Applying Geometric Thick Paths to Compute the Maximum Number of Additional Train Paths in a Railway Timetable

Anders Peterson [a], Valentin Polishchuk [a], Christiane Schmidt [a]
[a] Communications and Transport Systems, ITN, Linköping University, Norrköping, Sweden. Email: firstname.lastname@liu.se

Abstract
Railway timetabling is a prominent research area in railway research. The timetable is usually shown as a time-space diagram. However, even algorithms that try to adapt/add to an existing timetable rely mainly on mixed integer programming, but do not use the geometric representation of the timetable. In this paper, we consider the problem of determining residual train paths in an existing timetable. We aim to restrict possible disturbance on existing (passenger) traffic, and, hence, insert train paths of a specified minimum temporal distance to other trains. We show how we can extend algorithms for thick paths in polygonal domains to compute the maximum number of trains with a specified robustness to insert.

Keywords
Railway Timetabling, Residual Capacity, Polygonal Domains, Geometric Thick Paths

1 Introduction

Both passenger traffic and freight traffic volumes in Sweden significantly increased over the last 20 years—from 1996 to 2016 by 82% (from about 7 to 12.8 billion passenger kilometers) and by 23% (from about 55 to about 68 million tonne-kilometers), respectively, see Trafikanalys (2017a,b). Over the last years the freight volume transported via railway within the EU has stagnated, but both road congestion and oil prices make road transport more expensive and less attractive. In contrast, railway transport is safer and more environmental friendly. However, already with the current traffic load, railway infrastructure is often overloaded. This is particular true for marshalling yards: trains that are already completed occupy highly demanded space until their departure. To free this capacity, both the freight operator and the infrastructure manager (IM) often agree in their goal to depart ahead of schedule. Today, such a request is answered manually by looking a few stations ahead, and if the completed freight train will not interrupt operations on this limited considered stretch, an earlier departure will be permitted. This procedure hardly takes into account the already congested rail network, where freight traffic interacts with passenger traffic with high requirements on punctuality. The early departed freight train might be stuck at a sidetrack before its destination for long stretches of time due to the limited spatial and temporal horizon considered for the decision by the IM. Having several early departing freight trains only worsens this situation. Similarly, early arrivals contribute to congestion, when no track capacity is available at the destination yard.

To make sure both that the existing (passenger) traffic is not affected by the train path of the freight train and that the freight train actually obtains a feasible train path to its destination, it is essential to optimize the process. In Ljunggren et al. (2018), we proposed an algorithm that computes a maximum robust train path for inserting a single additional train (at a time). Here, we aim to determine how many additional trains with certain properties can be added to the existing timetable, that is, we aim to determine the residual capacity for additional train paths within given time windows. This could be particularly interesting for adding freight trains, but also adding passenger trains can be of interest.

Timetabling is a problem that has been extensively studied, in the majority a new timetable, or a larger part of it, is constructed from scratch, see, e.g., Hansen and Pachl (2014); Liebchen (2008) for an overview.

Adding a new train to an existing timetable was considered, e.g., by Burdett and Kozan (2009). Flier et al. (2009)(see also Flier (2011)) present a shortest path model using a time-expanded graph, which integrates linear regression models based on extensive historical delay data, that gives Pareto optimal train paths w.r.t. travel time and risk of delay. Ingolotti et al. (2004) consider adding new trains to a heterogeneous, heavily loaded railway network, and aim to minimize the traversal time for each additional train. Cacchiani et al. (2010) also consider the problem of inserting a single freight train into an existing schedule of fixed passenger trains. They assume that the operator specifies an ideal timetable that the IM can modify, which also includes the use of a different path. Cacchiani et al. aim to add the maximum number of new freight trains, such that their timetable is as close as possible to the ideal one. To do so, they use a heuristic algorithm based on a lagrangian relaxation of an Integer Linear Program (ILP).

UIC (2004) has developed a compression technique for computing capacity utilization. This technique is widely used for assessing capacity utilization in the railway network. For example the Swedish infrastructure manager routinely makes an annual report about the network congestion (Trafikverket (2018)). The corresponding analysis for year 2011 has also been presented in English, see Grimm (2012). The UIC 406 compression technique is an easy and effective way of estimating the capacity consumption, but it is possible to expound it in different ways leading to different estimates. Landex et al. (2006), who explain how the method has been implemented in Denmark, show the importance of choosing the right length of the line sections and examine how line sections with multiple tracks are considered. Also Lindner (2011) discusses some aspects of this problem. In particular, the UIC 406 code calculates a capacity consumption, that is, it evaluates how much of the available capacity is consumed by the existing traffic. It first compresses the timetable, that is, the existing train paths on the considered line section are shifted as close together as possible. At this stage, they represent trains running within a certain time interval, but no longer are considered during the actual time they occupy the line. After this shifting certain blocking time elements and indirect occupancies are integrated to obtain the capacity utilization. The extension of the compression technique given in the second edition (UIC (2013)) mainly concerns how the method can be applied at station areas and in complex nodes. For the aggregated type of analysis in an annual report, we believe that the UIC compression technique is well suited. However, for determining the actual number of executable train paths between two nodes, which can be added to the existing timetable, the method is insufficient. In this paper, we address the problem to obtain as many train paths that such a network part still allows. Moreover, we allow a trade-off between adding further trains and influencing the existing trains as little as possible: the required temporal distance to the existing train is an input parameter to our computation. When we use the minimal required temporal distance we can add more trains than with a larger temporal distance, however, this comes at the price of a higher impact on other trains. Additionally, we may add train paths over topological different routes.

Pellegrini et al. (2017) and Lucchini et al. (2001) considered the saturation problem: an existing (possibly empty) timetable and a set of saturation trains are given, and the goal is to add as many trains to the timetable as possible. Lucchini et al. (2001) use the CAPRES method—in which stations and junctions are modeled as a graph on which a constraint program is solved—to determine how many freight trains can run on the North-South rail corridor in Switzerland. Pellegrini et al. (2017) used a MILP approach, . In the saturation problem, various train types (possibly with number of trains per type) are considered, while we assume a specific type, but aim at disturbing the passenger traffic as little as possible, and obtain a trade-off with the temporal distance to other trains. CAPRES uses heuristics, Pellegrini et al. (2017) output the best feasible solution found until a time limit is reached, while we present an optimal solution.

A timetable is usually shown as a time-space diagram. However, even when we only aim at inserting something into an existing timetable, or make some limited adaptations to it, this geometric representation is not used in algorithms (while it is used in the practical, mainly manual, process). We present a roadmap on how we will make use of this geometric representation in Section 2. There exist various results on thick paths and flows within a polygonal domain, we present basic definitions and the results important for this paper in Section 3. In Section 4 we describe our general approach, and detail in Subsection 4.1 how we construct our polygonal domain, in Subsection 4.2 how to extend the path computation to our needs, and in Subsection 4.3 how we combine these to compute the maximum number of additional trains.

2 Roadmap for Our Strategy

We aim to insert additional trains to a given timetable, where we consider the existing trains as fixed. When we consider the time-space diagram of the given timetable (where we consider time on the x-, and space on the y-axis), inserting new trains means to route paths from their start to their end station. However, these paths cannot be arbitrarily close to each other: we need to keep a certain temporal distance to consecutive trains on any track. Let d_s and d_o denote this temporal distance for trains running in the same and for trains running in the opposite direction as the trains to be inserted, respectively (these values may coincide, but will usually not). So, instead of thinking of the existing trains as line segments in the time-space representation of the timetable, we can think of them as "blown-up" line segments (blown up by the temporal distance), that is polygons. Similarly, the trains that we route are not just curves in $\mathbb{R}^2$, but *thick paths*, where the thickness represents the temporal distance we need to keep to neighboring trains, d. For our algorithm, we can choose d according to the minimal necessary temporal distance between trains, or—with the motivation of disturbing the existing trains as little as possible, e.g., because we insert freight trains shortly before operation, see Ljunggren et al. (2018)—we can choose a larger value.

We will use concepts from Computational Geometry that allow routing such thick paths. Of course, if we would just "blow up" all existing trains, no thick paths could overtake any other train at a station, as the line segments representing the stations and the existing trains would constitute obstacles. To enable such options, we will need to make certain adaptations to the "blown up" time-space representation, we show how to construct the appropriate polygonal domain for our problem in Subsection 4.1. We are given a time-window for possible departure on the start station and a time-window for possible arrival at the end station, these will coincide with special edges, the *source* and *sink*, of the polygonal domain, between which we need to route the thick paths.

To determine the maximum number of trains that we can insert into a timetable, we then need to determine the maximum number of thick paths that we can route in that polygonal domain. However, we do not want to route arbitrary thick paths, for example, paths that are parallel to the y-axis, would mean that our trains run with infinite speed. We are given a maximum speed, and this limits the slope of the feasible train paths. If we denote time along the x-axis, we of course aim for x-monotone paths (as we should not allow our trains to go back in time, implementing none-x-monotone paths will result in definite problems). Hence, we aim for thick (non-crossing) x-monotone paths of a limited slope.

Polishchuk (2007) presented an algorithm to compute the maximum number of (x-) monotone thick non-crossing paths, we will describe this algorithm in Section 3.

We need to extend the algorithm for x-monotone thick paths to compute thick paths of a limited slope, see Subsection 4.2. We will then combine this general algorithm (Subsection 4.2) and the constructed polygonal domain (Subsection 4.1) to determine the maximum number of additional train paths in Subsection 4.3.

3 Routing a Maximum Number of Thick Paths through a Polygonal Domain

Various authors studied maximum flows in geometric domains Hu (1969); Hu et al. (1992); Strang (1983); Mitchell (1990); Eriksson-Bique et al. (2014), Mitchell (1990) presented efficient algorithms for computing maximum flows in polygonal domains. Here, we are, however, interested in routing thick paths through a domain, and not a flow, see Polishchuk (2007). A *thick path* is the Minkowski sum of a "normal" (zero-thickness, or "thin") path and a disk (for some metric). Hence, we try to wire a maximum number of "threads" (of a specific thickness) between given edges of the domain. The domain is usually given by a polygon. We detail the necessary notation in Subsection 3.1, and present a polynomial-time algorithm by Arkin et al. (2010) (see also Polishchuk (2007)) to compute the maximum number of thick paths in Subsection 3.2.

3.1 Notation

We use the notation given by Polishchuk (2007). A polygon can either be simple, or contain holes. Both are possible inputs: We are given a polygonal domain, Ω, defined by the outer simple polygon, P, and a set $\mathcal{H}$ of h holes $H_1, \ldots, H_h$ within P. (In case of a simple polygon, we have $\mathcal{H} = \emptyset$.) For any set $Q \subset \mathbb{R}^2$ we let $\delta(Q)$ denote its boundary; if $Q \neq \delta(Q)$, that is, if Q has interior points, we will assume that Q is open (i.e., $\forall p \in \delta(Q), p \notin Q$).

Two edges on $\delta(P)$ are specifically marked: they are the *source* and the *sink* in our domain—that is, the edges from and to which we want to route the thick paths. We denote them by Γ_s and Γ_t. $\delta(P) \setminus (\Gamma_s \cup \Gamma_t)$ has two connected components, the "top" T, and the "bottom" B.

A "thin" (normal) path π is a simple curve. For $r > 0$ we let $\mathcal{C}_r$ denote the open disk of radius r centered at the origin; we use $\mathcal{C} = \mathcal{C}_1$. For a set $S \subset \mathbb{R}^2$ we let $(S)^r$ denote the Minkowski sum $S \bigoplus \mathcal{C}_r$, with $S \bigoplus \mathcal{C}_r = \{x + y | x \in S, y \in \mathcal{C}_r\}$. A *thick path* is the Minkowski sum of a "thin" (normal) reference path, that is, a curve in $\mathbb{R}^2$, and a unit disk (or, more general, a disk of radius r): a thick path Π with reference path π is the Minkowski sum of π and $\mathcal{C}$, that is, $\Pi = (\pi)^1$, such that Π does not intersect P's exterior.

3.2 Computing the Maximum Number of Thick Paths

We now aim to find the maximum number of thick paths from Γ_s to Γ_t, where the paths should avoid all the obstacles (holes) and be non-crossing, that is, for any wo paths Π_i, Π_j we have $\Pi_i \cap \Pi_j = \emptyset$. This requires the interiors of paths to be disjoint, thick paths may share boundary.

No path can run outside Ω. We add B and T to the set of holes $\mathcal{H}$, that is, $H_0 = T, H_{h+1} = B$. Arkin et al. (2010) used the concept introduced by Mitchell (1990), and we follow this idea: we assume that Ω has been "perforated" at Γ_s and Γ_t, and that Riemann flaps were glued to Ω at these two perforated edges. This circumnavigates complications from a thick path protruding through Γ_s and Γ_t.

Arkin et al. (2010) showed that the maximum number of x-monotone thick non-crossing paths can be found in $\mathcal{O}(nh + n \log n)$. The routing of these thick non-crossing paths, or wires, is different to just routing the maximum number of self-overlapping thick paths (for example, there exist domains in which only the latter exist at all).

A single self-overlapping thick path from Γ_s to Γ_t that avoids all obstacles can be found by first building the offset of all holes by 1 (that is, building the Minkowski sum $(H_i)^1 \forall i$), and then solving the "usual" shortest path problem in the presence of these new, offset obstacles. The path found by this procedure yields the reference path for an optimal thick path Liu and Arimoto (1995); Chen et al. (2001). This does not translate to the case of wires (non-crossing thick paths).

The idea of the algorithm by Arkin, Mitchell and Polishchuk is to use an adaptation of the so-called "grass fire" analogy from Mitchell (1990): the free space $\Omega \setminus \mathcal{H}$ is grass over which fire travels at speed 1. All the holes are highly flammable, that is, once they are ignited, the fire moves through them with infinite speed. We start setting the bottom on fire. The *wavefront* at time τ is the boundary of the burnt grass by time τ. Whenever the fire has not hit a hole after burning for 2 time units, we can route a thick path through the burnt grass: Arkin et al. showed that a thick path does exist when the fire has burnt for 2 time units, and that routing a thick path at that point does not hamper the construction of any of the other paths in a maximum set of such paths. Once a thick path has been routed, we use the wavefront as the new bottom, and start over.

If a hole H is hit after $\tau < 2$, Arkin et al. define e to be the segment of length τ connecting H to B (or, to T, as they started the fire at the top, we will use the bottom in our application and adapted the description accordingly). We split the free space along e and around $\delta(H)$ (thus, the hole is no longer a hole), and glue a Riemann sheet to each copy of e, where we place a circular segment of radius 2 with e as chord in each. Then we continue the grass fire by igniting H and a belt of thickness τ around it, as the fire flips to the other side of H and runs there.

Polishchuk and Mitchell (2007) proved a continuous version of the discrete network Flow Decomposition Theorem, the Continuous Flow Decomposition Theorem (CFDT),

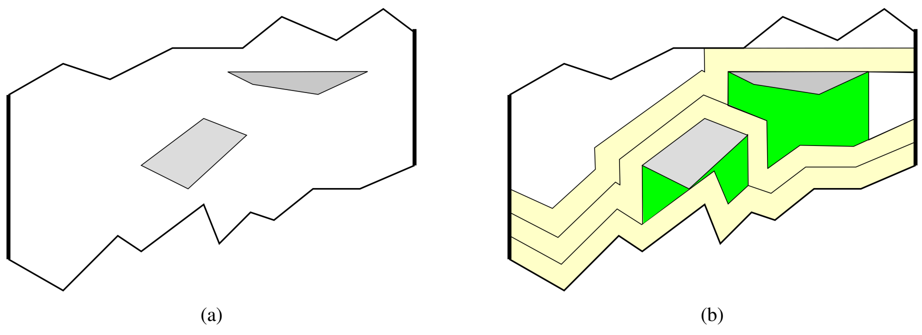
Figure 1: (a) A polygonal environment with two obstacles (gray), and the source and sink shown in bold. (b) shows the wavefronts after 2 time units each (which induce the x-monotone paths). Waterfalls are depicted in green.

which states that the support of a minimum-cost flow can be decomposed into a set of thick paths; the size of the decomposition is linear in the size of the description of the flow (Theorem 5.5. in Polishchuk (2007)). This enables the proof that the above algorithm actually routes the maximum number of thick non- crossing paths. Its runtime is $\mathcal{O}(nh + n \log n)$.

Monotone Thick Paths Polishchuk also aimed for x-monotone thick paths, where a thick path Π is x-monotone if its reference path π is x-monotone (each vertical line intersects π in at most one point). Each x-monotone thick paths is a monotone simple polygon.

To compute the maximum number of monotone thick non-crossing paths, Polishchuk extended the algorithm for the maximum number of thick non- crossing paths. First, we need a monotone $\delta(P)$. Hence, we need to add "waterfalls" (following notation from Arkin et al. (1989)): the inner x-monotone hull of P is the largest x-monotone polygon that is contained in P; to compute it (see Polishchuk (2007)), we can sweep a vertical line in x direction, for every vertex $v \in P$, connect v to the first point of P hit when going up from v, and when going down. Whenever we hit a hole (with the wavefront of our burning fire), we use the waterfalls to outer-monotonize the hole. See Figure 1(a) for an exemplary polygonal domain, and Figure 1(b) for the waterfalls of the obstacles (note that P was already monotone) and the maximum number of thick paths.

4 Inserting a Maximum Number of Trains in a Timetable

We consider the existing trains as fixed, are given a time-window for possible departure on the start station and a time-window for possible arrival at the end station, and a maximum speed for the trains to be inserted. Moreover, we are given a minimum temporal distance, d, that we need to keep between consecutive inserted trains on any track.

The idea is that thick paths through the timetable reflect train paths with a certain temporal distance to neighboring trains. That is, we still use the algorithms with the "grass fire" analogy and route trains when the fire burnt without hitting obstacles—given by other trains–for d time units. However, to be able to do so, we need to complete two steps:

1. The timetable is given by line segments for trains and stations, we need to make

certain adaptations to generate our polygonal domains: stations are not obstacles and we need to remove these lines; we need to keep a required safety distance to the existing trains, hence, we will extend the line segments to polygonal obstacles, such that a train path passing this new obstacles keeps the safety distance to the existing train, et cetera. We describe the construction in Subsection 4.1.

2. The algorithm presented by Arkin et al. (2010) is for x-monotone paths. For example, applying it to our problem, this would allow for paths parallel to the y-axis, that is, our trains would run with infinite speed. We are given a maximum speed, and this limits the slope of the feasible train paths. Hence, we need to extend the algorithm for x-monotone thick paths to compute thick paths of a limited slope, see Subsection 4.2.

Finally, we can combine this general geometric algorithm with our specific polygonal domain to compute the maximum number of trains in Subsection 4.3.

4.1 Construct Polygonal Domain from Timetable

We are given: a starting station s_0 and an end station s_M for the trains to be inserted; time windows $w_s = [w_s^a, w_s^e]$ for earliest arrival and latest departure of the trains at station s for all, or some of, the stations; the train-specific running times $t_{i,i+1}$ for the trains from station i to $i+1$ $\forall i \in \{0, \dots, M-1\}$, given by a maximum possible speed (defining the maximum slope for our thick paths); the timetable of all trains in the set $\mathcal{T}$: all trains that run in $[w_0^a - \varepsilon_1, w_M^e + \varepsilon_2]$, where ε_i is defined such that the trains that depart before or arrive after a possible path for new trains at any station are included; the required temporal safety distances d_s, d_o between any other train τ and inserted trains. For $s = 0$ the time window describes all possible departure times from the origin, and for $s = M$ the time window describes all possible arrival times at the destination. A time window at an intermediate station may also be given, e.g., due to staff schedule or wagon coupling/uncoupling, here we concentrate on the case with time windows at $s = 0, s = M$, the other case can easily be integrated in the construction, by using the algorithm between any consecutive time windows, given that the intermediate station with a time window has enough side tracks. The minimum number of train paths over all consecutive sections will determine the total number of additional train paths to be inserted.

The following construction of the polygonal domain Ω depends on d and the cone for the allowed slope (that is, different values will result in different polygonal domains for the same timetable). See Figure 2 for an exemplary construction, the time-space diagram of the considered timetable is given in Figure 2(a) (where the bold lines denote the given departure and arrival time windows).

1. **Extend the time windows by $\frac{d}{2}$ to both sides to create Γ_s and Γ_t, let $\Gamma_s = [p_1, p_2], \Gamma_t = [p_3, p_4]$, see Figure 2(b).** The train path, our reference path π, can depart anyway in $w_0 = [w_0^a, w_0^e]$ and arrive anyway in $w_M = [w_M^a, w_M^e]$, if we cut our polygon at the end of edges that represent these time windows any thick path would be restricted to that interval, hence, the train path could only depart in $w_0 = [w_0^a + d/2, w_0^e - d/2]$ and arrive in $w_M = [w_M^a + d/2, w_M^e - d/2]$. This would not be the correct solution, hence, we add $\frac{d}{2}$ to both ends of the time windows to create our source and sink edge.

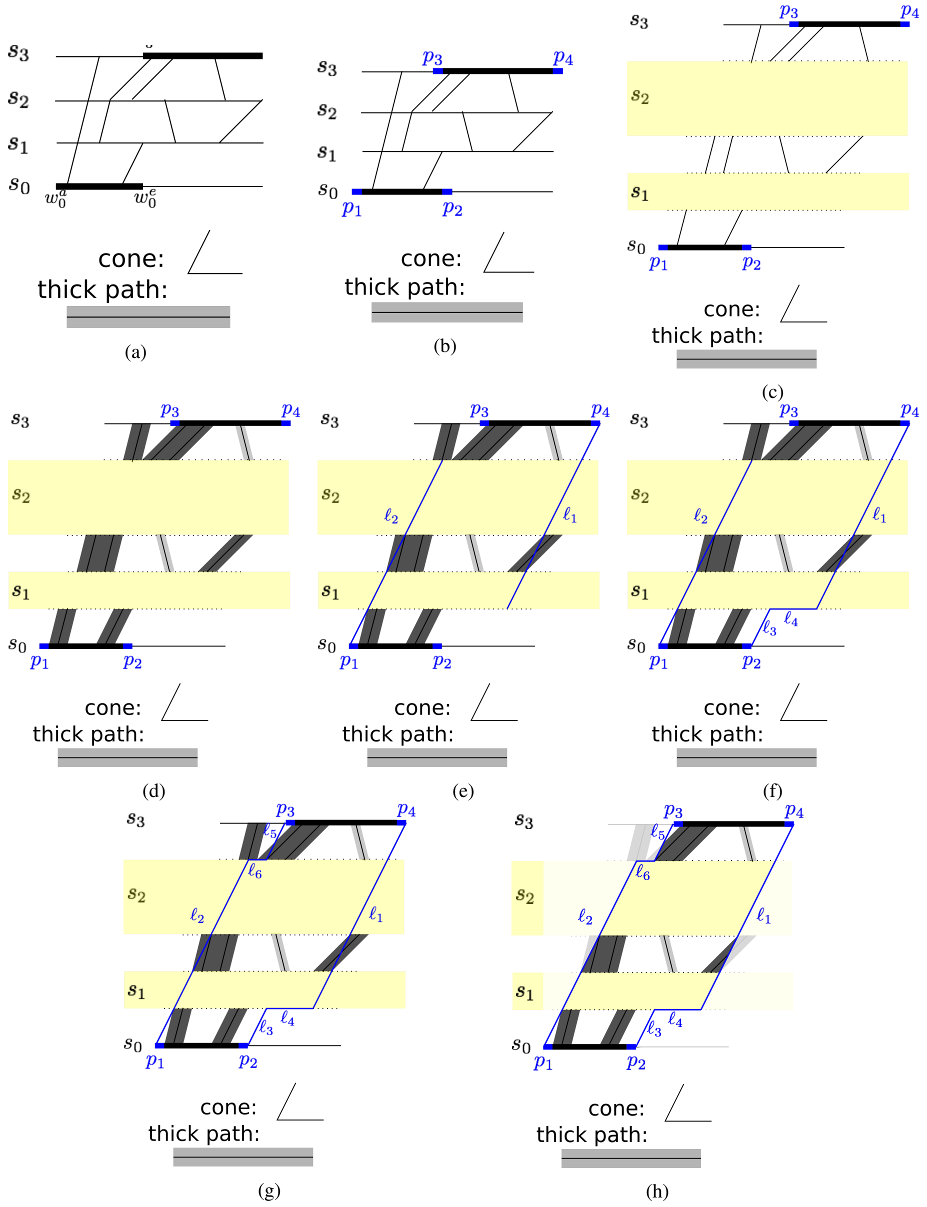

Figure 2: Example for the construction of the polygonal domain. The dotted lines and yellow blocks are given just for visual help and are not present in the domain. The width of the thick paths is shown in light gray below the diagrams. (a) Time-space diagram for the timetable we consider with given time windows $w_0 = [w_0^a, w_0^e]$ and $w_M = [w_M^a, w_M^e]$, $M = 3$. (b) Extension of the time windows by $\frac{d}{2}$ to both sides to create Γ_s and Γ_t (the bold blue and black line segments together constitute Γ_s and Γ_t). (c) "Cut open" (intermediate) stations s_1, s_2, insert a vertical distance (yellow): for s_1 we assume two side tracks, for s_2 no such limit exists. Moreover, the stations are shifted horizontally according to the procedure described in Figure 3. (d) Construction of the set of potential holes $\mathcal{H}_p$ by inserting the security distance (d_s, d_o) around the existing trains (d_s is shown in dark gray, d_o in light gray). (e) Insert ℓ_1, ℓ_2, (f) insert ℓ_3, ℓ_4, (g) insert ℓ_5, ℓ_6. (h) The polygonal domain Ω obtained by intersecting the potential holes from $\mathcal{H}_p$ with $\delta(P)$ to construct the holes $\mathcal{H}$, and P.

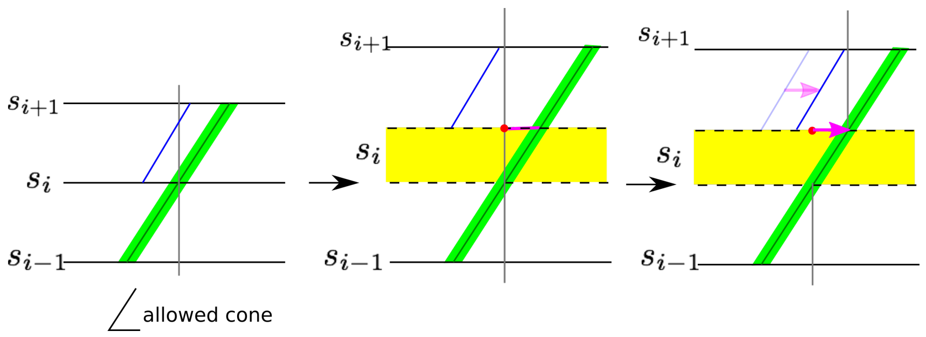

Figure 3: Time-space diagram with three stations s_{i-1}, s_i, s_{i+1}. As we cut the diagram open in step 2 (insert the yellow vertical distance), and we have a limited slope for allowed paths, we also need to shift consecutive stations horizontally. The green train arrives and departs s_i at the point in time denoted by the gray vertical line. With the added vertical distance, the red point of departure cannot be reached, hence, all stations above are shifted by the pink distance.

2. **"Cut" the diagram "open" at intermediate stations ($s_1, \ldots, s_{M-1}$), delete the vertical line for the station, insert an appropriate vertical distance ("blow up" each station) and shift the next stations according to the inserted vertical distance, see Figure 2(c).** Consider Figure 3: If we keep the stations as is, the lines would block any trains, and no train can stay at a siding at a station (and hence switch from the "left" of a train to the "right" of a train). To allow this, we cut each station open, and insert a vertical distance between arrival at station s and departure from station s (shown in yellow in Figure 3). If the station s has exactly k sidetracks, we insert a vertical distance of $k \cdot d$, if no such limit exists, we can insert a vertical distance of $\min\{|\Gamma_s|, |\Gamma_t|\}$ (which would allow the maximum possible number of additional trains to stay at a station). For the case of k sidetracks, if one or several tracks is occupied, we insert a height d rectangles to block the according height. Just inserting a vertical distance is not enough: We shift consecutive stations horizontally, as we do not allow the paths to run in parallel to the y-axis, which they would have to do when just passing through a blown-up station. So, we shift the consecutive station to the right, such that this path can be reached with limited slope. Our new trains, one of which is shown in green in Figure 3 has a limited velocity, which for us translates to the limited slope of our path. The green train arrives at station s_i at the point in time denoted by the gray vertical line. However, when we insert the yellow vertical distance, the limited slope path cannot reach the same point in time at which it should depart s_i, marked by a red point. Hence, we need to shift all stations above, and all existing trains departing s_i by the pink distance in Figure 3.

3. **Insert the security distance (d_s, d_o) around the existing trains, that is, construct the potential holes $\mathcal{H}_p$, see Figure 2(d) (for the example, we use $d_s = d$, and $d_o = \frac{d}{2}$ and denote these in dark and light gray, respectively).**

4. **Construct a line segment of maximal slope from p_4 down to s_1, ℓ_1, and from p_1 up to s_{M-1}, ℓ_2, see Figure 2(e).**

5. **Construct a line segment of maximal slope from p_2 up to s_2, ℓ_3, and a horizontal**

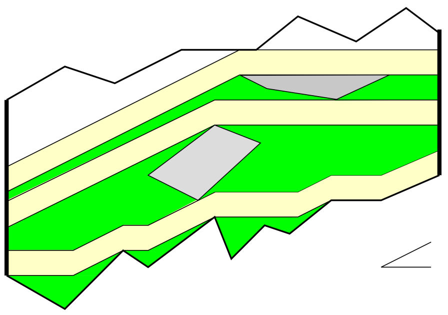

Figure 4: Paths with a limited slope that are restricted to directions in the cone shown in the lower right corner. Waterfalls are depicted in green, the wavefronts after d time units each (which induce the $\mathcal{C}$-respecting paths) are shown in light yellow. The polygonal domain is the domain from Figure 1(a).

line segment ℓ_4 from the end of ℓ_3 to the intersection with ℓ_1, see Figure 2(f).

6. **Construct a line segment of maximal slope from p_3 down to s_{n-1}, ℓ_5, and a horizontal line segment ℓ_6 from the end of ℓ_5 to the intersection with ℓ_2, see Figure 2(g).**
7. **The edges $\Gamma_s, \Gamma_t, \ell_1, \ldots, \ell_6$ define $\delta(P)$.** If we would have no other trains, we could route any thick path within the constructed polygon P.
8. **Intersect the potential holes from $\mathcal{H}_p$ with $\delta(P)$, the part of the potential holes in the interior of P determines the holes $\mathcal{H}$, together with P they constitute Ω, the polygonal domain, see Figure 2(h).**

4.2 Thick Paths with Limited Slope

If we think of our train paths (with temporal buffer around them) as thick paths, we do not just aim for thick paths, but for a path with a limited slope, that is, within a cone limited by the x-axis and a line somewhere between the x- and the y-axis.

We show that we can adapt the waterfall construction from Subsection 3.2, shown in Figure 1, to restrict our paths to the given cone, see Figure 4. Note that, while P in the example is already x-monotone, we need to use waterfalls from the bottom to make the first path feasible.

Let $\mathcal{C}$ be a cone limited by the x-axis and a line somewhere between the x- and the y-axis. We call a thick path $\mathcal{C}$-respecting if its reference path π is $\mathcal{C}$-respecting, i.e., if every line that is orthogonal to a half-line within $\mathcal{C}$ intersects π in at most one point. Accordingly, we call a flow $\mathcal{C}$-respecting if each of its streamlines is $\mathcal{C}$-respecting.

We extend the algorithm from Polishchuk for monotone thick path to find $\mathcal{C}$-respecting paths: First we make B $\mathcal{C}$-respecting: We use a different type of waterfalls than Polishchuk (2007), Arkin et al. (2010) and Mitchell (1990). First, we sweep a horizontal line in the y direction. For every vertex $v \in B$, we connect v to the first point of P hit when going right from v. Additionally, we sweep a line of the maximum slope in $\mathcal{C}$ orthogonally to this direction. For every vertex $v \in B$, we connect v to the first point of P hit when going left from v. After this procedure B is $\mathcal{C}$-respecting.

Then we run the shortest path maxflow algorithm from Mitchell (1990) to fill the free space with flowlines. As the new bottom B is $\mathcal{C}$-respecting, all flowlines are also $\mathcal{C}$-respecting.

When a hole is hit by a wavefront, we make the hole outer-$\mathcal{C}$-respecting. Essentially, this means that we outer-monotonize a hole w.r.t to two directions, see Arkin et al. (1989) for efficient algorithms for monotonization of the holes. If a waterfall during this process hits another hole, this holes is also made outer-$\mathcal{C}$-respecting. We assign the wavefront and the boundaries of the new, outer-$\mathcal{C}$-respecting holes to the new bottom B. We the make the new bottom $\mathcal{C}$-respecting and continue the grass fire. See Figure 4 for an example of this process.

Theorem 4.1. *A representation of the maximum number of $\mathcal{C}$-respecting thick non-crossing paths can be found in $\mathcal{O}(nh + n \log n)$ time.*

4.3 Maximum Number of Trains

Now, we can compute the maximum number of trains to be inserted into a timetable by computing the maximum number of $\mathcal{C}$-respecting thick non-crossing paths in the polygonal domain constructed in Subsection 4.1. See Figure 5 for an example for the construction of the maximum number of $\mathcal{C}$-respecting thick non-crossing paths for the polygonal domain constructed in Figure 2, and the resulting maximum number of train paths.

Let s denote the number of stations in our domain, and t the number of existing trains. We have $\mathcal{O}(ts)$ holes, and $\mathcal{O}(ts)$ vertices (because we have at most four vertices per train in between two stations). This yields (using Theorem 4.1):

Corollary 4.1.1. *A representation of the maximum number of train paths can be found in $\mathcal{O}(t \cdot s \cdot t \cdot s + t \cdot s \cdot \log(ts)) = \mathcal{O}(t^2 s^2)$ time. (Or, if we consider the number of times some train departs from some stations, x, in $O(x^2)$.)*

Paths of Different Thicknesses. Note that if we want to compute paths of different thickness, that is, train paths with different temporal buffers, we can use the same algorithm if the order of the trains is given. If the order of the paths is not given, Kim et al. (2012) showed the problem to be NP-hard.

5 Conclusion

We showed how to convert the time-space diagram of a timetable into a polygonal domain Ω, such that finding a maximum number of $\mathcal{C}$-respecting thick non-crossing paths in Ω gives the maximum number of additional trains that can be inserted into the timetable. To compute this, we extended a known algorithm to compute the maximum number of x-monotone thick non-crossing paths to an algorithm that can compute the maximum number of $\mathcal{C}$-respecting thick non-crossing paths in a polygonal domain. In general, this provides an application of using the geometric representation of a timetable with a geometric algorithm. In the future, it would of course be interesting to study what other geometric concepts could be extended to this geometric representation, that is, which railway problems could be solved using geometric algorithms. Moreover, future work will include the application of our algorithm to real-world scenarios. This includes the more general problem of converting headway based capacity measures on macro level to real blocking times.

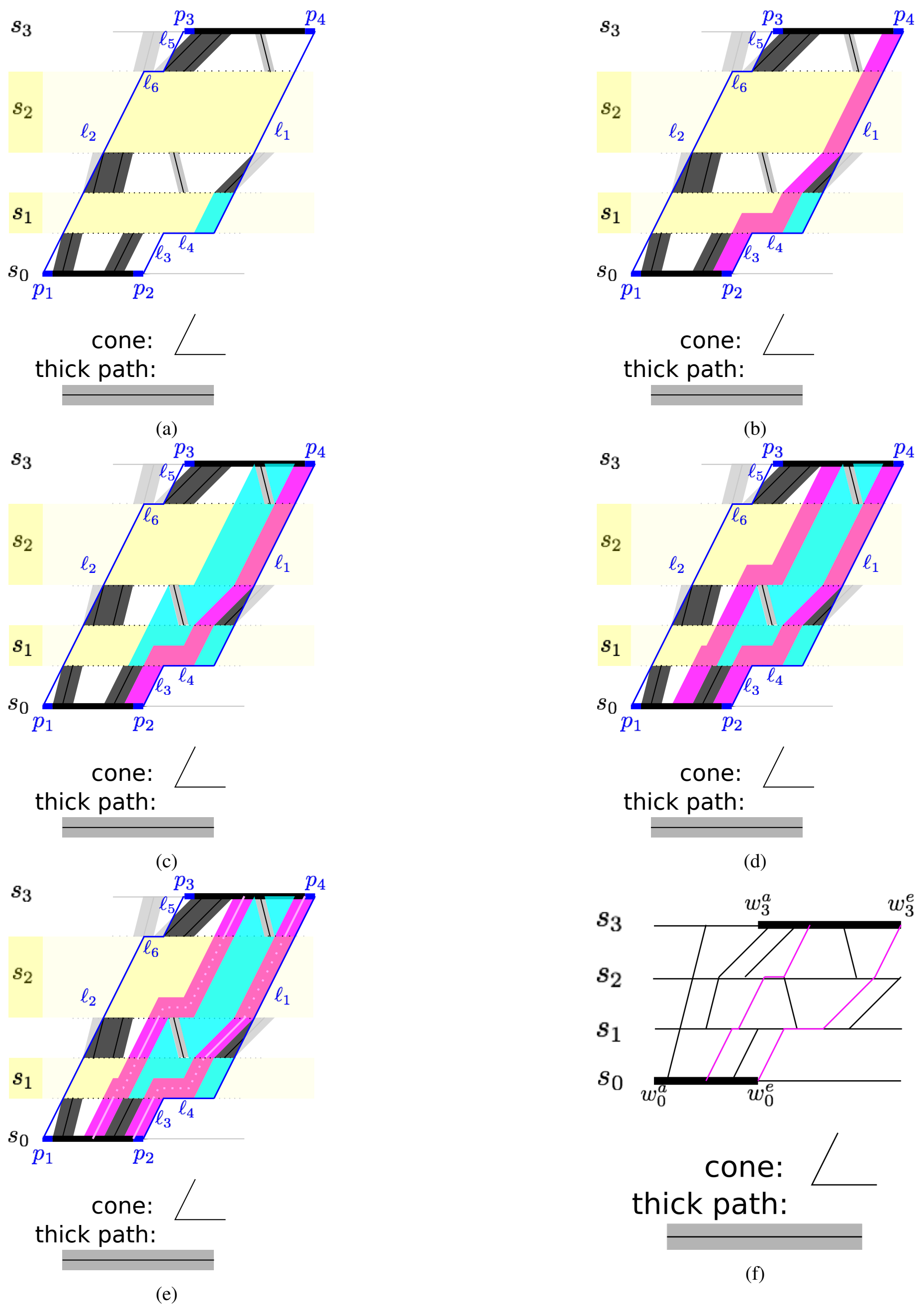

Figure 5: Example for the construction of thick paths with limited slopes for the polygonal domain constructed in Figure 2. Waterfalls are shown in turquoise, the wavefront covered after d time units each in pink. (a)-(d) Waterfalls and wavefronts/thick path construction. (e) The light pink line is the (thin) reference path (the dotted part will not be considered, as the blown-up stations are not part of the original timetable). (f) The obtained train paths inserted in the original timetable Figure 2(a).

Acknowledgements
This research is a result of a collaboration between Linköping University and Trafikverket, and part of the EU H2020 project Shift2Rail/FR8Hub (Grant Agreement No. 777402), and partially funded by Trafikverket (Dnr TRV 2016/75881). The authors are grateful to Magnus Wahlborg and Fredrik Lundström (Trafikverket) for helpful discussions.

References

E. M. Arkin, R. Connelly, and J. S. B. Mitchell, 1989. On monotone paths among obstacles with applications to planning assemblies. In *Proceedings of the Fifth Annual Symposium on Computational Geometry, Saarbrücken, Germany, June 5-7, 1989*, pages 334–343.

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

R. Burdett and E. Kozan, 2009. Techniques for inserting additional trains into existing timetables. *Transportation Research Part B: Methodological*, 43(8):821 – 836.

V. Cacchiani, A. Caprara, and P. Toth, 2010. Scheduling extra freight trains on railway networks. *Transportation Research Part B: Methodological*, 44(2):215 – 231.

D. Z. Chen, O. Daescu, and K. S. Klenk, 2001. On geometric path query problems. *Internat. J. Comput. Geom. Appl.*, 11(6):617–645.

S. D. Eriksson-Bique, V. Polishchuk, and M. Sysikaski, 2014. Optimal geometric flows via dual programs. In *30th Annual Symposium on Computational Geometry, SOCG'14, Kyoto, Japan, June 08 - 11, 2014*, page 100.

H. Flier, 2011. Optimization of railway operations: Algorithms, complexity, and models.

H. Flier, T. Graffagnino, and M. Nunkesser, 2009. Scheduling additional trains on dense corridors. In *8th International Symposium on Experimental Algorithms (SEA 2009), Dortmund, Germany, June 4-6, 2009*, pages 149 –160.

M. Grimm, 2012. The analysis of congested infrastructure and capacity utilisation at Trafikverket. In *8thWIT Transactions on the Built Environment*, 127:359-367.

I. A. Hansen and J. Pachl, 2014. *Railway Timetable & Traffic: Analysis - Modelling - Simulation*. Eurailpress in DVV Media Group, 2nd edition edition.

T. Hu, 1969. *Integer programming and network flows*. Addison-Wesley, Reading, MA.

T. C. Hu, A. B. Kahng, and G. Robins, 1992. Solution of the discrete plateau problem. *Proceedings of the National Academy of Sciences*, 89(19):9235–9236.

L. Ingolotti, F. Barber, P. Tormos, A. Lova, M. A. Salido, and M. Abril, 2004. *An Efficient Method to Schedule New Trains on a Heavily Loaded Railway Network*, pages 164–173. Springer Berlin Heidelberg, Berlin, Heidelberg.

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

A. Landex, A.H. Kaas, B. Schittenhelm, and J. Schneider-Tilli, 2006. Practical use of the UIC 406 capacity leaflet by including timetable tools in the investigations. *WIT Transactions on The Built Environment*, 88:643-652.

C. Liebchen, 2008. The first optimized railway timetable in practice. *Transportation Science*, 42(4):420–435.

T. Lindner, 2011. Applicability of the analytical uic code 406 compression method for evaluating line and station capacity. *Journal of Rail Transport Planning & Management*, 1(1):49–57.

Y.-H. Liu and S. Arimoto, 1995. Finding the shortest path of a disc among polygonal obstacles using a radius-independent graph. *IEEE Trans. Robot. Autom.*, 11(5):682–691.

F. Ljunggren, K. Persson, A. Peterson, and C. Schmidt, 2018. Maximum robust train path for an additional train inserted in an existing railway timetable. In *CASPT 2018*.

L. Lucchini, A. Curchod, R. Rivier, 2001. Transalpine rail network: a capacity assessment model (CAPRES). In *Swiss Transport Research Conference (STRC) 2001*.

J. S. B. Mitchell, 1990. On maximum flows in polyhedral domains. *J. Comput. Syst. Sci.*, 40(1):88–123.

P. Pellegrini, G. Marlière and J. Rodriguez, 2017. RECIFE-SAT: A MILP-based algorithm for the railway saturation problem In *Journal of Rail Transport Planning & Management*, 7(1):19–32.

V. Polishchuk, 2007. Thick non-crossing paths and minimum-cost continuous flows in polygonal domains.

V. Polishchuk and J. S. B. Mitchell, 2007. Thick non-crossing paths and minimum-cost flows in polygonal domains. In *Proceedings of the 23rd ACM Symposium on Computational Geometry, Gyeongju, South Korea, June 6-8, 2007*, pages 56–65.

G. Strang, 1983. Maximal flow through a domain. *Math. Program.*, 26(2):123–143.

Trafikanalys, 2017a. Rail traffic 2016.

Trafikanalys, 2017b. Railway transport 2017 quarter 3.

Trafikverket, 2018. Järnvägens kapacitet 2017 (in Swedish). Report, publication number 2018:050.

UIC, 2004. Capacity. leaflet 406, International Union of Railways, 1st ed.

UIC, 2013. Capacity. leaflet 406, International Union of Railways, 2nd ed.