Scheduling Three Trains is NP-Complete

Christian Scheffer*

Abstract

We consider the Train Scheduling Problem which can be described as follows: Given m trains via their tracks, i.e., curves in the plane, and the trains' lengths, we want to compute a schedule that moves collision-free and with limited speed the trains along their tracks such that the maximal travel time is minimized. We prove that the Train Scheduling Problem is NP-complete already for three trains.

Furthermore, we extend our NP-completeness construction to the Aircraft Scheduling Problem which means from the case of three trains, i.e., subcurves, to the case of three aircrafts, i.e., disks or squares moving on curves.

1 Introduction

In this paper, we consider a parallel motion planning problem, the Train Scheduling Problem which is naturally motivated from practice and defined as follows: Consider k given *trains* each one defined as a pair which is made up of a curve in the plane, called the *track* of the train and a value, called the *length* of the train. We want to compute a schedule moving collision-free and with bounded velocity all trains along their tracks from their tracks' start points to their tracks' end points such that the maximal travel time called the makespan is minimized.

Furthermore, we consider the Aircraft Scheduling Problem which considers aircrafts, i.e., squares or disks, instead of trains, i.e., subcurves of the tracks.

1.1 Our Results

1. We show that the Train Scheduling Problem is NP-complete already for three trains, see Theorem 1.

2. We establish that the Aircraft Scheduling Problem is NP-complete already for three aircrafts, see Theorem 6.

*Department for Computer Science, Westfälische Wcollision-freeilhelms-Universität Münster
christian.scheffer@uni-muenster.de

1.2 Related Work

Multi-robot coordination is one of the most famous and traditional interfaces between robotics and computational geometry. Due to the amazing large landscape of parallel motion planning topics and corresponding results, we refer to surveys as [7, 8, 9] for detailed overviews.

In their pioneering work, Hopcroft, Schwartz, and Sharir [6] show that even the simple Warehouseman's Problem which requires to coordinate a set of rectangles from a start configuration to a target configuration inside a rectangular box is PSPACE-hard.

In a previous paper [4] accepted to the International Symposium on Computational Geometry 2018, we consider the variant of our Aircraft Scheduling Problem such that the aircrafts' movements are not restricted to curves but to the common Euclidean plane. Amongst others, we showed that this 2D variant is NP-complete for arbitrary many vehicles and gave a constant factor approximation for the case that the aircrafts are sufficiently separated. Furthermore, in [2] we demonstrate a practical realization of our approaches.

In a recent paper [13], we show that there is no FPTAS neither for the Train Scheduling Problem nor for the Aircraft Scheduling Problem but do not answer the question whether there is an efficient algorithm for constant many vehicles.

O`Donnell and Lozano-Perez [11] consider the Path Coordination Problem which corresponds to our Aircraft Scheduling Problem and give a $\mathcal{O}(q^2 \log q)$ runtime algorithm for coordinating two robots at which only forward movements are allowed and q is the maximal number of segments on the considered trajectories. Akella and Hutchinson [1] consider Trajectory Coordination Problems in which both the traveling curves and the velocity at which the robots traverse the curves are known. They showed that it is NP-complete to compute departure times for arbitrary many robots such that a minimum-time collision-free robot coordination is achieved.

Reif and Sharir [12] consider Dynamic Movers Problems in which a given polyhedral body B has to be moved collision-free within some 1D, 2D, or 3D space by translations and rotations from a start position to a target position amid a set of obstacles that rotate and move along known trajectories. They provide PSPACE-hardness of the 3D dynamic movement problem if the body B has to hold a velocity bound and NP-hardness if the body's velocity is unbounded. Furthermore, Reif and Sharir [12] consider Asteroid Avoidance Problems as a special variant of Dynamic Movers Problems in which neither the moving body B nor the obstacles may rotate. In particular, Reif and Sharir provide a near-linear time algorithm for the 1-dimensional Asteroid Avoidance Problem in which each of the obstacles is a polyhedron traveling with fixed (possible distinct) translational velocity along a 1-dimensional line. Reif and Sharir provide an efficient algorithm for the two-dimensional Asteroid Avoidance Problem if the number of the obstacles is a constant and for the three-dimensioal Asteroid Avoidance Problem a single exponential time and a polynomial space algorithm for a convex polyhedron B and arbitrary many obstacles.

2 Preliminaries

A train is a pair (H, L_h) where $L_h \in \mathbb{R}_{>0}$ is the *length* of the train and H is the *track* of the train which is defined as a curve $H : [0,1] \to \mathbb{R}^2$. We simultaneously denote by H, the function $H : [0,1] \to \mathbb{R}^2$ and its image $\{p \in \mathbb{R}^2 \mid \text{there is a } t \in [0,1] \text{ with } p = H(t)\}$. The length $|T|$ of a track $T : [0,1] \to \mathbb{R}^2$ in the ambient space is defined as its length w.r.t. the Euclidean norm, i.e., $|T| := \int_0^1 ||T'(t)||_2 \, dt$. A *$k$-fleet* is an k-tuple of trains. Two trains (H, L_h) and (X, L_x) *collide* for the parameters λ_h and λ_x if the subcurves of H and X with midpoints $H(\lambda_h)$ and $X(\lambda_x)$ and lengths L_h and L_x are intersecting each other. A *reparametrization* of a train (H, L_h) is a continuous and piecewise linear function $\alpha : [0, +\infty) \to [0,1]$ such that (1) $\alpha(0) = 0$, (2) there is a minimal value $\lambda \geq 0$ with $\alpha(\mu) = 1$ for all $\mu \geq \lambda$, and (3) the speed of the train is upper-bounded by 1, i.e., for each point in time $t \in [0, +\infty)$, both left and right derivative of $H \circ \alpha$ have Euclidean length at most 1. A schedule for an k-fleet $((T_1, L_1), \ldots, (T_k, L_k))$ is a tuple $(\alpha_1 : [0, M_1] \to [0,1], \ldots, \alpha_k : [0, M_k \to [0,1])$ such that (1) α_i is a reparametrization for the train (T_i, L_i) for all $i \in \{1, \ldots, k\}$ and (2) T_i and T_j do not collide for the parameters $\alpha_i(t)$ and $\alpha_j(t)$ for all $i \neq j \in \{1, \ldots, k\}$ and $t \geq 0$. The *makespan* of the schedule $(\alpha_1 : [0, M_1] \to [0,1], \ldots, \alpha_k : [0, M_k \to [0,1])$ is defined as the maximum $M_{\max}$ of the $M_1, \ldots, M_k$. Wl.o.g., all travel times are equal to $T_{\max}$ by extending α_i with $\alpha_i(t) = \alpha_i(M_i)$ for all $M_i < t < M_{\max}$. Given a k-fleet F, the Train Scheduling Problem asks for a schedule with minimal makespan.

The parameter space P of a k-fleet $((T_1, L_1), \ldots, (T_k, L_k))$ is defined as $P := [0, |T_1|] \times \cdots \times [0, |T_k|]$. The *forbidden* or *black* space $\mathcal{B}$ of P is the union of all parameter points $p = (\lambda_1, \ldots, \lambda_k) \in P$ such that there are two trains T_i and T_j that collide with the parameters λ_i and λ_j. The *allowed* or *white space* $\mathcal{W}$ is defined as $P \setminus \mathcal{B}^\circ$ where $\mathcal{B}^\circ$ denotes the interior of $\mathcal{B}$. Note that the white space $\mathcal{W}$ is closed.

A *path* is a curve $\pi : [0,1] \to P$ and the *length* $\ell(\pi)$ of π is defined as its length w.r.t. the maximum metric, i.e., $\ell(\pi) := \int_0^1 ||\pi'(t)||_\infty \, dt$. An a-b-path *in the free space diagram of* $((T_1, L_1), \ldots, (T_k, L_k))$ is a path $\pi \subset \mathcal{W}$ between a and b. If not other stated, a path in the free space diagram is a path $\pi \subset \mathcal{W}$ connecting the points $(0, \ldots, 0)$ and $(|T_1|, \ldots, |T_k|)$.

3 Scheduling Three Vehicles is NP-complete

In this section, we show that surprisingly the Train Scheduling Problem already for three trains, Train (3) for short, and the Aircraft Scheduling Problem already for three aircrafts, Aircraft (3) for short, are NP-complete. We start with the hardness proof for Train (3).

Theorem 1 Train (3) *is NP-complete.*

We show that Train (3) is NP-complete by proving that it is NP-complete to decide wether there is a schedule with a makespan no larger than M where M is an input value. Given an M, w.l.o.g., we set $s := (0,0,0)$ and $t := (M, M, M)$. It is obvious that in an optimal schedule for each point in time there is a train that travels with speed 1. Thus we obtain:

Observation 2 *For a given fleet F, there is a schedule with makespan M if and only if there is an s-t-path of length M w.r.t. the maximum metric in the free space diagram of F.*

In Section 3.1, we construct an instance $\mathcal{I}$ of a 3D-shortest path problem that implies a polynomial time reduction from 3-SAT to a 3D-shortest path problem that is NP-complete. In Section 3.5, we give a reduction of 3-SAT to Train (3) by providing a construction of an instance for Train (3) whose optimal makespan is equal to the shortest path distance of $\mathcal{I}$.

3.1 An NP-Completeness Construction for 3D-Shortest Paths

We consider the three-dimensional Euclidean space $\mathbb{R}^3$ and refer to the three corresponding axes and coordinates as h-, x-, and y-axis and -coordinates. For $a \in \{h, x, y\}$, the *a-length* of a point set $A \subseteq \mathbb{R}^3$ is defined as $\max_{p,q \in A} |p.a - q.a|$ where $p.a$ and $q.a$ denote the a-coordinates of p and q. Furthermore, the *a-distance* between two connected point sets $A, B \subset \mathbb{R}^3$ is defined as $\min_{p \in A, q \in B} |p.a - q.a|$.

Definition 3 *A* plank *is an axis-aligned cuboid $R \subset \mathbb{R}^3$ whose h-, x- or y-length is long enough to be assumed infinity. The* width *and* height *of a plank are the maximum and minimum of the lengths of R in the remaining two axes directions. A plank R is*

- horizontal *if the h- and x-lengths of R are the height of R and inifinity*
- vertical *if the h- and y-lengths of R are the height of R and infinity, and*
- perpendicular *if the y- and h-lengths of R are the height of R and infinity.*

The orientation *of R is horizontal, vertical, or perpendicular.*

Next, we define the shortest path problem to which we reduce 3-SAT.

Definition 4 *An instance $\mathcal{I} =: (s, t, L, \xi, \mathcal{R})$ of* 3DPLANKS *asks if there is a shortest path of length $L \in \mathbb{R}_{\geq 0}$ w.r.t. the maximum metric between the points $s, t \in \mathbb{R}^3$ and among the set $\mathcal{R}$ of horizontal, vertical, or perpendicular planks that have all a height of ξ.*

For the polynomial-time reduction of 3-SAT to 3DPLANKS, we apply the *path encoding technique* as already used for hardness results of other 3D-shortest path problems [3, 10]. However, in the context of our problem setting we need to ensure important new aspects, see Properties **(P1)**-**(P8)**, because our construction needs to be realisable by the free space diagram of three trains.

In the remainder of this section we show that 3DPLANKS is NP-complete. First we prove that 3DPLANKS is in NP. After that we give the construction of $\mathcal{I}$ and its analysis.

The piecewise linear environment implies that a shortest path is piecewise linear. Thus, the length of a given path can be calculated within polynomial time w.r.t. the complexity of the environment. Hence, we obtain that 3DPLANKS is in NP.

3.2 Outline of the Construction

We consider shortest paths between two points s and t at which st induces a line that has a slope close to 1 w.r.t. x-, y-, and h-coordinate, see Figures 1.

We construct an instance with exponential many topological different shortest path classes representing all possible variable assignments for a given 3-SAT formula F. Thus, we first construct a sequence of n *path splitter gadgets*, see the green gadget in Figure 1, at which each path splitter gadget doubles the number of incoming shortest path classes.

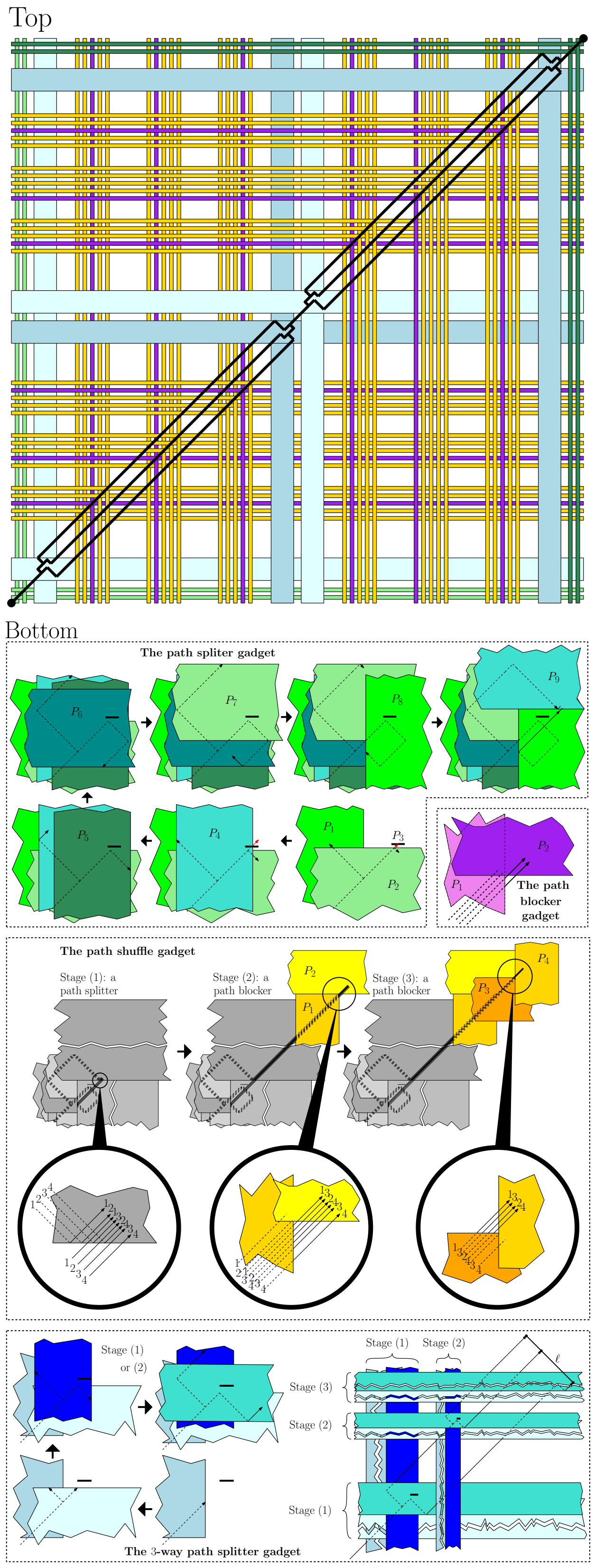

Figure 1: (Top) Construction of $\mathcal{I}$ made up of (3-way) splitter (blue and green), blocker (lila), and shuffle gadgets (orange). (Bottom) Detailed illustration of the used gadgets.

Next it follows a sequence of m clause filters. A clause filter realizes a clause $C_i = (\ell_{i1} \vee \ell_{i2} \vee \ell_{i3})$ and is made up of three parallel literal filters at which parallel means that each literal filter is passed by an individual tube containing 2^n shortest path classes. In order to produce these tubes, a clause filter starts with a 3-*way path splitter gadget*, see the blue gadget in Figure 1, and ends with an *inverted* 3-way path splitter gadget which merges three input tubes of 2^n shortest path classes into one tube of 2^n shortest path class.

Inside a clause filter C_i, each literal filter represents a literal ℓ_{ij} and is made up of a sequence of n *path shuffle gadgets* (see the orange gadgets in Figure 1) which is interrupted by one *path blocker gadget* (see the lila gadget in Figure 1). The path blocker gadget blocks all shortest path classes whose represented bit assignment for $b_1, \dots, b_n$ contradicts ℓ_{ij}. In particular, the shortest path classes inside each literal filter lie inside a thin diagonal tube. The prefixed sequence of path shuffle gadgets ensures that all shortest path classes corresponding to bit assignments that contradict ℓ_{ij} lie either on the top left side of the tube or on the bottom right side of the tube. Correspondingly, the path blocker gadgets increases the length of all these shortest path classes to be blocked, i.e., blocks them from being a shortest path of length L between s and t. Finally, the postposed sequence of path shuffle gadget rebuilds the configuration of the shortest path classes inside the tube.

Finally, the bundle of all remaining shortest path classes are merged by a sequence of n inverted path splitting gadgets. By the above discussion it follows that F is satisfiable if and only if there is a shortest path of length L between s and t which we call property **(P1)**.

The first sequence of n path splitter gadgets generates 2^n shortest path classes lying inside a tube of width $\varepsilon \ll 1$ which is maintained for all three copies inside each clause filter.

Each gadget is made up of $\mathcal{O}(1)$ planks, see Figure 1 for an overview and the following section for more details. All in all we have $2n + m(2 + n + 1)$ path gadgets which implies that $\mathcal{I}$ has $\mathcal{O}(mn)$ planks which we call property **(P2)**.

3.3 Detailed Construction of the Path Gadgets

In the following, we discuss the approaches of path splitter gadgets, path blocker gadgets, path shuffle gadgets, and 3-way path splitter gadgets separately, see Figure 1. The inverted versions of the path splitter and the 3-way path splitter gadget are constructed in inverted order.

The input to the path splitter gadget is a thin bundle of shortest path classes, see the green gadget in Figure 1. The produced output is a bundle containing two copies of the input bundle. A perpendicular plank blocks paths from being a shortest path by enforcing the "unwanted" paths to take a detour around the perpendicular plank, see the black bar and the red arrow in the green gadget of Figure 1.

The path blocker gadget blocks either an upper or lower part of the input bundle of shortest paths from being an overall shortest path.

The path shuffle gadget realizes a perfect shuffle to all input shortest path classes and is made up of three stages, see the orange gadget in Figure 1: A path splitter gadget which is colored in gray and two path blocker gadgets colored in yellow, gold, and orange.

The 3-way path splitter gadget produces three instances π_1, π_2, and π_3 of the input shortest path classes for a clause filter representing a clause $C_i = (\ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3})$, see the blue gadget of Figure 1. Each instance π_1, π_2, and π_3 represents one of the three literals $\ell_{i,1}$, $\ell_{i,2}$, and $\ell_{i,3}$ which are logically linked by an "or". Thus, a plank in the clause filter corresponding to C_i is only allowed to have an influence to either the shortest paths in π_1, π_2, or π_3. In order to ensure that, we construct the 3-way path splitter gadget such that the distance between two points from two different bundles of π_1, π_2, and π_3 on a diagonal line ℓ is a constant times larger than the widths of the planks used in the clause filter of C_i, see Figure 1. In particular, the 3-way path splitter gadget is made up of three stages: (1) Four planks, splitting the input shortest path class into two classes, (2) four planks, splitting the upper class of Stage (1) into two shortest path classes π_1 and π_2, and (3) two planks extending the length of the second shortest path class π_3 of Stage (1) about a distance equal to the length extension caused by Stage (2) for π_1 and π_2.

In the following section, we prove that the remaining properties **(P3)**-**(P8)** of Theorem 5 are fulfilled by $\mathcal{I}$.

Theorem 5 3DPLANKS *is NP-complete. In particular, for each 3-SAT formula Φ with n variables and m clauses, there is an instance $\mathcal{I} = \mathcal{I}(\Phi) = (s, t, L, \xi, \mathcal{R})$ of* 3DPLANKS *(see Figure 1) such that*

- ***(P1)**: the shortest s-t-path has a length of L if and only if Φ is satisfiable,*
- ***(P2)**: there are $\mathcal{O}(mn)$ planks,*
- ***(P3)**: all planks have the same height ξ,*
- ***(P4)**: the minimal width of a plank is 1,*
- ***(P5)**: the minimal h-distance of two planks that are not perpendicular is $\Omega(1)$,*
- ***(P6)**: all planks have a width of $\mathcal{O}(mn)$,*
- ***(P7)**: the maximal x-distance between two vertical planks of the same path gadget is $\mathcal{O}(mn)$, and*
- ***(P8)**: the maximal y-distance between two consecutive horizontal planks of the same path gadget is $\mathcal{O}(mn)$.*

3.4 Properties (P3)-(P8)

Property **(P3)** is trivially ensured by explicitly using planks of a common height. Furthermore, w.l.o.g. we assume that the minimal width of planks used in $\mathcal{I}$ is 1 which is property **(P4)**. Otherwise, we scale the whole construction of $\mathcal{I}$ which maintains that all planks have the same height.

For each path gadget, we ensure that the h-distance between two planks that are not perpendicular is $\Omega(1)$. As the input and output path bundles of all path gadgets are diagonal, we can construct $\mathcal{I}$ such that the length of the (shortest) subpath between two consecutive path gadgets is in $\Theta(mn)$. Analogously, we ensure that the shortest path distance between two stages of the same path shuffle or 3-way path splitter gadget is $\Theta(mn)$. Thus we can ensure in our overall construction that the h-distance between any pair of planks that are not perpendicular is at least $\Theta(1)$ which is property **(P5)**.

In our reduction from 3-SAT to 3Dplanks, we apply that some planks are only passed at one side. We guarantee that by choosing the widths of these planks "sufficiently large" (see below for details) such that passing the plank at a forbidden side would cause a detour which prevents the path from being shortest. In the following, we discuss the details of that approach for each type of path gadgets separately.

- The path splitter gadget is constructed such that doubling the input shortest path classes causes a detour of constant length. In order to enforce that a shortest path passes through all $2n$ path splitter gadgets despite a detour of length $\Theta(n)$, we choose the widths of the planks P_1, P_2, P_8, and P_9 of all $2n$ path splitter gadgets as $\Theta(n)$. Furthermore, we choose the widths of the planks P_4, P_5, P_6, and P_7 as $\Theta(1)$ to ensure that a shortest path passes the planks P_4, P_5, P_6, and P_7 on the required sides of the planks, as illustrated in Figure 1.

- A path blocker gadget simply needs to ensure that shortest path classes that represent variable assignments that are forbidden by the represented literal are blocked from being an overall shortest path. Thus, it suffices to choose the width of all planks of all path blocker gadgets as $\Theta(1)$.

- Each path splitter of a path shuffle gadget causes a detour of constant lengths. Inside each clause filter, a shortest path passes through n path shuffle gadgets resulting in summed detour of $\mathcal{O}(n)$. In order to enforce that a shortest path inside each clause filter passes through all n path shuffle gadgets of a literal filter, we choose the widths of the first, the second, and the last two planks of the path splitter part of the path shuffle gadget as $\Theta(n)$. The widths of the remaining planks are chosen equal to the widths of the corresponding planks in the path splitter and path blocker gadgets.

- The 3-way path splitter gadget ensures that the distance between two points from different output bundles is $\Theta(n)$. This results in a detour of length $\Theta(n)$ caused by each 3-way path splitter gadget. In order to ensure that a shortest path passes each plank of a clause filter only at the intended side we choose the width of each one-sided passed plank larger than the entire detour length caused by all m clause filters, i.e., as $\Theta(mn)$.

From the above discussion it follows that the widths of all planks used in our overall construction of $\mathcal{I}$ are upper-bounded by $\mathcal{O}(mn)$ which is property **(P6)**.

Let P_1 and P_2 be two vertical planks of the same path gadget such that there is not another vertical plank lying between P_1 and P_2 w.r.t. the h-axis. We distinguish wether P_1 and P_2 belong to a path splitter gadget or not: If P_1 and P_2 belong to a path splitter gadget, our construction of $\mathcal{I}$ ensures that the x-distance between P_1 and P_2 is 0. As a path splitter gadget is made up of $\mathcal{O}(1)$ planks with widths no larger than $\mathcal{O}(1)$ we obtain that the x-distance between P_1 and P_2 is upper-bounded by $\mathcal{O}(1)$. If P_1 and P_2 belong to path shuffle or a 3-way path splitter gadget, we combine that the path distances between different stages of the path gadget is upper-bounded by $\mathcal{O}(mn)$, that the planks have a width of $\mathcal{O}(mn)$, that each stage is made up of $\mathcal{O}(1)$ planks, and that the path shuffle and the 3-way path splitter gadget are made up of three stages. Thus, we obtain that the x-distance between P_1 and P_2 is upper-bounded by $\mathcal{O}(mn)$. In both cases, we obtain that the x-distance between P_1 and P_2 is upper-bounded by $\mathcal{O}(mn)$ which is property **(P7)**.

A symmetric argument implies that the y-distance between two consecutive horizontal planks belonging to the same path gadget is in $\mathcal{O}(mn)$ which is property **(P8)** concluding the proof of Theorem 5.

3.5 Reduction of 3-SAT to Train (3)

We construct a 3-fleet with optimal makespan equal to the shortest path distance of the instance $\mathcal{I}$ constructed in Section 3.1. A triple $(\lambda_h, \lambda_x, \lambda_y)$ of parameters for the three trains of a 3-fleet F is forbidden if at least two trains collide with their parameters independent from the parameter of the third train. This means the forbidden space $\mathcal{B}$ of F is the union of a set of axis-aligned planks at which each single plank corresponds to an intersection point of two curves, see Figure 2(a)+(b).

The lengths of the planks in the axes directions corresponding to the colliding trains are equal to the lengths of the colliding trains. Furthermore, the plank extends

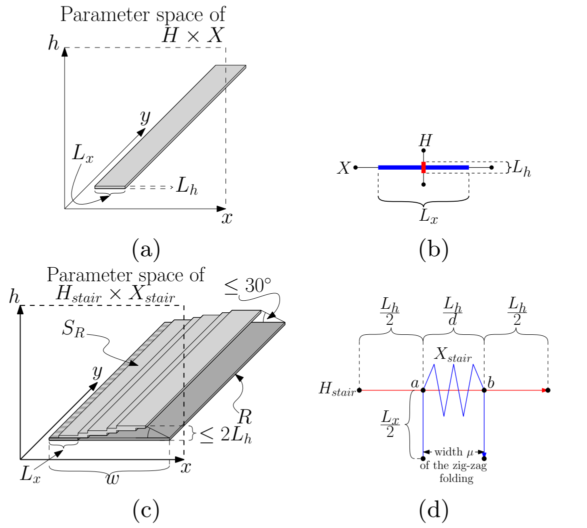

Figure 2: (a) A vertical plank as part caused by an intersection point of the curves H and X that are illustrated in (b). The length of the plank in y-axis direction is infinite because a collision of the trains on H and X is independent from the position of the train on Y. (c) Replacing a vertical plank R by a vertical stairway S_R, and (d) the curves H_{stair} and X_{stair}.

in parallel to the axis corresponding to the third train through the whole parameter space of H, X, and Y. Thus, we occasionally say that a plank has a length of infinity (w.r.t. the axis corresponding to the train which is not necessarily involved in the collision).

The forbidden space of F is piecewise linear impying that a shortest path π' inside the free space diagram is piecewise linear, i.e., π' can be represented by a polynomial sequence of edges it flips over. This implies, that the length of π' can be determined in polynomial time. Thus, Observation 2 implies that TRAIN (3) is in NP.

In order to prove that TRAIN (3) is NP-hard we consider an arbitrary 3-SAT formula Φ and the corresponding instance $\mathcal{I} := \mathcal{I}(\Phi) := (s, t, L, \xi, \mathcal{R})$ of 3DPLANKS constructed in Section 3.1. We construct a 3-fleet $F := ((H, L_h), (X, L_x), (Y, L_y))$ and a value L' such that verifying if there is an optimal schedule for F with maximal travel time no larger than L' is equivalent to verifying if there is a shortest path with length no larger than L for $\mathcal{I}$. As the construction of $\mathcal{I}$ induces a polynomial time reduction from 3-SAT to 3DPLANKS, it follows that the construction of F induces a polynomial time reduction from 3-SAT to TRAIN (3).

Straight forwardly substituting the planks of $\mathcal{I}$ by planks that are caused by intersection points of the trains (H, L_h), (X, L_x), and (Y, L_y) is not possible because in the construction of $\mathcal{I}$ we use different horizontal planks that have different widths while all horizontal planks in the forbidden space of (H, L_h), (X, L_x), and (Y, L_y) have a width of L_y. Furthermore, there is the same issue with vertical and perpendicular planks. Thus, we replace each single plank R of $\mathcal{I}$ by a so called *stairway* S_R and give an approach how to construct three trains whose forbidden space modulo translations is equal to S_R, see Figures 2(c)+(d) and Figure 6 for illustrations and Section A for a definition of the used stairways.

By assembling all resulting curves corresponding to stairways, we obtain the trains (H, L_H), (X, L_x), and (Y, L_y), see Figure 5, concluding the proof of Theorem 1.

3.6 Reduction of 3-SAT to AIRCRAFT (3)

We remark that scheduling three aircrafts, i.e., squares or disks instead of trains, i.e., subcurves is also NP-complete. Generally speaking, we use the 3D-shortest path instance $\mathcal{I}$ and substitute planks of $\mathcal{I}$ by *(curved) wedges*, see Figure 3 and Figures 7, 8, 9, and 10.

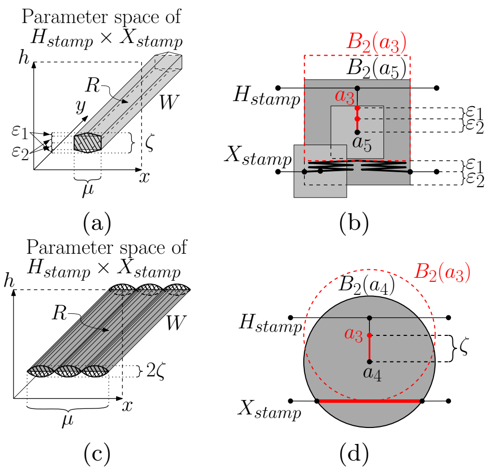

Figure 3: (a)+(c): In the case of square-shaped and disk-shaped aircrafts, we substitute planks by wedges and curved wedges. (b)+(d): In a fixed configuration, two aircrafts collide if and only if the centre of the first aircraft lies inside the square $B_2(c)$ with radius 2 and centre c in the midpoint of the second aircraft.

Theorem 6 AIRCRAFT (3) *is NP-complete for disk-shaped and square-shaped aircrafts.*

4 Conclusion

We presented hardness results for parallel motion planning problems considering objects to moved collision-free along their tracks. Our hardness constructions involve curves that are quite dense in the following manner: Driemel et al. [5] say that a curve is *c-packed* for a $c \geq 0$ if the total intersection of the curve with any ball of radius $r > 0$ is no larger than cr. The curves constructed in our hardness proof are not c-packed for any constant c. Thus, we ask the question whether there is an efficient algorithm for scheduling three trains or aircrafts along c-packed curves.

References

[1] S. Akella and S. Hutchinson. Coordinating the motions of multiple robots with specified trajectories. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002, May 11-15, 2002, Washington, DC, USA*, pages 624–631, 2002.

[2] A. T. Becker, S. P. Fekete, P. Keldenich, L. Lin, and C. Scheffer. Coordinated motion planning: The video. In C. Tóth and B. Speckmann, editors, *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 74:1–74:6, 2018. Video available at https://youtu.be/0OrG72sX5gk.

[3] J. F. Canny and J. H. Reif. New lower bound techniques for robot motion planning problems. In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 49–60, 1987.

[4] E. D. Demaine, S. P. Fekete, P. Keldenich, H. Meijer, and C. Scheffer. Coordinated Motion Planning: Coordinating a Swarm of Labeled Robots with Bounded Stretch. In C. Tóth and B. Speckmann, editors, *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 29:1–29:15, 2018.

[5] A. Driemel, S. Har-Peled, and C. Wenk. Approximating the fréchet distance for realistic curves in near linear time. *Discret. Comput. Geom.*, 48(1):94–127, 2012.

[6] J. E. Hopcroft, J. T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the warehouseman's problem. *The International Journal of Robotics Research*, 3(4):76–88, 1984.

[7] L. E. Kavraki and S. M. LaValle. Motion planning. In *Springer Handbook of Robotics*, pages 139–162. 2016.

[8] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[9] S. M. LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[10] J. S. B. Mitchell and M. Sharir. New results on shortest paths in three dimensions. In *Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004*, pages 124–133, 2004.

[11] P. A. O'Donnell and T. Lozano-Pérez. Deadlock-free and collision-free coordination of two robot manipulators. In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation, Scottsdale, Arizona, USA, May 14-19, 1989*, pages 484–489, 1989.

[12] J. H. Reif and M. Sharir. Motion planning in the presence of moving obstacles. *J. ACM*, 41(4):764–790, 1994.

[13] C. Scheffer. Train scheduling: Hardness and algorithms. In *Proceedings of the 14th International Conference on Algorithms and Computation (WALCOM)*, pages 342–347, 2020.

A Details on the Hardness of Scheduling Three Trains

A.1 A Schematic Illustration of the Hardness Construction for Three Trains

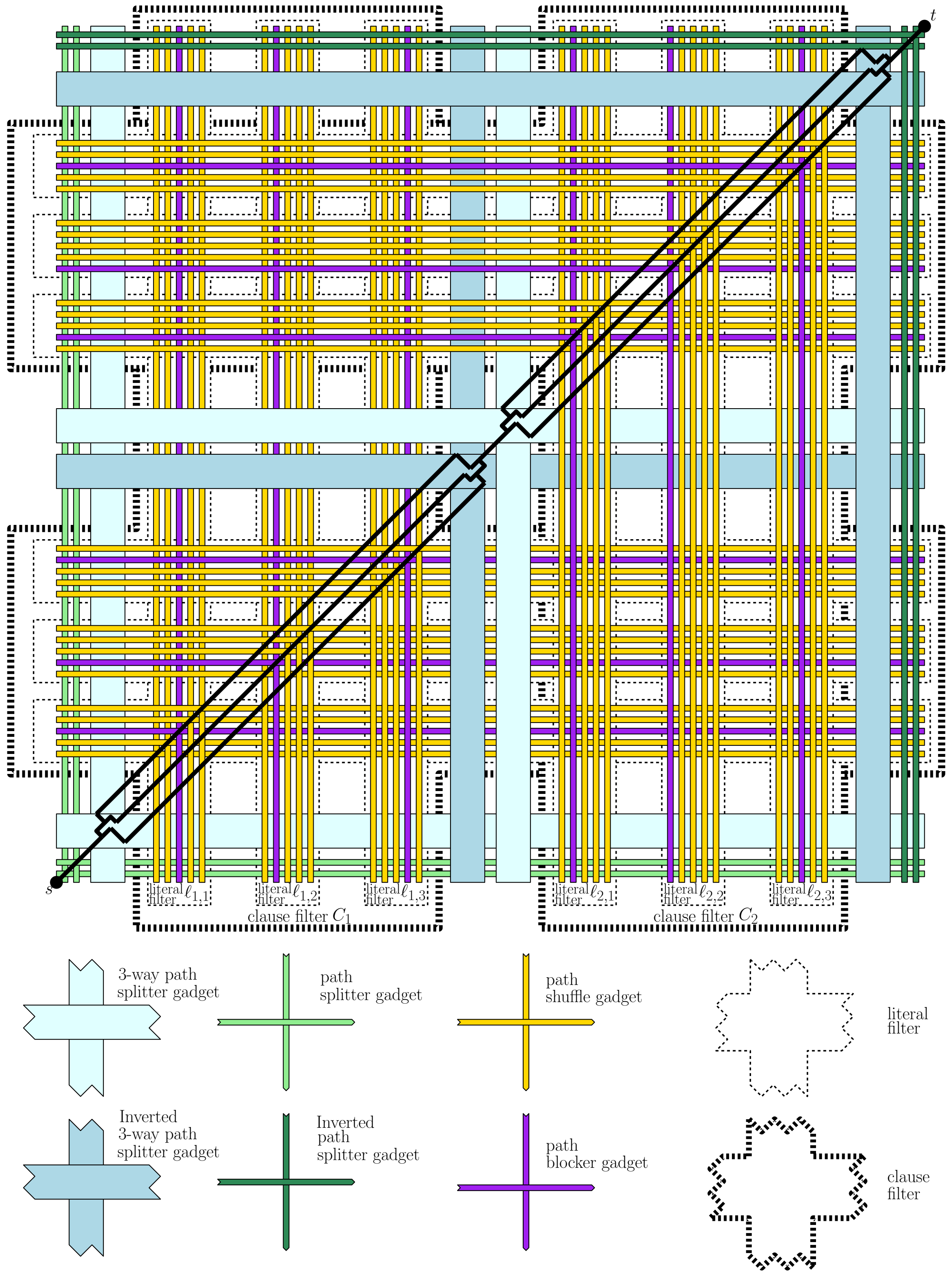
t
s
literal filter $\ell_{1,1}$
literal filter $\ell_{1,2}$
literal filter $\ell_{1,3}$
clause filter C_1
literal filter $\ell_{2,1}$
literal filter $\ell_{2,2}$
literal filter $\ell_{2,3}$
clause filter C_2
3-way path splitter gadget
path splitter gadget
path shuffle gadget
literal filter
Inverted 3-way path splitter gadget
Inverted path splitter gadget
path blocker gadget
clause filter

Figure 4: The whole NP-hardness construction corresponding to a given 3-SAT formular $\Phi = (\ell_{1,1} \vee \ell_{1,2} \vee \ell_{1,3}) \wedge (\ell_{2,1} \vee \ell_{2,2} \vee \ell_{2,3})$.

A.2 Illustrations of the Schedule Gadgets and the Entire Construction

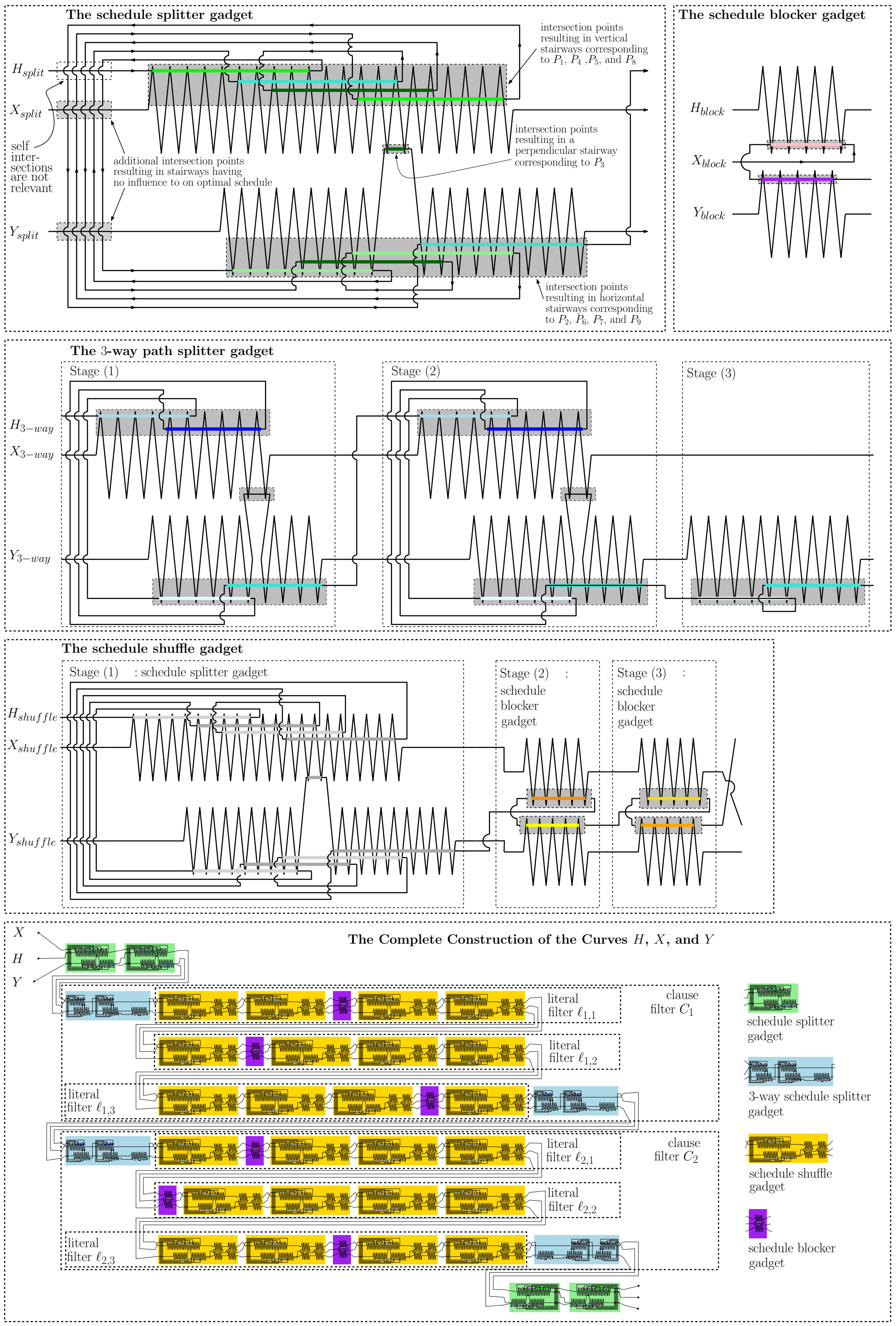
The schedule splitter gadget
H_{split}
X_{split}
Y_{split}
self intersections are not relevant
additional intersection points resulting in stairways having no influence to on optimal schedule
intersection points resulting in vertical stairways corresponding to P_1, P_4, P_5, and P_8
intersection points resulting in a perpendicular stairway corresponding to P_3
intersection points resulting in horizontal stairways corresponding to P_2, P_6, P_7, and P_9
The schedule blocker gadget
H_{block}
X_{block}
Y_{block}
The 3-way path splitter gadget
Stage (1)
Stage (2)
Stage (3)
H_{3-way}
X_{3-way}
Y_{3-way}
The schedule shuffle gadget
Stage (1) : schedule splitter gadget
Stage (2) : schedule blocker gadget
Stage (3) : schedule blocker gadget
$H_{shuffle}$
$X_{shuffle}$
$Y_{shuffle}$
The Complete Construction of the Curves H, X, and Y
X
H
Y
literal filter $\ell_{1,1}$
clause filter C_1
literal filter $\ell_{1,2}$
literal filter $\ell_{1,3}$
literal filter $\ell_{2,1}$
clause filter C_2
literal filter $\ell_{2,2}$
literal filter $\ell_{2,3}$
schedule splitter gadget
3-way schedule splitter gadget
schedule shuffle gadget
schedule blocker gadget

Figure 5: The approach for constructing the curves H, X, and Y for the 3-SAT formular $\Phi = (\ell_{1,1} \vee \ell_{1,2} \vee \ell_{1,3}) \wedge (\ell_{2,1} \vee \ell_{2,2} \vee \ell_{2,3})$. Each path gadget results in a corresponding schedule gadget. The order in which the schedule gadgets are passed is the same in which the corresponding path gadgets are passed, see Figure 4.

A.3 Technical Details of the Construction

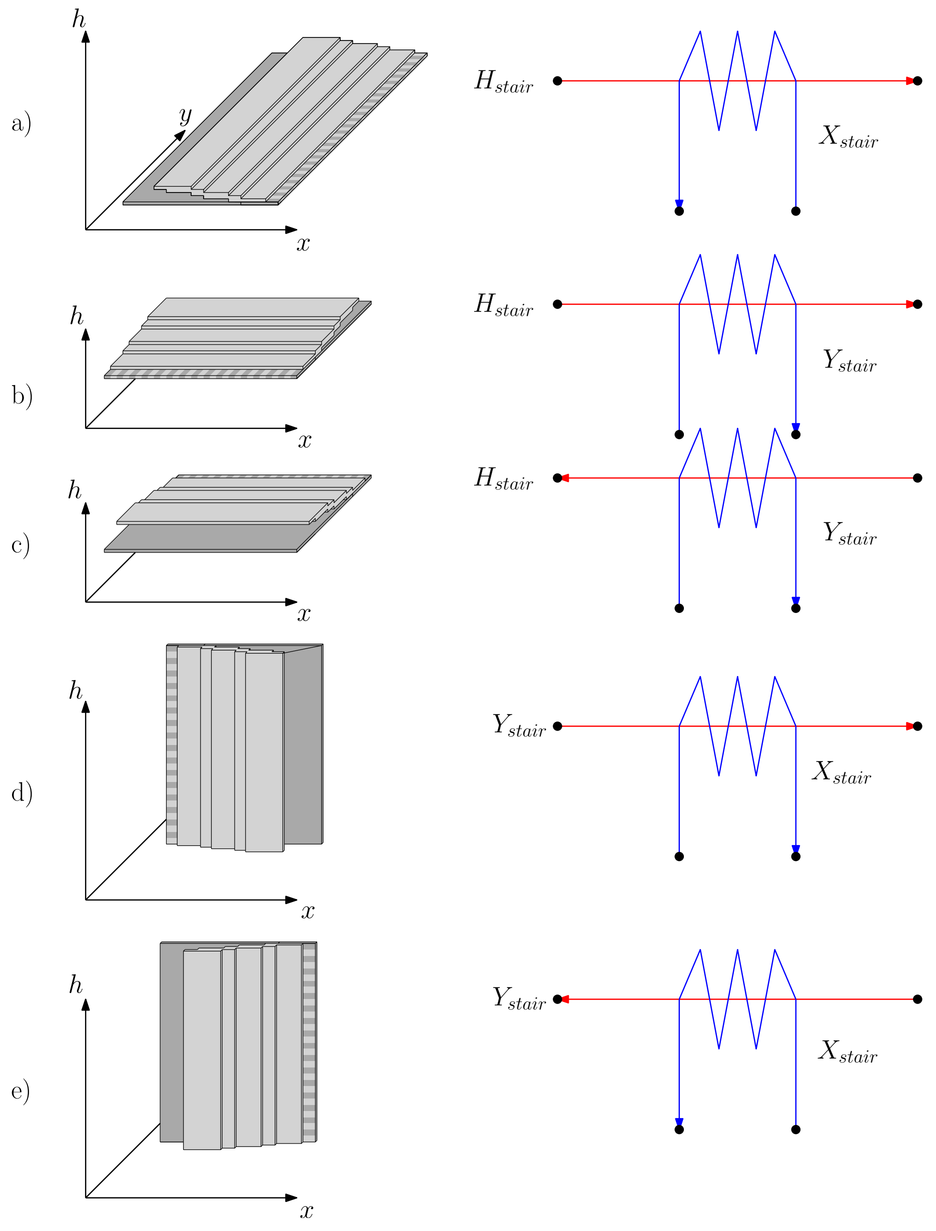

Figure 6: Remaining five types of stairways and the corresponding pairs of trains.

Definition 7 *Let $\ell \in \mathbb{R}_{\geq 0}$.*

- *An* increasing vertical stairway *S corresponding to a vertical plank R with width w and height ξ is the connected union of a sequence of vertical planks with height ξ and which are increasing w.r.t. h- and x-coordinates, as illustrated in Figure 2(c).*
- *A* decreasing *vertical stairway corresponding to vertical plank R with width w and height ξ is constructed analogously to an increasing vertical stairway by mirroring the construction of an increasing vertical stairway w.r.t. x-coordinates, see Figure 2(c).*
- *An* increasing or decreasing horizontal *stairway corresponding to a horizontal plank R with width w and height ξ is constructed analogously to an increasing or decreasing vertical stairway by rotating the whole construction of an* increasing or decreasing vertical stairway *by 90° around the h-axis, see Figures 6(c)+(e).*
- *An* increasing or decreasing perpendicular *stairway corresponding to a perpendicular plank R with width w and height ξ is constructed analogously to vertical stairway by rotating the whole construction of a* increasing or decreasing vertical stairway *by 90° around the x-axis, see Figures 6(g)+(i).*

"Increasing" or "decreasing" is the direction *of S and "horizontal", "vertical", or "perpendicular" is the* orientation *of S.*

By the following lemma, we give an approach how to construct a triple of trains whose forbidden space in their free space diagram is a stairway of required direction and orientation.

Lemma 8 *Let R be a plank with width w and direction d and let $L_h, L_x, L_y \in \mathbb{R}_{\geq 0}$. We can construct three curves H_{stair}, X_{stair}, and Y_{stair} such that the forbidden space of the trains (H_{stair}, L_h), (X_{stair}, L_x), and (Y_{stair}, L_h) is a stairway S with direction d, height no larger than $\frac{3}{2}L_h$, and corresponding to R (except for translations) such that S and R have the same orientation. The curves H_{stair}, X_{stair}, and Y_{stair} have $\mathcal{O}\left(\frac{w}{L}\right)$ segments, where $L = L_x$ if R is vertical, $L = L_y$ if R is horizontal, and $L = L_h$ if R is perpendicular.*

Proof. W.l.o.g. we assume that R is an increasing vertical plank. The cases in which R is horizontal or perpendicular and/or the required direction d of S is decreasing are symmetric.

In Figure 2(d), we illustrate the construction of three trains (H_{stair}, L_h), (X_{stair}, L_x), and (Y_{stair}, L_y). In particular, (Y_{stair}, L_y) is omitted from the illustration because for a vertical stairway we only need intersection points between H_{stair} and X_{stair}. Thus, the following construction works for every curve Y_{stair} that intersects neither H_{stair} nor X_{stair}.

The curve H_{stair} is a segment being separated into three segments by two points $a, b \in H_{\text{stair}}$ at which the middle one is called the *needle* of H_{stair}, see Figure 2(d). The segments' lengths are $\frac{L_h}{2}$, $\frac{L_h}{d}$, and $\frac{L_h}{2}$ for an arbitrarily chosen constant $d \geq 2$ resulting in $L_h + \frac{L_h}{d}$ as the length of H_{stair}. Thus, the height of the resulting stairway will be $L_h + \frac{L_h}{d} \leq \frac{3}{2}L_h$.

The curve X_{stair} is the concatenation of three curves. The two end parts of X_{stair} are segments that are adjacent to a and b and have a length of $\frac{L_x}{2}$. The middle part of X_{stair} is a zig-zag folding starting and ending in a and b such that the middle part of H_{stair} strings the middle part of X_{stair} like a needle.

We choose the lengths of the segments of the middle part of X_{stair} between $\frac{L_x}{c}$ and $\frac{L_x}{2}$ for an arbitrarily chosen constant $c > 2$. This implies that two trains with lengths L_h and L_x collide if the centres of the trains lie on the middle parts of H_{stair} and X_{stair}.

We construct X_{stair} by combining $\Theta(\frac{w}{L_x})$ segments of length $\Theta(L_x)$. Thus the forbidden space of (H_{stair}, L_h) and (X_{stair}, L_x) is a connected stairway which is made up of $\Theta(\frac{w}{L_x})$ vertical planks. This concludes the proof. □

Note that the shapes of the four end parts of curves constructed in the proof of Lemma 8 do not have to be segments. In particular, the only important thing are their lengths.

We use the approach of Lemma 8 to substitute planks from $\mathcal{I}$ by stairways without changing the length of shortest path between s and t.

Lemma 9 *We can construct three trains (H, L_h), (X, L_x), and (Y, L_y) such that substituting the union all planks of $\mathcal{I}(F)$ by the forbidden space of (H, L_h), (X, L_x), and (Y, L_y) does not change the length of a shortest path between* $\mathfrak{s}$ *and* $\mathfrak{t}$. *The curves H, X, and Y are made up of $\mathcal{O}(m^2n^2)$ segments where m and n are the numbers of clauses and variables of F.*

Proof. We choose L_h as the height ξ of the planks from $\mathcal{I}$ and L_x and L_y as the minimal width 1 of the planks of $\mathcal{I}$.

Consider an arbitrarily chosen path gadget of $\mathcal{I}$. In the following we give an approach how to construct a triple of curves H_{gadget}, X_{gadget}, and Y_{gadget} such that substituting the union of the planks of the given gadget by the forbidden space of (H_{gadget}, L_h), (X_{gadget}, L_x), and (Y_{gadget}, L_y) does not change the length of a shortest path in $\mathcal{I}$. By applying this approach to all path gadgets, we get a sequence of triples of curves.

Finally, the concatenations of all these subcurves yields the curves H, X, and Y, see Figure 5, such that a shortest path in the free space diagram of (H, L_h), (X, L_x), and (Y, L_y) has the same length L as a shortest path in $\mathcal{I}$.

First, we give a construction of a tripple $(H_{\text{split}}, X_{\text{split}}, Y_{\text{split}})$ of curves, *schedule splitter gadget*, corresponding to a path splitter gadget. Roughly spoken, for each plank of the path splitter gadget, we apply the approach of Lemma 8 resulting in a stairway for each plank of the path splitte gadget, see Figure 5. This results in three curves H_{split}, X_{split}, and Y_{split}, where the shapes of X_{split} and Y_{split} are formed by long zig-zag foldings such that X_{split} lies above Y_{split}. The curve H_{split} threads sequentially parts of the zig-zag folding of X_{split} from above and parts of the zig-zag folding of Y_{split} from below like a needle. The corresponding needles of H_{split} are highlighted by green colors in Figure 5.

Each needle results in a stairway. The order in which H_{split} threads parts of X_{split} and Y_{split} is the increasing order of the corresponding stairways w.r.t. their h-coordinate. The combination of the Properties **(P4)**-**(P8)** implies that the curves H_{split}, X_{split}, and Y_{split} can be constructed as the concatenation of $\mathcal{O}(mn)$ segments.

The approaches for schedule blocker gadgets, schedule shuffle gadgets, and 3-way schedule splitter gadgets are similar, see Figure 5.

As $\mathcal{I}$ is made up of $\Theta(mn)$ path gadgets, we can construct H, X, and Y made up of $\mathcal{O}(m^2n^2)$ segments. This concludes the proof. □

B Details on the Hardness Construction for Three Aircrafts

B.1 Definition of the Aircraft Scheduling Problem

In this section, we change our focus from scheduling trains to scheduling aircrafts that are roughly spoken represented by disks or squares instead of subcurves. A collision between two aircrafts is defined by two intersecting disks or squares that represent the colliding aircrafts. Note that $3D$ shapes might also represent safety zones around the aircrafts.

An aircraft is a pair (H, L_h) where $L_h \in \mathbb{R}_{>0}$ is the *radius* of the aircraft and H is the *track* of the aircraft which is defined as a curve $H : [0,1] \to \mathbb{R}^2$. We simultaneously denote by H, the function $H : [0,1] \to \mathbb{R}^2$ and its image $\{p \in \mathbb{R}^2 \mid$ there is a $t \in [0,1]$ with $p = H(t)\}$. The length $|T|$ of a track $T : [0,1] \to \mathbb{R}^2$ in the ambient space is defined as its length w.r.t. the Euclidean norm, i.e., $|T| := \int_0^1 ||T'(t)||_2 \, dt$. An *k-fleet* is an k-tuple of aircrafts. Two *disk-shaped* aircrafts (H, L_h) and (X, L_x) *collide* for the parameters λ_h and λ_x if the disks with midpoints $H(\lambda_h)$ and $X(\lambda_h)$ and radii L_h and L_x are intersecting each other. Two *square-shaped* aircrafts (H, L_h) and (X, L_x) *collide* for the parameters λ_h and λ_x if the squares with midpoints $H(\lambda_h)$ and $X(\lambda_h)$ and sidelenghts $2L_h$ and $2L_x$ are intersecting each other. A *reparametrization* of an aircraft (H, L_h) is a continuous and piecewise linear function $\alpha : [0, +\infty) \to [0,1]$ such that (1) $\alpha(0) = 0$, (2) there is a minimal value $\lambda \geq 0$ with $\alpha(\mu) = 1$ for all $\mu \geq \lambda$, and (3) the speed of the aircraft is upper-bounded by 1, i.e., for each point in time $t \in [0, +\infty)$, both left and right derivative of $H \circ \alpha$ have Euclidean length at most 1. A schedule for a tuple of aircraft $((T_1, L_1), \ldots, (T_k, L_k))$ is a tuple $(\alpha_1 : [0, +\infty) \to [0,1], \ldots, \alpha_k : [0, +\infty) \to [0,1])$ such that (1) α_i is a reparametrization for the aircraft (T_i, L_i) for all $i \in \{1, \ldots, k\}$ and (2) the disk-shaped or square shaped aircrafts (T_i, L_i) and (T_j, L_j) do not collide for the parameters $\alpha_i(t)$ and $\alpha_j(t)$ for all $i \neq j \in \{1, \ldots, m\}$ and t. A *timetable* for $((T_1, L_1), \ldots, (T_k, L_k))$ is a tupple $((D_1, A_1), \ldots, (D_k, A_k))$ where D_i and A_i are the required *departure* and *arrival time* of the aircraft (T_i, L_i) such that $0 \leq D_i \leq A_i < +\infty$.

A schedule $S = (\alpha_1, \ldots, \alpha_k)$ for an k-fleet $((T_1, L_1), \ldots, (T_k, L_k))$ *meets* a timetable $T = ((D_1, A_1), \ldots, (D_k, A_k))$ if $T_i(\alpha_i(\lambda)) = T_i(0)$ and $T_i(\alpha_i(\mu)) = T_i(1)$ hold for all $\lambda \in [0, D_i]$, $\mu \geq A_i$, and $i \in \{1, \ldots, k\}$. Furthermore, S *meets approximately* T if $T_i(\alpha_i(\lambda)) = T_i(0)$ and $T_i(\alpha_i(\mu)) = T_i(1)$ hold for all $\lambda \in [0, D_i]$, $\mu \geq cA_i$, and $i \in \{1, \ldots, k\}$.

Given a k-fleet F and a corresponding timetable, the Aircraft Scheduling Problem for disk-shaped aircrafts and square-shaped aircrafts asks if there is a schedule S for F such that S meets the given timetable. A *c-approximation* for Train Scheduling Problem is an algorithm that computes a schedule that meets approximately the required timetable.

The parameter space P of an m-fleet $((T_1, L_1), \ldots, (T_k, L_k))$ of k is defined as $P := \mathbb{R}^k_{\geq 0}$. The *forbidden* or *black* space $\mathcal{B}$ of P is the union of all parameter points $p = (\lambda_1, \ldots, \lambda_k) \in P$ such that there are two aircrafts (T_i, L_i) and (T_j, L_j) that collide with the parameters λ_i and λ_j. The *allowed* or *white space* $\mathcal{W}$ is defined as $P \setminus \mathcal{B}^\circ$ where $\mathcal{B}^\circ$ denotes the interior of $\mathcal{B}$. Note that the white space $\mathcal{W}$ is closed.

B.2 The Schedule Gadgets for Squared Aircrafts

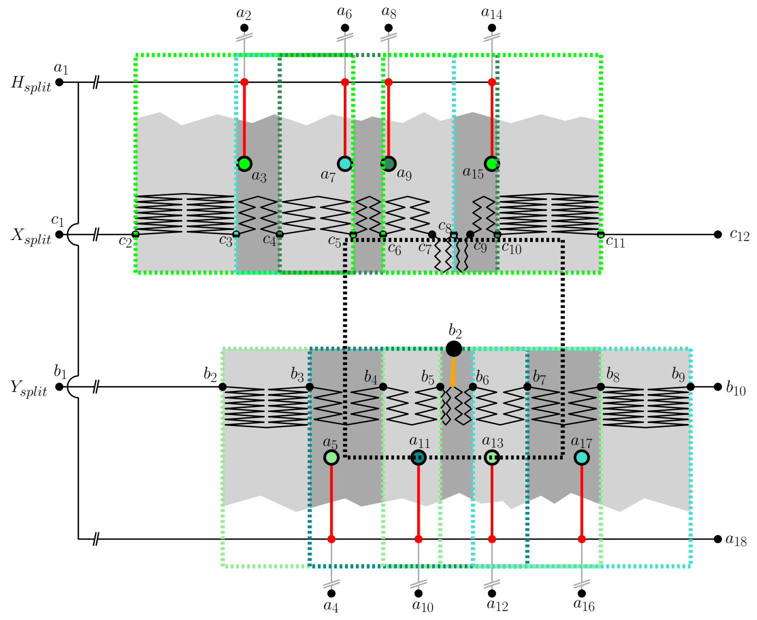

Figure 7: Three curves H_{split}, X_{split}, Y_{split} constructing a schedule splitter gadget by stamping out subcurves which correspond the planks and blocking squares used in the construction of the path splitter gadget. The red subcurves are the stampers, the gray subcurves are the corresponding synchronizers, and the alternating light and dark gray shaded areas are the tubes in which the zig-zag folding of X_{split} and Y_{split} might have different hights.

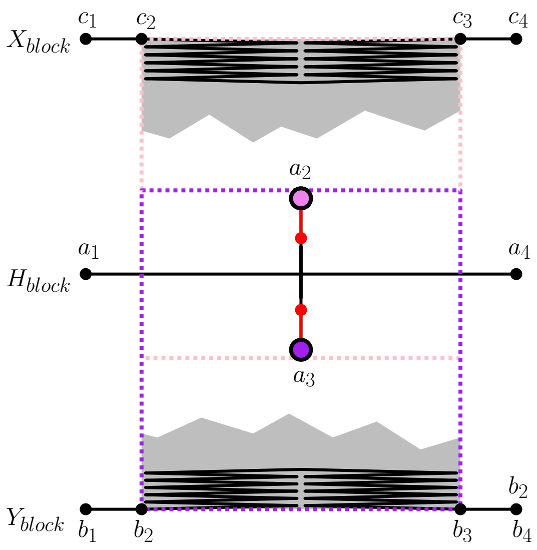

Figure 8: Three curves $H_{\text{block}} \subset H$, $X_{\text{block}} \subset X$, and $Y_{\text{block}} \subset Y$ constructing a schedule blocker gadget by stamping out subcurves which correspond the planks used in the construction of the path blocker gadget.

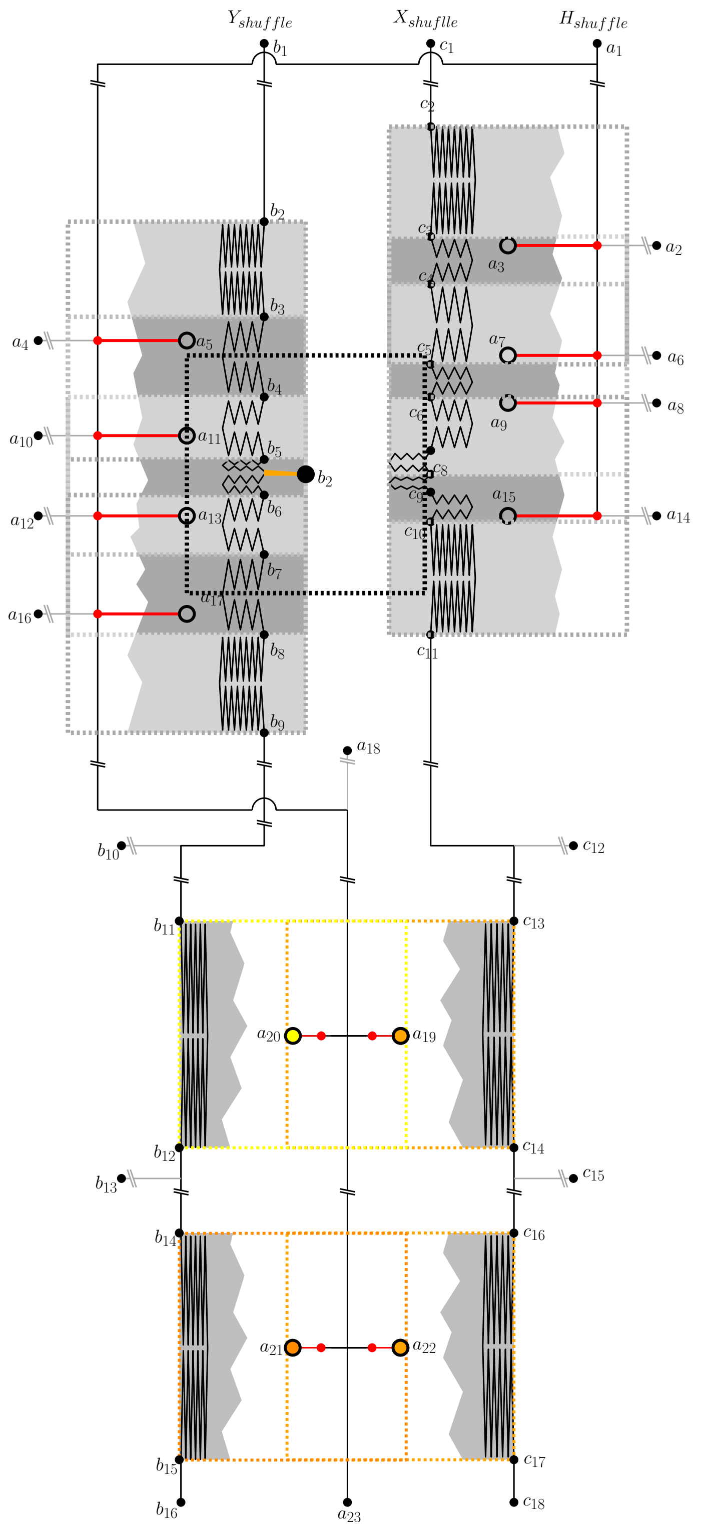

Figure 9: Three curves H_{shuffle}, X_{shuffle}, Y_{shuffle} constructing a schedule shuffle gadget by stamping out subcurves which correspond the planks and blocking squares used in the construction of the path shuffle gadget. The red subcurves are the stampers, the gray subcurves are the corresponding synchronizers, and the alternating light and dark gray shaded areas are the tubes in which the zig-zag folding of X_{shuffle} and Y_{shuffle} might have different hights.

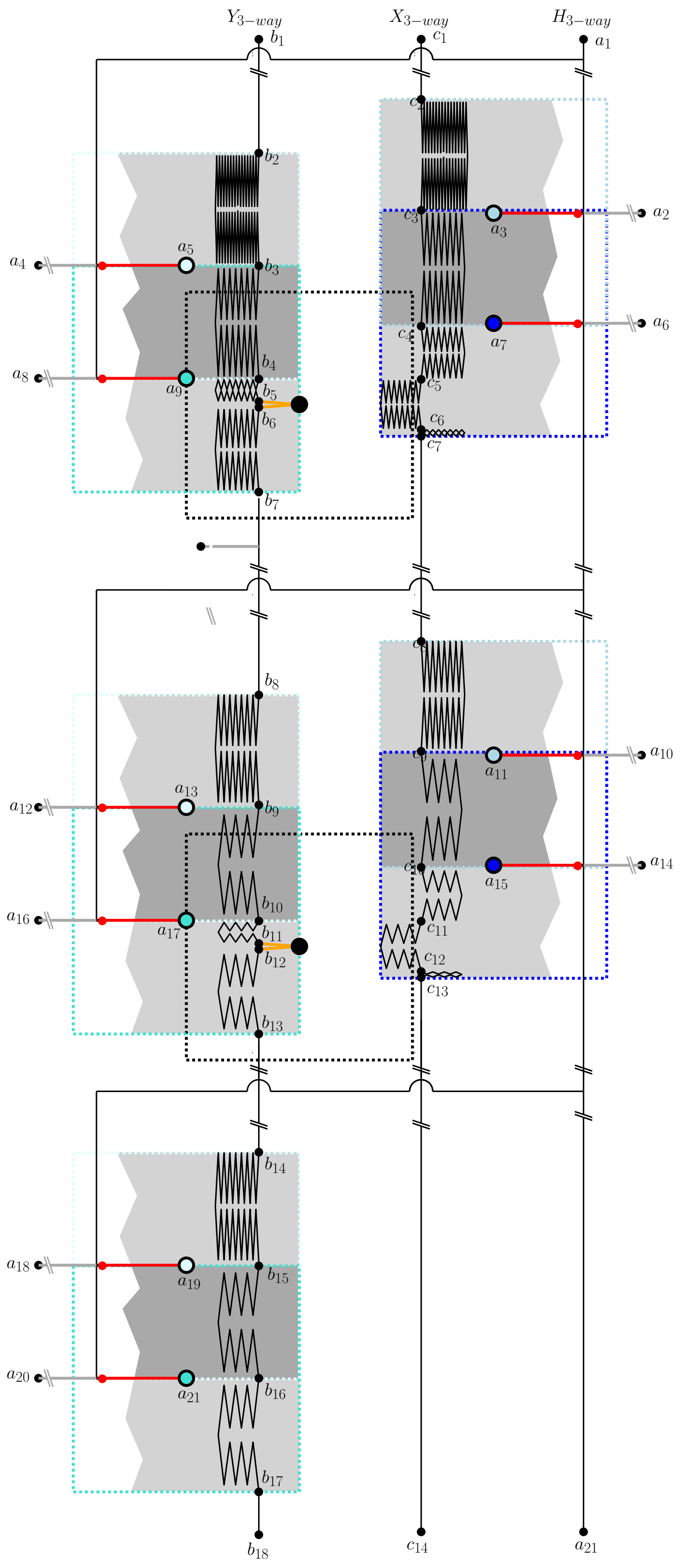

Figure 10: Three curves $H_{3-\text{way}} \subset H$, $X_{3-\text{way}} \subset X$, and $Y_{3-\text{way}} \subset Y$ constructing a 3-way schedule splitter gadget by stamping out subcurves which correspond the planks and blocking squares used in the construction of the 3-way path splitter gadget.

B.3 Chronological Order of the Schedule Splitter and Blocker Gadgets

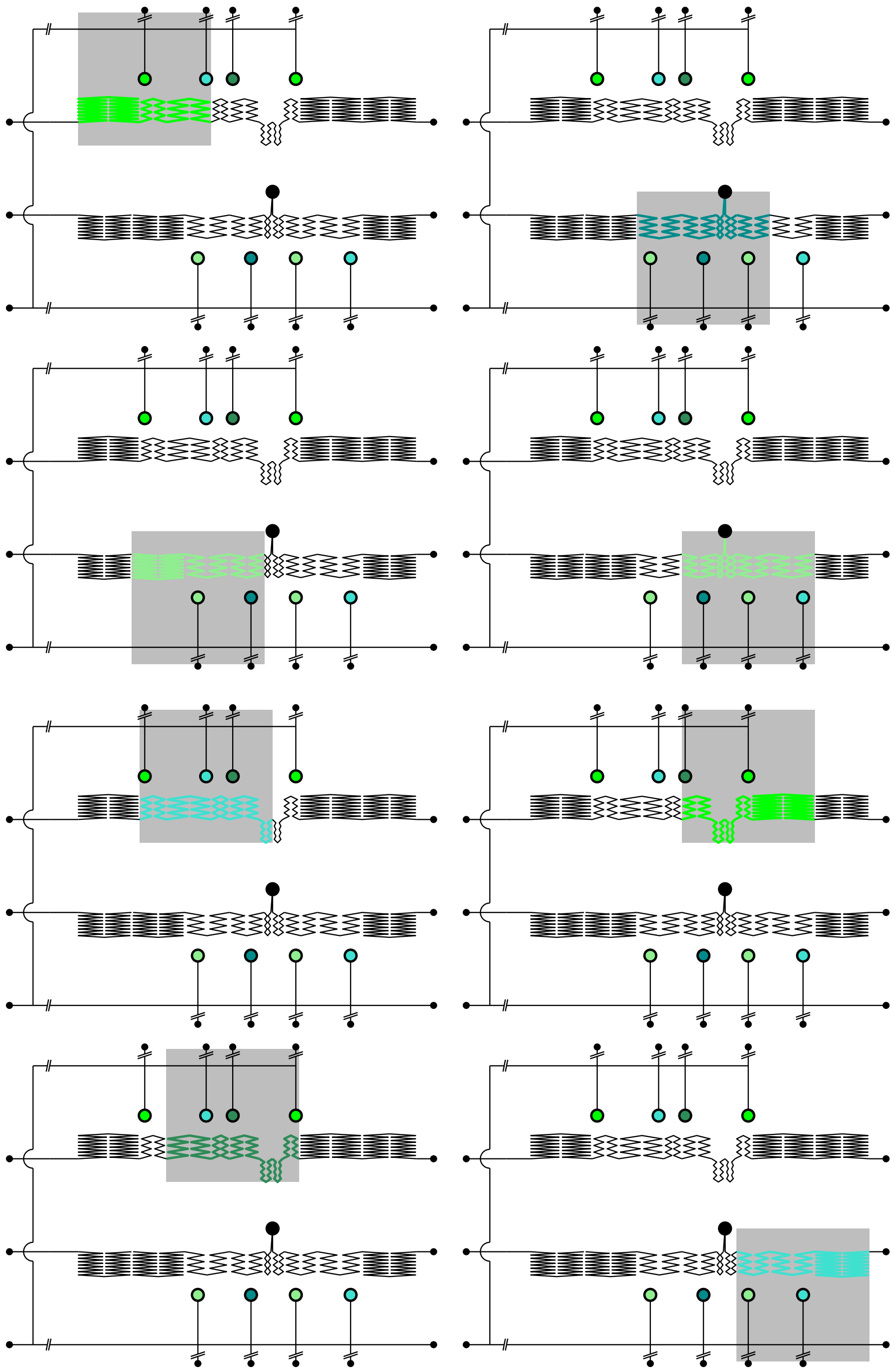

Figure 11: Chronological order in which subcurves from H_{split}, X_{split}, and Y_{split} are stamped out for the construction of a path splitter gadget in the free space diagram of H_{split}, X_{split}, and Y_{split}.

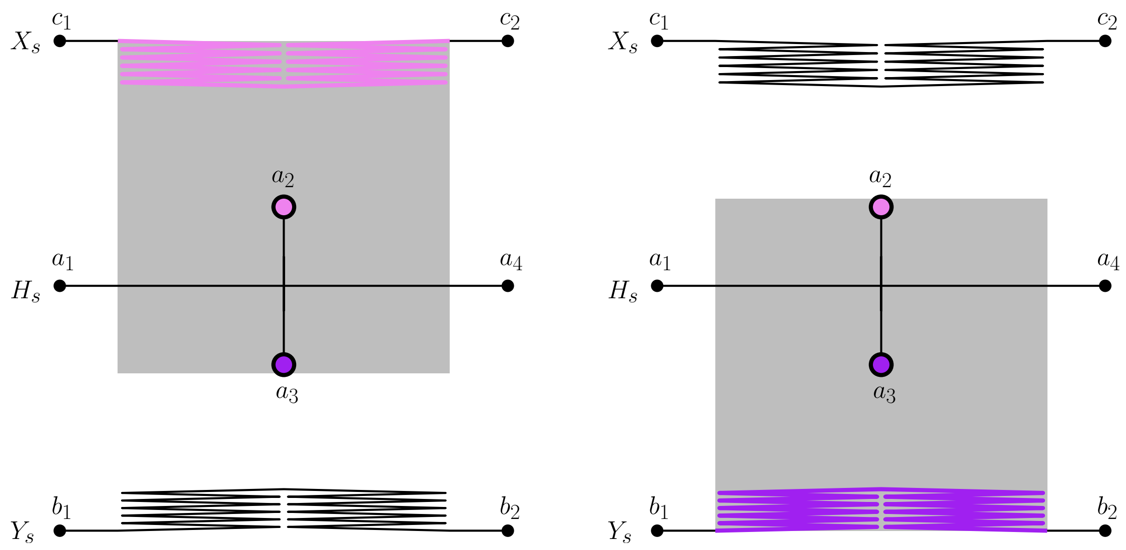

Figure 12: Chronological order in which subcurves from H_{block}, X_{block}, and Y_{block} are stamped out for the construction of a path blocker gadget in the free space diagram of H_{block}, X_{block}, and Y_{block}.