Optimal Geometric Flows via Dual Programs

Sylvester Eriksson-Bique* Valentin Polishchuk† Mikko Sysikaski‡

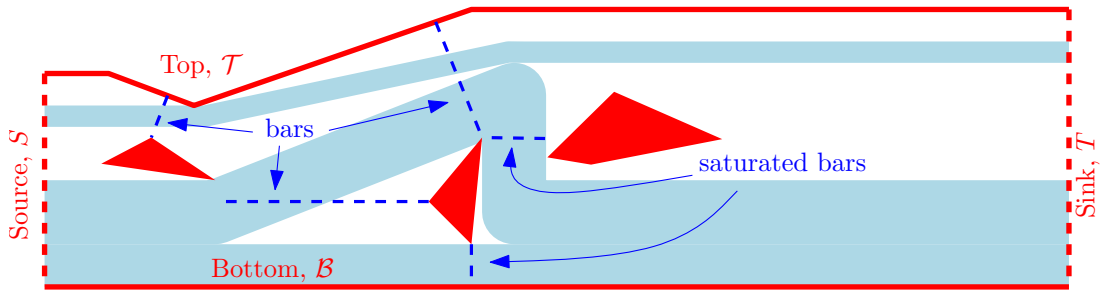
Abstract

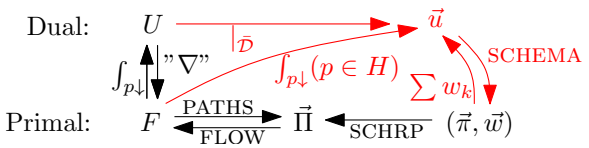
Considering potentials in the dual of a planar network has proved to be a powerful tool for computing planar maximum flows. In this paper we explore the use of potentials for giving algorithmic and combinatorial results on *continuous flows in geometric domains* – a (far going) generalization of discrete flows in unit-capacity planar networks.

A continuous flow in a polygonal domain is a divergence-free vector field whose magnitude at every point is bounded by a given constant – the domain's permeability. The flow enters the domain through a source edge on the outer boundary of the domain, and leaves through a sink edge on the boundary. The value of the flow is the total flow amount that comes in through the source (the same amount comes out of the sink, due to the vanishing divergence). The cost of the flow is the total length of its streamlines. The flow is called monotone if its streamlines are x-monotone curves.

Our main result is an algorithm to find (an arbitrarily close approximation to) the *minimum-cost* monotone flow in a polygonal domain, by formulating the problem as a convex program with an interesting choice of variables: one variable per *hole* in the domain. Our approach is based on the following flow of ideas: flow is the gradient of a potential function; a potential function can be extended from free space to holes so that it is constant over each hole; instead of the potential function (a value per point in the domain) we can thus speak of a potential vector (a value per hole); a potential uniquely (up to homotopic equivalence) defines "thick" paths in the domain; the paths define a flow. We show that the flow cost is convex in the potential; this allows us to employ the ellipsoid method to find the mincost flow, using holes potentials as variables. At each ellipsoid iteration the separation oracle is implemented by feasibility checks in the "critical graph" of the domain – as we prove, the graph defines linear constraints on the potentials.

Using potentials and critical graphs we also prove MaxFlow/MinCut theorems for geometric flows – both for monotone and for general ones. Formulating and proving the monotone version is a new result. The MaxFlow/MinCut theorem for non-monotone flows has been proved earlier by two different methods; the potentials technique provides a third proof.





*Courant Institute, NYU. ebs@cims.nyu.edu
†Communication and Transport Systems, ITN, Linköpng University. valentin.polishchuk@liu.se
‡Google, Inc. firstname.lastname@gmail.com

1 Introduction

Computing optimal flows in networks is one of the central problems in discrete optimization. At the same time, in its very classical meaning the word "flow" (as understood, say, in calculus or in physics) refers to a vector field defined over a *continuous* domain – in contrast to the network flows which are *discrete* by nature. The vast majority of both theoretical and experimental work on continuous flows is being done in the realm of *numerical simulation* of fluid dynamics – in particular, with finite element analysis; see, e.g., [22] and references thereof. The research is heated by enormous practical importance of computing fluid flow through porous media, e.g., in reservoir engineering and similar areas of hydrogeology. Other applications motivating the study of continuous flows include fibered material design, swarm robotics, evacuation planning, battlefield operations, ship routing, video games and simulators, air traffic management. In all these motion planning domains the task is to route small objects (agents) through a polygonal environment. Finding and describing a specific path for each individual object may be unreasonable due to the prohibitively large number of the agents; the solution then is to find an optimal flow and let each object travel through the domain simply by following one of the flow streamlines. In many of these applications it is desirable that objects' paths do not backtrack, i.e., that the paths are monotone in some direction.

1.1 Problem Formulation

Let $\mathcal{D}$ be a polygonal domain with holes (obstacles). The domain is defined by a simple polygon $\mathcal{P}$ (the *outer polygon* of $\mathcal{D}$) and a set of holes – pairwise-disjoint simple polygons lying inside $\mathcal{P}$; the domain $\mathcal{D}$ is $\mathcal{P}$ minus the holes. Two edges, S, T on the boundary of the outer polygon are designated as the *source* and the *sink* (we can work also with multiple source/sink edges, but for ease of exposition restrict the model to 1 source and 1 sink). S and T split the boundary of $\mathcal{P}$ into two parts called the *bottom* $\mathcal{B}$ and the *top* $\mathcal{T}$ (see the figure on the title page). Let n, h denote the number of vertices and holes in $\mathcal{D}$. Assume that vertices of $\mathcal{D}$ have integral coordinates and let L be the largest integer.

The domain $\mathcal{D}$ is uniformly capacitated in the sense that the amount of flow that can go through any point of the domain is bounded by 1; in other words, the permeability of any point is 1 (this is formalized below – see the third constraint in (1)). We are also given a number $V \geq 0$ – the target flow value. Our goal is to ***find a monotone source-sink flow of value*** V ***in*** $\mathcal{D}$ ***with minimum cost***. Formal definitions of the flow, its value and cost follow.

A *flow* in $\mathcal{D}$ is a function $F : \mathcal{D} \mapsto \mathbb{R}^2$ with the following properties:

$$\begin{aligned} \operatorname{div} F(p) &= 0 \qquad \forall p \in \mathcal{D} \\ F(p) \cdot \mathbf{n}(p) &= 0 \qquad \forall p \in \partial\mathcal{D} \setminus \{S \cup T\} \\ |F(p)| &\leq 1 \qquad \forall p \in \mathcal{D} \end{aligned} \tag{1}$$

where $\mathbf{n}(p)$ is the unit normal to the boundary of $\mathcal{D}$ at point p. That is, a flow is a vector field $F(p) = (F^x(p), F^y(p))$. There are no source/sinks inside the domain (the first constraint in (1)), the flow enters/exits $\mathcal{D}$ only through the source/sink (the second constraint in (1)), and the flow respects the capacity constraints (the third constraint in (1)). A flow is *x-monotone* if its x-component is non-negative ($\forall p \in \mathcal{D}, F^x(p) \geq 0$); we will omit the prefix *x-* and call an x-monotone flow just *monotone*. Moreover, unless stated otherwise, any flow in this paper will be monotone; that is, by a *flow* we will understand an (x-)monotone flow.

Similarly to the discrete network flow, the *value* of a continuous flow F is the total flow coming in from the source, i.e., $\int_S F \cdot \mathbf{n} \, \mathrm{d}s$. Since inside $\mathcal{D}$ the flow is divergence-free (flow conserves inside $\mathcal{D}$), by the divergence theorem, the value is equal to the total flow out of the sink, $-\int_T F \cdot \mathbf{n} \, \mathrm{d}t$.

A *streamline* of F through a point $s \in S$ is a curve ℓ_s such that F is tangent to ℓ_s at every point of the curve; for a monotone flow, each streamline is an x-monotone curve (i.e., a vertical line intersects any streamline in at most one contiguous interval). Strang [50] and Mitchell [40] suggested that the *cost* of a flow may be defined as: (I) the area of the support, $\operatorname{supp} F$, of the flow, where $\operatorname{supp} F = \{p \in \mathcal{D} : |F(p)| > 0\}$; or, (II) the "total length of the streamlines", $\int_{s \in S} |\ell_s| ds$; or, (III) the

length of the longest streamline $\max_{s \in S} |\ell_s|$. Other natural definitions of flow cost include integrals of flow's absolute value or its square over the domain: (IV) $\int_{\mathcal{D}} |F|$; or, (V) $\int_{\mathcal{D}} |F|^2$.

Similarly to [40, 42], unless otherwise stated, we consider only *0/1 flows*, i.e., flows that saturate every point of the support: $\forall p \in \operatorname{supp} F$, $|F(p)| = 1$ (we justify the focus on 0/1 flows in Section 2.2). For a 0/1 flow, definitions (I), (II) and (IV) become equivalent, and these are the definitions that we will use; for concreteness, assume that the cost is always defined according to (I).[1] Minimizing $\int_{\mathcal{D}} |F|^2$ (definition (V)) is a variant of the Dirichlet problem for the Poisson equation widely studied in the boundary-value problems literature, see e.g., [13]. Under definition (III) our problem is NP-hard, as we argue in Appendix G.

1.2 Related work

In comparison to the well-trod field of discrete network flows, much less is known about computing continuous flows in geometric domains. Atkinson and Vaidya [7] and Agarwal, Efrat and Sharir [1] considered the Euclidean *transportation problem* (a generalization of the minimum-weight bipartite Euclidean matching problem) in which a commodity has to be transferred, at a minimum cost, from a set of point sources to a set of point sinks. The problem considered here may be viewed as the continuous version of the *transshipment problem*; in our model the source and the sink are edges of the domain.

The three celebrated results in the discrete network flows area are MaxFlow/MinCut Theorem, Menger's Theorem, and Flow Decomposition Theorem. Note that the theorems are *combinatorial* results, which do not automatically lead to *algorithms* for computing optimal flows and disjoint paths in networks. On the algorithmic frontier, the two central problems associated with network flows are finding maximum flows and minimum-cost flows.

The Continuous MaxFlow/MinCut Theorem was proved by Strang [50]. As with the discrete theorem, Strang's result does not immediately imply an efficient algorithm for finding maximum flows and minimum cuts in continua. Mitchell [40] showed how to compute the MaxFlow by filling the domain with "bottommost" streamlines and how to find the MinCut using the "critical graph" of $\mathcal{D}$ [21].

The continuous counterpart of Menger's Theorem was proposed in [6]; the same paper gave an algorithm to compute the maximum number of disjoint source-sink "thick paths" in a polygonal domain.

The continuous version of the Flow Decomposition Theorem was established in [42]. The theorem reduces the mincost flow problem to that of computing a set of optimal thick paths, but it does not immediately lead to an efficient way of finding the widths and the homotopy types of the paths (in fact, the theorem does not even suggest how many paths there are in the decomposition); thus except for special cases in which the paths' widths and homotopy types can be inferred (e.g., in simple polygons – the case considered in [42]), the theorem does not provide an *algorithm* for computing mincost flows. In this paper we make a first step towards solving the mincost flow problem in polygonal domains with holes, giving an algorithm for monotone flows. (In [42] it was also shown that optimal thick paths in a polygonal domain can be found by scrolling through the path's homotopy types; the algorithm works also for monotone paths, but the scrolling still takes time exponential in the number of holes.)

Conducting further investigations "into the mapping between discrete graph algorithms and their continuous analogues in geometry" was put on the agenda by Mitchell, who, in turn, obtained the suggestion to consider this class of problems from Papadimitriou [40]. The techniques developed in this paper can be viewed as a geometric extension of Hassin's ingenious idea to use dual-graph potentials to compute maximum flows in planar graphs [25]; see also [31], and [10, 14] for recent work. We note that in discrete networks, the potentials were used almost exclusively for finding *maximum* flows; the exceptions are [14, Section 4] where potentials were used implicitly to compute minimum-cost flows with well-behaved cost functions and the recent near-linear-time algorithm for outerplanar networks

[1] One may also consider a generalization to non-uniform costs case, where pushing flow through different parts of $\mathcal{D}$ incurs different costs. This is a much harder problem; in fact, in this setting it is not even known how to find just 1 "thick" path of minimum weighted area (finding 1 *thin* path of minimum weighted length is the *weighted region problem*, solved in [41]).

announced in [29]. In contrast, we exploit potentials to compute *minimum-cost* continuous flows. Our use of potentials for geometric *maximum* flows is non-algorithmic; we employ them to (re)prove MaxFlow/MinCut theorems for flows in continua.

Kohn and Strang studied mincost continuous flows in [34, 35] (one technical difference between their setting and ours is that they assume the flow's normal component is given everywhere on the boundary); high-dimensional generalizations were considered in [49]. The treatment in [34, 35, 49] is purely mathematical, and the developed techniques do not provide algorithms for computing mincost flows [33].

Multicommodity versions of geometric flows were studied in [32]. Applications of combinatorial and algorithmic results on continuous flows to air traffic management are described in [36, 52]. In another practical domain—wireless sensor networks (WSNs)—continuous flows model message passing in a dense network; in particular [45] defines and studies *geographic* MaxFlow and MinCut through a WSN.

The term "continuous flow" was used also for flows in discrete networks that have time-varying parameters – edge capacities, node storage, link delays, etc. [4,46]. In contrast, in our setting, "continuous flow" refers to a continuous vector field (the flow) in a *static* polygonal environment (the continuum).

1.3 Our contributions

We present an algorithm that, for an arbitrary $0 < \varepsilon < 1$, outputs a flow whose cost is at most ε larger than the minimum cost of any flow with the same value V (that is, we give an *additive-ε* approximation to the mincost flow). The algorithm runs in $O(nh^3 \log^2 \frac{nL}{\varepsilon})$ time.

We also apply our techniques to *maximum* flows: we formulate and prove the MaxFlow/MinCut theorem for monotone continuous flows, and give a new proof of the MaxFlow/MinCut theorem for general, non-monotone flows (the latter theorem was earlier proved in [40, 50] using different methods).

1.4 Overview of the approach

The Continuous Flow Decomposition Theorem allows us to reformulate our problem as that of finding an optimal set of thick non-crossing paths. We find the optimal paths in two steps: first find optimal homotopy types and widths of the paths, and then find optimal paths of given homotopy types and widths. The second step is an instance of the homotopic routing problem, which can be solved efficiently using existing tools (Section 2.1).

The tricky part is finding optimal homotopy types and widths of the paths. Here we turn attention to the dual of a continuous flow – the potential function, which is a scalar function defined at each point of the domain; the gradient of the potential function determines the flow vector at the point, and the level sets of the function are the streamlines.

Our main insight—using one variable per hole in the domain to indirectly define a flow (Section 3)—hinges on the following two observations (one simple, one non-trivial):

• The potential function can be extended to the holes of the domain by assigning a constant to every hole. (This is obvious, since the flow does not penetrate holes.)

• If the potentials of all holes are known, then the homotopy types of the flow streamlines are uniquely defined. (This requires a proof; we give it in Lemma 6, using the "bottommost fill" paradigm of Mitchell [40].)

The above observations allow us to formulate our flow problem in terms of the holes potentials.

Our main technical result is established next: the mathematical program for mincost flow, which uses holes potentials as variables, is convex. Specifically, we prove that:

• The flow cost as a function of the potentials is convex. The convexity proof (Lemma 8) is a simple application of the triangle inequality.

• The feasible region of the program is a polyhedron. This is proved by defining a version of the domain's "critical graph" [6, 21, 40] and showing that the program constraints correspond to saturation of the graph

edges by flow (Lemmas 9 and 10).[2]

Finally, the convex optimization problem can be solved using standard methods (modulo many non-trivial technical details needed to adapt the methods to our specific case); see Section 5.

As a by-product we establish the MaxFlow/MinCut theorem for monotone flows, by observing that finding *maximum* monotone flow can also be formulated as a convex program for holes potentials and that the program is exactly the dual of a shortest path problem in our critical graph (Theorem 12). Last but not least, we prove that the same convex program models finding maximum *non-monotone* flows; this gives a new proof of the MaxFlow/MinCut theorem for non-monotone flows (in comparison with the monotone case, in which the proof directly follows from our treatment of mincost flows, the proof for general flows requires some additional work; see Appendix C).

Remark 1. Recall that the discrete network flow problem is also a convex program – a linear program (LP) whose variables represent flow along arcs of the network. One may extend this LP to a convex program for minimum-cost monotone continuous flows: variables in the program would represent flow along edges of the visibility graph of the domain. Such "primal" approach would follow the standard paradigm in geometric route optimization – employing the visibility graph to discretize the search space. Our, dual approach leads to a more efficient algorithm for the mincost flow (in particular, we have only h variables as opposed to $O(n^2)$ in the primal, visibility-based program).

2 The tools

This section sets up the requisite tools for our treatment of flows; we describe the tools only as they are relevant for us (not necessarily in their full strength and generality).

Thick paths For a set $\mathcal{S} \subset \mathcal{D}$ and a number $r > 0$ let $\langle \mathcal{S} \rangle^{(r)}$ denote the Minkowski sum of $\mathcal{S}$ with the radius-r open disk centered at the origin. Let π be a source-sink path in $\mathcal{D}$ that leaves the source and enters the sink perpendicularly to the boundary of the domain; that is, π connects a point $s \in S$ to a point $t \in T$, and π is perpendicular to S at s and to T at t. Let $w > 0$ be a number, and assume that $\langle \pi \rangle^{(w/2)}$ does not intersect the holes. A *w-thick path* Π is the part of $\langle \pi \rangle^{(w/2)}$ that lies inside $\mathcal{D}$: $\Pi = \langle \pi \rangle^{(w/2)} \cap \mathcal{D}$. With this definition, a thick path is a "curved rectangular strip" of width w with opposite sides of length w residing on the source and the sink ([42] calls the strip the *canonical part* of the Minkowski sum $\langle \pi \rangle^{(w/2)}$). The path π is called the *centerline* of Π, and w is the *width* of Π; the *length* of Π is the length of π, and the *homotopy type* of Π is that of π. A thick path is *locally shortest* if it cannot be shortened without changing its homotopy type; we will often call a locally shortest path just *shortest*. A thick path is *x-monotone* if its centerline is an x-monotone path; we will omit the prefix *x-* and call an x-monotone path just *monotone*. Moreover, unless stated otherwise, any path in this paper will be monotone; that is, by a *path* we will understand an (x-)monotone path.

2.1 Shortest homotopic routing

A *wire* is a (monotone, thin) polygonal S-T path in $\mathcal{D}$. A *schema* $(\vec{\pi}, \vec{w})$ is a collection $\vec{\pi} = (\pi_1, \ldots, \pi_K)$ of non-crossing wires, together with a collection $\vec{w} = (w_1, \ldots, w_K)$ of widths – one width per wire. (The terms are borrowed from VLSI where schema represents a schematic drawing of how wires should go on a chip – the layout.) The Continuous Homotopic Routing Problem (CHRP) [16, 20, 37, 38] is to route a set of K thick paths, with widths $(w_1, \ldots, w_K)$, homotopically equivalent to $(\pi_1, \ldots, \pi_K)$; the thick paths should be pairwise-disjoint and should not intersect the holes. For our purposes, we will need to find the K locally shortest width-$\vec{w}$ thick paths homotopically equivalent to $\vec{\pi}$, i.e., solve the *Shortest* Continuous Homotopic Routing Problem. We abbreviate the problem to SCHRP and let $\textsc{schrp}(\vec{\pi}, \vec{w})$ denote the solution to the problem.

[2] In discrete network flows parlance, "saturation of edges by flow" refers to edges of the network – those that carry the flow. Edges of critical graph do not carry continuous flow (on the contrary, they define cuts through the domain); still, in the literature on continuous flows, saturation of a critical graph edge means the same thing: flow through the edge cannot be increased without violating the edge capacity [6, 40].

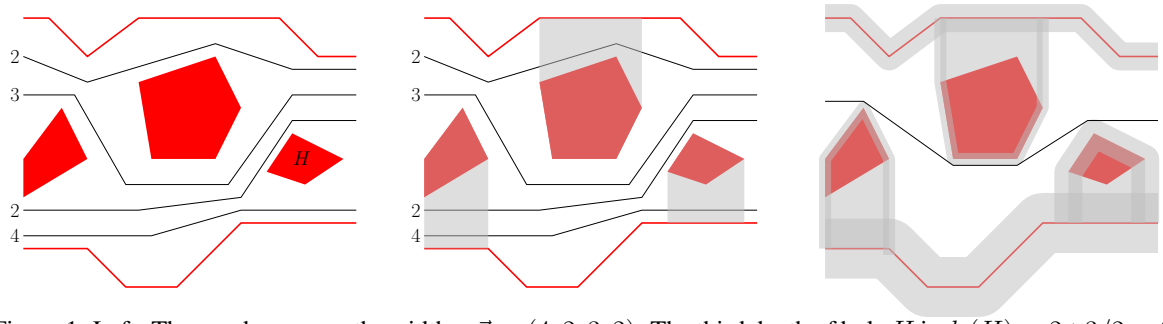


Figure 1: Left: The numbers are paths widths; $\vec{w} = (4, 2, 3, 2)$. The third depth of hole H is $d_3(H) = 2+3/2 = 3.5$. Middle: Bridge holes to $\mathcal{B}$ and $\mathcal{T}$ to enforce the correct homotopy type for the third path. Right: Inflate each hole H by $d_3(H)$ to obtain the free space for the centerline of the thick path.

We compute $\text{SCHRP}(\vec{\pi}, \vec{w})$ path-by-path using tools from [15, 26, 39, 42]. We determine "depths" of the holes w.r.t. $\vec{\pi}$ [42], i.e., for every hole H and every $k = 1 \dots K$, we determine the total width, $d_k(H)$, of paths from $\vec{\pi}$ that separate H from π_k (Fig. 1, left). Then, for each $k = 1 \dots K$, we repeat the following: bridge the holes below π_k to $\mathcal{B}$, and the holes above π_k – to $\mathcal{T}$, creating a simple polygon to which the kth thick path from $\text{SCHRP}(\vec{\pi}, \vec{w})$ will belong (Fig. 1, middle); build "conservative" obstacles [42] for (the centerline of) the kth thick path by inflating each hole H by $d_k(H)$ (Fig. 1, right); find the shortest S-T path in the free space (which is a simple splinegon) using bounded-degree splinegonal decomposition (analog of triangulation) of the free space [15, 39]. The details are in Appendix A; the final result is:

Lemma 1. *If each wire has $O(n)$ edges, $\text{SCHRP}(\vec{\pi}, \vec{w})$ can be found in $O(nh \log K + Kn \log n)$ time.*

Remark 2. A problem, closely related to SCHRP is *Fat-Edge Drawing* (FED) [17, 18]: fatten, as much as possible, a given set of non-crossing paths while keeping the thick paths non-overlapping. SCHRP is *almost* a special case of FED; the *only* difference is the presence of obstacles – [17] dealt with paths growing amidst *point* obstacles, and extending their algorithm to non-point obstacles (as we have in our case) is mentioned as an open problem in [17] (we conjecture that the extension is, in fact, possible; see, e.g., [51]). FED amidst non-point obstacles is slightly more general than our SCHRP – in the FED, the paths endpoints can be anywhere, not necessarily sliding on two segments S and T (the results on FED can be extended easily to the sliding endpoints). Our approach described above is in a sense "orthogonal" to the algorithm of [17] – we inflate the paths one-by-one, not all-at-once; the reason we chose not to use FED is efficiency – FED algorithm takes cubic time (it is possible though that for our special case one might work out a more efficient variant of FED solution).

2.2 Flow Composition and Decomposition

A discrete network flow can be composed from (and decomposed into) a set of source-sink paths in the network. We now recall analogous statements for continuous flows, leading to a reformulation of our mincost flow problem in terms of thick paths.

Flow from thick paths Let $\vec{\Pi} = (\Pi_1, \dots, \Pi_K)$ be a set of shortest thick non-crossing paths. Starting from $\vec{\Pi}$, one can define a 0/1 flow using the construction suggested by Mitchell [40]: The centerline of a shortest thick path is an alternating sequence of segments and circular arcs (segments are bitangent to adjacent arcs); consequently, a thick path is an alternating sequence of rectangular and circular parts. The flow over each rectangular part is constant; the streamlines are parallel to the segment of the centerline. In each circular part the streamlines are circular arcs cocentric with the arc of the centerline. (See Fig. 4 in Appendix B). We use $\text{FLOW}(\vec{\Pi})$ to denote the flow obtained from $\vec{\Pi}$.

Locally optimal flows are 0/1 Consider an arbitrary, not necessarily 0/1 flow $\tilde{F}$ (we will use $\tilde{F}$ to denote a not-necessarily-0/1 flow). Apply SCHRP to streamlines of $\tilde{F}$ viewed as infinitesimally-thick paths; let $\vec{\Pi} = (\Pi_1, \dots, \Pi_K)$ be the solution of the SCHRP. Each path Π_k is "glued" out of homotopically equivalent streamlines of $\tilde{F}$. (The gluing, formalized in [42, Lemma 5.2], means that k "touching" w-thick paths can be treated as a single kw-thick path; the streamlines form a continuum of 0-width paths, but

one can apply the gluing using a limiting argument as in [42, Section 5]). In their turn, the paths $\vec{\Pi}$ can be composed into a 0/1 flow $F = \text{FLOW}(\vec{\Pi})$. Clearly, the values of $\tilde{F}$ and F are the same and streamlines of $\tilde{F}$ and F have the same homotopy types: the streamlines of F are "pulled taut" streamlines of $\tilde{F}$, and the cost of F cannot be decreased without changing the homotopy type of at least one streamline. We say that F is the *locally optimal* flow given the homotopy types of streamlines; it is easy to see that F is unique (unless some paths in $\vec{\Pi}$ are straight). The total length of the streamlines of F is not larger than that of $\tilde{F}$ (as proved in [42], each streamline is as short as possible given the very existence of the other streamlines), and hence, by the coarea formula $\int_{\mathcal{D}} |F| \leq \int_{\mathcal{D}} |\tilde{F}|$. Summarizing, we have:

Lemma 2. *For any flow $\tilde{F}$ there exists the locally optimal 0/1 flow F that has the same value, whose streamlines have the same homotopy types as streamlines of $\tilde{F}$, and such that $\int_{\mathcal{D}} |F| \leq \int_{\mathcal{D}} |\tilde{F}|$.*

A different, non-constructive proof of Lemma 2 may be obtained by following arguments in [35, 49].

Thick paths from flow The inverse of the flow composition operation $\text{FLOW}(\vec{\Pi})$ is flow decomposition, asserted by the following theorem:

Theorem 3. (Continuous Flow Decomposition Theorem, [42, Theorem 5.5]) *The support of a minimum-cost flow can be decomposed into a set of shortest thick non-crossing paths.*

The theorem holds not only for the (globally) minimum-cost flow, but also for locally optimal flows; this can be seen by following the proof of the CFDT in [42]. We use $\text{PATHS}(F)$ to denote the thick paths into which (the support of) a flow F decomposes.

2.2.1 Problem reformulation

The flow composition and decomposition are inverses of each other in the sense that $\text{FLOW}(\text{PATHS}(F)) = F$ and $\text{PATHS}(\text{FLOW}(\vec{\Pi})) = \vec{\Pi}$. Moreover, the total width of the paths in $\text{PATHS}(F)$ equals the value of F, and the cost of F is equal to the area of the thick paths. Thus we may restate our minimum-cost flow problem as: ***Find a set of thick non-crossing paths with total width V, having minimum total area.***

2.3 Potential function

Any "nice" scalar function $U(x, y)$ is the *potential function* for the divergence-free flow $\tilde{F} = (\frac{\partial U}{\partial y}, -\frac{\partial U}{\partial x})$; this is a cornerstone relation in several branches of physics, including fluid dynamics. Conversely, any flow $\tilde{F}$ defines a potential U by setting the potential of a point equal to the total flow below the point: $U(p) = \int_{p\downarrow} \tilde{F}^x$ where $p\downarrow$ is the vertical ray going down from p. Because at any point (x, y), it holds that $\tilde{F}(x, y)$ is perpendicular to $\nabla U(x, y)$, streamlines of $\tilde{F}$ are level sets of U: the total flow below a point p on a streamline ℓ is the same for any $p \in \ell$ and equals the total thickness of the streamlines running below ℓ.

Relations between notions described in this section are shown with black color in the chart at the bottom of the title page.

2.4 Bottommost fill

To find a *maximum* flow in $\mathcal{D}$, Mitchell [40] proposed to fill the domain with streamlines of the flow, starting from the bottom $\mathcal{B}$ (i.e., the first streamline follows $\mathcal{B}$). For any $t \geq 0$, the *level-t* streamline ("level" in the sense of level set of the potential function of the flow) is the locus of points at distance t from $\mathcal{B}$ in the *0/1 metric* that assigns cost 1 for traveling through $\mathcal{D}$ and assigns weight 0 for traveling through holes. The bottommost fill is nicely explained in [40] using the "grassfire" analogy: Imagine that $\mathcal{D}$ is grass over which fire travels at unit speed; imagine also that the holes in the domain are made of highly flammable material so that as soon as any point on a hole is touched by fire, the whole hole is immediately ignited. Ignite $\mathcal{B}$ at time $t = 0$; then at any t the firefront will be the level-t streamline (Fig. 2). When streamlines reach a hole j, an edge ij of the "critical graph" of $\mathcal{D}$ is saturated by the flow – no more flow can be pushed between i and j, and so the flow begins running above j. (Throughout this paper we use variations of the bottommost fill, in which we start routing flow above j earlier than in the standard bottommost fill, i.e., before saturating an edge ij.)

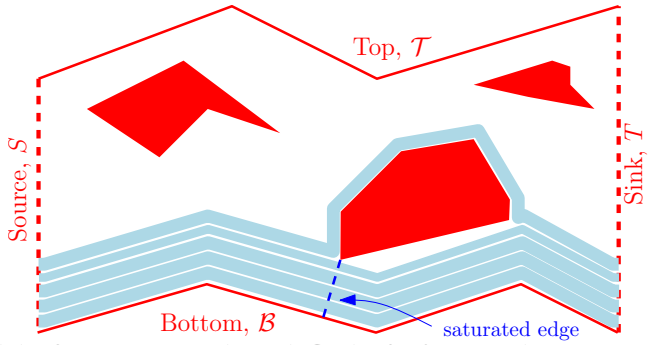


Figure 2: $\mathcal{B}$ is ignited and the fire propagates through $\mathcal{D}$; the firefronts make up streamlines of the flow. When a hole is touched by the fire, an edge of the "critical graph" of $\mathcal{D}$ is saturated – no more flow can go below the hole, and the streamlines start going around it.

Remark 3. The bottommost fill is a continuous counterpart of Ford and Fulkerson's uppermost path algorithm [19] for st-planar graphs (those with source and sink on the same face), which Hassin [25] later recognized as the planar-dual of Dijkstra's algorithm, and which Borradaile and Klein [10] recently generalized to arbitrary planar networks.

2.5 The ellipsoid method

The main technical result below is that our problem can be formulated as a convex program: minimize $\textsc{cost}(\vec{u})$, where $\textsc{cost} : \mathbb{R}^N \mapsto \mathbb{R}$ is a convex function, subject to a set $\mathcal{C}$ of linear constraints on the components of the vector $\vec{u}$. We solve the program by following the *ellipsoid method* [9, 44, 48] – an archetypal cutting-plane technique.[3] The method is based on iterative use of the following oracle: Given $\vec{u} \in \mathbb{R}^N$, check whether $\vec{u}$ is feasible (satisfies the constraints in $\mathcal{C}$); if no – exhibit a constraint from $\mathcal{C}$ violated by $\vec{u}$, if yes – evaluate $\nabla\textsc{cost}(\vec{u})$ (the gradient of the objective function at $\vec{u}$). The oracle thus outputs a hyperplane H (if $\vec{u}$ is feasible, H is perpendicular to $\nabla\textsc{cost}(\vec{u})$ – it is called the *objective-function* cut; otherwise, H is the violated constraint – it is called the *feasibility* cut), which is used to update the ellipsoid for the next iteration (the update is carried out in $O(N^2)$ time via explicit formulas).
Remark 4. If $\textsc{cost}$ is not differentiable at $\vec{u}$, a *subgradient* of $\textsc{cost}$ must be used in place of the gradient (a vector $\vec{v}$ is a subgradient of $\textsc{cost}$ at $\vec{u}$ if $\forall\vec{p}, (\vec{p} - \vec{u}) \cdot \vec{v} \leq \textsc{cost}(\vec{p}) - \textsc{cost}(\vec{u})$; e.g., if $\textsc{cost}$ is differentiable at $\vec{u}$, then $\nabla\textsc{cost}(\vec{u})$ is a unique subgradient). Our cost function $\textsc{cost}$ will be piecewise-differentiable; thus, the weak subgradient calculus (i.e., finding *one* subgradient) is particularly simple in our case: just use the gradient from any differentiable piece incident to point of non-differentiability. Section D.2 gives the details.

The rate of convergence of the ellipsoid algorithm is as follows. For a set $S \subset \mathbb{R}^N$ let $\mathrm{vol}(S)$ denote the (N-dimensional) volume of S.

Theorem 4. [44, Theorem 7.2.2] *If the feasible set $\mathcal{C}$ defines a full-dimensional polyhedron (i.e., $\mathrm{vol}(\mathcal{C}) > 0$) and if an ellipsoid E_0 containing $\mathcal{C}$ is known initially, then after $2N \ln \frac{\mathrm{vol}(E_0)}{\mathrm{vol}(\mathcal{C})\varepsilon^N}$ ellipsoid iterations a feasible solution $\vec{u} \in \mathcal{C}$ will be found such that $\textsc{cost}(\vec{u}) \leq \min_{\vec{v}\in\mathcal{C}} \textsc{cost}(\vec{v}) + \varepsilon(\max_{\vec{v}\in\mathcal{C}} \textsc{cost}(\vec{v}) - \min_{\vec{v}\in\mathcal{C}} \textsc{cost}(\vec{v}))$.*

3 The vector of hole potentials

The discussion in the previous section revolved around relations between potential functions, flows, thick paths and schemas. We now introduce a new, fifth element to the four elements shown in black in the chart at the bottom of the title page; the element and its relations to the old notions are shown in red.

[3] While the ellipsoid method is famous for its worst-case running time bounds (it was the first to prove polynomial-time solvability of LP), it is not most efficient in practice. We chose the method because of the theoretical guarantees it provides; for an implementation, one may opt for other approaches, e.g., by possibly embedding our techniques into interior-point algorithms which are known to converge, in practice, after a small number of iterations, fairly independent of problem size [11, p. 8].

Definition 5. A *potential* is a vector $\vec{u} = (u_{\mathcal{B}}, u_1, \ldots, u_h, u_{\mathcal{T}})$ of $h+2$ non-negative numbers, with $u_{\mathcal{B}} = 0, u_{\mathcal{T}} = V$. A flow $\tilde{F}$ (not necessarily 0/1) is *compatible* with $\vec{u}$ if for any hole H the amount of the flow sent below the hole is equal to its potential: $u_H = \int_{p\downarrow} \tilde{F}^x$ where p is an arbitrary point in H. Similarly, a schema $(\vec{\pi}, \vec{w})$ is *compatible* with $\vec{u}$ if for any hole H the total width of wires that run below H equals u_H. A potential $\vec{u}$ is called *compatible* if there exists a flow compatible with $\vec{u}$.

The potential vector is simply an extension of a potential function to holes (or a restriction of the function to holes, if one thinks of the potential function as being defined everywhere): since flow must vanish on any hole H, the potential function has to remain constant over H; thus, one can speak about the potential, u_H, of the whole hole. From now on, the term *potential* will mean a vector $\vec{u}$ of holes potentials (distinguishing from a *potential function* $U : \mathcal{D} \mapsto \mathbb{R}^2$). We emphasize that compatibility of flow with a potential is defined for an arbitrary flow – not necessarily a 0/1 flow; this will be crucial in proving one of our main results – convexity of the objective function in our program for mincost flow (Lemma 8).

Potential is connected to the "primal" notions of wires and flows via the compatibility relation. In particular, any schema uniquely defines holes potentials compatible with the schema (just sum up the widths of wires running below each hole). More interestingly, any compatible potential $\vec{u}$ uniquely defines homotopy types and widths of paths in a schema compatible with $\vec{u}$; the next lemma gives a constructive way to do it, based on the bottommost fill (Section 2.4):

Lemma 6. *A schema compatible with $\vec{u}$ (if one exists) can be computed in $O(nh)$ time.*

Proof. Let $\mathcal{H}$ be the set of holes with 0 potential; bridge each hole in $\mathcal{H}$ to the bottom $\mathcal{B}$ and remove the bridge from $\mathcal{D}$ – this puts the holes outside the domain and thus transforms $\mathcal{D}$ into a domain $\mathcal{D}_1$ whose bottom, $\mathcal{B}_1$, goes above the holes in $\mathcal{H}$, but still below all other holes. E.g., in Fig. 1, middle, the bridges are the gray areas below the two holes connected to $\mathcal{B}$ (the exact shape of a bridge does not matter – it can be a segment or the whole area below the hole). No wire in any schema compatible with $\vec{u}$ is allowed to go between $\mathcal{H}$ and $\mathcal{B}$. On the other hand, at least one wire must go above $\mathcal{H}$ but below all the other holes; thus at least one wire must be homotopically equivalent to $\mathcal{B}_1$. So route the first wire π_1 along $\mathcal{B}_1$, and assign width $w_1 = x_1$ to it, where x_1 is the smallest positive potential in $\vec{u}$.

Recursively, take the holes having second smallest potential x_2, bridge them to $\mathcal{B}_1$, and run a wire of width $w_2 = x_2 - x_1$ along the bottom of the new domain. Continuing this way, we obtain a schema compatible with $\vec{u}$. Since we had no flexibility in deciding the homotopy types and width of the wires (both were dictated by $\vec{u}$), our construction shows that homotopy types and widths of wires in any schema compatible with $\vec{u}$ are uniquely defined by $\vec{u}$.

As for the running time, we can find the homotopy type of one wire in linear time by removing trapezoids below 0-potential holes from the trapezoidal decomposition of $\mathcal{D}$ and finding the lowermost path in the dual graph; this is iterated for all $O(h)$ values in the potential vector. □

We denote the schema constructed in the above proof by $\textsc{schema}(\vec{u})$. Let $F = \textsc{flow}(\textsc{schrp}(\textsc{schema}(\vec{u})))$ be the flow composed from thick paths obtained by inflating $\textsc{schema}(\vec{u})$. Clearly, F is compatible with $\vec{u}$ and moreover, F is the locally optimal flow among flows whose streamlines have homotopy types given by $\textsc{schema}(\vec{u})$. From Lemma 2 we have:

Lemma 7. *Let $\tilde{F}$ be any flow compatible with $\vec{u}$. Then $\int_{\mathcal{D}} |F| \leq \int_{\mathcal{D}} |\tilde{F}|$.*

3.1 Another reformulation

The above discussion allows us to reformulate our problem as: ***Find the potential $\vec{u}$ such that the area of the support of*** $\textbf{FLOW}(\textbf{SCHRP}(\textbf{SCHEMA}(\vec{u})))$ ***is minimized***.[4] Since $\textsc{flow}(\vec{\Pi})$ is a 0/1 flow, the area

[4] We now confess that the potential function is exogenous to our method; all we need to justify the reformulation in terms of $\vec{u}$ is the directed cycle through $\vec{u}$ and F in the chart at the bottom of the title page. We nevertheless decided to keep U in the picture since it explains and "motivates" the appearance of the potential vector $\vec{u}$. On a similar note we could get along without the continuous flow decomposition theorem (the PATHS arrow in chart is not needed for the cycle through $\vec{u}$ and F) or even simply *define* a "geometric flow" as a collection of non-crossing thick paths (and use the theorem to justify such a definition); still, we believe that the gradual reformulation of the problem (first via thick paths, and then via potentials) helps to follow our thoughts.

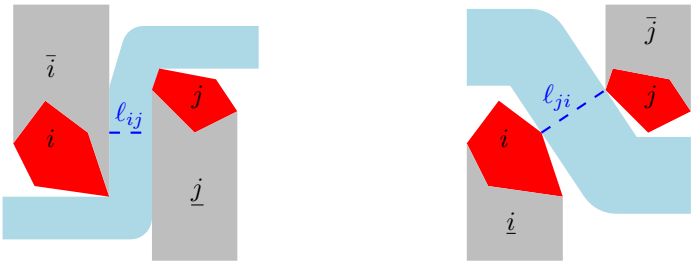


Figure 3: The maximum amount of monotone flow between i and j may be different depending on which hole is passed from above and which is from below: $\ell_{ij} \neq \ell_{ji}$. (In the standard critical graph G for general flows, the length of ij in this instance would be ℓ_{ji}.) Saturated bars are dashed.

of its support equals its integral over the domain. We thus write our problem as

$$\min \int_{\mathcal{D}} |\textsc{flow}(\textsc{schrp}(\textsc{schema}(\vec{u})))| \qquad \text{s.t.} \quad \vec{u} \text{ is compatible} \tag{2}$$

The next subsection proves that (2) is a convex program.

3.2 The convexity

We first prove convexity of the objective function in (2):

Lemma 8. *Let $\vec{u}_1, \vec{u}_2$ be two potentials; suppose each of $\vec{u}_1, \vec{u}_2$ is compatible. Let $F_i = \textsc{flow}(\textsc{schrp}(\textsc{schema}(\vec{u}_i)))$, $i = 1, 2$; let $\vec{u} = (\vec{u}_1 + \vec{u}_2)/2$, and let $F = \textsc{flow}(\textsc{schrp}(\textsc{schema}(\vec{u})))$. Then $2\int_{\mathcal{D}} |F| \leq \int_{\mathcal{D}} |F_1| + \int_{\mathcal{D}} |F_2|$.*

Proof. Let $\tilde{F} = (F_1 + F_2)/2$ (the flow $\tilde{F}$ may be non-0/1). Since F_i is compatible with $\vec{u}_i$, $\tilde{F}$ is compatible with $\vec{u}$: for any hole H and $p \in H$, $\int_{p\downarrow} \tilde{F} = (\int_{p\downarrow} F_1^x + \int_{p\downarrow} F_2^x)/2 = (u_{1H} + u_{2H})/2 = u_H$. Thus, by Lemma 7, we have: $2\int_D |F| \leq 2\int_{\mathcal{D}} |\tilde{F}| = \int_{\mathcal{D}} |F_1 + F_2| \leq \int_{\mathcal{D}} |F_1| + \int_{\mathcal{D}} |F_2|$. □

Next, we prove that the constraints in (2) are linear using a modification of the *critical graph* G of $\mathcal{D}$, which is a complete graph on holes, $\mathcal{B}$ and $\mathcal{T}$ [21]. For general (non-monotone) flows the length of an edge ij of G is equal to the distance between the holes i and j; the length upper bounds the flow that may go between i and j [40]. For monotone flows, we define the critical graph $\vec{G}$ analogously; however, $\vec{G}$ will be directed because the length of an edge ij may not equal to that of ji. Specifically, for a hole H, let $\bar{H}$ (resp. $\underline{H}$) denote H together with the area above (resp. below) H (see, e.g., gray areas in Fig. 1, middle); the length of the edge ij of $\vec{G}$, denoted ℓ_{ij}, is the distance between $\bar{i}$ and $\underline{j}$. We call edges of $\vec{G}$ *bars* (because they bar the flow, just like in the non-monotone case), and say that a bar ij is *saturated* if the flow across the bar is ℓ_{ij} (Fig. 3; see also the figure on the title page).

Suppose that $u_i > u_j$ in a potential $\vec{u}$. Since the value u_H is the total amount sent below H by a compatible flow, any flow compatible with $\vec{u}$ must send exactly $u_i - u_j$ units above j but below i (cf. proof of Lemma 6). On the other hand, the amount of flow above j but below i cannot exceed ℓ_{ij}. Thus, the bar ij enforces the constraint $u_i - u_j \leq \ell_{ij}$ on the potentials:

Lemma 9. *If $\vec{u}$ is compatible, then for every bar ij, $u_i - u_j \leq \ell_{ij}$.*

To show linearity of the constraints in (2) it remains to prove the converse of Lemma 9:

Lemma 10. *If for every bar ij $u_i - u_j \leq \ell_{ij}$, then $\vec{u}$ is compatible.*

Proof. The lemma is proved constructively with the bottommost fill (Section 2.4), similarly to Lemma 6: Every hole H with $u_H = 0$ is bridged to $\mathcal{B}$ with $\underline{H}$; let $\mathcal{B}_1$ be the bottom of the obtained domain. Offset $\mathcal{B}_1$ inside the domain by x_1 where x_1 is the smallest positive potential in $\vec{u}$, and let $\mathcal{B}_1'$ be the upper boundary of the offset. By the standard bottommost-fill arguments [6, 40] one can show that (1) x_1 units of flow can be routed between $\mathcal{B}_1$ and $\mathcal{B}_1'$, and (2) no bar ij between $i \in \mathcal{H}$ and $j \notin \mathcal{H}$ is oversaturated by the flow. Recursively, route $x_2 - x_1$ of flow between $\mathcal{B}_1'$ and the holes with the next-smallest potential x_2, and so on. □

By Lemmas 9 and 10, we can rewrite (2) as

$$\min \int_{\mathcal{D}} |\textsc{flow}(\textsc{schrp}(\textsc{schema}(\vec{u})))| \qquad \text{s.t.} \quad u_i - u_j \leq \ell_{ij} \qquad \forall (i,j) \tag{3}$$

This is the ***final reformulation*** of our problem.

4 MaxFlow/MinCut Theorems

In this section we temporarily switch attention from minimum-cost to *maximum* flows and demonstrate the power of the potentials method by proving MaxFlow/MinCut theorems – both for monotone flows and for general ones. The theorem for general flows reads (see Appendix C for our new proof):

Theorem 11. [40] *The maximum value of an S-T flow in $\mathcal{D}$ is equal to the length of the shortest $\mathcal{B}$-$\mathcal{T}$ path in the critical graph G.*

Here we state and prove a similar result for monotone flows:

Theorem 12. [Monotone MaxFlow/MinCut theorem] *The maximum value of a* monotone *S-T flow in $\mathcal{D}$ is equal to the length of the shortest $\mathcal{B}$-$\mathcal{T}$ path in the* directed *critical graph $\vec{G}$.*

Tools set up in Section 3 allow us to give a 1-line proof of Theorem 12: To formulate the maxflow problem as a math program, change the objective function in (3), so the program reads

$$\max \quad u_{\mathcal{T}} - u_{\mathcal{B}} \qquad \text{s.t.} \quad u_i - u_j \le \ell_{ij} \qquad \forall (i,j) \tag{4}$$

– but this is exactly the familiar dual program for shortest $\mathcal{B}$-$\mathcal{T}$ path in $\vec{G}$ (see, e.g., [2, program (9.9)]).

5 Applying the ellipsoid method

We now switch back to our main problem—computing mincost monotone flow—which we reformulated as a convex program (3). We use the ellipsoid algorithm to solve it (see Section 2.5 for a recap of the algorithm's essentials).

The oracle Finding a separating hyperplane amounts, at every iteration, to checking whether a given potential is feasible (compatible). If no, we do feasibility cut – either while solving the SCHRP or by explicitly checking the constraints (Appendix D.1). If yes, we do objective-function cut using automatic differentiation [8, 12, 23, 43], with weak subgradient calculus implemented via symbolic perturbation (Appendix D.2).

Feasible set volume Successful application of the ellipsoid method crucially depends on ensuring that the feasible set $\mathcal{C}$ of the program is full-dimensional, and, moreover, on exhibiting a lower bound on $\mathrm{vol}(\mathcal{C})$ (see Theorem 4). A standard way to achieve the full-dimensionality is to change the variables to a basis in the affine subspace spanned by $\mathcal{C}$; we could easily do that with our program (3). On the contrary, off-the-shelf techniques for bounding $\mathrm{vol}(\mathcal{C})$ away from 0 do not work for us: the techniques derive volume bounds from the encoding size of integers/rationals in the input (see, e.g., [47, Section 2.2]), but in our case (even though we assume that the coordinates of $\mathcal{D}$ are specified with integers) the right-hand-sides of the constraints in (3) may contain square roots since they represent lengths of edges of the critical graph $\vec{G}$. We therefore give a lower bound on $\mathrm{vol}(\mathcal{C})$ by exploiting specific properties of our problem. Specifically, we identify two cases when the feasible set of the program (3) collapses: (a) pairs of holes are forced to have the same potential because the distance between them in $\vec{G}$ is 0, and (b) the flow is a maxflow. In the first case we merge holes whose potentials are forced to be equal. In the second case we fix potentials of holes along the mincut. We prove that after the hole merging and potential fixing has been done, the feasible set (for the remaining, non-fixed potentials) is full-dimensional; we do it constructively, by exhibiting an interior point of the set. Moreover, we show that there is a non-negligible "slack" in the amount of flow sent along different-homotopy paths in a flow compatible with the point. This implies that there is a slack also in the values of compatible potentials; quantifying the slack gives a lower bound on $\mathrm{vol}(\mathcal{C})$. The details are in Appendix E; here we state the final result (recall from Section 1.1 that n, L denote the number of vertices of $\mathcal{D}$ and the largest integer coordinate of a vertex): $\mathrm{vol}(\mathcal{C}) \ge \frac{(2\varepsilon)^h}{(nh^2L)^h h!}$.

Number of iterations Theorem 4 implies that $O(h^2 \log \frac{nL}{\varepsilon})$ iterations suffice for ε-approximation: the maximum possible flow cost is L^2, we have at most h variables ($N \le h$), $\mathrm{vol}(\mathcal{C})$ is bounded as above, and a radius-L ball may be used as the starting ellipsoid E_0 (see Appendix F for the details).

Remark 5. Instead of the radius L-ball centered at the origin, as a warm-start ellipsoid we could use an approximation to the smallest-volume ellipsoid containing the feasible set; finding the optimal such

ellipsoid is hard, but finding an approximation is easy: just take a dilated copy of the largest ellipsoid inscribed into the feasible set [5, 30, 53] – by the famous result of John [28] the volume of the ellipsoid is not much larger than that of the set. Such an approach, however, would be less efficient asymptotically.

Time per iteration At each ellipsoid iteration we start from computing $\text{SCHRP}(\text{SCHEMA}(\vec{u}))$; by Lemma 1 it takes $O(nh \log n)$ time since $\text{SCHEMA}(\vec{u})$ has at most $h + 1$ wires. Feasibility of $\vec{u}$ can be checked either while solving the SCHRP or by checking the $O(h^2)$ constraints written out explicitly (Section D.1). After SCHRP is solved, the area of the thick paths can be calculated in time proportional to their complexity: since each path has $O(n)$ complexity and there are at most $h + 1$ paths, the cost of $\vec{u}$ can be computed in $O(nh)$ time; the same time is spent taking the gradient with autodiff (Section D.2). Since we have $O(h)$ variables, ellipsoid update takes $O(h^2)$ time (Section 2.5). Overall, one iteration completes in $O(nh \log n)$ time.

From the two preceding paragraphs we obtain our final result:

Theorem 13. *An ε-approximate flow can be found in $O(nh^3 \log^2 \frac{nL}{\varepsilon})$ time.*

Concluding remarks Extending the potential method for finding maximum discrete flows in planar networks, we explored the use of holes potentials for continuous flows in polygonal domains. We showed how to find a monotone flow with (almost) minimum cost, using holes potentials as variables in a convex program. We also proved MaxFlow/MinCut theorems for monotone and general flows. It would be interesting to see if the potentials method can be applied to finding non-monotone mincost flows: almost all discussion in this paper applies to the general flows as well, the only exception is that in general a potential does not define homotopy types uniquely (see, e.g., Fig. 5 in Appendix C). We believe that the general case may be solved using a "primal" program, with variables associated with edges of the visibility graph of the domain; however, proving convexity of the program seems hard in this case.

Acknowledgements We thank the anonymous referees for their suggestions, Mika Göös, Irina Kostitsyna and Gunter Röte for helpful discussions, and Joe Mitchell for introducing us to flows in continua. This work was done while the authors were affiliated with Helsinki Institute for Information Technology HIIT, CS Dept of University of Helsinki, and supported by the Academy of Finland grant 1138520 and University of Helsinki Research Funds.

References

[1] P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIJCOMP*, 29:912–953, 1999.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. Prentice Hall, 1993.

[3] N. M. Amato. An optimal algorithm for finding the separation of simple polygons. In *WADS*, 1993.

[4] E. J. Anderson, P. Nash, and A. B. Philpott. A class of continuous network flow problems. *Math Oper Res*, 7(4):501–514, 1982.

[5] K. M. Anstreicher. Improved complexity for maximum volume inscribed ellipsoids. *SIAM Journal on Optimization*, 13(2):309–320, 2002.

[6] E. M. Arkin, J. S. B. Mitchell, and V. Polishchuk. Maximum thick paths in static and dynamic environments. *CGTA*, 43(3):279–294, 2010.

[7] D. S. Atkinson and P. M. Vaidya. Using geometry to solve the transportation problem in the plane. *Algorithmica*, 13(5):442–461, 1995.

[8] C. H. Bischof and M. Bücker. Computing derivatives of computer programs. In *Modern Methods and Algorithms of Quantum Chemistry*, pages 315–327. NIC-Directors, 2000.

[9] R. G. Bland, D. Goldfarb, and M. J. Todd. The ellipsoid method: A survey. *Operations Research*, 6(29):1039–1091, 1981.

[10] G. Borradaile and P. N. Klein. An O(n log n) algorithm for maximum st-flow in a directed planar graph. *J. ACM*, 56(2), 2009.

[11] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[12] M. Bücker and P. Hovland. `http://www.autodiff.org/`.

[13] A. Cellina. On a constrained Dirichlet problem. *SIAM Journal on Control and Optimization*, 41(2):331–344, 2002.

[14] E. W. Chambers, J. Erickson, and A. Nayyeri. Homology flows, cohomology cuts. *SIAM J. Comput.*, 41(6):1605–1634, 2012.

[15] D. Z. Chen and H. Wang. Computing shortest paths among curved obstacles in the plane. *CoRR*, abs/1103.3911, 2011.

[16] R. Cole and A. Siegel. River routing every which way, but loose. In *FoCS*, 1984.

[17] C. A. Duncan, A. Efrat, S. G. Kobourov, and C. Wenk. Drawing with fat edges. *Int. J. Found. Comput. Sci.*, 17(5):1143–1164, 2006.

[18] A. Efrat, S. Kobourov, M. Stepp, and C. Wenk. Growing fat graphs. In *SoCG*, 2002.

[19] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, 1956.

[20] S. Gao, M. Jerrum, M. Kaufman, K. Mehlhorn, and W. Rülling. On continuous homotopic one layer routing. In *SoCG*, pages 392–402, 1988.

[21] L. Gewali, A. Meng, J. S. B. Mitchell, and S. Ntafos. Path planning in $0/1/\infty$ weighted regions with applications. *ORSA J. Comput.*, 2(3):253–272, 1990.

[22] J. Glimm and D. H. Sharp. Complex fluid mixing flows. *SIAM News*, 39(5), 2006.

[23] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Frontiers in Applied Mathematics Series. SIAM, 2000.

[24] D. Halperin and E. Packer. Iterated snap rounding. *Comput. Geom.*, 23(2):209–225, 2002.

[25] R. Hassin. Maximum flow in (s, t) planar networks. *Inf. Process. Lett.*, 13(3):107, 1981.

[26] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *CGTA*, 4:63–98, 1994.

[27] H. v. d. Holst and J. C. d. Pina. Length-bounded disjoint paths in planar graphs. *Discr. Appl. Math.*, 120(1-3):251–261, 2002.

[28] F. John. Extremum Problems with Inequalities as Subsidiary Conditions. In K. O. Friedrichs, O. E. Neugebauer, and J. J. Stoker, editors, *Studies and Essays: Courant Anniversary Volume*, pages 187–204. Wiley-Interscience, New York, 1948.

[29] H. Kaplan and Y. Nussbaum. Min-cost flow duality in planar networks. *CoRR*, abs/1306.6728, 2013.

[30] L. Khachiyan. On the complexity of approximating extremal determinants in matrices. *J. Complexity*, 11(1):138–153, 1995.

[31] S. Khuller, J. Naor, and P. N. Klein. The lattice structure of flow in planar graphs. *SIAM J. Discrete Math.*, 6(3):477–490, 1993.

[32] J. Kim, J. S. B. Mitchell, V. Polishchuk, S. Yang, and J. Zou. Routing multi-class traffic flows in the plane. *CGTA*, 45(3):99–114, 2012.

[33] R. Kohn and G. Strang. Personal communication.

[34] R. Kohn and G. Strang. Optimal design and relaxation of variational problems. *Communications on Pure and Appl. Math.*, 39(1):113–137, 1986.

[35] R. Kohn and G. Strang. The constrained least gradient problem. In R. Knops and A. Lacey, editors, *Nonclassical Continuum Mechanics*, pages 226–243. Cambridge Univ. Press, 1987.

[36] J. Krozel, J. S. B. Mitchell, V. Polishchuk, and J. Prete. Capacity estimation for level flight with convective weather constraints. *Air Traffic Control Quarterly*, 15(3):209–238, 2007.

[37] C. E. Leiserson and F. M. Maley. Algorithms for routing and testing routability of planar VLSI layouts. In *SToC*, 1985.

[38] F. M. Maley. *Single-Layer Wire Routing and Compaction*. MIT Press, 1990.

[39] E. A. Melissaratos and D. L. Souvaine. Shortest paths help solve geometric optimization problems in planar regions. *SIAM J. Comput.*, 21(4):601–638, 1992.

[40] J. S. B. Mitchell. On maximum flows in polyhedral domains. *J. Comp. Syst. Sci.*, 40:88–123, 1990.

[41] J. S. B. Mitchell and C. H. Papadimitriou. The weighted region problem: finding shortest paths through a weighted planar subdivision. *J. ACM*, 38:18–73, 1991.

[42] J. S. B. Mitchell and V. Polishchuk. Thick non-crossing paths and minimum-cost flows in polygonal domains. In *SoCG*, pages 56–65, 2007. Full version: `http://webstaff.itn.liu.se/~valpo40/pages/thesis.pdf`, Ch. 1–5.

[43] U. Naumann. *The Art of Differentiating Computer Programs: An Introduction to Algorithmic Differentiation*. Number 24 in Software, Environments, and Tools. SIAM, Philadelphia, PA, 2012.

[44] A. Nemirovski. Standard numerical methods for nonlinear continuous optimization. Technical report, Technion – Israel Institute of Technology, Minerva Optimization Center, 1999.

[45] S. Neumayer, A. Efrat, and E. Modiano. Geographic max-flow and min-cut under a circular disk failure model. In *INFOCOM, 2012 Proceedings IEEE*, pages 2736–2740. IEEE, 2012.

[46] A. Orda and R. Rom. On continuous network flows. *Oper. Res. Letters*, 17, 1995.

[47] S. Rebennack. Ellipsoid method. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, pages 890–899. Springer, 2009.

[48] N. Z. Shor. The development of numerical methods for nonsmooth optimization in the USSR. In *History of Math. Prog.*, pages 135–139. North-Holland, 1991.

[49] P. Sternberg, G. Williams;, and W. P. Ziemer. The constrained least gradient problem in R^n. *Transactions of the American Mathematical Society*, 339(1):403–432, 1993.

[50] G. Strang. Maximal flow through a domain. *Math. Program.*, 26:123–143, 1983.

[51] K. Verbeek. Homotopic c-oriented routing. In W. Didimo and M. Patrignani, editors, *Revised Selected Papers from Graph Drawing 2012*, volume 7704 of *LNCS*, pages 272–278. Springer, 2013.

[52] S. Yang, J. S. B. Mitchell, J. Krozel, V. Polishchuk, J. Kim, and J. Zou. Flexible airlane generation to maximize flow under hard and soft constraints. *Air Traffic Control Quarterly*, 19:1–26, 2011.

[53] Y. Zhang and L. Gao. On numerical solution of the maximum volume ellipsoid problem. *SIAM Journal on Optimization*, 14(1):53–76, 2003.

Appendix

A SCHRP algorithm

For $k \in \{1 \dots K\}$, let π_k be the kth wire in $\vec{\pi}$, and let $\bar{\pi}_k$ be the centerline of the kth thick path in the solution to the SCHRP (so $\bar{\pi}_k$ is homotopically equivalent to π_k). For a hole H, let $d_k(H)$ denote the *k-th depth* of H, which is equals $w_k/2$ plus the total width of those paths from $\vec{\pi}$ that separate H from π_k (Fig. 1, left). We compute the depths hole-by-hole. Specifically, assume the paths in $\vec{\pi}$ are numbered 1 through K in the order they appear along S from bottom to top. Add the bottom $\mathcal{B}$ of the domain to the set $\vec{\pi}$ as path number 0: $\pi_0 = \mathcal{B}$. Add the top $\mathcal{T}$ as path number $K+1$: $\vec{\pi}_{K+1} = \mathcal{T}$. For $i, j \in \{0, \dots K+1\}$ such that $i < j$, let $\mathcal{D}_i^j$ be the part of $\mathcal{D}$ between π_i and π_j. For each hole H we determine in which polygon $\mathcal{D}_l^{l+1}$ it lies; this is enough to infer $d_k(H)$ (total width of paths lying between π_k and H) in constant time (assuming the total widths of paths from i to j were precomputed for all i, j, which takes $O(h^2)$ time). We determine l by binary search. There are $\log K$ steps in the search, and one step is a point-in-polygon query "Does an arbitrary point from H lies inside a polygon D_i^j for some i, j?" If each path in $\vec{\pi}$ has $O(n)$ edges, then D_i^j has $O(n)$ vertices, so the query can be answered in $O(n)$ time. Overall we spend $O(h^2)$ to precompute partial sums of paths widths, $O(n \log K)$ time per hole to deptermine in which polygon it lies, and then $O(1)$ time per hole H per k to determine the depth $d_k(H)$; altogether computing the depths takes $O(h^2 + Kh + nh \log K) = O(Kh + nh \log K)$ time.

To define the free space for the path $\bar{\pi}_k$, we bridge the holes below π_k to $\mathcal{B}$, and the holes above π_k – to $\mathcal{T}$ (Fig. 1, middle). Let $\mathcal{O}_k$ denote the set of obstacles obtained after inflating all holes by their kth depths ($\mathcal{O}_k$ are called *conservative* obstacles in [42] since each hole H is inflated so as to leave just enough space for the appropriate thick paths to squeeze in between H and the kth thick path, $\langle \bar{\pi}_k \rangle^{(w_k/2)}$, from the SCHRP solution); Fig. 1, right. In [42] it was shown that $\bar{\pi}_k$ is the shortest S-T path (homotopically equivalent to π_k) that avoids $\mathcal{O}_k$. (The necessity of $\bar{\pi}_k$ avoiding $\mathcal{O}_k$ is obvious – otherwise there is not enough space for the thick paths to squeeze in, without overlap, between the thick path $\langle \bar{\pi}_k \rangle^{(w_k/2)}$ and some hole; [42] also showed the sufficiency.) It was also shown in [42] that $\mathcal{O}_k$ is a union of $O(n)$ pseudodisks, and thus the free space for $\bar{\pi}_k$ is an $O(n)$-complexity splinegon which can be built in $O(n \log n)$ time.

B Flow from thick paths

C Proof of Theorem 11

Unlike in the monotone case, for non-monotone flows a potential does not define a compatible schema uniquely (Fig. 5). In fact, it is not straightforward even to define the very notion of "above/below" for flows and obstacles. To remedy this we consider *circulations*, i.e., flows whose every streamline is a (simple) closed curve. For that we imagine that the source S and the sink T are connected with a tube running outside of the domain. Now we can have a non-zero circulation around any hole and around the top $\mathcal{T}$; however no streamline may go around the bottom $\mathcal{B}$. We consider only circulations in which the flow goes counterclockwise along every streamline. There is an obvious one-to-one correspondence between flows in the original domain and circulations in the domain with the tube; in particular, the value

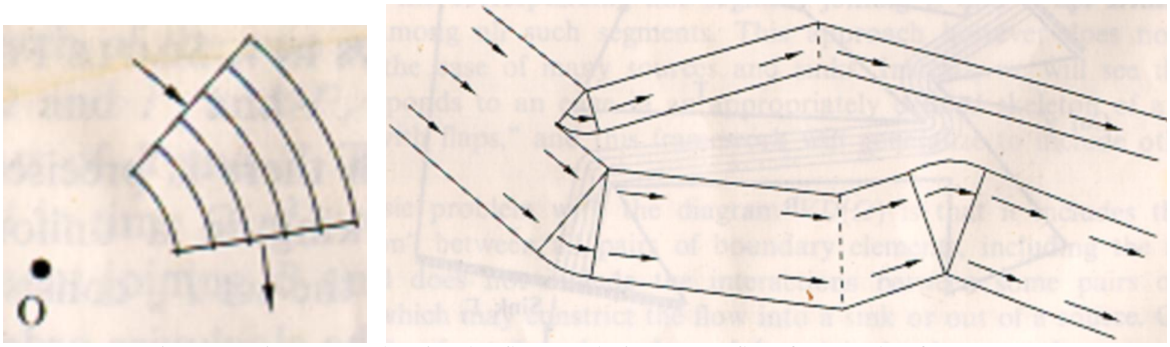


Figure 4: Figure 4.2 (the circular flow) and Figure 4.3 (flow from paths) from [40].

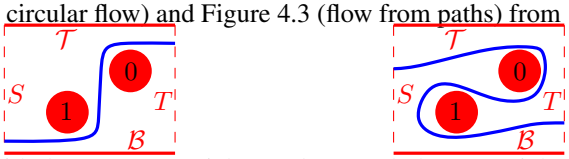


Figure 5: Two schemas compatible with the same potential. Numbers are holes potentials; $u_{\mathcal{B}} = 0, u_{\mathcal{T}} = 1$.

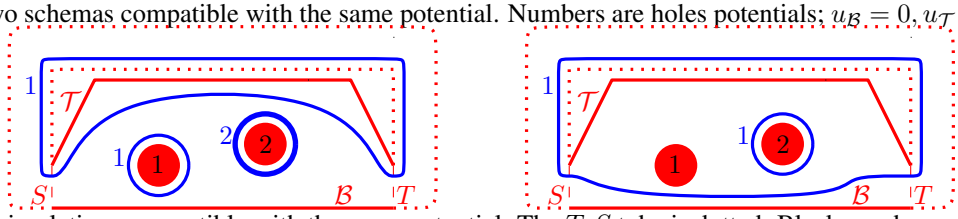


Figure 6: Two circulations compatible with the same potential. The T-S tube is dotted. Black numbers are holes potentials; $u_{\mathcal{B}} = 0, u_{\mathcal{T}} = 1$. Blue numbers are widths of the closed thick paths.

of a circulation is still defined as the total amount of flow that comes in though the source S (this is, of course, equal to the flow through the tube).

With the above, we can give a new definition of compatibility between flows and potentials:

Definition 14. A *potential* is a vector $\vec{u} = (u_{\mathcal{B}}, u_1, \ldots, u_h, u_{\mathcal{T}})$ of $h + 2$ non-negative numbers, with $u_{\mathcal{B}} = 0$. A flow $\tilde{F}$ is *compatible* with $\vec{u}$ if the total flow around each hole H is u_H (Fig. 6).

The analog of Lemma 9 clearly holds (with ℓ_{ij} being the length of edge ij in the undirected critical graph G):

Lemma 15. *If there exists a circulation compatible with $\vec{u}$, then for every pair $\{i, j\}$ of holes $u_i - u_j \leq \ell_{ij}$.*

Indeed, $|u_i - u_j|$ units of a compatible flow must go between holes i and j; if $|u_i - u_j| > \ell_{ij}$, there is not enough space for the flow to squeeze in between the holes.

It remains to prove the converse of Lemma 15:

Lemma 16. *If $|u_i - u_j| \leq \ell_{ij}$ for every pair $\{i, j\}$ of holes, then there exists a circulation compatible with $\vec{u}$.*

Proof. The lemma is proved constructively with a kind of "bottommost fill"; we put bottommost fill in quotes because our proof does not follow the vanilla bottommost fill paradigm (Section 2.4): instead of filling $\mathcal{D}$ with flowlines that run close to the bottom $\mathcal{B}$, we route the flowlines (or, actually, circulationlines) close to *every* hole *simultaneously*. In particular, our "bottommost fill" does not treat $\mathcal{B}$ (nor $\mathcal{T}$) differently from any other hole.

If we ignore the capacity constraints, one way to create a flow compatible with $\vec{u}$ is to circulate flow u_H counterclockwise around every hole H.

Let us now take capacity constraints (i.e., distances between holes) into account. If H is the only hole, to circulate flow W around H, it is sufficient to have space $\langle H \rangle^{(W)}$ for the flow – the flowlines are the (outer) boundaries of sets $\langle H \rangle^{(w)}$ for w varying from 0 to W.

Now consider two holes, H_1 and H_2 with potentials u_1 and u_2 and distance $\ell \geq |u_1 - u_2|$. If $u_1 + u_2 \leq \ell$, we may just circulate the flows around the holes separately. Otherwise we have to circulate flow $u_i - c$ around hole H_i and flow c around both holes, for some amount c. Since the flows may not overlap, we should have $(u_1 - c) + (u_2 - c) \leq \ell$, or $2c \geq u_1 + u_2 - \ell$. If we choose $c = \frac{u_1+u_2-\ell}{2}$,

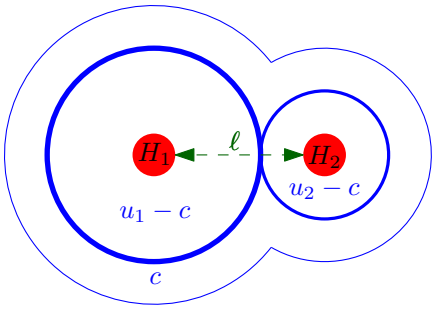


Figure 7: $(u_1 - c) + (u_2 - c) = \ell$; $u_i - c$ of flow goes around H_i, and c of flow – around both holes.

we may circulate the flow around H_1 and H_2 inside the region $\langle H_1 \rangle^{(u_1)} \cup \langle H_2 \rangle^{(u_2)}$: first circulate flow $u_i - c$ around H_i and then route c flow around the resulting connected shape (Fig. 7).

When we have several holes (as well as bottom $\mathcal{B}$ and top $\mathcal{T}$) and a potential vector $\vec{u}$, we may use a similar strategy of inflating and merging the holes to form a flow compatible with $\vec{u}$. Let $V = u_{\mathcal{T}}$ be the potential of the top. Increase the flow around holes continuously. Start inflating hole i at time $t_i = V - u_i$, and inflate the hole at unit speed. At time t, hole i is inflated to $\langle i \rangle^{(w_i(t))}$ where $w_i(t) = \max(0, t - t_i)$, and flow $w_i(t)$ circulates around the hole. When two inflated holes bump into each other we merge them into one and start increasing flow around the merged hole; refer to Fig. 7. (Note: the time t starts at $-\infty$ and not at 0 because some holes may have potential higher than V).

Since we start increasing flow around hole i at time $V - u_i$, at time V the hole has potential u_i as required. We now prove that our construction does not get stuck before time V. First of all observe that the only possible times when we cannot continue the inflation are "events" time, when holes bump into each other. The events are of 2 types: (1) when two inflated holes collide (each possibly being a merger of few other holes), and (2) when an inflated hole (possibly a merger of others) bumps into a hole H_i that has not yet started to inflate. (Assume w.l.o.g. that all event times are distinct.)

We cannot get stuck at a type-1 event: when the event happens we just merge the holes and continue routing flow around the merged holes. Now consider a type-2 event: say, an inflated hole j bumps into a non-inflated hole i. The event time t is less than $V - u_i$ – the time when i will start inflating (so at time t no flow goes around i yet, and we get stuck since we cannot yet merge the holes and start routing flow around the merger; in other words, hole j started inflating too early because the potential u_j is too high). But at time t, the hole j has been inflated only by the width $w_j(t) = t - t_j = (V - u_i) - (V - u_j) = u_j - u_i \le \ell_{ij}$; thus by the time t, the inflated j could not have reached i – a contradiction. □

From Lemmas 15 and 16, and observing that the value of the flow is equal to the circulation $u_{\mathcal{T}}$ around the top $\mathcal{T}$, we have that the maximum flow problem may again be formulated as the program (4) (with ℓ_{ij} being the length of edge ij in the critical graph G). This finishes our proof of Theorem 11.

D Oracle details

We describe how to find a separating hyperplane, which amounts to checking at every iteration whether a given potential $\vec{u}$ is feasible (compatible). If $\vec{u}$ is not compatible, we have to produce a feasibility cut, otherwise – produce an objective-function cut.

D.1 Feasibility cut: on-the-fly in SCHRP or by explicit constraints check

The ellipsoid algorithm (just like any other separation-based method) does not need to have the constraints listed upfront; all that is needed is a separation oracle that produces a violated constraint for an infeasible point. In our case, the oracle can be embedded into the SCHRP algorithm: if we are not able to route the centerline for the kth thick path (for some $k = 1 \dots K$), we detect which bridged-and-inflated holes $\langle \underline{i} \rangle^{(d_k(i))}$, $\langle \bar{j} \rangle^{(d_k(j))}$ (refer to Fig. 1, right) have separated the source from the sink (because $\langle \underline{i} \rangle^{(d_k(i))} \cap \langle \bar{j} \rangle^{(d_k(j))} \neq \emptyset$); this tells us that the bar ij is oversaturated by the would-be flow compatible with the current potential, which gives us the violated constraint $u_i - u_j \le \ell_{ij}$. Still, if one wants to

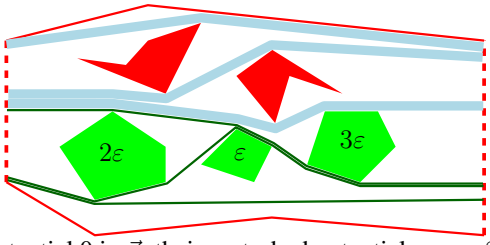


Figure 8: Lightgreen holes have potential 0 in $\vec{u}$; their perturbed potentials are $\varepsilon, 2\varepsilon, 3\varepsilon$. The flow obtained from the perturbed potential has green paths, each of thickness ε, in addition to the blue paths that were obtained from $\vec{u}$. We compute the cost of all the paths (blue+green) as a function of ε; the gradient of the function is also a function of ε. The subgradient at $\vec{u}$ is the limit of that function at $\varepsilon = 0$.

write the constraints of (2) explicitly, one can build the areas $\underline{H}, \bar{H}$ for each hole H (overall $O(nh)$ time trivially), and compute the values ℓ_{ij} for all i, j in total $O(nh)$ time using the linear-time algorithm for finding the distance between simple polygons [3] (the total running time is $O(nh)$ because it is $O(\sum_i \sum_j (n_i + n_j)) = O(2\sum_i \sum_j n_j) = O(2\sum_i n) = O(hn)$ where n_j is the complexity of hole j).

D.2 Objective-function cut: via automatic differentiation

We do not know a general formula for the cost, $\int_{\mathcal{D}} |\textsc{flow}(\textsc{schrp}(\textsc{schema}(\vec{u})))|$, of a potential $\vec{u}$ (i.e., formula that would hold for any potential $\vec{u}$), nor do we know how to split the potentials space into cells such that within one cell the formula would be the same; this is because the formula for the cost varies depending on the homotopy types of the thick paths $\textsc{schrp}(\textsc{schema}(\vec{u}))$ and on the vertices of $\mathcal{D}$ on which the paths bend. Nevertheless, for a given potential $\vec{u}$ computation of the cost is a program producing the cost formula valid in some neighborhood of $\vec{u}$; thus the gradient of the cost can be obtained using *automatic differentiation with back-propagation (autodiff)*, which takes asymptotically the same time as computing the cost itself [8, 12, 23, 43].

In the autodiff, computation is viewed as calculation of a sequence of values $x_1, \ldots, x_N$, where N is the number of steps in the algorithm and x_N is the output. In our case, x_N is the cost of $\vec{u}$, and the first values $(x_1, \ldots, x_h) = \vec{u}$ are the input to the algorithm. Each other value x_i is a temporary value computed from some set $\phi(i)$ of the earlier values. The derivative of x_N w.r.t. x_i is taken using the chain rule ($\partial x_N/\partial x_i = \sum_{j:i\in\phi(j)} \partial x_N/\partial x_j \cdot \partial x_j/\partial x_i$); this way all the derivatives of the output can be computed in reverse order from N to 1 starting from the base case $\partial x_N/\partial x_N = 1$. Autodiff splits the computation into small steps such that during one step the value x_j is computed from the previous values using a constant-size formula, so that the partial derivative $\partial x_j/\partial x_i$ can be also computed in constant time. Thus overall the gradient of the cost can be computed within the same time bound as computing the cost itself.

Weak subgradient calculus (finding a subgradient)

It remains to show how to compute a subgradient of the cost at a point where the cost is not differentiable. Let us look closer at when we may need to do this.

First, if no two holes have equal potentials in $\vec{u}$, then small changes in the potential will not lead to appearance or disappearance of paths in the solution; hence the formula for the cost will locally stay the same, implying that cost is differentiable at $\vec{u}$ (in this abstract we ignore the degenerate case when the boundary of a thick path touches a vertex of $\mathcal{D}$ but does not bend on it, so a small change in the path thickness changes the formula for the cost). As a corollary we obtain that non-differentiable points have measure 0 in the space of potentials. Next, if $\vec{u}$ is on the boundary of the feasible set, then at least one bar is saturated, i.e., $u_i - u_j = \ell_{ij}$ for some i, j, and we may use the constraint $u_i - u_j \le \ell_{ij}$ as a feasibility cut. It follows that we may need to compute a subgradient only when $\vec{u}$ is strictly inside the feasible set (and when some holes have equal potentials). But when $\vec{u}$ is strictly feasible, a small perturbation keeps the potential feasible, and since non-differentiable points have measure 0, the perturbation will "likely" move the potential to a differentiable point.

We could have stopped at the above argument and say that we apply "small random" perturbation

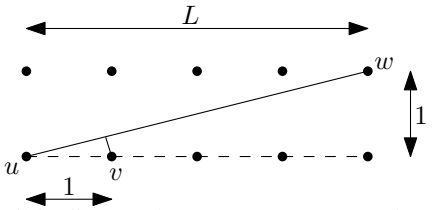


Figure 9: The smallest possible distance between a vertex v and an edge uw is $\frac{1}{\sqrt{L^2+1}} > \frac{1}{2L}$.

to compute the gradient at the perturbed potential and use it as an approximation to the subgradient at the non-differentiable potential $\vec{u}$. Such an option, however, would bring further questions: How small should the perturbation be, what kind of randomness to use, etc. To avoid dealing with such questions, we perform the perturbation symbolically, obtaining an *exact* subgradient at $\vec{u}$ as the limit of the gradient at a differentiable point close to $\vec{u}$, as detailed next.

As observed above, the non-differentiability stems from equal-potential holes in $\vec{u}$. Assume for simplicity that a set $\mathcal{H}$ of holes has potential 0 in $\vec{u}$, and the other holes potentials are distinct; our method extends directly to the general case. To break the equality of potential, assign values $\varepsilon, 2\varepsilon, \ldots, |\mathcal{H}|\varepsilon$ to the holes in $\mathcal{H}$ arbitrarily; this moves $\vec{u}$ to a perturbed point $\vec{u}^\varepsilon$ (Fig. 8). Compute the cost of the perturbed solution; during the computation treat ε symbolically, as a variable; in particular, treat it as an input variable for the autodiff. Since in $\vec{u}^\varepsilon$ the potentials of all holes are different, the cost is differentiable at $\vec{u}^\varepsilon$; using autodiff, we obtain the gradient of the cost at $\vec{u}^\varepsilon$ as a function of ε. The limit of this function at $\varepsilon = 0$ is the desired subgradient at $\vec{u}$.[5]

E Feasible set volume: the details

We start from a useful observation that follows from integrality of coordinates of vertices of $\mathcal{D}$; the proof is in Figure 9 (see also [24, Section 2]):

Lemma 17. *The smallest non-0 distance between holes in $\mathcal{D}$ is larger than $\frac{1}{2L}$.*

The next two subsections identify two cases in which $\mathrm{vol}(\mathcal{C}) = 0$ and describe what to do if these cases are encountered. In the last subsection we prove that these are the only cases in which the feasible set is not full-dimensional, and bound $\mathrm{vol}(\mathcal{C})$ away from 0. We also introduce the "resolution" of $\frac{\varepsilon}{nL}$ with which we look at V; the resolution allows us to ignore cases when V is too small or when V is too close to the maxflow.

E.1 $\ell_{ij} = \ell_{ji} = 0 \Longrightarrow$ Holes merging

One reason why the feasible set of program (3) may collapse is because there may be holes i, j with both $\underline{i} \cap \bar{j} \neq \emptyset$ and $\underline{j} \cap \bar{i} \neq \emptyset$, which implies $\ell_{ij} = \ell_{ji} = 0$ and enforces $u_i = u_j$. To deal with this we merge such holes into one by a vertical segment (Fig. 10) and have a single variable in our program for the potential of the merged hole. Algorithmically, we can discover such cases in $O(nh^2)$ time in a preprocessing step: build $\underline{H}, \bar{H}$ for every holes H (linear time), compute ℓ_{ij} for all tuples (i, j) of holes ($O(nh)$ time, see end of Section D.1), merge each tuple with $\ell_{ij} = \ell_{ji} = 0$; repeat (there are at most h repetitions because at least 2 holes merge at each).

[5]The same technique could be applied, instead of the autodiff, also at a point where cost is differentiable: To take partial derivative of cost w.r.t. u_H, change u_H to $u_H + \varepsilon$; this will change the widths of the thick paths Π_k, Π_{k+1} from SCHRP(SCHEMA($\vec{u}$)) that are immediately above and below the hole H (width ε will be flipped from one of the paths to the other). Write the formula for the total area of the new collection of thick paths (note that changing the widths of Π_k, Π_{k+1} may influence areas of the other paths in the collection, just like in Fig. 8 green paths influence the original, blue ones); again, treat ε symbolically. The formula will consist of $O(nh)$ terms (one term per segment or circular arc in the centerline of each thick path); each term is an elementary function. Differentiate the formula w.r.t. ε, and substitute $\varepsilon = 0$ to obtain the gradient. Such approach, however, will take $O(nh \log n)$ time per hole H (at least when implemented directly, hole-by-hole); thus overall, autodiff outperforms this by a factor of h.

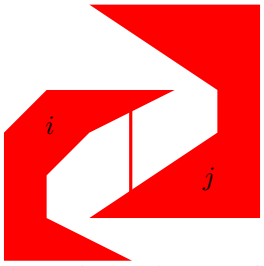


Figure 10: No monotone flow can go between i and j, so the potentials of the holes must be the same; we merge the holes into one.

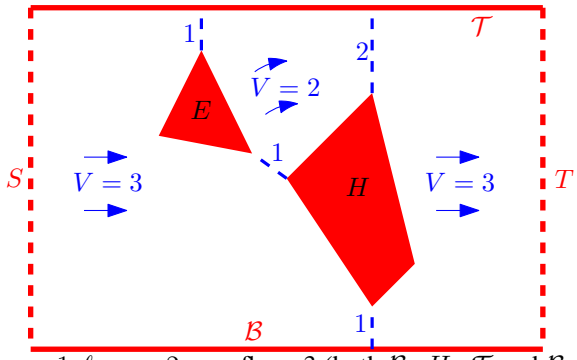


Figure 11: $\ell_{H\mathcal{B}} = \ell_{EH} = \ell_{\mathcal{T}E} = 1, \ell_{\mathcal{T}H} = 2$; maxflow=3 (both $\mathcal{B}-H-\mathcal{T}$ and $\mathcal{B}-H-E-\mathcal{T}$ are mincuts). The problem splits into three: finding mincost flows of value 3 from S to $\mathcal{B}-H-E-\mathcal{T}$ and from $\mathcal{B}-H-\mathcal{T}$ to T, and mincost flow of value 2 from $\mathcal{B}-H-E-\mathcal{T}$ to $\mathcal{B}-H-\mathcal{T}$.

E.2 V = maxflow $\Longrightarrow$ Domain splitting

Another (in fact, the other, as we prove next) case when the feasible set has 0 volume is when V is equal to the maximum flow value. In this case potentials along a mincut (i.e., potentials of the holes lying on a shortest $\mathcal{B}$-$\mathcal{T}$ path in the critical graph $\vec{G}$) are locked to their shortest-path distances from $\mathcal{B}$ in $\vec{G}$; we therefore set these potentials and no longer treat them as variables in our program. In addition, we split our problem into two: finding mincost flow from S to the mincut and finding mincost flow from the mincut to T; since any maxflow crosses the mincut perpendicularly to the mincut's bars, the solutions to the subproblems match at the mincut. We then check if $\mathcal{D}$ has still another mincut in each of the subproblems (it is possible that the domain has several mincuts); if it is the case, we recursively subdivide each subproblem. Note that if $\mathcal{D}$'s mincuts overlap, the target flow value in a subproblem may be less than V; refer to Fig. 11.

After potentials of the holes along the mincut(s) are fixed, these potentials are no longer variables in our program, i.e., we treat only the potentials of the other holes as variables in the program. We call the holes with non-set potentials *free*.

Overall we have at most $h + 1$ subproblems and the total number of free holes in all subproblems is at most h. Since subproblems are completely independent, the lower bound on the volume of the feasible set of a subproblem carries over to the original problem (it does not matter whether we solve every subproblem separately or whether we solve the original problem whose feasible set is the Cartesian product of the feasible sets for the subproblems). From now on we concentrate on a single subproblem. Abusing the notation, we will use h, V to denote the number of free holes and the target flow value within the subproblem and in general, we will call the subproblem just *the problem*; e.g., $\mathcal{D}$ will denote the domain for the subproblem, etc. There are two exceptions: n, L still denote the number of vertices and the largest integer coordinate in the original, unsplit domain.

E.3 Very large or very small V

Before giving the lower bound on the volume of the feasible set, we get rid of two more cases. In the first one the (sub)problem is "complicated" and our bound on $\mathrm{vol}(\mathcal{C})$ is not good enough; we split the problem further into two subproblems (while increasing the cost of the solution by at most 2ε). In the

second case the (sub)problem is "simple" but there is no bound on $\mathrm{vol}(\mathcal{C})$; we do not solve the problem at all (again, increasing the cost by at most 2ε).

$V >$ maxflow$-\frac{\varepsilon}{nL} \Longrightarrow$ Further splitting Let M be the value of the maximum flow (equivalently, the mincut length) in $\mathcal{D}$. Below, we will obtain a lower bound on $\mathrm{vol}(\mathcal{C})$ in terms of $M - V$. Even though we know that $V < M$, our bound may be arbitrarily bad (i.e., arbitrarily close to 0) because V may be arbitrarily close to M. To address this we impose a "resolution" of $\frac{\varepsilon}{nL}$ on V. Specifically, if $V > M - \frac{\varepsilon}{nL}$, we fix the potentials along the mincut and split the domain along it (just as we did above in the case $V = M$). Due to the possibly suboptimal setting of the potentials, some flow may go on the "wrong side" of some holes in comparison with the optimal solution. However, since the total amount of the wrongly routed flow is at most $\frac{\varepsilon}{nL}$, the total increase in the cost of the solution is at most 2ε – (the centerlines of) wrongly routed paths have total length at most $2nL$ (they consist of at most n segments and n arcs, each of length at most L) and hence the total area of the paths is at most $\frac{\varepsilon}{nL} 2nL$. (This argument is equivalent to saying that the cost is a Lipschitz function – a standard condition used to bound the approximation error in convex optimization.)

$V < \frac{\varepsilon}{nL} \Longrightarrow$ Arbitrary flow To obtain a lower bound on the volume of the feasible set we must get rid of (sub)problems in which V is very small (because, obviously, $V \to 0$ implies $\mathrm{vol}(\mathcal{C}) \to 0$). Specifically, if $V < \frac{\varepsilon}{nL}$ we route the flow arbitrarily (e.g., using the bottommost fill). Since we are routing so little flow, it will have little effect on the cost – as above, routing $\frac{\varepsilon}{nL}$ of flow incorrectly may result in the cost increase of at most 2ε.

Remark 6. Even though the discussion above focused on a single subproblem, in fact, the resolution brings up an additional error of at most 2ε *overall* for all subproblems; this is because n, L relate to the original, unsplit domain $\mathcal{D}$. (We could also multiply the resolution error by $h+1$ – the maximum possible number of subproblems; this would have no effect on the asymptotic running time of our algorithm.)

E.4 The lower bound

We are finally ready to show that (under the above assumptions – holes merged, problem split, $\frac{\varepsilon}{nL} < V < M - \frac{\varepsilon}{nL}$) the volume of the feasible set can be lower-bounded:

Lemma 18. $\mathrm{vol}(\mathcal{C}) \geq \frac{(2\varepsilon)^h}{(nh^2L)^h h!}$.

Proof. We use bottommost fill with "early jump-overs" to (quantifiably) undersaturate bars and to ensure that all holes have (quantifiably) different potentials. Assume first that in $\vec{G}$ the (shortest-path) distances from $\mathcal{B}$ to all holes are distinct. Let $\delta = \frac{\varepsilon}{nhL}$. Perform the bottommost fill, but start routing on the other side of a hole as soon as either (a) a bar leading to the hole is δ short of saturation, or (b) the total amount of flow that remains to be routed is $\delta(h' - 1)$ where h' is the number of holes that are currently above the flow (Fig. 12). That is, while a lot of flow remains to be routed (more than $\delta(h' - 1)$), keep routing as in (a) – which is exactly the bottommost fill but with the capacity of each bar *reduced* by δ. When $\delta(h' - 1)$ of flow remains to be routed, switch to (b) and never come back to (a): route δ of flow and jump over the hole; this is repeated $h' - 1$ times (such routing is still like bottommost fill but with the capacity of each bar *equal* to δ). Since $V < M - \frac{\varepsilon}{nL}$, routing as in (a) is possible (even if we never get to (b)); since $V > \frac{\varepsilon}{nL}$, there is enough flow to do (b) (even if never do (a) and start from (b) right away), and since $\delta < \frac{1}{2L}$ there is enough space between holes to route as in (b) – by Lemma 17, δ is smaller than the minimum distance between holes.

Let $\vec{u}$ be the potential compatible with the flow. By construction, it is possible to flip up to δ flow from one side of any hole to the other (by Lemma 17, 2δ is still smaller than the minimum distance between the holes, so even if we routed as in (b), extra δ of can be safely added between holes). Thus for any $i = 1 \dots h$, the points $\vec{p}_i = \vec{u} + \delta\vec{e}_i$ and $\vec{q}_i = \vec{u} - \delta\vec{e}_i$ are feasible, where $\vec{e}_i$ is the unit vector along the ith axis in the potentials space $\mathbb{R}^h$. Since the feasible set is convex, it contains the convex hull of the points $\{\vec{p}_i, \vec{q}_i\}_{i=1}^h$. The convex hull is bounded by 2^h $(h-1)$-simplicies; the volume between $\vec{u}$ and each simplex is $\frac{1}{h!}\det(\delta\vec{e}_1, \delta\vec{e}_2, \dots, \delta\vec{e}_h) = \delta^h/h!$. Thus the volume of the convex hull is $(2\delta)^h/h!$.

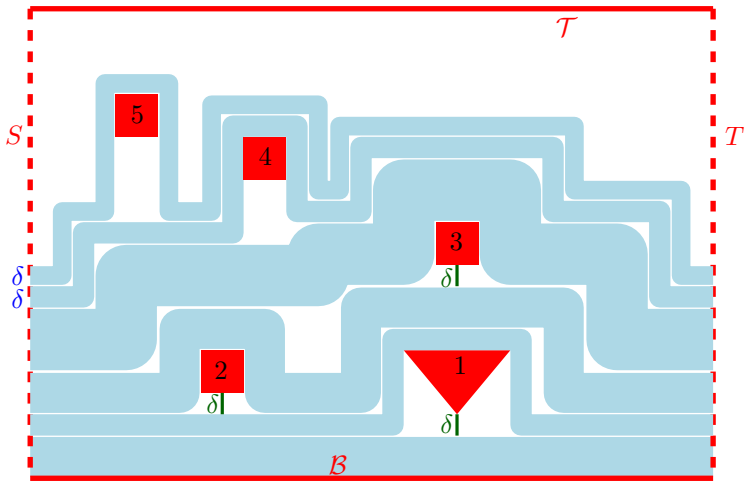


Figure 12: Free space of length δ (darkgreen) is left on each bar in the shortest-paths tree from $\mathcal{B}$ in $\vec{G}$ up to hole 3. Some amount flow is routed between 3 and 4, and then just enough flow remains to route δ uniformly between the remaining holes.

We now lift the assumption that all holes have distinct shortest-path labels in $\vec{G}$. Perform the same bottommost fill as above. When the fill reaches holes with the same label, arbitrarily order the holes and route $\delta' = \delta/h$ between every pair of consecutive holes in the order (Fig. 13); since $\delta' h$ is smaller than the smallest distance between holes, no bar gets saturated. Analogously to the above, in the obtained flow δ' units can be flipped from one side of any hole to the other, and hence the feasible set has volume at least

$$\frac{(2\delta')^h}{h!} = \frac{(2\delta)^h}{h^h h!} = \frac{(2\varepsilon)^h}{(nh^2L)^h h!}$$

□

Remark 7. We do not have to find the potential $\vec{u}$ algorithmically (even though we could); showing its existence suffices.

F Number of iterations

To bound the number of iterations we use Theorem 4 which we restate here for ease of reference:

Theorem 19. *If the feasible set $\mathcal{C}$ defines a full-dimensional polyhedron (i.e., $\mathrm{vol}(\mathcal{C}) > 0$) and if an ellipsoid E_0 containing $\mathcal{C}$ is known initially, then after $2N \ln \frac{\mathrm{vol}(E_0)}{\mathrm{vol}(\mathcal{C})\varepsilon^N}$ ellipsoid iterations a feasible solution $\vec{u} \in \mathcal{C}$ will be found such that*

$$\textsc{cost}(\vec{u}) \quad \leq \quad \min_{\vec{v}\in\mathcal{C}} \textsc{cost}(\vec{v}) \quad + \quad \varepsilon(\max_{\vec{v}\in\mathcal{C}} \textsc{cost}(\vec{v}) - \min_{\vec{v}\in\mathcal{C}} \textsc{cost}(\vec{v}))$$

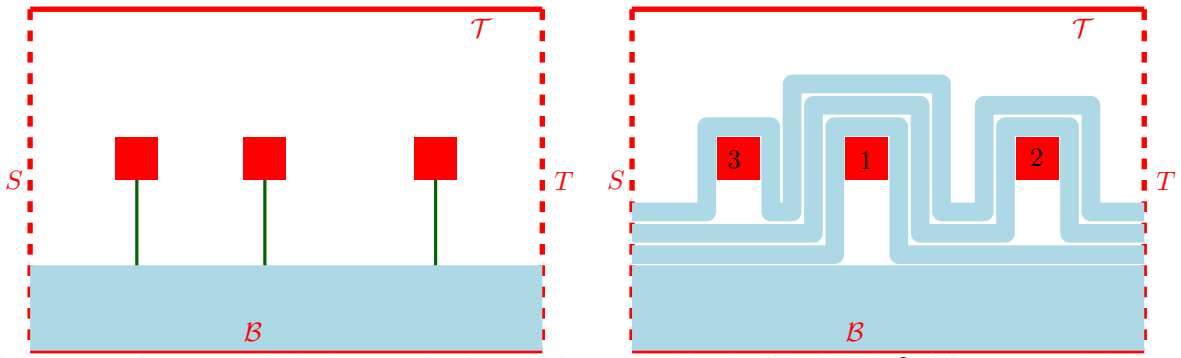


Figure 13: Left: Same-label holes are (simultaneously) reached by the fill – length δ of free space (darkgreen) remains on the bars leading to the holes. Right: the holes are ordered arbitrarily and δ' of flow is routed between the holes in the order.

Let $\mathsf{opt} = \min_{\vec{v}\in\mathcal{C}} \textsc{cost}(\vec{v})$ denote the cost of the optimal solution to our problem.

Corollary 20. *A flow with cost at most* $\mathsf{opt} + \varepsilon$ *will be found after* $O(h^2 \log \frac{nL}{\varepsilon})$ *ellipsoid iterations.*

Proof. Clearly, the cost of any flow in $\mathcal{D}$ cannot exceed L^2 (recall from Section 1.1 that L denotes the largest coordinate of $\mathcal{D}$ – the "size" of the domain), i.e., for us $\max_{\vec{v}\in\mathcal{C}} \textsc{cost}(\vec{v}) \le L^2$. Using Theorem 19 with $\frac{\varepsilon}{L^2}$ in place of ε, we obtain that $2N \ln \frac{\mathrm{vol}(E_0)\, L^{2N}}{\mathrm{vol}(\mathcal{C})\varepsilon^N}$ iterations are enough to get within ε of opt.

In our case the number of variables N is the number of holes whose potential is not set (after hole merging and mincut-potentials setting has been performed); this number is, of course, at most h. Since by Lemma 18, $\mathrm{vol}(\mathcal{C}) \ge \frac{(2\varepsilon)^h}{(nh^2L)^h h!}$, the number of iterations is at most $2h \ln \frac{\mathrm{vol}(E_0)(nh^2L^3)^h h!}{(2\varepsilon^2)^h}$.

Finally, since no potential can be larger than L, the optimal solution lies within radius-L ball centered at the origin in $\mathbb{R}^h$ – the ball is our initial ellipsoid E_0. Thus, the number of required iterations is at most

$$2h \ln \frac{\frac{\pi^{h/2}L^h}{\Gamma(h/2+1)}(nh^2L^3)^h h!}{(2\varepsilon^2)^h} = 2h \ln \frac{(\sqrt{\pi}nh^2L^4)^h h!}{(2\varepsilon^2)^h \Gamma(h/2+1)} = O\left(h^2 \log \frac{nL}{\varepsilon}\right)$$

□

G The minmax version

Minimizing the length of the longest streamline of the flow is NP-hard; this can be seen by adapting the proof of hardness of finding length-bounded disjoint paths in a planar graph (previously, similar adaptations were used to show hardness of the minmax version of thick non-crossing paths problems [6, 42]).

Holst and Pina [27] proved that it is NP-hard to find a given number of disjoint bounded-length paths in a planar graph. The proof is by reduction from 3SAT: Given a 3SAT instance with n variables and m clauses, lay out a $(2n+m)\times(2n)$ grid of rhombi (e.g., squares turned by 45^o); the grid, together with supersource and supersink vertices r, s forms a graph G (Fig. 14). The first 2 rows in the grid correspond to the variable u_1, rows 3 and 4 – to variable u_2, and so on. The columns 1 through n correspond to variables u_1 through u_n; symmetrically, the columns $2n+m$ down to $n+m+1$ correspond to u_1 through u_n. The columns $n+1$ through $n+m$ correspond to the clauses.

Most of the edges in G have weight 1. Weight 3 is assigned to the left edges of rhombus in ith column and row $2i-1$ for each $i = 1 \dots n$ (if one were to draw a line through these rhombi, the line would have slope 2); symmetrically, weight 3 is assigned to the right edges of rhombus in column $2n+m-i+1$ and row $2i-1$. Finally, in each clause column j (columns n through $n+m$) we have three weight-2 edges corresponding to the literals in the clause: if u_i is unnegated in clause j, then the right edge of rhombus in row $2i-1$ has weight 2; if clause j contains $\bar{u}_i$, then the left edge the rhombus has weight 2.

Hoslt and Pina argue that there exist $2n+m$ r-s disjoint paths with each path having length at most $2n+4$ iff the 3SAT instance is feasible: In any set of $2n+m$ disjoint r-s paths in G, let *i-path* be the path that goes through the ith vertex at the bottom of the graph, i.e., through the vertex $(2i-1, 0)$. The i-path must stay in the ith column (if the path "leaks" into a neighboring column, there is not enough space for the other paths); the only freedom the path has is how to "wiggle" through the column, i.e., whether to turn left or right when climbing from one row to the next. But even this freedom is not total: when going through rows $2i-1, 2i$ the path must use the left sides of one of the rhombi and the right sides of the other rhombus (otherwise, the path uses two edges of weight 3, which it cannot afford because then its length becomes $2n+6 > 2n+4$). It follows that in a single row, the paths in different columns must wiggle in the same way – ether all to the right, or all to the left (Fig. 15). Finally, in any clause column $j \in \{n+1, \dots, n+m\}$ the j-path can afford to go only through two weight-2 edges, or else its length becomes $2n+5 > 2n+4$; the omitted weight-2 edge corresponds to the literal satisfying clause j.

We turn G into a polygonal domain by assuming that each edge of G is a corridor of width 1 (to emulate length-2 and length-3 edges of G, the corresponding corridors wiggle a bit to gain the necessary length). To ensure that there exists a flow of value $2n+m$, we make a wide opening at r, followed by

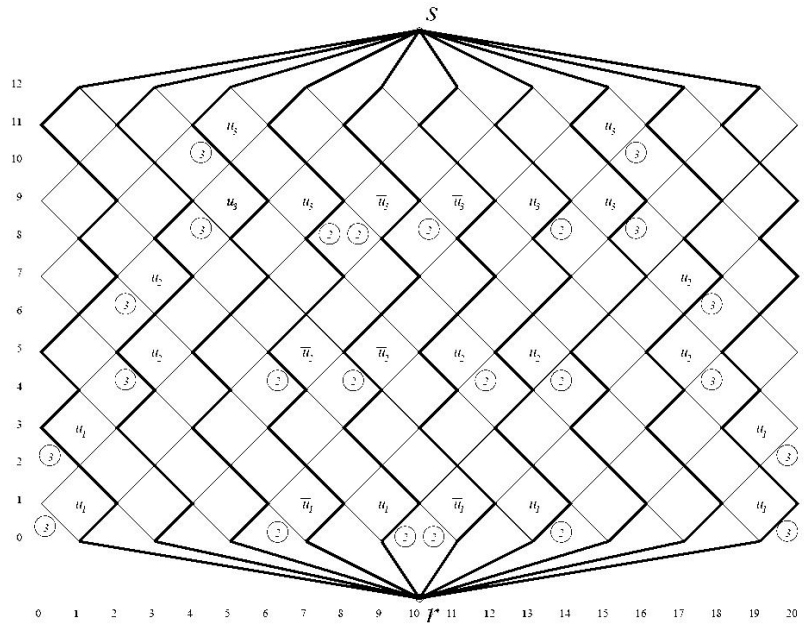


Figure 1: Instance of the maximum length-bounded disjoint paths problem associated with the clause $C = \{\{\bar{u}_1, \bar{u}_2, u_3\}, \{u_1, \bar{u}_2, \bar{u}_3\}, \{\bar{u}_1, u_2, \bar{u}_3\}, \{u_1, u_2, u_3\}\}$.

Figure 14: Fig. 1 from [27] – the graph G formed by a grid of squares, and a set of paths in the graph (bold).

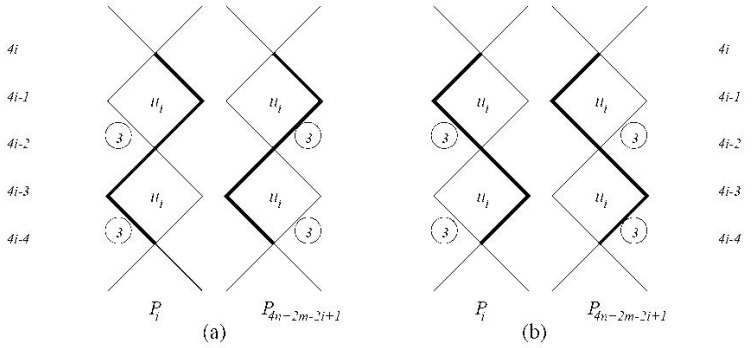


Figure 2: Figure (a) shows the subpaths corresponding to the assignment $t(u_i) :=$ true and Figure (b) shows the subpaths corresponding to the assignment $t(u_i) :=$ false.

Figure 15: Fig. 2 from [27]: on the left are subpaths that set the u_i true; on the right – false.

$2n + m$ corridors; we do similar transformation at s. We claim that there exists a flow of value $2n + m$ with each streamline having length at most $2n + 4$ iff the 3SAT instance is feasible.

The proof repeats the one of Holst and Pina almost verbatim: Let an *i-streamline* be a streamline that goes through the ith opening at the bottom of the domain (the total thickness of i-streamlines is 1). Any i-streamline must stay within the ith column; if an i-streamline leaks into a neighboring column, there is not enough space for the other streamlines (this is easy to see by looking at the cross-section of the domain by a horizontal line that cuts between neighboring rows of rhombi – the free space along the line is a set of $2n + m$ unit segments, so a whole unit of flow must pass through every such segment). Again, no streamline is allowed to pass through both weight-3 edges (i.e., length-3 corridors) in the ith column (and symmetrically, in the $2n + m - i + 1$st column); just as above this implies that in each row, i-streamlines are consistent – either going to the right or going to the left, setting the truth assignment for u_i. Also as above, in any clause column, no streamline is allowed to go through all three length-2 edges; the omitted edge corresponds to the satisfying literal.

By construction, all streamlines are monotone (in the figures above, reproduced from [27], they are y-monotone, but the construction can be rotated by 90 degrees).