ROUTING MULTI-CLASS TRAFFIC FLOWS IN THE PLANE[*]

Joondong Kim[†] Joseph S. B. Mitchell[‡] Valentin Polishchuk[§] Shang Yang[¶]
and Jingyu Zou[†]

August 28, 2011

Abstract

We study a class of multi-commodity flow problems in geometric domains: For a given planar domain P populated with obstacles (holes) of $K \geq 2$ *types*, compute a set of thick paths from a "source" edge of P to a "sink" edge of P for vehicles of K distinct *classes*. Each class k of vehicle has a given set, $\mathcal{O}_k$, of obstacles it must avoid and a certain width, w_k, of path it requires. The problem is to determine if it is possible to route N_k width-w_k paths for class k vehicles from source to sink, with each path avoiding the requisite set $\mathcal{O}_k$ of obstacles, and no two paths overlapping. This form of multi-commodity flow in two-dimensional domains arises in computing throughput capacity for multiple classes of aircraft in an airspace impacted by different types of constraints, such as those arising from weather hazards.

We give both algorithmic theory results and experimental results.

We show hardness of many versions of the problem by proving that two simple variants are NP-hard even in the case $K = 2$. If $w_1 = w_2 = 1$, then the problem is NP-hard even when $O_1 = \emptyset$. If $w_1 = 2, w_2 = 3$, then the problem is NP-hard even when $O_1 = O_2$. In contrast, the problem for a single width and a single type of obstacles is polynomially solvable.

We present approximation algorithms for the multi-criteria optimization problems that arise when trying to maximize the number of routable paths. We also give a polynomial-time algorithm for the case in which the number of holes in the input domain is bounded.

Finally, we give experimental results based on an implementation of our methods and experiment with enhanced heuristics for efficient solutions in practice. Our algorithms are being utilized in simulations with NASA's Future Air traffic management Concepts Evaluation Tool (FACET). We report on experimental results based on applying our algorithms to weather-impacted airspaces, comparing heuristic strategies for searching for feasible path orderings and for computing short multi-class routes. Our results show that multi-class routes can feasibly be computed on real weather data instances on the scale required in air traffic management applications.

[*]This work grew out of discussions during our participation in the *ATM-Wx Impact Modeling Workshop 2: ATM Weather Constraint Modeling*, at the National Center for Atmospheric Research, Boulder, CO, March 3-4, 2008. We thank William Chan (NASA Ames), Nathan Downs (Metron Aviation), Jimmy Krozel (Metron Aviation), Tenny Lindholm (STAR), Joseph Prete (Metron Aviation), and Bob Sharman (STAR) for their input in formulating the algorithmic model and providing aviation expertise and weather data for our experiments. We thank Nathan Downs (Metron Aviation) for assistance in FACET simulations. FACET is provided by NASA Ames. We thank Esther Arkin, Petteri Kaski, Jukka Suomela, Girish Sabhnani and Charles Ward for helpful discussions. We thank the anonymous reviewers for their suggestions that improved the presentation of the results. This work was partially funded by NASA Ames, Metron Aviation, the National Science Foundation (CCF-0431030, CCF-0528209, CCF-0729019), and Academy of Finland (grant 138520). Preliminary version appeared in the abstracts of the 18th Annual Fall Workshop on Computational Geometry, 2008.

[†]Google Inc., `jdkim,jasonzou@google.com`

[‡]Department of Applied Mathematics and Statistics, Stony Brook University, `jsbm@ams.sunysb.edu`

[§]Helsinki Institute for Information Technology, Department of Computer Science, University of Helsinki, `polishch@hiit.fi`

[¶]Department of Computer Science, Stony Brook University, `syang@cs.sunysb.edu`

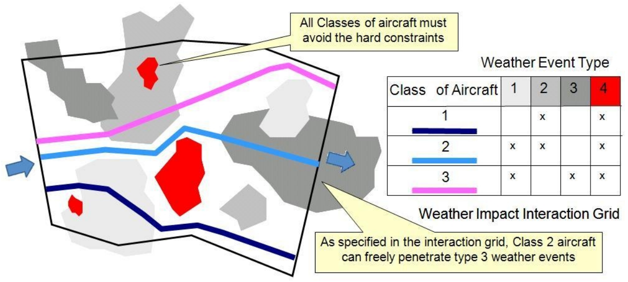
1 Introduction

Many applications of path planning in polygonal domains—VLSI routing, robotics, air traffic management (ATM), sensor networks—call for finding multiple disjoint thick paths serving as "lanes" along which non-point objects may move without conflicting with each other. Studying disjoint thick paths is also of theoretical interest as it leads to developing geometric counterparts of classical network flow results: the Max-flow Min-cut, the Flow Decomposition, and Menger's theorems. In this sense, the present paper addresses the geometric version of the *multi-commodity* network flow problem.

Motivation. Our problem statement is natural in any multiple path routing setting involving separation standards and different constraints on paths. We suspect it has several possible applications; however, our specific motivation comes from the domain of ATM. The aircraft differ in their capabilities, which impacts which regions of airspace they can traverse (e.g., due to hazardous weather conditions) and how much separation is needed between parallel "flows" of aircraft. In particular, one weather system can serve as an obstacle for one class of aircraft while being safely passable by another class of better equipped (or larger) aircraft. A good route planner must take this into account by possibly permitting "stronger" aircraft to fly through certain weather conditions, which serve as obstacles to "weaker" aircraft. In general, each class of aircraft is restricted to avoid certain types of airspace. See Fig. 1. (Note: our figures are best viewed in color.)

Our goal is to provide algorithms for decision support tools for the Next Generation Air Transportation System [19], specifically, to perform capacity estimation to determine how constraints, such as weather events, impact throughput capacity of airspace. The problem studied here is that of determining the maximum number of air lanes of multi-class aircraft that can permeate an airspace, given the constraints implied by weather forecast data. It is not expected that the routes computed with our algorithms will be the actual routes flown; many other complex issues affect the exact routes (jetways, winds, controller workload, etc). Rather, our goal is to be able to compute the maximum theoretical throughput possible across, e.g., a "flow-constrained area" (FCA), given a mixture of classes of aircraft and types of constraints, so that this information can be used as part of a decision support tool for traffic flow management.

Related Work. While we know of no prior results on the multi-class geometric flow routing problem studied here, multicommodity flows in discrete networks have been a subject of extensive research [3]. There is also an abundance of related work on computing multiple (single-class) paths and flows in geometric domains. A classification of existing approaches to multiple paths planning is given by van den Berg and Overmars [20]. In *prioritized* planning the paths are found one-by-one; all routed paths are declared as obstacles for a new path. The algorithms that do not use prioritized planning range from centralized over roadmap-based to decoupled (see [20] for details). A polynomial-time algorithm for finding a maximum number of thick paths in a polygonal domain is presented in [2]. The geometric versions of the Max-flow Min-cut, the Flow Decomposition, and Menger's theorems were established in [2, 14, 15, 18].


All Classes of aircraft must avoid the hard constraints
Weather Event Type
Weather Impact Interaction Grid
As specified in the interaction grid, Class 2 aircraft can freely penetrate type 3 weather events

Class of Aircraft	1	2	3	4
1		x		x
2	x	x		x
3	x		x	x

Figure 1: Example of the multi-class routing problem from ATM: A flow-constrained area having four different types of weather constraints impacting three different classes of aircraft.

In the ATM literature, there has been considerable interest in capacity estimation using weather forecast data. Our multi-class flow problem arises when weather forecast data is translated to impact on airspace by "causality analysis" [9], which considers how different classes of aircraft respond to various types of weather hazards, such as convection, in-flight icing, turbulence, and visibility. The use of geometric flow theory for capacity estimation of a sector or an FCA in ATM is discussed in [8, 11–13, 16].

Summary of Contributions

1. We introduce the multi-class routing problem and prove hardness of its most basic versions: Routing two classes of vehicles among two types of obstacles, and routing paths of two distinct widths among one type of obstacle. We also prove some hardness of approximation results and the (likely) nonexistence of a fixed-parameter tractable algorithm.

2. We give approximation algorithms for multi-class routing and a polynomial-time algorithm for the case in which the number of holes in the input domain is bounded.

3. We give experimental results based on an implementation that is currently being used in simulations with NASA's Future Air traffic management Concepts Evaluation Tool (FACET). We devise, implement, and compare efficient heuristics for exact solutions in practical settings.

2 Problem Formulation and Overview of the Results

The input to our problem is a polygonal domain P, consisting of an outer polygon and polygonal obstacles (holes). Let n denote the (total) number of vertices of P, and h denote the number of holes. Two edges of the outer polygon are designated as the *source* and the *sink*.

A *w-thick path* is the Minkowski sum of a usual (thin) source-sink path and disk of radius $w/2$ centered at the origin. As is common in the thick-paths literature [2, 14, 15], we assume that the polygonal domain is augmented by attaching Riemann flaps to the source and the sink so that the endpoints of the path belong to the source and the sink.[1]

The holes in the domain, as well as the paths sought are of one of K *types*. The width of a type k path is w_k, $k = 1 \dots K$. A path of type k must avoid the holes, $\mathcal{O}_k$, of type k, but may pass freely through the holes of the other types. The decision version of our problem is: Given a set of numbers $N_1, \dots, N_K$, determine if it is possible to route N_k width-w_k paths of type k from source to sink, so that no two paths overlap. This is the *Multi-Class paths problem*.

Note that the number of paths that exist in a domain may be exponential in the input size; e.g., there may exist $\Omega(M)$ width-1 paths in a $2 \times M$ rectangle, specified with $O(\log M)$ bits. By the Continuous Flow Decomposition Theorem [15], thick paths can be encoded succinctly by representing a "bundle" of paths of total thickness W by one W-thick path. Our positive results should be understood in the sense that the paths can be found in pseudopolynomial time, or that the *representations* of the paths can be found in strongly polynomial time.

Type Sequence and Uppermost Paths. The source and sink edges split the boundary of the outer polygon of the domain into two parts — the *top* T and *bottom* B (Fig. 2). We will assume that in any collection of paths, the paths are numbered in the order as they are encountered when going along the source or sink from T to B. Let $\tau = (t_1, \dots, t_M)$, $t_m \in \{1 \dots K\}$ be a sequence of path types. We say that τ is the *type sequence* of a collection of M paths if the mth path in the collection is of type t_m.

If the type sequence of the paths sought is specified, the Multi-Class paths problem can be solved in polynomial time by routing *uppermost paths*, as we now explain. Paths $(\Pi_1, \dots, \Pi_M)$ are called *uppermost* [2, 14] if Π_1 runs "as close as possible" to T, and Π_m runs "as close as possible" to Π_{m-1}, for $m = 2 \dots M$.

[1]Alternatively, we may consider only the *canonical parts* of the thick paths [15], which are the "rectilinear strips" of width w with opposite sides of length w residing on the source and the sink.

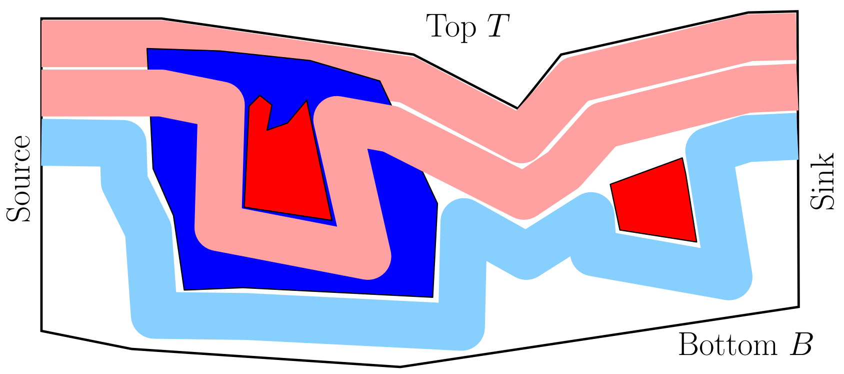


Figure 2: Uppermost paths for the type sequence (red, red, blue).

Specifically, imagine that the domain is grass, over which fire travels at speed 1; imagine also that, at the mth stage of our algorithm (when routing Π_m) the type-t_m holes, $\mathcal{O}_{t_m}$, are saturated with a highly flammable material so that whenever fire touches such a hole, the entire hole is ignited instantaneously. Ignite T at time 0, and let P' be the part of the domain that has burned by time w_1 (see [2, Theorem 2.2] for details of simulating the fire – handling events, attaching Riemann flaps, etc.). The (first) uppermost path is a w_1-thick path routed within P'. The second uppermost path is the thick path of type t_2 routed by iteratively treating the lower boundary of the first uppermost path as T and using obstacle set $\mathcal{O}_{t_2}$; the other uppermost paths are defined recursively in a similar manner.

We obtain the following straightforward extension of Theorem 2.2 in [2]:

Theorem 1. *If there exist M paths with type sequence τ, then there exist uppermost paths with type sequence τ, and a representation of the paths can be found in $O(nh + n \log n + M)$ time.*

For the case in which the holes are all of the same type, and paths only differ by their widths, we also have a "query" version of Theorem 1 (the benefit of the query version is the running time). In contrast with Theorem 1, the algorithm for Theorem 2 does not produce the paths; it just tests whether the routing is possible. (We have been unable to obtain a result similar to Theorem 2 in the case that holes are of two or more different types; this remains open.)

Theorem 2. *If all holes are of the same type, a graph can be built in time $O(nh)$ such that given a type sequence τ, it can be tested in $O(M + h^2 \log M)$ time whether it is possible to route the M paths with type sequence τ, by solving a variation of the shortest path problem in the graph.*

See Section 3 for the proof of the theorem.

For most of the paper we speak about two types of paths ($K = 2$), and concentrate on two cases:

Red/Blue paths problem: The paths have the same width ($w_1 = w_2 = 1$), but the holes are of two types, *Red* and *Blue*.

Two Widths paths problem: The holes are of one type, but the paths have different widths, $w_1 = 2, w_2 = 3$.

We focus on these special cases only for ease of presentation, and actually lose no generality by considering them: we give hardness proofs for these restricted cases, and our algorithms extend to the case of an arbitrary number of types of paths of many different widths.

Our main hardness result, proved in Section 4, is as follows.

Theorem 3. *(i) Given two integers, r and b, and a polygonal domain with red and blue holes, it is NP-hard to decide if it is possible to route r red and b blue pairwise-disjoint width-1 paths. (ii) Given two integers, N_1 and N_2, and a polygonal domain with (monochromatic) holes, it is NP-hard to decide whether it is possible to route N_1 width-2 and N_2 width-3 pairwise-disjoint paths.*

In fact, our hardness proof for (i) holds even when only blue obstacles are used in the reduction. Thus, Theorem 3(i) can be strengthened to show that an even more special case is hard:

Corollary 4. *Given two integers, r and b, and a polygonal domain with* only blue *holes, it is NP-hard to decide if it is possible to route r red and b blue pairwise-disjoint width-1 paths.*

Since finding an exact solution to our bicriteria optimization problem is NP-hard, we turn the attention to approximation algorithms. Suppose that there exist r red and b blue paths in P. There are (at least) three approaches to approximation of the problem:

(i) approximate the maximum number of routable blue paths while making sure that r red paths are routed;

(ii) approximate the numbers of paths of both colors, i.e., give an algorithm to route αr red and βb blue paths, for some $\alpha, \beta \in (0, 1)$; and,

(iii) approximate the maximum number of routable blue paths while making sure the total number of routed paths stays equal to $r + b$.

We leave approach (i) as an open problem. In Theorem 7 we show that (ii) is possible for essentially any α, β with $\alpha + \beta = 1$. The reduction employed in the proof of Theorem 3 shows that (iii) is NP-hard:

Corollary 5. *Let P a polygonal domain with n vertices in which there exist r red and b blue paths. Let $\gamma = \Omega(n^{1/6-\varepsilon})$ be a number, for some $\varepsilon > 0$. Unless P=NP, one cannot find in polynomial time a set of $r - \gamma b$ red and γb blue paths in the domain.*

In the Red/Blue (resp., Two Widths) paths problem, a type sequence is just a sequence of R's and B's (resp., 2's and 3's). In certain cases, one can go through all possible type sequences in polynomial time. For example, suppose that in the Red/Blue paths problem, the number of color changes in the type sequence is bounded by a number L. Then the number of different type sequences, for a total of M red and blue paths, is $O(M^{O(L)})$ – polynomial for a constant L. For each of the sequences, we can use Theorem 1 to route the red/blue paths or to conclude that it is not possible. Define a *switch* to be a change from R to B, or from B to R (resp., from 2 to 3, or from 3 to 2) in the type sequence for the Red/Blue (resp., Two Widths) paths problem. Then, we have

Lemma 6. *If the number of switches is at most L, both the Red/Blue and the Two Widths paths problems can be solved in $O(\mathrm{poly}(n)\, M^{O(L)})$ time.*

One may hope to have a fixed-parameter tractable, $O(f(L)\ \mathrm{poly}(n, M))$-time algorithm (where f could be exponential). However, in our reduction from INDEPENDENT SET to the Red/Blue paths problem (proof of Theorem 3), the existence of an independent set of size L in the graph implies a solution to the Red/Blue paths problem with at most $L + 1$ switches. Thus, an $O(f(L)\ \mathrm{poly}(n, M))$ algorithm for the Red/Blue paths problem would imply an $O(f'(L)\ \mathrm{poly}(n))$ algorithm for INDEPENDENT SET in an n-vertex graph, which is unlikely to exist due to $W[1]$-completeness of the problem [5]. Thus, our problem is $W[1]$-hard with respect to the number of switches.

On the positive side, we show that *any* sequence of paths can be approximated by a sequence with a small number of switches. Choosing the *best* type sequence out of all sequences with few switches, and appealing to Lemma 6, we obtain an approximation algorithm for the Red/Blue paths problem. Specifically, in Section 5 we prove

Theorem 7. *Let r and b, $r \leq b$, be two integers such that there exist r red and b blue thick paths in P. Let $L \leq r + b$ and $\ell \leq L$ be two arbitrary integers such that r/L and b/L are integers. One can find $\frac{\ell}{L} r$ red and $\frac{L-\ell}{L} b$ blue paths in $O(\mathrm{poly}(n, r^L))$ time.*

An analogous statement holds for the Two Widths paths problem.

As another application of Lemma 6, we give a polynomial-time algorithm for the case when h, the number of holes in the domain, is small. In this case, the number of relevant "threadings" of the paths is small, and within a subsequence of paths of common threading, it is enough to have at most one switch in the type sequence. We prove in Section 6:

Theorem 8. *If the number of holes in the domain is bounded ($h = O(1)$), both the Red/Blue and the Two Widths paths problems can be solved in polynomial time.*

In the remainder of the paper we give proofs of the above theorems. In Section 7, we discuss an implementation and experimental results.

3 Testing type sequence feasibility in the monochromatic case

In this section the holes and the paths are all of the same color; the paths differ only in width (i.e., a type sequence is a sequence of numbers – the paths widths). We build a data structure to answer efficiently the following queries: "Given a type sequence τ, is it possible to route a set of paths with type sequence τ?":

Theorem 2 (stated again). *If all holes are of the same type, a graph can be built in time $O(nh)$ such that given a type sequence τ, it can be tested in $O(M + h^2 \log M)$ time whether it is possible to route the M paths with type sequence τ, by solving a variation of the shortest path problem in the graph.*

Note that here we do not find the paths themselves; we only test whether the routing is possible in principle.

Proof. We build the graph on the holes of the domain. Specifically, the *critical graph* of the domain [2, 7, 14] has a vertex for each hole, for T, and for B; the length of an edge between two vertices is equal to the Euclidean distance between the corresponding holes (Fig. 3). The graph can be built in $O(nh)$ time by using the linear time algorithm in [1] to compute the Euclidean distance between every pair of holes. The length of a shortest T-B path in the graph is equal to the value of the maximum flow [14]. In the *thresholded* version of the graph the length of each edge is thresholded down to the nearest integer; the length of a shortest T-B path in the graph is equal to the maximum number of width-1 paths that can be routed though P. See [2, 7, 14] for details.

We use a Dijkstra-like algorithm to find a shortest T-B path in the critical graph, conforming to the sequence τ. We label the vertices of the graph by positions in τ. The *permanent* label $\ell(v)$ assigned to a vertex v after the algorithm completes, is the largest index such that the paths $1, \ldots, \ell(v)$ can be routed between T and the hole corresponding to v.

We start with assigning permanent label $\ell(T) = 0$ and *temporary* labels $\ell(\cdot) = \infty$ for the other vertices (just like in Dijkstra's algorithm, the temporary label of each vertex is an upper bound on its permanent label). Next, we propagate the labels, Dijkstra-style from the vertex v with the smallest label. For each edge (v, u) out of v, we see how far along τ it is possible to go by that edge, and update the label of u:

$$\ell(u) \quad \leftarrow \quad \min \left\{ \quad \ell(u) \quad , \quad \arg\max_{m} \sum_{i=\ell(v)+1}^{m} w_{t_i} \leq d(u, v) \quad \right\}$$

where $d(u, v)$ is the length of the edge (u, v) in the critical graph, and w_{t_i} is the width of the path of type t_i. The propagation along an edge takes $O(\log M)$ time (after storing an array of τ's partial sums).

Just like in Dijkstra's algorithm, the induction on the label shows that the final label of B is the length of the longest subsequence of τ (starting from t_1) that can be routed through the domain. □

4 Hardness results

In this section we show that even very simple versions of our geometric multicommodity flow problem are NP-hard. For instance, an important special case of the Red/Blue paths problem is when the holes are *nested*,

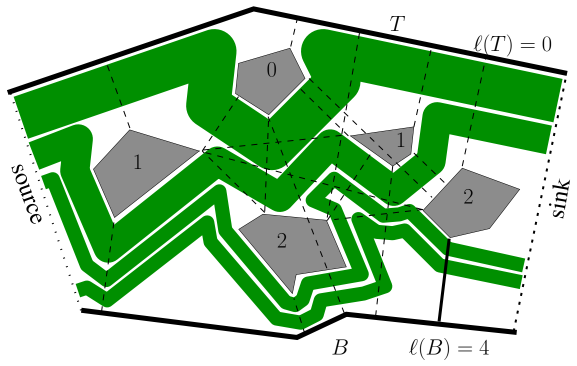


Figure 3: The edges of the critical graph are dashed. The numbers are labels of holes, corresponding to a sequence of widths $\tau = (3, 2, 1, 1)$. For another sequence, say, $\tau' = (3, 2, 4)$, the label of B would have been 2 since there would not be room for the third path.

i.e., when the set of the red holes is a subset of blue, modeling the situation when the capabilities of one class of vehicle is a subset of the capabilities of the other class. (Then, the red paths must avoid only the red obstacles, while the blue paths must avoid *both* red and blue.) We now show that even this special case of the Red/Blue paths problem is NP-hard; in Section 4.1 we show the hardness of the Two Widths paths problem.

We reduce from INDEPENDENT SET [6]. Let G be a graph with n vertices and m edges. In the independent set problem, the question is: Given an integer k, do there exist k vertices of G no two of which are connected by an edge? (In this section n denotes the number of vertices in G, and k denotes the size of the independent set.) We construct from G an instance of the Red/Blue paths problem as follows.

For each vertex of G we create a *vertex gadget* (Fig. 4(a)). We create a column of n aligned vertex gadgets, one on top of another, forming the *vertex part* of the construction (Fig. 4(b)).

For each edge of G we create an *edge gadget*. To build the gadget, we first create an $8n$-by-$8n$ square with top and bottom sides being red segments; we put n equally spaced blue segments of height 4 along the right side of the square (Fig. 4(c)). If edge e is incident to a vertex i, we add length 1 to the top and the bottom of the ith obstacle in the edge gadget corresponding to e (thus, there are exactly two stretched obstacles in each edge gadget). Finally, the top and the bottom boundary of each edge gadget are shifted by 1 up and down (Fig. 4(d)).

To finish the construction, we put the vertex part and the m edge gadgets side by side from left to right; we align the obstacles in the edge gadgets with the rightmost obstacles in the vertex part. We then insert a vertex part between consecutive edge gadgets; this way, the paths are always aligned in the same way before entering any edge gadget (Fig. 5). Overall, our construction has, from left to right: vertex part – edge 1 gadget – vertex part – edge 2 gadget – vertex part – $\cdots$ – vertex part – edge m gadget. Since the paths are non-crossing, the ordering of the paths from top to bottom is the same in every gadget.

We now prove that there exists an independent set of size k in G if and only if $8(n-k)$ red and $4k$ blue paths can be routed all the way from the left of the construction to the right.

First, suppose there is an independent set of size k in G. Route four blue paths through each vertex gadget that corresponds to a vertex in the independent set; route eight red paths through the other gadgets. We claim that the paths may pass through all edge gadgets. Indeed, consider any edge gadget. If the gadget did not have obstacles with additional length, the paths could go through the gadget in the same way they came out of the vertex gadgets. Adding height 1 to the top and bottom of an obstacle in the gadget may have a pair of blue paths shifted up and down, causing also shifting of the other paths. But since the blue paths go through vertex gadgets that collectively correspond to an independent set, there is at most one pair of shifted blue paths within one edge gadget. Thus, the shifted paths will fit into the extra space at the top and the bottom. This proves that if there is an independent set of size k in G, there exist $8(n-k)$ red and $4k$ blue paths in our instance of the Red/Blue paths problem.

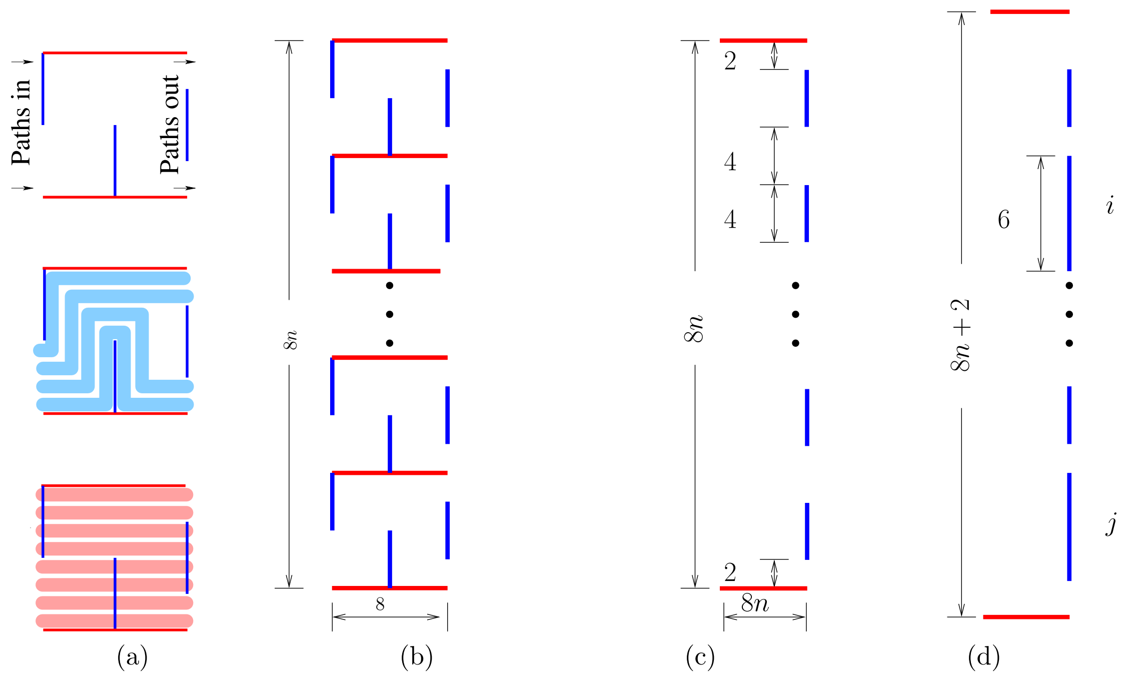


Figure 4: (a) Top: the vertex gadget is an 8-by-8 square with top and bottom sides being red segments; there are three blue obstacles inside the gadget, each is a vertical segment of height 4. (a) Middle and bottom: if the paths going through the gadget are of one color, they are either (at most) four blue paths or (at most) eight red paths. (b): the vertex part — n stacked vertex gadgets. (c): each edge gadget is built from an $8n$-by-$8n$ square with n blue obstacles, each being a height-4 blue segment. (d): in the gadget for an edge (i, j), we stretch ith and jth obstacles by extending them upwards and downwards by length 1; also, the top and the bottom sides of each edge gadget are shifted by 1 up and down respectively.

On the other hand, suppose that there exist $8(n-k)$ red and $4k$ blue paths. We first show that no vertex gadget has both red and blue paths passing through it. Let B_0 (resp., B_1, B_2, B_3, B_4) be the set of gadgets through which 0 (resp., 1, 2, 3, 4) blue paths pass. As can be seen by inspection (Fig. 6), the number of red paths going through a gadget in B_0 (resp., B_1, B_2, B_3, B_4) is at most 8 (resp., 4, 2, 1, 0). Thus, the total number of red paths is at most

$$8|B_0| + 4|B_1| + 2|B_2| + |B_3| = 8n - 4|B_1| - 6|B_2| - 7|B_3| - 8|B_4| = 8n - 8k - 2|B_1| - 2|B_2| - |B_3|,$$

where the first equality uses $|B_0|+|B_1|+|B_2|+|B_3|+|B_4| = n$, and the second uses $|B_1|+2|B_2|+3|B_3|+4|B_4| = 4k$. Since we assumed that there are exactly $8n - 8k$ red paths, we have:

Lemma 9. *$|B_1| = |B_2| = |B_3| = 0$.*

By Lemma 9, there are k vertex gadgets filled with blue paths, and $n - k$ gadgets filled with red. Suppose that two gadgets, corresponding to endpoints of an edge e, are both filled with blue paths. Then the $8(n-k)$ red and $4k$ blue paths will not fit through the edge gadget corresponding to e, since the topmost and the bottommost paths in the gadget will have to shift up and down by 2, and there is no space for it. Thus, the $4k$ blue paths go through vertex gadgets corresponding to an independent set of size k in G.

This proves

Theorem 4(i) (stated again). *Given two integers, r and b, and a polygonal domain with red and blue holes, it is NP-hard to decide if it is possible to route r red and b blue pairwise-disjoint width-1 paths.*

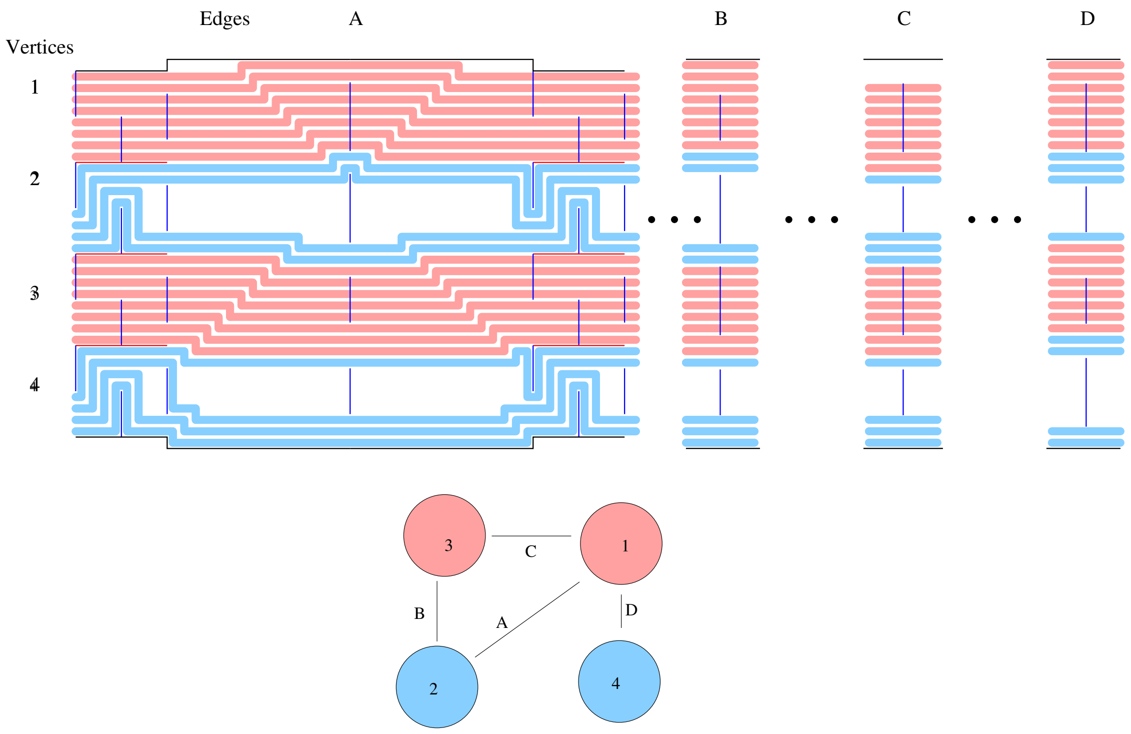


Figure 5: The vertex parts and the edge gadgets are put one after another. The edge gadgets are not to scale. Stretching the obstacles shifts the paths by 1 up and down; the shifted paths fit fine into the gadgets because the top and the bottom of the gadgets were shifted by 1 too. The outer polygon of the domain is shown black. This example shows the construction for the graph at the bottom.

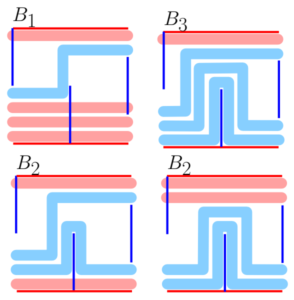


Figure 6: The maximum number of red paths going through a gadget with 1, 2, and 3 blue paths.

Only blue obstacles In the above construction we can replace all red obstacles with blue obstacles. Indeed, the only place where the red obstacles appear is the vertex gadget. Lemma 9 remains true after the replacement, and so does the claim that independent sets of size k in G are in one-to-one correspondence with $8(n-k)$ red and $4k$ blue paths. Thus, the Red/Blue paths problem is hard even when restricted to instances with only blue obstacles:

Corollary 4 (stated again). *Given two integers, r and b, and a polygonal domain with* only blue *holes, it is NP-hard to decide if it is possible to route r red and b blue pairwise-disjoint width-1 paths.*

Hardness of approximation Assume there exist r and b red and blue paths in the problem instance constructed from a graph G with N vertices and M edges. Let now $n = O(NM) = O(N^3)$ be the complexity of the domain constructed from G. We know from the reduction that there exists an independent set of size $b/4$ in G. If we could find $r - \gamma b$ red and γb blue paths, for some γ, we could find an independent set of size $\gamma b/4$. This is not possible in polynomial time (unless P=NP) for $\gamma = \Omega(N^{1/2-\varepsilon})$, where $\varepsilon > 0$ is an arbitrary positive number [4]. This means that one cannot approximate, to within any factor $\gamma = \Omega(n^{1/6-\varepsilon})$, the maximum number of routable blue paths while keeping the total number of routed paths equal to $r + b$:

Corollary 5 (stated again). *Let P a polygonal domain with n vertices in which there exist r red and b blue paths. Let $\gamma = \Omega(n^{1/6-\varepsilon})$ be a number, for some $\varepsilon > 0$. Unless P=NP, one cannot find in polynomial time a set of $r - \gamma b$ red and γb blue paths in the domain.*

4.1 Hardness of the Two Widths paths problem

The reduction showing NP-hardness of the Two Widths paths problem is very similar to the reduction used above for the Red/Blue paths problem. The vertex gadget is a 6-by-6 square with top and bottom sides being obstacles. If the paths going through the gadget are of the same thickness, they are either (at most) three width-2 path or (at most) two width-3 paths. As in the proof of hardness of the Red/Blue paths problem, n vertex gadgets are stacked one on top of another forming the *vertex part* (Fig. 7).

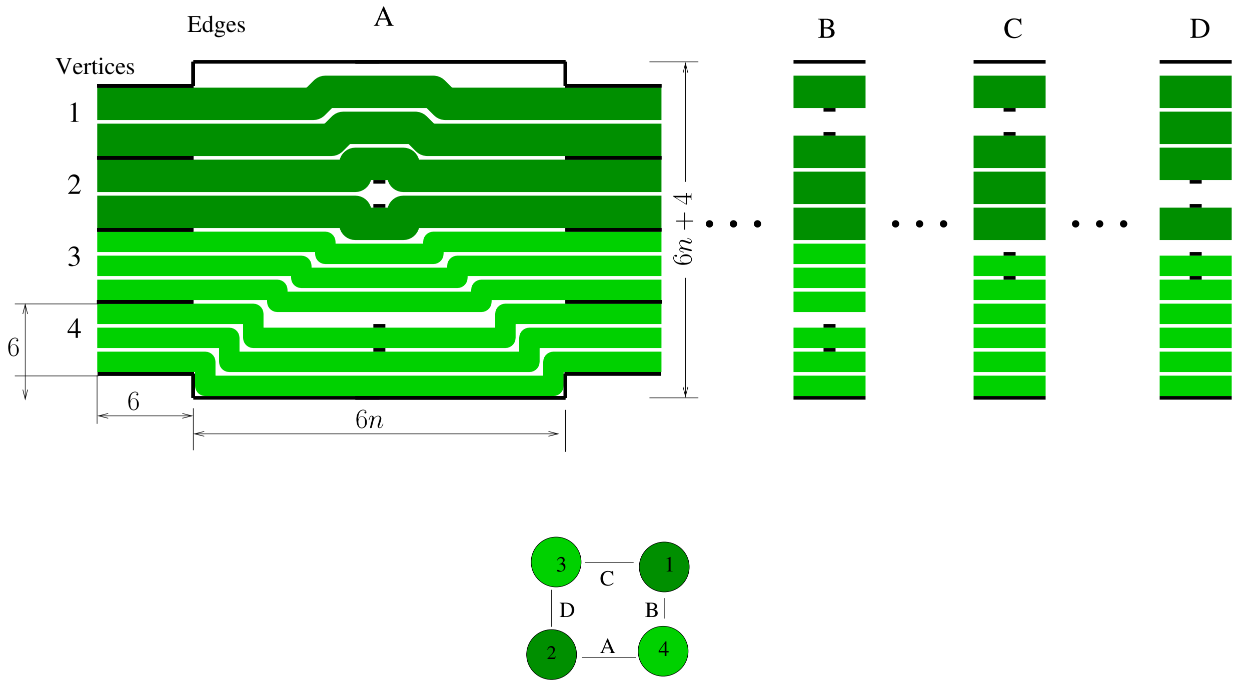


Figure 7: Construction for the proof of Theorem 3(ii). The edge gadgets are not to scale. This example shows the construction for the graph at the bottom.

Each edge gadget is a $6n$-by-$(6n + 4)$ rectangle with top and bottom sides being obstacles. In the gadget for an edge (i, j), we put a pair of point obstacles at distance 2 apart, aligned with vertex gadgets i and j.

The rest of the construction is identical to the one for the Red/Blue paths problem. The edge gadgets are put next to the vertex part, one-by-one from left to right. We also insert a vertex part between consecutive edge gadgets to ensure the paths are aligned in the same way before entering any edge gadget. There is an independent set of size k in G if and only if we can route $3(n - k)$ width-2 and $2k$ width-3 paths; the wider paths must go through vertex gadgets, corresponding to an independent set.

Theorem 3(ii) (stated again). *Given two integers, N_1 and N_2, and a polygonal domain with (monochromatic) holes, it is NP-hard to decide whether it is possible to route N_1 width-2 and N_2 width-3 pairwise-disjoint paths.*

5 Approximation

We now turn to positive results. In this section we show how to approximate any type sequence by a sequence with few switches. Choosing the best sequence with the few switches allows one to approximate simultaneously the number of paths of each color in polynomial time:

Theorem 7 (stated again). *Let r and b, $r \leq b$, be two integers such that there exist r red and b blue thick paths in P. Let $L \leq r + b$ and $\ell \leq L$ be two arbitrary integers such that r/L and b/L are integers. One can find $\frac{\ell}{L} r$ red and $\frac{L-\ell}{L} b$ blue paths in $O(\text{poly}(n, r^L))$ time.*

Proof. Consider the collection of r red and b blue paths. Consider also $L + 1$ (thin) source-sink paths $\pi_0^*, \pi_1^*, \ldots, \pi_L^*$ (where $\pi_0^* = T$, $\pi_L^* = B$) that split the domain so that there is exactly $(r + b)/L$ thick paths (both red and blue) between π_{i-1}^* and π_i^* for each $i = 1 \ldots L$ (Fig. 8). Call the part of the domain between π_{i-1}^* and π_i^* the ith *slot*. Let $\mathcal{R}^* = (r_1^*, \ldots, r_L^*)$, $\mathcal{B}^* = (b_1^*, \ldots, b_L^*)$, be the sequences of the numbers of red and blue paths in the slots; $r_1^* + \cdots + r_L^* = r$, $b_1^* + \cdots + b_L^* = b$, $r_1^* + b_1^* = \cdots = r_L^* + b_L^* = (r + b)/L$.

We go through all $O(r^L)$ possible representations of r as a sum of L integers each less than or equal to $(r + b)/L$. The integers specify one possibility for how many of the r paths reside in each of the L slots. For each representation $\mathcal{R} = (r_1, \ldots, r_L)$, we run the algorithm RedBluePaths that routes successively either r_i red uppermost paths or $\frac{r+b}{L} - r_i$ blue uppermost paths; the choice of the color is determined by whether r_i is one of the ℓ largest numbers in $\mathcal{R}$:

```
Algorithm RedBluePaths($\mathcal{R}$)
Input. $r, b \in \mathbb{N}$; domain in which there exist $r$ red and $b$ blue paths; integers $L, \ell$;
sequence of integers $\mathcal{R} = (r_1, \ldots, r_L)$, such that $r_1 + \cdots + r_L = r$.
Output. A collection of $\ell r/L$ red and $(L - \ell)b/L$ blue paths (if one exists).

1  $\max_\ell(\mathcal{R}) \leftarrow$ indices of $\ell$ largest integers in $\mathcal{R}$
2  for $i = 1$ to $L$
3  if $i \in \max_\ell(\mathcal{R})$
4  route $r_i$ uppermost red paths
5      else
6  $b_i \leftarrow \frac{r+b}{L} - r_i$
7  route $b_i$ uppermost blue paths
8  endif
9  endfor
```

Now, one of the representations examined will be $\mathcal{R}^*$. Let Π_i^* be the set of uppermost paths routed by RedBluePaths($\mathcal{R}^*$) in the ith **for** loop; Π_i^* is a collection of either r_i^* red *or* b_i^* blue uppermost paths, depending on whether $i \in \max_\ell(\mathcal{R})$ or not. Since the first slot contains r_1^* red *and* b_1^* blue paths, the uppermost paths Π_1^* (which are *either* r_1^* reds *or* b_1^* blues) will stay within the slot:

Fact 1. *The lower boundary of* Π_1^* *is* above π_1^*.

The second slot contains r_2^* red *and* b_2^* blue paths. Again, r_2^* red *or* b_2^* blue *uppermost* paths, routed within the slot (i.e., treating π_1^* as the top of the domain), will stay within the slot. RedBluePaths($\mathcal{R}^*$) actually routes Π_2^* treating the lower boundary of Π_1^* as the top. Thus, by Fact 1, Π_2^* will not cross π_2^*. By induction, we obtain that RedBluePaths($\mathcal{R}^*$) will successfully route the paths without crossing $\pi_L^* = B$.

The number of red paths routed by RedBluePaths($\mathcal{R}^*$) is

$$\sum_{i \in \max_\ell(\mathcal{R})} r_i^* \quad \geq \quad \ell r/L \quad .$$

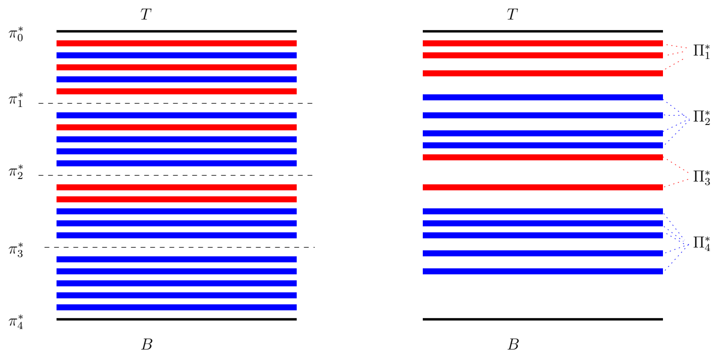


Figure 8: Left: In ith slot there exist r_i^* red *and* b_i^* blue paths; $r_i^* + b_i^* = (r+b)/L$. Right: RedBluePaths($\mathcal{R}^*$) routes *either* r_i^* red *or* b_i^* blue uppermost paths; thus, the paths do not go below the boundary of the slot.

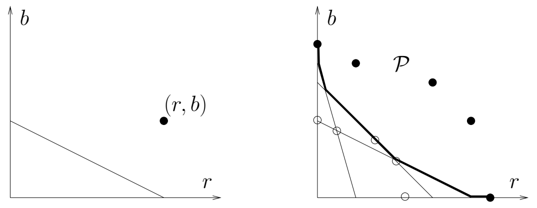


Figure 9: Left: The line segment $(0,b)$-$(r,0)$ is the dual of (r,b). Right: We obtain the upper envelope of the segments dual to the Pareto optimal pairs (filled circles); the half-optimal solutions (hollow circles) are below the envelope.

The number of blue paths routed is

$$\sum_{i \notin \max_\ell(\mathcal{R})} b_i \quad = \quad b - \sum_{i \in \max_\ell(\mathcal{R})} b_i \quad \geq \quad b - \ell b/L$$

□

Approximating Pareto-optimal solutions. Applying Theorem 7 to all possible values of (r, b) such that there may exist r red or b blue paths, we obtain an approximation to the *Pareto frontier* of optimal solutions to the problem. Specifically, for a pair of numbers (r, b) let the *dual* of the pair be a line segment from $(0, b)$ to $(r, 0)$ in the (r, b)-plane (Fig. 9). Let $\mathcal{P}$ be the set of *Pareto optimal* pairs (r, b), i.e., such that there exist r and b red and blue paths through the domain ((r, b) is feasible), but $(r, b+1)$ and $(r+1, b)$ are not feasible pairs. We say that pairs $(r/2, b/2)$ for $(r, b) \in \mathcal{P}$, are *half-optimal* solutions.

Let r^* (resp., b^*) be the maximum number of red (resp., blue) paths that can be routed in the domain without routing any blue (resp., red) paths. For each pair $(r, b) \in [0, r^*] \times [0, b^*]$ and each $\ell = 0, 1, \dots, r$ we apply Theorem 7 with $L = r$. This gives a set of line segments, which includes the segments dual to the points in $\mathcal{P}$. Since for each pair $(r, b) \in \mathcal{P}$ we obtain at least $(r/2, b/2)$ paths (by setting $\ell/L = 1/2$), half-optimal solutions lie below the upper envelope of the segments.

6 Small number of holes

In this section we show that our problems are tractable when the number of holes is constant:

Theorem 8 (stated again). *If the number of holes in the domain is bounded ($h = O(1)$), both the Red/Blue and the Two Widths paths problems can be solved in polynomial time.*

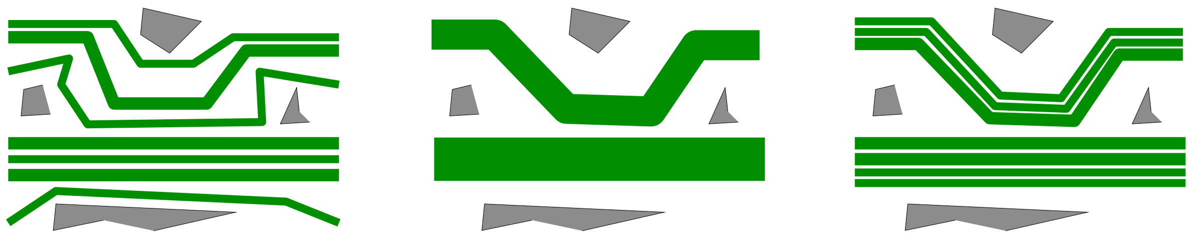


Figure 10: Two bundles (left) may be transformed into two thick paths (middle). Then the order of width-2 and width-3 paths may be changed so that there is one switch per bundle.

Proof. We first prove the theorem for the Two Widths paths problem. Take an optimal collection of paths in the Two Widths paths problem. Fix the start and the destination of each path, and consider the collection of *shortest* paths with the fixed starts and destinations. A *threading* of a source-sink path is a vector of length h, indicating for each hole whether the hole is above or below the path [15]. Call a (maximal) set of paths of the same threading a *bundle*. By the Continuous Flow Decomposition Theorem [15], each bundle can be pulled taut so that it becomes *one* thick path, with the thickness equal to the total width of the paths in the bundle (Fig. 10).

Number the bundles in the order as they appear along the source when going from T to B. If a hole H is above bundle i, then it is also above bundle $i+1$. Thus, the number of threadings of the bundles – and hence the number of the bundles – is at most $h+1$. Within one bundle, the paths may be reordered so that there is at most one switch in the paths' type sequence: first all width-2 paths, and then all width-3 paths.

Moreover, if bundle i only has one type of paths, there is no switch in the bundle. Otherwise, the paths in the bundle can be ordered so that there is no switch when going to the bundle i from the bundle $i-1$ (e.g., if the bundle $i-1$ had width-2 paths below width-3 ones, then the bundle i can have width-2 paths above width-3 ones). This way we have at most one switch per bundle, and the total number of switches is at most $h+1$.

Theorem 8 for the Two Widths paths problem follows now from Lemma 6. The above proof extends verbatim to the case when there are more than two, but a constant number, of path widths. On the contrary, if the number of widths is large the problem becomes NP-hard by a reduction from 3-PARTITION (Fig. 11).

The proof for the Red/Blue paths problem is similar. Suppose that the first path is red. So the type sequence τ starts from some number, m, of reds, and then follow some blues. Consider the first time the type sequence switches back to red; say this happens when going from kth path to $(k+1)$-st (τ_k = blue, τ_{k+1} = red). Do the "bubble sort" on the paths: Try swapping the $(k+1)$st and the kth paths; if this is possible, do the swapping (so that the kth path is red) and try swapping the kth and the $(k-1)$st path, etc. Continuing this way, either the red path floats all the way up to become the $(m+1)$st path, or it gets stuck; in the former case, try swapping the $(k+2)$nd (red) path with the $(k+1)$st (blue) path. In the end, either you have reduced the number of swaps by 2 (because the first layer of blue paths "drowned" down the next layer of blues), or you get stuck. Refer to Fig. 12.

But what does it mean to "get stuck"? It means that a red path ρ cannot be switched with a blue path β. This can only be due to either ρ passing through a blue hole B or β passing through a red hole R (or both). We charge the blue-red switch to the holes, and continue the bubble sort starting from the next blue-red switch; whenever we are stuck we again charge the blue-red switch to the holes that prevent the switch. It is easy to see that no hole is charged twice. Indeed, two consecutive charged holes are either of different colors or are separated by paths of their common color.

Thus, the number of blue-red switches is at most the number of the holes, and overall the number of the switches is at most $2h$. The claim of the theorem follows now from Lemma 6. □

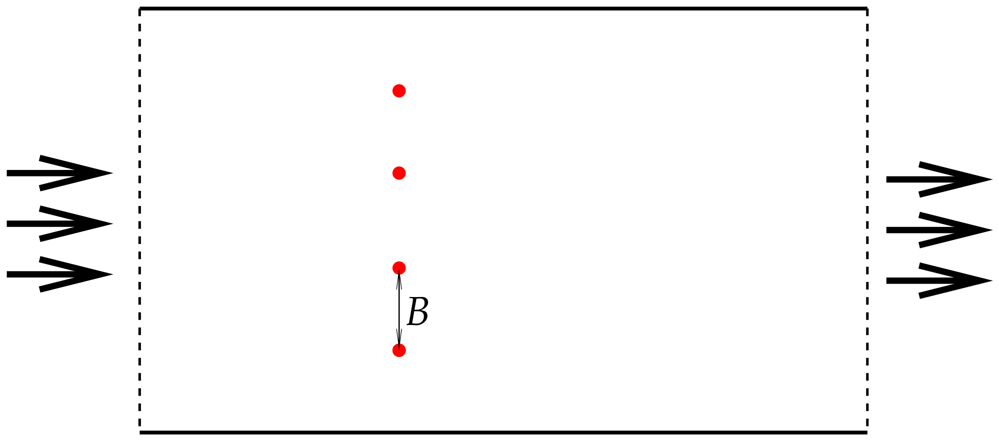


Figure 11: The 3-PARTITION problem asks if the numbers $a_1 \dots a_{3n}$ ($\sum a_i = nB$, $B/4 < a_i < B/2$) can be split into n groups of 3, such that the sum of numbers in each group is B. The polygonal domain created from a 3-PARTITION instance has $n-1$ holes; each hole is a point. The distance between ith and $(i+1)$st hole is B. The paths' thicknesses are $a_1 \dots a_{3n}$; all paths can be packed into the domain if and only if the 3-PARTITION instance is solvable.

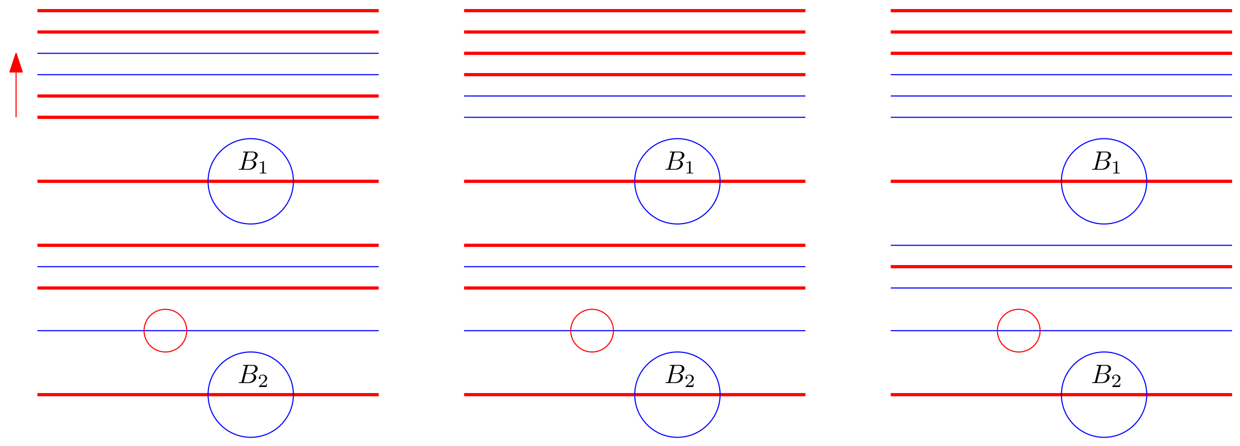


Figure 12: Left: Initial type sequence. Middle: Two red paths floated up, but the third is stuck because it intersects a blue hole B_1. Right: Two charged blue holes B_1, B_2 are separated by blue paths: the second layer of blue paths is below B_1 but is above B_2.

7 Practical Heuristics: Implementation and Experiments

We have implemented algorithms for multi-class routing and applied them to weather data for use in capacity estimation experiments and in NASA's FACET simulation tool. The FACET-based scenarios are performed in collaboration with colleagues at Metron Aviation and are reported separately [10, 21]; here, we report experimental results involving algorithmic design choices.

The implementation is in C++. The user interface allows one to import weather data, specify the outer boundary of P, mouse-in polygonal obstacles, and select algorithm parameters. The algorithms implemented are based on uppermost (or bottommost) filling of P with thick paths, according to a type sequence τ. Paths are inserted one by one, checking for feasibility according to the type of the path and the types of the obstacles. For purposes of these experiments, rather than doing offsetting using Voronoi methods, we use a simple method of computing bottommost routes by means of a depth-first search in a search grid that

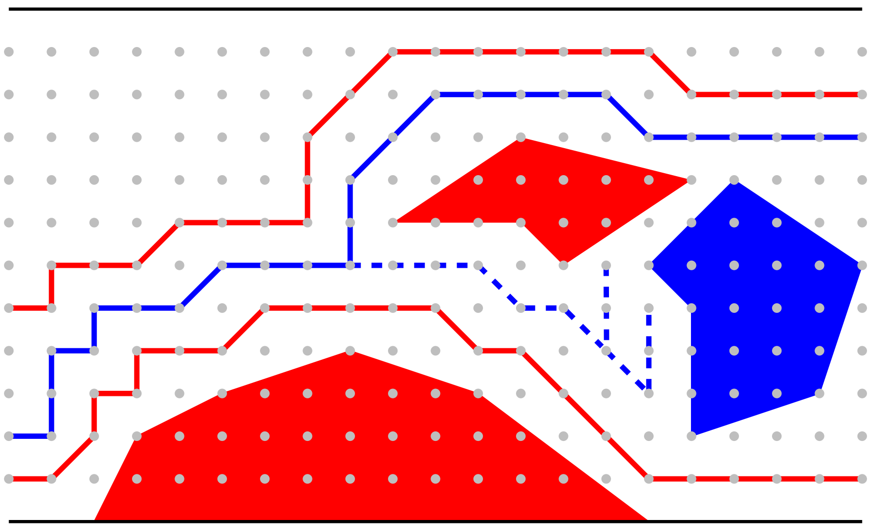


Figure 13: Routing bottommost paths in the grid. The grid step is equal to the path width. Starting at the lowest available grid point at the sink, the path proceeds to the next grid point, with the priority to turn to the right as much as possible; the constraints are that the path stays away from obstacles and the already routed paths, and that it is x-monotone. The routing continues until the sink is reached. If the sink is not reached, the search retracts (e.g., the DFS retracts twice when routing the second, blue path).

is superimposed over the domain (Fig. 13); this permits us to easily adapt to a wide variety of constraints, including weather data of various types, turn constraints, directionality constraints, etc. It also allows us to use sets of constraints (obstacles) that may not be disjoint from each other, as we simply have to have a predicate that tests if a given segment connecting two grid points satisfies the requisite lane width and obstacle type constraints.

For the experiments reported here, we imposed a monotonicity constraint that routes be x-monotone (in the direction of the flow of air traffic from a western source edge to an eastern sink edge). The monotonicity constraint is often imposed on ATM-relevant routes, as one does not fly routes that are highly non-monotone, with many switchbacks.

Data Sets. We used two kinds of data: real weather data, and simulated sets of obstacles. For real weather data, we used segmentations at two levels, a high threshold for red obstacles, and a lower threshold for blue obstacles. The weather data is of three varieties: convective weather (vertically integrated liquid), icing data, and turbulence data (Graphical Turbulence Guidance) [17]. Real weather data is usually composed of big blocks of weather constraints staying near each other, and the red constraints are typically inside the blue constraints. The example shown in Fig. 14 is based on a sample of convective weather data.

For simulated data, we randomly generate two sets of quadrilaterals, with one set designated as red obstacles and the other set as blue obstacles; quadrilaterals from the two sets possibly overlap with each other, and/or cross the boundary of the outer polygon. In more details, for each set, we use a uniform distribution in an axis-aligned bounding box of P to generate a pre-specified number of center points; around each center point, we first generate an L_1-metric circle of radius R from a fixed uniform distribution, and then perturb the coordinates of each vertex by $U(-R, R)$ – a random amount uniformly distributed in $(-R, R)$. An example is shown in Fig. 16(a).

Enumeration of Type Sequences. We experimented with different ways of enumerating the set of $\binom{r+b}{r}$ type sequences. If (r, b) is feasible, i.e., there exist r red and b blue paths, we hope, by choosing a smart way of enumeration, that we hit the first feasible type sequence after only a small number of steps.

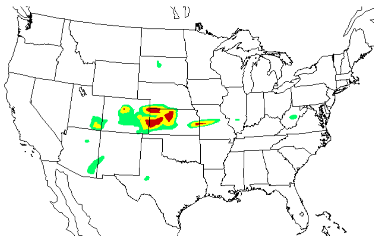
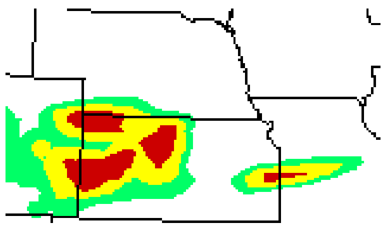
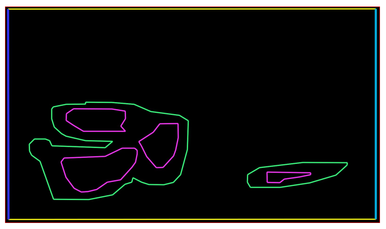
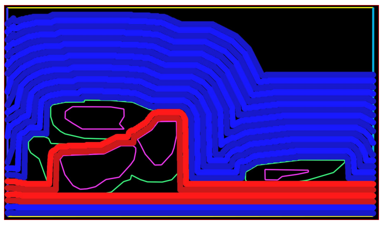
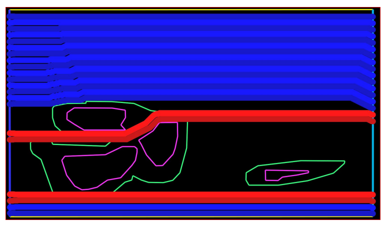
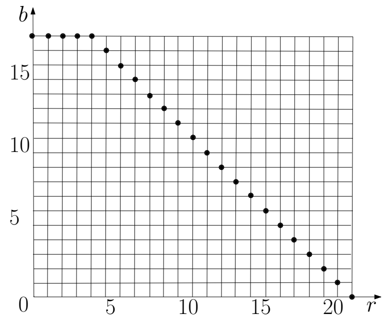



(a) (b) (c)

Figure 14: Example of a real weather test case: (a). The Map of Continental United States, (b). The Region of Interest: Nebraska, Kansas and part of Wyoming, Colorado, Iowa and Missouri, (c). The test case extracted from (b): Yellow and Green hazards are light weather constraints and red ones are severe weather constraints. We consider yellow hazards to be blue constraints and red hazards to be red constraints. The test case is a typical one extracted from real weather data: the constraints are forming large clusters, staying near to each other, and the red constraints are typically inside the blue ones.

(a) (b) (c)

Figure 15: Example of routing in the test case from Fig 14(c): (a). Result of (bottommost) routing 4 red paths and 17 blue paths, (b). result of heuristic shortening, (c). the Pareto frontier of the routing instance.

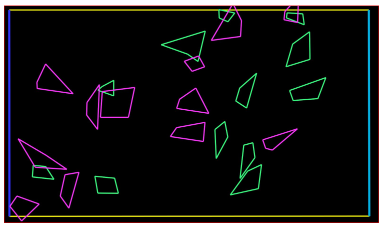
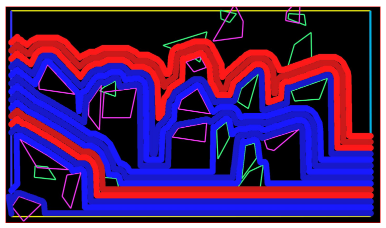
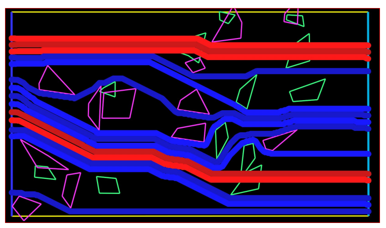



(a) (b) (c)

Figure 16: Example of routing in a randomly generated test case: (a). The instance, (b). Result of (bottommost) routing 5 red paths and 9 blue paths, (c). result of heuristic shortening.

Because only red paths may go through blue obstacles, intuitively it is beneficial to have red paths go through blue obstacles whenever possible in order to leave enough free space (without any obstacle) for the blue paths. Therefore, in a feasible routing scheme, it is likely that the red paths stay together inside a blue obstacle, so that the blue ones can stay together in the "free" space. This leads to an intuition that, in most cases, a type sequence with a smaller number of color changes has a greater chance of being feasible. Hence, we experimented with the following enumeration strategies:

(1) Starting with the initial sequence of r R's, followed by b B's, we enumerate the type sequences in lexicographically increasing order.

(2) Break into subcases according to the number, L, of color changes in the subsequence, and enumerate in one of the following ways: (a) increasing L, lex-increasing; (b) decreasing L, lex-decreasing; (c) increasing L, lex-decreasing; (d) decreasing L, lex-increasing.

If (r, b) is infeasible, the hope is to be able to conclude so *without* having to try *all* permutations. To achieve this goal, we implemented a method of pruning the search tree using red and blue *indices*: for each node of the search grid, we first compute the maximum number of red/blue paths that can pass above it. If, during our bottommost fill algorithm according to a type sequence we ever pass through a node such that its blue index is less than the number of blue paths remaining in the type sequence, we terminate early, concluding that any sequence that begins with the prefix of the sequence just tested cannot lead to a successful routing of r red and b blue paths.

A Heuristic for Short Paths. While it remains an open problem to compute a set of thick paths to minimize the sum (or the maximum) of the path lengths, we implemented a method for local optimization of paths for a given feasible pair (r, b). In addition to studying the length optimization problem, our goal was to compute paths that could be used in simulation experiments with FACET on real data; thus, we were expected to generate routes that were at least plausibly flyable (and bottommost fill routes fail this criterion).

Our heuristic does the following tautening of the routes computed by bottommost fill for a feasible pair (r, b) with a given type sequence. First, the topmost route (route number $r + b$, denoted γ_{r+b}) is shortened by computing a shortest thick route, γ^*_{r+b}, of the width corresponding to the class of the route, from source to sink, lying between route γ_{r+b-1} and T, the top of P, while avoiding the relevant obstacles. (Note that the homotopy type of γ^*_{r+b} may be quite different from that of γ_{r+b}.) Then, iteratively, each route γ_i is shortened by computing a shortest source-to-sink thick path that lies between γ_{i-1} and γ^*_{i+1}. Fig. 15(a),(b) and Fig. 16(b),(c) show the results of the bottommost routing and the tautening process.

Experimental Results

We ran each data set multiple times, for each enumeration strategy, iterating over choices of (r, b), thereby obtaining the Pareto frontier; see Fig. 15(c). In order to test instances that were *not* feasible, we also ran the algorithm for choices of (r, b) just above the Pareto frontier (i.e., at $(r, b^*(r) + 1)$). We measure (a) the number of type sequences tested before termination, (b) whether the pair (r, b) is feasible or not, and (c) the path length (sum of lengths and max of lengths) of the solution after local tautening.

We tested over 2000 test cases on our data sets. For both feasible and infeasible pairs (r, b), we recorded the average number of type sequences tested for each of the enumeration strategies. For feasible pairs, we differentiate the test cases by *heavy loads* and *light loads*, where heavy loads test cases refer to the pairs (r, b) on the Pareto frontier and light loads cases refer to pairs (r, b) that are inside the Pareto frontier. For each of the test cases, we tested six (r, b) pairs corresponding to three light loads cases, two heavy loads cases, and one infeasible case.

The test results are shown in Table 1. For all strategies, we see that we obtain early termination with success far sooner than the overall average value of $\binom{r+b}{r}$. This suggests the practical efficiency of our implementation, since testing a given type sequence is a very fast operation (essentially, a depth-first search, in the grid, for the bottommost paths).

We see that for feasible pairs, though, the enumeration strategies 2(a) and 2(c) are the overall winners (this is inline with Theorem 8 as the number of holes is small). They are better strategies than simply using lexicographic order, which proves our conjecture above that enumerating type sequences in order of increasing L (the number of color changes) allows one to spot a feasible type sequence more quickly. In heavy load cases, where there are usually only few successful type sequences, the strategies 2(b) and 2(d) perform poorly. In real weather data sets, the weather constraints are typically forming large clusters, staying near to each other, and the red constraints are typically inside the blue constraints. Consequently, the red paths are more likely to be adjacent to each other so that they take more space occupied by blue constraints. Therefore, for real weather test cases, the strategies 2(b) and 2(d) perform extremely bad compared to strategies 2(a) and 2(c).

Data Sets	**Random Data Sets**			**Real Weather Data Sets**		
Strategy	**FCHL**	**FCLL**	**INF**	**FCHL**	**FCLL**	**INF**
(1)	9.38 (0.4s)	4.09 (0.3s)	58.48 (5.2s)	3.26 (0.3s)	3.04 (0.3s)	50.87 (3.1s)
(2a)	4.25 (0.4s)	2.27 (0.2s)	58.48 (5.2s)	3.80 (0.3s)	2.85 (0.2s)	50.87 (3.2s)
(2b)	15.28 (1.8s)	6.67 (0.5s)	58.48 (5.2s)	31.23 (2.1s)	17.13 (1.6s)	50.87 (3.2s)
(2c)	3.81 (0.3s)	2.20 (0.2s)	58.48 (5.2s)	3.15 (0.2s)	2.65 (0.2s)	50.87 (3.2s)
(2d)	15.33 (1.8s)	6.96 (0.5s)	58.48 (5.2s)	31.79 (2.1s)	17.31 (1.5s)	50.87 (3.2s)
$\binom{r+b}{b}$	245.92	55.75	320.35	957.65	212.39	4077.06

Table 1: The comparison among different enumeration strategies. FCHL stands for feasible cases (heavy loads), FCLL stands for feasible cases (light loads) and INF stands for infeasible test cases. We provide the average number of type sequences tested to find a feasible one or report infeasible, and the average running time of our algorithms (in seconds, in parentheses). The first column shows the strategy used, where (1) stands for lexicographically increasing order, (2a) for increasing L, the color changes, lex-increasing, (2b) for decreasing L, lex-decreasing, (2c) for increasing L, lex-decreasing, (2d) for decreasing L, lex-increasing; $\binom{r+b}{b}$ is the binomial coefficient, i.e., the overall number of possible type sequences that one would have to test using brute-force enumeration – it provides the baseline to which our methods are compared. The remaining columns show the average number of type sequences tested to find a feasible one for randomly generated data sets and real weather data sets.

For infeasible pairs (r, b), a naive approach tests all $\binom{r+b}{r}$ type sequences. With our early termination method to do pruning of the enumeration, though, we find that, on average, a small fraction, 39.55%, of the total number of possible sequences needs to be tested before discovery of infeasibility. (It is easy to see that all five of the enumeration strategies search the same subset of sequences, so we do not differentiate by strategy.)

We also examined the path lengths obtained by our heuristic shortening method. We found that there is little difference among the results according to different enumeration strategies (about a 10% variability). We also found that the total length of the routes are fairly close (within about 10%-15%) to the lower bound on possible length (computed using $(r + b)$ times the length of a single shortest path from source to sink). We do not provide lengths related data because the path lengths highly depend on the size of the region of interest.

8 Conclusion

We considered routing multiple types of paths in a polygonal domain containing obstacles of multiple types. The very basic versions of the problem have been proved to be NP-hard. We presented approximation algorithms for different variations of the problem, as well as efficient heuristic to find the paths amidst real-world and synthesized obstacles.

We left open approximating the maximum number of blue paths that can be routed while ensuring that a specified number of red paths exists in the domain. Another natural problem to study is maximizing the total number of all-type paths routed. For the Two Widths problem, we were not able to show hardness in the case when the width of the thinner path divides perfectly the width of the thicker ones; say, if $w_1 = 1, w_2 = 2$. On the experiments frontier, it would be interesting to investigate alternative heuristics for minimizing path lengths.

References

[1] N. M. Amato. An optimal algorithm for finding the separation of simple polygons. In *WADS '93: Proceedings of the Third Workshop on Algorithms and Data Structures*, pages 48–59, London, UK, 1993. Springer-Verlag.

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

[3] C. Chekuri. Multicommodity flow, well-linked terminals and routing problems. In M.-Y. Kao, editor, *Encyclopedia of Algorithms*. Springer, 2008.

[4] P. Crescenzi and V. Kann. A compendium of NP optimization problems. In *Complexity and Approximation. Combinatorial Optimization Problems and Their Approximability Properties*. Springer, Berlin, 1999.

[5] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.

[7] 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.

[8] A. Klein, L. Cook, B. Wood, and D. Simenauer. Airspace capacity estimation using flows and weather-impacted traffic index. In *Integrated Communications, Navigation and Surveillance (ICNS'08)*, pages 1–12, 2008.

[9] J. Krozel, W. McNichols, J. Prete, and T. Lindholm. Causality analysis for aviation weather hazards. In *AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, Anchorage, AK, Sept. 2008.

[10] J. Krozel, J. S. B. Mitchell, V. Klimenko, T. Lindholm, J. Prete, N. Downs, and S. Krishna. Translation of weather information to traffic flow management (TFM) impact: Weather translation model for TFM decisions including FACET demonstrations. Technical Report 34N0707-002-R0, Metron Aviation, October 2008.

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

[12] J. Krozel, J. Prete, J. S. B. Mitchell, J. Kim, and J. Zou. Capacity estimation for super-dense operations. In *AIAA Guidance, Navigation, and Control Conf.*, Aug 2008.

[13] D. Michalek and H. Balakrishnan. Identification of robust routes using convective weather. In *Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM)*, 2009.

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

[15] J. S. B. Mitchell and V. Polishchuk. Thick non-crossing paths and minimum-cost flows in polygonal domains. In *Proceedings 23rd ACM Symposium on Computational Geometry*, pages 56–65, 2007.

[16] J. Prete, J. Krozel, J. S. B. Mitchell, J. Kim, and J. Zou. Flexible, performance-based route planning for super-dense operations. In *AIAA Guidance, Navigation, and Control Conf.*, Aug 2008.

[17] R. Sharman, C. Tebaldi, G. Wiener, and J. Wolff. An integrated approach to mid- and upper-level turbulence forecasting. *Weather and Forecasting*, 21:268–287, 2006.

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

[19] H. Swenson, R. Barhydt, and M. Landis. Next generation air transportation system (NGATS) air traffic management (ATM)-airspace project. Tech. Report, Version 6.0, NASA, 2006.

[20] J. P. van den Berg and M. H. Overmars. Prioritized motion planning for multiple robots. In *Proceedings IEEE/RSJ Int. Conf. on Intelligent Robots and Systems - IROS'05*, pages 2217–2222, 2005.

[21] 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(3):1–26, 2011.