those that result from Voronoi tesselations), and consider a triangle ABC in that polygon:
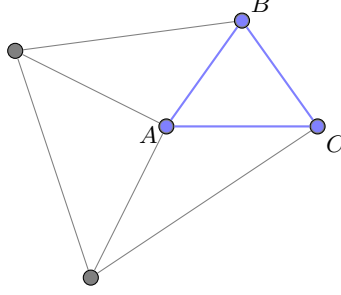

FIG. 6. Diagram of a triangle of a Voronoi cell. A is the sink node and B and C are vertices resulting from the tessaration.

The area of such a triangle is trivial to calculate, and thus the total area is:

$$A = \sum_{T \in \text{triangles}} A_T, \tag{7}$$

where A_T is trivially computed with

$$A_T = \frac{1}{2} \left| (\boldsymbol{x}_A - \boldsymbol{x}_B) \times (\boldsymbol{x}_A - \boldsymbol{x}_C) \right|. \tag{8}$$

Likewise, we need the average distance from A to all points in the polygon. This becomes:

$$\langle L \rangle = \frac{1}{A} \sum_{T \in \text{triangles}} \int_{A_T} ||\boldsymbol{x}_A - \boldsymbol{x}|| \, \mathrm{d}A. \tag{9}$$

This integral is not easy, but can be done by using polar coordinates. We write

$$\boldsymbol{x}(r, \theta) = \boldsymbol{x}_A + r \begin{pmatrix} \cos \theta \\ \sin \theta, \end{pmatrix} \tag{10}$$

and the integral becomes

$$\mathcal{I} = \int_{A_T} ||\boldsymbol{x}_A - \boldsymbol{x}|| \, \mathrm{d}A = \int_{\theta_1}^{\theta_2} \mathrm{d}\theta \int_0^{R(\theta)} \mathrm{d}r \, r^2 = \frac{1}{3} \int_{\theta_1}^{\theta_2} R(\theta)^3 \, \mathrm{d}\theta, \tag{11}$$

where $R(\theta)$ is the distance from A to the line BC with angle θ,

$$R(\theta) = \frac{x_a(y_b - y_c) + x_b(y_c - y_a) + x_c(y_a - y_b)}{(y_c - y_b) \cos \theta + (x_b - x_c) \sin \theta}, \tag{12}$$

thus, our integral becomes

$$\mathcal{I} = \frac{1}{3} \left[x_a(y_b - y_c) + x_b(y_c - y_a) + x_c(y_a - y_b) \right]^3 \int_{\theta_1}^{\theta_2} \frac{1}{(\delta y \cos \theta - \delta x \sin \theta)^3} \, \mathrm{d}\theta,$$

where $\delta x = x_c - x_b$ and $\delta y = y_c - y_b$. This integral can be evaluated using Weierstrass substitution to give

$$\int \frac{1}{(\delta y \cos \theta - \delta x \sin \theta)^3} \, \mathrm{d}\theta = \frac{1}{\delta^3} \tanh^{-1} \left(\frac{\delta x + \delta_y \tan(\theta/2)}{\delta} \right)$$
$$+ \frac{\delta x \cos \theta + \delta y \sin \theta}{2\delta^2 (\delta y \cos \theta - \delta x \sin \theta)^2}, \tag{13}$$

where we defined $\delta = \sqrt{\delta x^2 + \delta y^2}$. Note that

$$\tan(\theta/2) = \frac{\sin \theta}{1 + \cos \theta}, \tag{14}$$

so this can all be written in terms of $\cos \theta$ and $\sin \theta$ (i.e. no need to actually calculate θ). Thus, the formula is complete by specifying

$$\cos \theta_1 = \frac{x_B - x_A}{||\boldsymbol{x}_B - \boldsymbol{x}_A||}, \qquad\qquad \sin \theta_1 = \frac{y_B - y_A}{||\boldsymbol{x}_B - \boldsymbol{x}_A||}, \tag{15}$$

$$\cos \theta_2 = \frac{x_C - x_A}{||\boldsymbol{x}_C - \boldsymbol{x}_A||}, \qquad\qquad \sin \theta_2 = \frac{y_C - y_A}{||\boldsymbol{x}_C - \boldsymbol{x}_A||}. \tag{16}$$

As Weierstrass substitution has problems at $\theta = \pi$, the integral is only valid when the triangle is aligned with node A west of nodes B, C — but this is always achievable by a simple rotation.

Solution for the 1-Dimensional case

Here, we show the derivations and explanations to find the optimal distribution of $\boldsymbol{x}$ in the simplified case of a one-dimensional (1D) leaf. We begin by considering a system with a single source $s_0 = 1$ at $x_0 = 0$, and n delivery nodes positioned at locations $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$, which get constrained to $x \in [0, L]$.

Similar to the 2D case, the sink values s_i represent the rate at which flow is absorbed or removed at each node x_i. The sink value is calculated based on the distances between the neighbouring nodes, normalized by the system length L. An exception to this is the first and last nodes, where the distance to the boundary is used instead.

$$s_i = -\frac{1}{2L}(x_{i+1} - x_{i-1}), \tag{17}$$

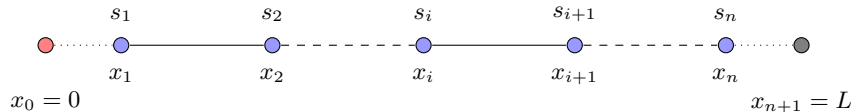with special attention to correct at x_0 and x_n.

FIG. 7. Diagram of the 1D transport networks with moving nodes (blue) and a fixed source (red).

Next, we compute the flow values F_i of the passing flow through each node. Since we are in 1D, we assume that the nodes are only connected to the nodes on their sides. Hence, the flow expression for each edge simplifies to

$$F_{i+1} = F_i - s_i = F_1 - \sum_{j \leq i} s_j, \tag{18}$$

given that $F_1 = 1$.

The conductivity at each node denoted C_i, is a function of the flow F_i. We can find the optimal conductivity for this system by minimizing the transport power P_t given by

$$P_t = \sum_{i \in \text{transport}} L_i \left(\frac{F_i^2}{C_i} + c_t C_i^\gamma\right), \tag{19}$$

by setting $\frac{\partial P}{\partial C} = 0$, we find the conductivity at node x_i to be:

$$C_i = \left(\frac{F_i^2}{c_t \gamma}\right)^{\frac{1}{1+\gamma}}. \tag{20}$$

This formula captures the non-linear dependence of conductivity on the square of the flow, modulated by the parameters c_t and γ.

The total power P of the system consists of two components: the transport power P_t and the delivery power P_d. The transport power is given by:

$$P_{\text{t}} = \sum_{i \in \text{transport}} L_i \left(\frac{F_i^2}{C_i} + c_t C_i^\gamma\right), \tag{21}$$

where L_i represents the length of the transport segment corresponding to node x_i, and the two terms inside the summation represent the power due to flow and the power related to the conductivity, respectively.

The delivery power, on the other hand, is the power used to deliver flow from the sink nodes to their surrounding areas, and it depends on the sink values s_i and the average delivery distance $\langle \ell_i \rangle$. The delivery power is expressed as:

$$P_{\text{delivery}} = \sum_{i \in \text{delivery}} \langle \ell_i \rangle \frac{s_i^2}{\hat{C}}, \tag{22}$$

where $\hat{C}$ represents the delivery conductivity, and $\langle \ell_i \rangle$ is the average distance over which the flow must be delivered.

The average delivery distance $\langle \ell_i \rangle$ for a node at position x_i is computed using the positions of the left and right domain edges, denoted x_L and x_R, respectively. The formula for the average delivery distance is the weighted average:

$$\langle \ell_i \rangle = \frac{(x_R - x_i)}{x_R - x_L} \cdot \frac{(x_R - x_i)}{2} + \frac{(xi - x_L)}{(x_R - x_L)} \cdot \frac{(x_i - x_L)}{2} = \frac{1}{2(x_R - x_L)}\left[(x_R - x_i)^2 + (x_i - x_L)^2\right] \tag{23}$$

This formula accounts for the spatial distribution of the flow and the relative positions of the nodes within the system.

Scaling for different network sizes

One-dimensional network

We analyze the scaling of the total power P given by

$$P = P_t + P_d, \tag{24}$$

with respect to the number of elements N on the 1 dimensional case using the expressions above. For this, we assume the domain to be fixed with N.

Starting by the transport term, we have

$$P_{\text{t}} = \sum_{i \in \text{transport}} L_i \left(\frac{F^2}{C_i} + c_t C_i^\gamma \right), \tag{25}$$

where $\mathcal{O}(F_i^2) \approx 1/N^2$, and thus $\mathcal{O}(C_i) = (\frac{1}{N^2})^{2/3}$ according to Eq. (20). We also assume c_t constant and $\gamma = 0.5$ as in the main text. Taking into account $L_i \propto \frac{L}{N}$, and splitting the terms we find:

$$\mathcal{O}(P_t) = \mathcal{O}(\Sigma) \, \mathcal{O}(L) \, \mathcal{O}(F^2) \, \mathcal{O}(C^{-1}) + \mathcal{O}(\Sigma) \, \mathcal{O}(L) \, \mathcal{O}(C^{1/2}) = N^{-\frac{2}{3}} = N^{-\frac{2}{3}} + N^{-\frac{2}{3}}. \tag{26}$$

Hence, the transport network scales as $\mathcal{O}(P_t) = N^{-2/3}$.

In the case of the delivery term, given by

$$P_d = \sum_{i \in \text{delivery}} \langle \ell_i \rangle \frac{s_i^2}{\hat{C}}, \tag{27}$$

where $\langle \ell_i \rangle \propto \frac{L}{N}$, $s_i^2 \propto \frac{1}{N^2}$ and $\hat{C}$ is constant, giving:

$$\mathcal{O}(P_d) = \mathcal{O}(\Sigma) \, \mathcal{O}(\langle \ell \rangle) \, \mathcal{O}(s^2) = N^{-2}. \tag{28}$$

The ratio between the transport and cost terms to the delivery term scales as:

$$\mathcal{O}(\Omega) = \frac{\mathcal{O}(P_d)}{\mathcal{O}(P_t)} = \frac{N^{-2}}{N^{-\frac{2}{3}}} = N^{-\frac{4}{3}} \tag{29}$$

Therefore, the ratio scales as $\mathcal{O}(\Omega) = N^{-4/3}$.

Two-dimensional network

In the case of two dimensions, such as the ones on the networks of the main paper, we need to approximate the scaling of the edges. This is due to the optimized conductivities making the graph sparse, resulting in a linear scaling of the edges.

Similarly to , we want to understand the scaling of the terms of the power by analiyzing how its components scale with N.

Here, the scaling of the flow $\mathcal{O}(F^2)$ is unknown because it depends on the network structure and therefore γ. Empirically $\mathcal{O}(F^2) \approx 1/N$ if $\gamma = 1/2$ and $\mathcal{O}(F^2) \approx 1/\sqrt{N}$ if $\gamma = 1/4$. This is approximate because in reality F^2 does not scale uniformly: some edges have $\mathcal{O}(1)$ and some have $\mathcal{O}(1/N)$.

Here we assume $\gamma = 1/2$ and take

$$\mathcal{O}(F^2) = N^{-1} \tag{30}$$

The remaining terms have known scaling with N:

$$C = \left(\frac{F^2}{c_t \gamma}\right)^{\frac{1}{1+\gamma}} \Rightarrow \mathcal{O}(C) = \mathcal{O}(c_t)^{-\frac{2}{3}} N^{-\frac{2}{3}} \tag{31}$$

$$\mathcal{O}(L) = N^{-\frac{1}{2}} \tag{32}$$

$$\mathcal{O}(\langle \ell \rangle) = N^{-\frac{1}{2}} \tag{33}$$

$$\mathcal{O}\left(\sum\right) = N \tag{34}$$

$$\mathcal{O}(s^2) = N^{-2} \tag{35}$$

So we find

$$\mathcal{O}(P_t) = \mathcal{O}(\Sigma)\, \mathcal{O}(L)\, \mathcal{O}(F^2)\, \mathcal{O}(C^{-1}) + \mathcal{O}(\Sigma)\, \mathcal{O}(L)\, \mathcal{O}(C^{1/2}) = N^{\frac{1}{6}} + N^{\frac{1}{6}} = N^{\frac{1}{6}} \tag{36}$$

and analogously

$$\mathcal{O}(P_d) = \mathcal{O}(\Sigma)\, \mathcal{O}(\langle \ell \rangle)\, \mathcal{O}(s^2) = N^{-3/2}. \tag{37}$$

Therefore, the resulting power ratio scales as

$$\mathcal{O}(\Omega) = \frac{\mathcal{O}(P_d)}{\mathcal{O}(P_t)} = \frac{N^{-3/2}}{N^{1/6}} = N^{-10/6} = N^{-5/3} \tag{38}$$

which changes the scaling w.r.t. the one dimensional case.

Differentiable Voronoi

In this section, we show an overview of the clipped Voronoi tessellation implementation used. Notably, our approach to make it differentiable relies on a pre-calculation using SCIPY, and leveraging that information to efficiency construct the resulting diagram.

Algorithm 1 Differentiable Voronoi Tessellation Clipped by a Boundary

---

Require: Set of points $x = \{(x_i, y_i)\}$, convex boundary polygon boundary
Ensure: Voronoi nodes and regions clipped within the boundary
 1: **procedure** DIFFERENTIABLE_VORONOI(x, boundary)
 2:     **Voronoi Pre-calculation (non-differentiable using SciPy and CPU)**
 3:     Extend x by scaling the boundary to avoid infinite regions:

$$x_{\text{ext}} = x \cup (\text{boundary} + 100 \cdot (\text{boundary} - \text{mean}(x)))$$

 4:     Compute initial Voronoi regions and Delaunay triangulation using `scipy.spatial`.
 5:     Identify regions needing clipping by checking intersections of Voronoi edges with boundary edges.

 6:     **Differentiable Clipped Voronoi (GPU friendly)**
 7:     Use Delaunay triangulation on x_{ext} to find Voronoi nodes as follows:
 8:     **for** each triangle defined by points $\{(a_x, a_y), (b_x, b_y), (c_x, c_y)\}$ **do**
 9:         Compute determinant:

$$D = 2 \cdot ((a_x - c_x) \cdot (b_y - c_y) - (b_x - c_x) \cdot (a_y - c_y))$$

10:         Compute coordinates (u_x, u_y) of the Voronoi node:

$$u_x = \frac{(a_x^2 + a_y^2)(b_y - c_y) + (b_x^2 + b_y^2)(c_y - a_y) + (c_x^2 + c_y^2)(a_y - b_y)}{D}$$

$$u_y = \frac{(a_x^2 + a_y^2)(c_x - b_x) + (b_x^2 + b_y^2)(a_x - c_x) + (c_x^2 + c_y^2)(b_x - a_x)}{D}$$

11:     **end for**
12:     Determine which regions require clipping based on intersection and inclusion conditions.
13:     **for** each Voronoi region **do**
14:         **if** region needs clipping **then**
15:             Clip the region to the boundary by finding intersections of edges (A, B) and (C, D) as follows:
16:             Calculate intersection parameter:

$$u_a = \frac{(D - C) \times (A - C)}{(B - A) \times (D - C)}$$

17:             If $0 \leq u_a \leq 1$, compute intersection point:

$$(x, y) = A + u_a \cdot (B - A)$$

18:             Sort clipped vertices by angle θ relative to the centroid for ordered polygon representation:

$$\theta = -\arctan 2(y - y_{\text{centroid}}, x - x_{\text{centroid}})$$

19:         **end if**
20:     **end for**
21:     **Return Voronoi Nodes and Regions**
22:     Concatenate and reorder clipped regions to form the final set of valid Voronoi polygons within the boundary.
23: **end procedure**

Since this implementation is written with GPU in mind, most of the loops are instead vector operations, and calculation occur on its majority in parallel. For calculating the gradients we rely on jax [16] automatic differentiation rules.

Implicit Function Theorem

During the optimization of the nodes positions, we perform a full optimization of the conductivities of the system. To do so, we use the dynamical model defined in the main text as

$$\frac{\mathrm{d}C_e}{\mathrm{d}t} = \left(\frac{F_e^2}{C_e^{\gamma+1}} - \gamma c_t \right) C_e + c_0 e^{-\lambda t}, \tag{39}$$

until it has reached the steady state $\boldsymbol{C}^*$. The steady state solution is equivalent to finding the fixed point of the equation $\frac{\mathrm{d}C_e}{\mathrm{d}t} = 0$. From now on, we are interested in solving for the implicit function $f(C^*, x) = 0$.

The result $\boldsymbol{C}^*$ is dependent on the positions of the nodes $\boldsymbol{x}$ in a non-trivial manner. Namely, $\boldsymbol{C}^*$ depends on $\boldsymbol{F}_e$, which is a function of $\boldsymbol{L}_e$ and $\boldsymbol{s}$, all of which dependent on $\boldsymbol{x}$. We are interested in finding $\frac{dC}{dx}$, and even though automatic differentiation would allow to apply the chain rule through the iterative solver, the memory and computational cost of calculating derivatives and storing them would make it computationally challenging. Luckily, since we are iterating over an implicit function, we can make use of the implicit function theorem [17, 27], which allows for backpropagation through a fixed-point iteration without requiring saving each iterative step into memory.

For simplicity of presentation, here we will consider that the power only depends on the optimal conductivities, i.e.

$$\frac{dP}{dx} = \frac{dP}{dC^*} \cdot \frac{dC^*}{dx}. \tag{40}$$

In order to compute $\frac{dC^*}{dx}$ (the gradients of interest here), we use its implicit form $f(C, x) = 0$ and obtain

$$\frac{df}{dx} = \frac{\partial f}{\partial C} \cdot \frac{dC^*}{dx} + \frac{\partial f}{\partial x} = 0. \tag{41}$$

Letting $A = \frac{\partial f}{\partial C}$ (the Jacobian of $\boldsymbol{f}$ with respect to $\boldsymbol{C}$) and $B = \frac{\partial f}{\partial x}$ (the Jacobian of $\boldsymbol{f}$ with respect to $\boldsymbol{x}$), we isolate $\frac{\partial C^*}{\partial x}$:

$$\frac{dC^*}{dx} = -A^{-1}B. \tag{42}$$

This Jacobian is memory intensive (here we are ignoring the dependencies on the solver of $\boldsymbol{F}$). However, we can rephrase this as a fixed-point problem using the adjoint vector w given by

$$w^T = v^T A. \tag{43}$$

where $v = \frac{\partial f}{\partial C^*}$. This means that we can solve for w iteratively if we express it in implicit form such that $w = f(w) = w^T - v^t A$ where we are around $f(w) = 0$, and have

$$\frac{df}{dx} = w^T B. \tag{44}$$

which is equivalent to finding $\frac{dC}{dx}$ in the regime where $f(C, x) = f(C^*, x) = 0$. Thus, by applying the implicit function theorem, we can compute the gradient with respect to parameters in the adaptation model for conductivity optimization, ensuring efficiency by avoiding the storage and backpropagation through each individual iteration step.

Our implementation of the fixed point approach can be found at `https://github.com/kirkegaardlab/gradnodes`.