# The Non-Stop Disjoint Trajectories Problem

Benno Hoch[a,*], Frauke Liers[a], Sarah Neumann[b], Francisco Javier Zaragoza Martínez[c]

[a] *FirstName.LastName@fau.de, Friedrich-Alexander-Universtität Erlangen-Nürnberg, Cauerstrasse 11, 91052 Erlangen, Germany*
[b] *Fraunhofer IIS, Nordostpark 93, 90411 Nürnberg, Germany*
[c] *franz@azc.uam.mx, Universidad Autónoma Metropolitana Azcapotzalco, Ciudad de México 02200, Mexico*

**Abstract**

Consider an undirected network with traversal times on its edges and a set of commodities with connection requests from sources to destinations and release dates. The non-stop disjoint trajectories problem is to find trajectories that fulfill all requests, such that the commodities never meet. In this extension to the $\mathcal{NP}$-complete disjoint paths problem, trajectories must satisfy a non-stop condition, which disallows waiting at vertices or along arcs. This problem variant appears, for example, when disjoint aircraft trajectories shall be determined or in bufferless packet routing. We study the border of tractability for feasibility and optimization problems on three graph classes that are frequently used where space and time are discretized simultaneously: the path, the grid, and the mesh. We show that if all commodities have a common release date, feasibility can be decided in polynomial time on paths. For the unbounded mesh and unit-costs, we show how to construct optimal trajectories. In contrast, if commodities have individual release intervals and turns are forbidden, then even feasibility is $\mathcal{NP}$-complete for the path. For the mesh and arbitrary edge costs, with individual release dates and restricted turning abilities of commodities, we show that optimization and approximation are not fixed-parameter tractable.

**Keywords:** disjoint paths; trajectory planning; combinatorial optimization; complexity

**Declarations of interest:** none.

## 1. Introduction

Conflict avoidance is a frequently encountered problem when objects that need to use shared and capacity-constrained resources move through a network. The respective mathematical models often need to determine paths that are in some sense disjoint. In many applications, such as traffic routing or packet routing, these tasks contain a temporal component for determining conflicts. In a base network, given by a base graph with traversal times, each object follows a trajectory that consists of a walk followed over time. In order not to exceed the capacities at any point in time, trajectories need to be pairwise disjoint.

We study a special version of this wide class of problems, namely the *non-stop disjoint trajectories problem* (NDTP), which disallows waiting at vertices or along edges of the base network. One

*Corresponding Author

main motivation to add the non-stop restriction to the disjoint trajectories problem is planning for aircraft, but also bufferless and direct routing [2, 20]. In this work, we aim at gaining a theoretical understanding of the NDTP in the setting of unit traversal times for connections in an (undirected) base network. Thereby, we consider $k$ connection requests together with a release date for each commodity when it *must* start its trajectory. A commodity disappears from the network once it has reached its destination. The task is to determine disjoint trajectories (for a formal definition see Section 2) for all commodities. We consider both feasibility and optimization tasks.

After all, the NDTP can be seen as a generalization of the vertex disjoint paths problem which was already proven to be $\mathcal{NP}$-complete for general $k$ by Karp [14]. Nevertheless, for special graph classes or structures, there are polynomial time approaches for related problems, e.g. for the packet routing problem on the grid [21]. In this work we want to study the border between tractable and intractable settings for the NDTP. Therefore, we mainly consider three settings of the NDTP with additional constraints on the characteristics of commodities. First, a basic variant where requests have *common* release dates and the selection of consecutive edges is *unrestricted* (CU). Second, we consider two variants where direct returns to the predecessor vertex in the base network are disallowed. Here, one setting is described by requests that have *common* release dates but the selection of consecutive edges is *restricted* (CR). In the other, requests can have *individual* release dates and selection of consecutive edges is *restricted* (IR). This is motivated by commodities representing real live vehicles that have to follow physical restrictions on their turning behavior.

We will consider three main graph classes for our study of computational complexity: The *path*, the *grid* with four-neighborhood, and the grid with eight-neighborhood (also called a *mesh*).

*Main Contributions:* We show that feasibility can be decided in polynomial time for all three settings on the path. In contrast, for three more application oriented generalizations of the NDTP on the path we show, that the feasibility problem already becomes an $\mathcal{NP}$-complete problem (Theorems 4.1, 4.3 and Proposition 4.4). For the grid with four- and eight-neighborhoods and a common destination we prove that, if the grid is large enough, each instance is feasible for settings CU and CR in Theorem 5.2. Further, we prove sharp lower bounds for three classes of relevant optimization objectives in Theorem 5.5. For setting IR on the grid with eight-neighborhood and individual edge costs we prove that minimization of the sum of costs for disjoint trajectories is not fixed-parameter tractable in Theorem 6.6.

*Outline:* Section 2 gives a formal statement of the problem under consideration and reviews related work. Section 3 studies feasibility of the NDTP on the path. Here, also some results for generalizations of the NDTP are presented. Section 5 considers the NDTP on the unit grid with four- and eight-neighborhoods and common destination, where feasibility and three optimization tasks are studied. Section 6 generalizes the grid with eight-neighborhood and allows for individual edge cost in the base network. The sum of costs objective is studied for setting IR. We conclude with Section 7, where we recapitulate some open questions and give an outlook to future research.

## 2. Problem Statement and Related Work

### *2.1. General definitions*

Consider a base network given by a simple, undirected graph $G = (V, E)$ with unit traversal times on its edges. We are given $k \in \mathbb{N}$ *connection requests* which, for each $1 \leq i \leq k$, consist of a *commodity* $i$, a *source* $s_i \in V$, a *destination* $d_i \in V \backslash \{s_i\}$, and a *release date* $r_i \in \mathbb{N}$. Further, let a planning horizon $T \in \mathbb{N}$ be given.

A *trajectory* $W$ of length $g$ is a sequence $(v_0, t^0), \ldots, (v_g, t^g)$ such that $v_j \in V$ for each $0 \leq j \leq g$, $e_j = \{v_{j-1}, v_j\} \in E$ for each $1 \leq j \leq g$, and $t^j = t^{j-1} + 1$ for each $1 \leq j \leq g$. In this case, we say that trajectory $W$ *traverses* each of $v_0, \ldots, v_g$ and $e_1, \ldots, e_g$. We say that trajectory $W$ *fulfills* connection request $i$ if $v_0 = s_i$, $v_g = d_i$, and $t^0 = r_i$.

In problems where space and time are discretized simultaneously, one often uses a *time-expanded* network $G^T = (V^T, E^T)$ with vertices $(v, t)$ for each $v \in V$ and $t = 0, 1, \ldots, T$. An arc exists from $(v, t)$ to $(w, t+1)$ in case the base graph $G$ contains an edge $e = \{v, w\} \in E$. Thus, a trajectory in the base network $G$ has a one-to-one correspondence to a path in the time-expanded network $G^T$, which we also refer to as trajectory. We note that $G^T$ is a directed acyclic graph.

We call two trajectories *disjoint*, if they do not share a position of the network at any time. Thus, two disjoint trajectories shall never use the same vertex in the time-expanded network. Having this, the impossibility of overtaking along an edge is already implied as we are given exactly one traversal time for each edge of the base network. If an edge of the base network is used by two trajectories in opposite directions, then it has to be ensured that commodities do not have a frontal crash while using this edge. These considerations are formally stated in the following definition.

**Definition 2.1** (Disjoint trajectories)**.** *Two given trajectories in $G^T$ are* disjoint *if they do not meet in $G^T$, that is:*

1. *They are vertex disjoint.*
2. *If one trajectory traverses $((v, t^1), (w, t^1 + 1)) \in E^T$ and the other trajectory traverses $((w, t^2), (v, t^2 + 1))$, then either $t^2 < t^1 + 1$ or $t^2 + 1 < t^1$ holds.*

*If two trajectories are not disjoint, we say that they are* in conflict*.*

**Definition 2.2** (Feasible trajectories)**.** *We say that $k$ trajectories $W_1, \ldots, W_k$ are* feasible *if they are pairwise disjoint and, for each $1 \leq i \leq k$, trajectory $W_i$ fulfills connection request $i$.*

**Definition 2.3** (Objectives)**.** *Let a base network be given by a base graph $G = (V, E)$ with unit traversal times and edge costs $c_e \in \mathbb{N}$ for each $e \in E$. We define the following objectives for $k$ connection requests on $G$ that have to be fulfilled:*

feas*: Decide whether feasible trajectories exist.*

minsum*: Minimize the sum of the costs of feasible trajectories.*

minmax*: Given minimal possible costs $c_1^*, \ldots, c_k^*$ for commodities to reach their respective destinations. Find feasible trajectories with costs $c_1, \ldots, c_k$ minimizing $\max_i\{c_i - c_i^*\}$.*

makespan*: Minimize the makespan, that is, the latest arrival time of any commodity.*

**Definition 2.4** (Settings)**.** *Let a base network be given by a base graph $G = (V, E)$ with unit traversal times. We define the following* settings *for $k$ connection requests on $G$ that have to be fulfilled:*

*CU All commodities $1 \leq i \leq k$ have a* common *release date $r_i = 0$. Selection of consecutive edges is* unrestricted*.*

*CR All commodities $1 \leq i \leq k$ have a* common *release date $r_i = 0$. Selection of consecutive edges is* restricted*. That is, commodities are not allowed to traverse the same edge of the base network in consecutive time steps.*

*IR* *Commodities* $1 \leq i \leq k$ *have* individual *release dates* $r_i \in \mathbb{N}$. *Selection of consecutive edges is* restricted*, as in CR.*

*In Section 4.1, we will consider a generalization of IR where we allow release intervals. This setting is called IIR.*

**Remark 2.5.** *Throughout the paper we assume that the planning horizon $T$ while being finite, is large enough to not impose any restriction.*

**Definition 2.6** (Non-stop disjoint trajectories problem)**.** *Let a base network be given by an undirected base graph $G = (V, E)$ with unit traversal times and costs $c_e \in \mathbb{N}$ for each $e \in E$, and $k$ connection requests together with a* setting *for the commodities, and an* objective*. Consider a finite, but large time horizon $T \in \mathbb{N}$ that does not impose any restrictions. The* non-stop disjoint trajectories problem (NDTP) *is to find $k$ feasible trajectories in the given setting that are optimal with respect to the given objective, or to detect infeasibility.*

**Definition 2.7** (Notation)**.** *We introduce a three-field notation, similar to what is used in the scheduling literature, in order to shorten the specification of an NDTP instance:*

$$Setting|Network|Objective - NDTP$$

*The network is thereby characterized by $G, c_e$, that is, the base graph $G$ and the edge costs $c_e$. If a field is filled with a $*$ this implies that the stated property holds true for all possible entries of this field. For example, $CR|G, *|$*minsum*$-NDTP$ specifies the NDTP on a base network $G$ with unit traversal times and arbitrary edge costs, where all connection requests have a common release date, movements of commodities are restricted, and the* minsum *objective has to be optimized.*

After the following overview over related work we start to study complexity on different graph classes that arise frequently in different applications: First the path, which appears in current practice for air traffic management, where aircraft are aligned to follow a predefined path when approaching an airport. Second the grid and the mesh, common structures not only when space has to be discretized, but also, e.g., for packet routing in parallel computing.

### *2.2. Related work*

The $k$ disjoint paths problem has received a lot of attention and there are many results regarding its complexity for special graph classes or fixed $k$. On the one hand there are hardness results, as for example for undirected graphs the $k$-vertex disjoint paths problem with $k$ being part of the input is shown to be $\mathcal{NP}$-complete even for planar graphs [19]. On the other hand, for many cases it is known that the disjoint paths problem is in $\mathcal{P}$. Deciding feasibility is in $\mathcal{P}$ for fixed $k$ in undirected graphs [22]. If all connection requests share either a common source or destination, the vertex disjoint path problem is in $\mathcal{P}$ [9]. For *fixed* $k$, polynomial time algorithms are available for finding $k$ vertex disjoint paths for directed acyclic graphs, even if no assumptions are made on the sources and destinations. The known algorithms have worst case asymptotic running times that are exponential in $k$. For a broad overview of complexity results see [15].

Disjoint trajectories where waiting at vertices is allowed also has received a lot of attention, especially in the context of robot motion, see [7, 27, 28].

In the context of packet-routing the non-stop property is also considered in bufferless or direct routing, see [2, 20]. Here, packets have to move non-stop and conflict-free along given routes, but

with the possibility to wait at the source. Hence, feasibility of instances is always guaranteed. Optimizing the routes as well leads to $\mathcal{NP}$-hard problems in general. For the unbounded grid and waiting allowed also along the trajectory, [21] provides some polynomial time optimal and approximation algorithms.

For unit traversal times, the NDTP is equivalent to the dynamic unsplittable multi-commodity flow problem with unit capacities on edges. A nice overview of dynamic flows with continuous time can be found in [24]. A good introduction to the time-discretized version of dynamic flows is given in [13]. The multi-commodity flow over time problem with continuous time is $\mathcal{NP}$-hard, even when restricted to series-parallel networks or to the case of only two commodities [11]. For a robust approach considering uncertain traversal times for edges and no-wait restrictions, even verifying a solution is $\mathcal{NP}$-hard [10].

As we consider undirected graphs as base networks, but the time expanded graphs are always acyclic directed graphs the NDTP lies in between those results. Hence, especially for special graph classes, such as the line, the grid, and the mesh, complexity of the different settings for the NDTP requires further investigation. We give further references to the literature for problems related to the modeling extensions on the line in the corresponding Section 4.

## 3. The NDTP on the Path

We start by studying the feasibility problem of all three introduced settings on the path, i.e. a representation of a (one-dimensional) line. In fact we will show, that all introduced settings can be decided in polynomial time. In an attempt to get closer to real world applications of the NDTP on the path such as air traffic management, we study afterwards the complexity of a generalized setting in Section 4.

For each $n \in \mathbb{N}$, let $\mathsf{P}_n = (V_n, E_n)$ be the path on $n$ vertices, that is

$$V_n = \{1, \dots, n\} \text{ and} \tag{1a}$$
$$E_n = \{\{v, v+1\} \mid v, v+1 \in V_n\}. \tag{1b}$$

For the settings CR and IR deciding whether a given instance on the path is feasible or not can be done in polynomial time: Due to the restricted movement condition on the path, commodities cannot change their direction of movement. Hence, for each commodity there is a unique trajectory if it wants to arrive at its destination. For each pair of commodities it can be checked in constant time whether their trajectories are disjoint or not. As there are $\mathcal{O}(k^2)$ possible pairs, feasibility can be checked in time $\mathcal{O}(k^2)$.

**Proposition 3.1.** *CR*$|\mathsf{P}_n, *|$feas-*NDTP and IR*$|\mathsf{P}_n, *|$feas-*NDTP can be decided in time* $\mathcal{O}(k^2)$.

For the setting CU, where turning along the path is allowed, trajectories are not unique and deciding whether an instance is feasible is not trivial anymore. In the following, we will show that CU$|\mathsf{P}_n, 1, *|$feas-NDTP can be decided in polynomial time. Namely, we present in Lemma 3.5 an iterative criterion to determine whether a commodity can possibly be the first to arrive to its destination. This criterion is used in Theorem 3.6 to show the desired property. Some preliminary results that are necessary for the proof are given next. These hold also for general base networks $G$.

In order to verify iteratively whether or not a general CU$|G,*|$feas-NDTP instance is feasible, we start with the following lemma. It states that any movement of commodities along disjoint trajectories can be undone if turning behavior is not restricted.

**Lemma 3.2.** *Consider a CU$|G,*|$feas-NDTP instance with $k$ connection requests and assume that at time $t$ the commodities are placed at pairwise distinct vertices $v_1^t, ..., v_k^t$. Further assume that for some $h > 0$ there are disjoint trajectories such that at time $t + h$ the commodities are at pairwise distinct vertices $v_1^{t+h}, ..., v_k^{t+h}$. Then there are also disjoint trajectories such that this is* undone*, that is, at time $t + 2h$ the commodities are again placed at $v_1^t, ..., v_k^t$.*

*Proof.* Let $W_i = \big((v_i^t, t), (v_i^{t+1}, t+1), \ldots, (v_i^{t+h-1}, t+h-1), (v_i^{t+h}, t+h)\big)$ be the trajectory for commodity $1 \leq i \leq k$. If $W_1, \ldots, W_k$ are pairwise disjoint, then it follows directly that also

$$W_i' = \big((v_i^{t+h}, t+h), (v_i^{t+h-1}, t+h+1), \ldots, (v_i^{t+1}, t+2h-1), (v_i^t, t+2h)\big)$$

for $1 \leq i \leq k$ are pairwise disjoint. □

We introduce this property to be later used it in the following way: Let an instance and an initial configuration of commodities be given. Assume we have a criterion that allows us to determine whether a commodity $j$ (or a non-empty subset $J \subseteq [k]$) can reach its destination first while all commodities follow disjoint trajectories. In this case, we do not have to consider where the other commodities actually are after $j$ (resp. $J$) reached its destination and disappeared from the network. Instead we can use Lemma 3.2 and *return* them to the initial configuration. Now, we can apply the criterion again on the remaining, smaller, instance where commodity $j$ (resp. $J$) no longer has to be considered.

This idea is formalized in the following theorem.

**Theorem 3.3.** *A CU$|G,*|$feas-NDTP instance with $k$ connection requests is feasible if and only if there are disjoint trajectories for the $k$ commodities such that:*

- *a non-empty subset $J \subseteq [k]$ of requests is fulfilled simultaneously at time $t$ and no other request is fulfilled earlier than $t$.*
- *the CU$|G,1,*|$feas-NDTP instance with the set $[k] \setminus J$ of requests is feasible.*

*Proof.* ($\Rightarrow$) Since the former instance is feasible, there are feasible trajectories $W_s$ for $s \in [k]$. Follow these trajectories until the first time $t$ at which at least one connection request has been fulfilled. Let $J$ be the subset of all connection requests fulfilled at $t$. By Lemma 3.2, the commodities corresponding to connection requests $[k] \setminus J$ can return to their original positions at time $2t$. The trajectories $W_s$ for $s \in [k] \setminus J$ show that the latter instance is feasible.

($\Leftarrow$) Follow the disjoint trajectories until the connection requests in $J$ are fulfilled at time $t$. By Lemma 3.2, the commodities corresponding to connection requests $[k] \setminus J$ can return to their original positions at time $2t$. Since the latter instance is feasible, there are feasible trajectories $W_s$ for $s \in [k] \setminus J$. Following these trajectories shows that the former instance is feasible. □

If a given NDTP instance is not feasible, we can sill use Lemma 3.2 and Theorem 3.3 to characterize the subset of requests of that instance that can reach their destination. For an NDTP instance, a subset $S \subseteq \{1, \ldots, k\}$ of connection requests is *realizable* if there are $k$ disjoint trajectories such that all connection requests in $S$ are fulfilled. Subset $S$ is a *maximal* realizable subset if no request can be added such that $S$ is still realizable. It turns out that maximal realizable sets are unique, as we show next.

**Corollary 3.4.** *Any instance of CU|G, ∗|*feas*-NDTP has a unique maximal realizable set.*

*Proof.* For a contradiction, assume that $S_1$ and $S_2$ are two distinct maximal realizable sets. There are orderings $i_1, \ldots, i_h$ of $S_1$ and $j_1, \ldots, j_q$ of $S_2$ in which those connection requests can be fulfilled. Without loss of generality, there is an index $\ell$ such that $i_1 = j_1, \ldots, i_\ell = j_\ell$ and $j_{\ell+1} \notin S_1$. Therefore, let $t$ be the time when $j_{\ell+1}$ is fulfilled, following the disjoint trajectories implied by $S_2$. Now, the remaining commodities can return to their initial configuration according to Lemma 3.2. Let the remaining commodities now follow their trajectories as they are implied to fully realize $S_1$. As those where feasible for the full instance, they are also for the remaining requests. Hence, in the end all requests in $S_1$ and $j_{\ell+1}$ are fulfilled, contradicting the maximality of $S_1$. □

The property given in Theorem 3.3 can be used to iteratively check feasibility of an instance. The remaining question for being able to apply the theorem is how to check whether there is a particular commodity that can reach its destination first. In the following we consider explicitly the path $\mathsf{P}_n$ and derive a criterion that can be checked in polynomial time.

For each $1 \leq i \leq k$, let $s_i \in V_n$ be the source and $d_i \in V_n$ be the destination of connection request $i$. We assume without loss of generality that $s_1 < \cdots < s_k$ and that $s_i \neq d_i$ for each $1 \leq i \leq k$. For each $1 \leq i < k$, let $g_i = s_{i+1} - s_i > 0$ be the *gap* between consecutive sources $s_i$ and $s_{i+1}$.

For each $1 \leq i \leq k$, let the *end of the line* for $i$ be $\Omega_i = 1$ and $D_i = \{1, \ldots d_i\}$ if $d_i < s_i$. Otherwise we set $\Omega_i = n$ and $D_i := \{d_i, \ldots, n\}$. For each $1 \leq i \leq k$, let $S_i$ be the set of sources between $s_i$ and $\Omega_i$, that is, let $S_i = \{s_1, \ldots, s_k\} \cap \{s_i, \ldots, \Omega_i\}$. Finally, let $\Gamma_i$ be the number of *even* gaps between consecutive sources in $S_i$.

**Lemma 3.5.** *Let a CU|*$\mathsf{P}_n, *$|*feas*-NDTP instance with k connection requests be given. A connection request $1 \leq i \leq k$ can be fulfilled first if and only if $|D_i| \geq |S_i| + \Gamma_i$.*

*Proof.* Let $1 \leq i < k$ and consider the trajectories of two consecutive commodities $i$ and $i+1$. The initial gap between these commodities is $g_i$. Each time they move, this gap either remains constant (if they move in the same direction), increases by two (if they move away from each other), or decreases by two (if they move towards each other). Hence, the gap between the two consecutive commodities $i$ and $i+1$ has always the parity of $g_i$. Furthermore, since their trajectories must be disjoint, their gap is always positive. In other words, if the gap is odd, then consecutive commodities can be in consecutive vertices. Otherwise, if the gap is even, then consecutive commodities have at least one empty vertex between them.

($\Rightarrow$) Assume that connection request $i$ is fulfilled first. This implies that all commodities with sources in $S_i$ are located simultaneously in distinct vertices of $D_i$. Moreover, at the same time, $D_i$ must contain at least $\Gamma_i$ empty vertices, one for each even gap between consecutive commodities. It follows that $|D_i| \geq |S_i| + \Gamma_i$.

($\Leftarrow$) Assume that connection request $i$ satisfies $|D_i| \geq |S_i| + \Gamma_i$. We construct disjoint trajectories for all commodities such that connection request $i$ is fulfilled. Without loss of generality, assume that $s_i > d_i$, which in turn implies that $\Omega_i = 1$, $D_i = \{1, \ldots, d_i\}$, and $S_i = \{s_1, \ldots, s_i\}$. For each time step, decide for each commodity $1 \leq j \leq k$ in ascending order, their next position as follows: Move commodity $j$ towards 1 if it is possible, otherwise move it towards $n$. Observe that this will *preserve* the gaps between consecutive commodities that move in the same direction and that it will *decrease* the gaps between consecutive commodities that move in opposite directions. Furthermore, for $1 \leq j < i$ the gaps between consecutive commodities $j$ and $j+1$ will eventually become 1 or 2, according to whether they were initially odd or even. But latest when this happens, commodity $i$

would have been able to reach position $|S_i| + \Gamma_i$. Since $s_i > d_i = |D_i| \geq |S_i| + \Gamma_i$, it follows that, latest at that point, commodity $i$ has reached its destination $d_i$. Note that, until that moment, commodities $i, \ldots, k$ have moved towards 1 at each time step. □

We prove now that the property given by Lemma 3.5 can be tested in linear time and, as a consequence, CU|$\mathsf{P}_n, *$|feas-NDTP can be solved in quadratic time.

**Theorem 3.6.** *Any instance of CU|$\mathsf{P}_n, *$|feas-NDTP with $k$ connection requests can be solved in time $\mathcal{O}(k^2)$.*

*Proof.* Sort the $k$ commodities in such a way that $s_1 < \ldots < s_k$. This can be done in time $\mathcal{O}(k \log k)$. Note that this order does not change, even if some commodities have reached their destinations and disappear.

Now we test whether some commodity $i$ satisfies the property given by Lemma 3.5. For this, we compute first all gaps $g_1, \ldots, g_{k-1}$ and the total number $\Gamma$ of even gaps. This can be done in time $\mathcal{O}(k)$. Second, for each $1 \leq i \leq k$, we compute $\Omega_i$, $|D_i|$, and $|S_i|$ as follows: If $d_i < s_i$, then $\Omega_i = 1$, $|D_i| = d_i$, and $|S_i| = i$. Otherwise, $\Omega_i = n$, $|D_i| = n - d_i + 1$, and $|S_i| = k - i + 1$. This can be done in time $\mathcal{O}(k)$. Third, for all $1 \leq i \leq k$, we compute $\Gamma_i$ as follows:

1. Let $\gamma = \Gamma$ and $g_k = 1$ which corresponds to an artificial odd gap after $s_k$.
2. For each $1 \leq i \leq k$:
   (a) If $s_i < d_i$, then $\Gamma_i = \gamma$, otherwise $\Gamma_i = \Gamma - \gamma$.
   (b) If $g_i$ is even, then decrease $\gamma$ by 1.

Observe that due to 2.b, the value of $\gamma$ is at every step the total number of even gaps between $s_i$ and $s_k$. These computations can be performed in time $\mathcal{O}(k)$. Finally, for each $1 \leq i \leq k$, we test whether $|D_i| \geq |S_i| + \Gamma_i$ or not. This can also be done in time $\mathcal{O}(k)$. Hence, in time $\mathcal{O}(k)$, we can decide whether some commodity $i$ satisfies the property or not.

In the latter case we stop, the instance is not feasible. In the former case, we delete commodity $i$ from consideration, we renumber the commodities $i + 1, \ldots, k$ to $i, \ldots, k - 1$, and we test again the property given by Lemma 3.5. Since this has to be done at most $k$ times, the total running time is $\mathcal{O}(k^2)$. □

We note that the renumbering at the end of the proof preserves the desired ordering of the sources and hence sorting again is unnecessary. Also, by Proposition 3.2, we can assume that the remaining commodities returned to their original sources. Furthermore, by Corollary 3.4, the described algorithm stops with a maximal realizable set.

Combining Proposition 3.1 and Theorem 3.6 for all considered settings on the path we showed that the feasibility problem can be solved in polynomial time, if the time horizon is large enough. Before moving on to the grid in Section 5 in the following section we consider some modeling extension for the path that are directed more towards applications where physical commodities are involved.

## 4. Modeling Extensions for the Path

As mentioned before, the NDTP has many real world applications such as air traffic management, where in current practice aircraft are at some point aligned to follow a one-dimensional path to the runway. Thereby, there is some flexibility to decide when an aircraft enters that path. In contexts such as this, the presented settings are too restrictive to allow for a representation of the real world necessities.

The positive results so far motivated us to consider such more realistic settings. Therefore, we study the version with restricted movement but individual release *intervals* instead of release dates. In the time discrete setting, due to the possibility to decide for a release time, the problem is related to the multiple choice problem with clique constraints as studied e.g. in [3, 4, 5, 18]. While it is in general $\mathcal{NP}$-complete, several structures that allow for polynomial time decisions are known that apply for example in the application of train timetabling. Further, there is a close relation to several scheduling problems with no-wait constraints between successive machines. For scheduling problems in general there is a wide variety for results, with those regarding $\mathcal{NP}$-completeness of the job-shop and flow-shop problem in [26] resp. [23] being some of the closest. None of the above can fully be transformed into a NDTP with release intervals, which is why we study its complexity in the following.

It turns out that, even for what before was the trivial case with restricted movement, the problem now becomes $\mathcal{NP}$-complete. This remains true even if in addition all commodities have to follow the same direction and for continuous time.

### *4.1. Release Intervals*

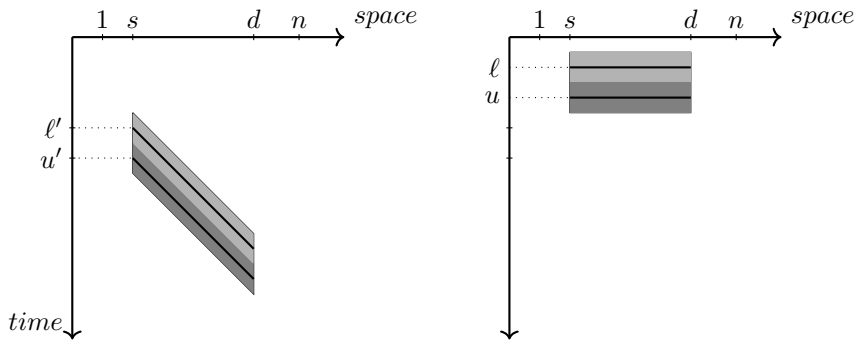
In this subsection we generalize the settings studied until now to allow for individual release intervals instead of exact release dates. This generalization is natural for many applications, but already deciding feasibility will turn out to be $\mathcal{NP}$-complete if movement of commodities is restricted. We call the setting under consideration IIR, that is, requests have *individual* release *intervals* and *restricted* movements of commodities as in CR. For each connection request $1 \leq i \leq k$, its release interval is $[\![\ell_i, u_i]\!] = \{\ell_i, \ell_i + 1, \ldots, u_i\}$, with $\ell_i \leq u_i \in \mathbb{N}$. If in a feasible solution the departure time for a commodity $i$ is set to $r_i \in [\![\ell_i, u_i]\!]$ we say that $i$ is *scheduled* at time $r_i$. A feasible solution is also called *scheduling*.

In the following we will prove $\mathcal{NP}$-completeness of IIR$|\mathsf{P}_n, *|$feas-NDTP even in case all commodities have to follow the same direction, that is, if $s_i \leq d_i$ holds for all connection requests.

We will use a geometric interpretation of trajectories. If we identify $\mathsf{P}_n$ with the Euclidean segment $[1, n]$ we can interpret the trajectories as segments of slope $+1$ or $-1$, see Figure 1(a). Trajectories are conflict free if their segments do not cross. Further, if for an IIR$|\mathsf{P}_n, *|$feas-NDTP instance all commodities have to move left to right, then a linear transformation on the time components of a time-expanded vertex $(s, t') \mapsto (s, t)$, namely $t = t' - s$ allows to interpret trajectories as horizontal segments, see Figure 1 (b). This transformation is used in the proof of Theorem 4.1, to facilitate the understanding of graphical representation and the statement of connection requests.

**Theorem 4.1.** *The IIR$|\mathsf{P}_n, *|$feas-NDTP is $\mathcal{NP}$-complete.*

*Proof.* We first show that the problem is in $\mathcal{NP}$: For IIR on $\mathsf{P}_n$ with unit traversal times the $k$ trajectories are described by $s_i, d_i$ and $r_i$. We can check in time $\mathcal{O}(k)$ whether the scheduled times lie within the valid intervals. Conflicts can be determined by checking all pairs of segments for intersection, which can be done in time $\mathcal{O}(k^2)$.




(a) Original representation (b) Transformed representation

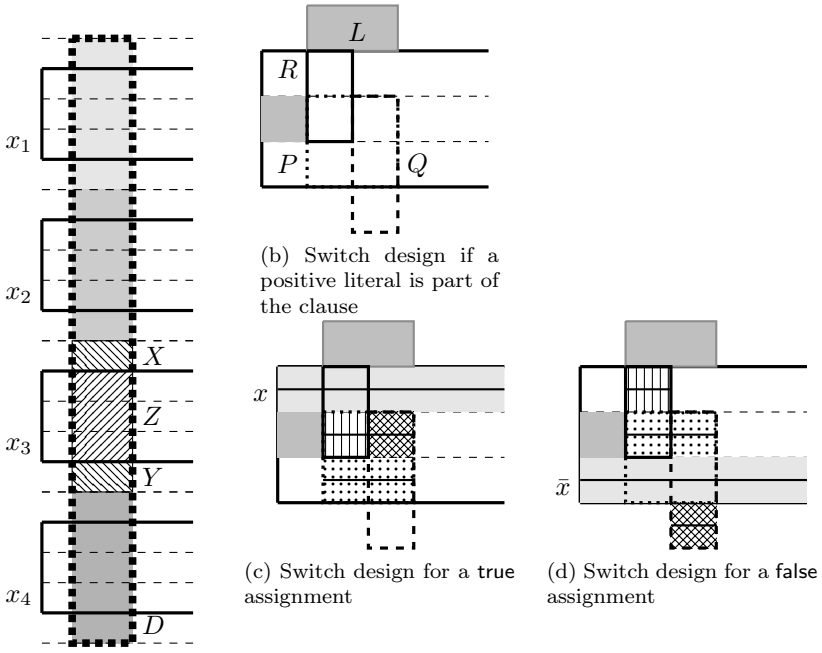
Figure 1: Geometrical interpretation for a commodity: This example has $u - \ell = 1$. Depicted as solid lines are the two possible trajectories, the shaded parallelograms illustrate the temporal separation between the two possibilities. For ease of notation in Theorem 4.1 we use $\ell'$ resp. $u'$ as earliest and latest release time in the original instance and $\ell, u$ for the transformed one.

We prove hardness by a reduction from 3-SAT which is well known to be $\mathcal{NP}$-complete (see Cook [6]). Let an instance with $q$ clauses $c^1, \ldots, c^q$ and $p$ variables $x_1, \ldots, x_p$ be given. Without loss of generality, assume that a literal and its negation do not appear in the same clause.

In order to improve readability we specify a connection request by $s(i), d(i), \ell(i), u(i)$ instead of $*_i$ .

For each variable $x_i$, $1 \leq i \leq p$, we introduce a commodity that has the source-destination pair $s(x_i) = 1$, $d(x_i) = 2q + 2$ and three possible departure times, given by $\ell(x_i) = 5(i-1) + 1$ and $u(x_i) = 5(i-1) + 3$. Thereby, choosing $\ell(x_i)$ represents that $x_i$ is set to true, choosing $u(x_i)$ represents that $x_i$ is set to false. We prevent a commodity from being scheduled at time $\ell(x_i)+1$ by introducing an additional blocking commodity $b_i$ with $s(b_i) = 1, d(b_i) = 2, \ell(b_i) = u(b_i) = \ell(x_i)+1$, which in a feasible solution necessarily has to be scheduled at this time. The latter is necessary to enable the correctness of so-called *switches* (to be explained later) which signal whether the literal included in a clause is set to true or not. This leads to a total of $2p$ commodities for the representation of the variables.

Each clause $c^j$, $1 \leq j \leq q$, is represented by a short segment of the path: One commodity $D^j$ with $s(D^j) = 2j,\ d(D^j) = 2j + 2,\ \ell(D^j) = 0,\ u(D^j) = 5p - 1$. If the corresponding IIR|$\mathsf{P}_n, *$|feas-NDTP has a feasible scheduling for the commodities, the scheduled time for decision commodity $D^j$ will signal a literal of the clause that has a true assignment. In order to achieve this, we need two categories of additional commodities: For each variable not included in the clause we need *locking* commodities that prevent $D^j$ from being scheduled at times that correspond to those variables. Further, for each variable in the clause we need further commodities that will form the so called *switch* implying whether the literal in the clause has a true assignment. First, we define the set of *locks* $X^j_*, Y^j_*, Z^j_*$ which are commodities that prevent commodity $D^j$ from starting its trajectory in time steps referring to variables not included in the clause. Thus, for each variable $x_i$ that is not part of clause $c^j$ all commodities of locks $X^j_i, Y^j_i, Z^j_i$ have $s, d$ values equal to those of $D^j$ and are defined as follows. Each lock $X^j_i$ and $Y^j_i$ consists of one commodity. $X^j_i$ has to start its path in the time step before $\ell(x_i)$, that is, it starts at $\ell(X^j_i) = u(X^j_i) = \ell(x_i) - 1$, and $Y^j_i$ at the time step after $u(x_i)$, i.e. at $u(x_i) + 1$. Further, lock $Z^j_i$ are two identical commodities that have the same $\ell, u$ as




(a) Example for an instance with four variables

(b) Switch design if a positive literal is part of the clause

(c) Switch design for a true assignment

(d) Switch design for a false assignment

Figure 2: Scheme for clause and switch design. For ease of exposition, we assume departure times are centered between the dashed lines.

the commodity corresponding to $x_i$ and block the two time steps remaining after the assignment of $x_i$. So far, these are $1 + 4(p-3)$ commodities per clause.

Next, for the variables included in the clause, we introduce commodities that form switches to determine whether the literal has a true assignment. In the overall scheme for representing clauses shown in Figure 2(a), those are marked by gray blocks. If the positive literal $x_i$ is included in a clause, the structure is depicted in Figure 2 (b)-(d). There is one commodity $R_i^j$ that has to be scheduled either at time $\ell(x_i)$ or at $\ell(x_i)+1$ with $s(R_i^j) = s(D^j),\ d(R_i^j) = s(D^j)+1$. Further, there is one commodity $P_i^j$ that has the same $s$ and $d$ coordinates as the commodity representing clause $j$. The possible departure times are given by $\ell(P_i^j) = \ell(x_i)+1,\ u(P_i^j) = u(x_i)$. The last part of the switch is commodity $Q_i^j$ with three possible departure times given by $\ell(Q_i^j) = \ell(x_i)+1$, $u(Q_i^j) = u(x_i)+1$, $s(Q_i^j) = s(D^j)+1$, and $d(Q_i^j) = d(D^j)$.

Additionally there is one commodity $L_i^j$ with $\ell(L_i^j) = u(L_i^j) = \ell(R_i^j) - 1$ and $s(L_i^j) = s(D^j)$ as well as $d(L_i^j) = d(D^j)$ to block this time step from being used as starting time of the decision commodity $D^j$ in a feasible scheduling.

These switches work as follows: If the commodity representing variable $x_i$ is scheduled on its earliest time, i.e. representing a true assignment, the commodities *can* be scheduled as shown in Figure 2(c). This leads to one empty time step where also $D^j$ can be scheduled which implies that

this clause is true. If the commodity is scheduled on its last departure time which we interpret $x_i$ is false, the commodities of the switch can only be scheduled as depicted in Figure 2(d): $P_i^j$ has to take the middle time step $\ell(x_i)+1$, $R_i^j$ is forced to its earliest departure time and the commodity $Q_i^j$ is forced to its latest departure time. Hence, not all of the commodities included in the switch and $D^j$ can be scheduled in the interval $[\![\ell(x_i)-1, u(x_i)+1]\!]$ corresponding to variable $x_i$ in clause $c^j$.

If a negative literal $\bar{x}_i$ is in clause $c^j$, then the switch can be symmetrically mirrored around the middle time step $\ell(x_i)+1$ of $x_i$. Thus, four commodities are needed for each switch. The total number of commodities we consider is $2p+q(3\cdot 3+1+4(p-3)) \in \mathcal{O}(qp)$, which is clearly polynomial in the input size.

Finally, we show that all commodities can be scheduled if and only if the given 3-SAT instance is satisfiable.

($\Rightarrow$) Let the IIR$|\mathsf{P}_n, *|$feas-NDTP instance be feasible. Now, set the variables $x_i$ of the 3-SAT instance according to the selected departure times of the commodity corresponding to this variable. Since each commodity $D^j$ can be scheduled, for each clause there is at least one literal with a true assignment (the case of Figure 2 (c)) and hence also the 3-SAT instance is satisfiable.

($\Leftarrow$) Let the 3-SAT instance be satisfiable. Choose the departure time of each commodity that represents a variable accordingly (first or third time step). For each clause $c^j$ select at least one literal that has a TRUE assignment. Schedule the corresponding commodities of the switch according to Figure 2(c) and schedule $D^j$ in the empty time step in this switch. As the locking commodities for non involved variables can be scheduled, it follows that each commodity is scheduled and the instance is feasible. □

**Remark 4.2.** *Our proof also shows $\mathcal{NP}$-completeness of the following version of the rectangle packing problem:*

***Rectangle Packing***

*Let a finite set of axis aligned bounding rectangles* B *in the Euclidean plane be given and, for each $b \in \mathsf{B}$, a set* P$(b)$ *of rectangles which have to be placed inside $b$. Is there a way to place all rectangles $\mathcal{P} = \bigcup_{b\in\mathsf{B}} \mathsf{P}(b)$ without intersections of their interiors?*

*Although there are complexity results for related problems, see for example [16], we are not aware of an earlier proof showing hardness of this rectangle packing problem.*

### *4.2. Continuous Time*

Not only in air traffic management and runway scheduling, but also for example train timetabling, separation distances are given as temporal distances instead of spatial ones. This is captured by the following problem version where space and time are considered to be continuous. Considering rational information for connection requests thereby is not a severe restriction for applications in general.

**Continuous $k$-NDTP on the line with unit speed:** IIR-$\epsilon|\mathbb{Q}, *|$feas-NDTP

Let $k$ connection requests be given, for each commodity $1 \le i \le k$ specified by a source $s_i \in \mathbb{Q}$, destination $d_i \in \mathbb{Q}$ and earliest and latest possible departure times $\ell_i, u_i \in \mathbb{Q}$ respectively, from $s_i$. Commodities have unit speed, that is, they traverse one unit of space in one unit of time, and they cannot change direction. A temporal safety distance of $\epsilon \in \mathbb{Q}^+$ between commodities has to be assured. That is, if a commodity $i$ is at location $p$ on the path at time $t$ no other commodity $j$ is allowed to also be at location $p$ in the time interval $[t-\epsilon, t+\epsilon]$.

Commodities appear in the system as soon as they start their trajectory (that is, at their actual departure time) and disappear instantaneously after having reached their destination. Can we decide for a feasible departure time for each commodity such that all requests are fulfilled by trajectories with safety distances?

The notation IIR-$\epsilon|\mathbb{Q},*|$feas reads similarly to the notation that has been used so far: Commodities have individual release intervals, movement is restricted and a common safety distance of $\epsilon$ is considered. The base network is given by the line with unit speed and requests have rational data. As we show in the following theorem this problem can be reduced to IIR$|\mathsf{P}_n,*|$feas-NDTP and hence is also $\mathcal{NP}$-complete.

**Theorem 4.3.** *IIR-$\epsilon|\mathbb{Q},*|$feas-NDTP is $\mathcal{NP}$-complete.*

*Proof.* First we show that IIR-$\epsilon|\mathbb{Q},*|$feas-NDTP is in $\mathcal{NP}$. For a trajectory given by a rational scheduled time, departure position, and arrival position, it can be checked in constant time whether it fulfills a request. The safety region around it is a parallelogram in a time-expanded representation. The gray shades in Figure 1(a) illustrate that. Thus, two trajectories are disjoint, if the interior of the corresponding parallelograms do not intersect, which can be checked in polynomial time. As there are $\mathcal{O}(k^2)$ pairs of trajectories the overall check can be done in polynomial time.

We prove hardness by a reduction to IIR$|\mathsf{P}_n,*|$feas-NDTP where all commodities have to go the same direction. This was proven to be $\mathcal{NP}$-complete in Theorem 4.1.

Let an IIR$|\mathsf{P}_n,*|$feas-NDTP instance with $k$ connection requests with $s_i \leq d_i$ for all $i$ be given. Directly transfer the connection requests to an instance of the IIR-0.5$|\mathbb{Q},*|$feas-NDTP. We show next, that this instance is feasible if and only if the IIR$|\mathsf{P}_n,*|$feas-NDTP instance was feasible.

($\Leftarrow$): Let a feasible solution for the IIR$|\mathsf{P}_n,*|$feas-NDTP instance be given. Using the geometrical interpretation given in Figure 1(a) we know that in the time-expanded network the line segments have a distance of at least 1 as the gray shades exactly represent the safety distances for $\epsilon = 0.5$. Feasibility of IIR-0.5$|\mathbb{Q},*|$feas-NDTP follows.

($\Rightarrow$): Let a feasible solution to the IIR-0.5$|\mathbb{Q},*|$feas-NDTP instance be given. Assume all commodities have integral departure times. In this case the same solution is trivially feasible for the IIR$|\mathsf{P}_n,*|$feas-NDTP having the geometrical representation of Figure 1 in mind.

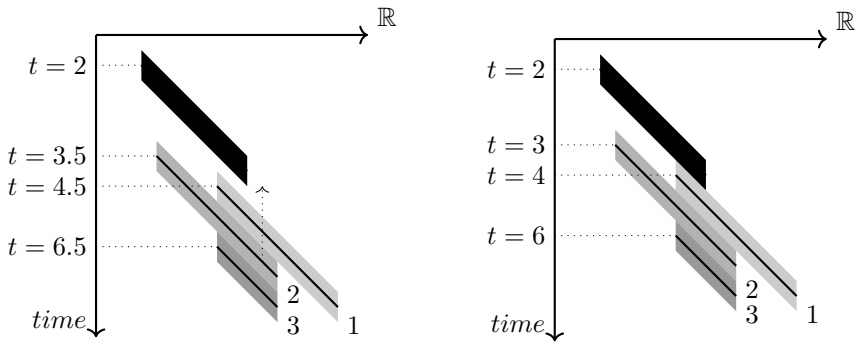
We can without loss of generality assume that the solution for the IIR-0.5$|\mathbb{Q},*|$feas-NDTP instance has integral departure times: If this is not true for a feasible solution there are $r \leq k, r \in \mathbb{N}$ so called *blocks* of commodities: Commodities in a block have non integral departure times with the same fractional part. Further, if a block consists of more than one commodity, for each commodity $i$ in the block there is at least one commodity $j$ in the same block, such that the corresponding trajectories use some common point of the line with a temporal distance of exactly $2\epsilon = 1$. Thus, all commodities gathered in a block have integral temporal distance at all common space positions.

Now, transform the given feasible solution into a feasible one with integral departure times: Consider one of those blocks. Let all commodities in this block depart earlier such that either

(i) departure times for all commodities in the block are integral. This is possible as $\ell_i \in \mathbb{N}$ for all $i$ according to IIR$|\mathsf{P}_n,*|$feas-NDTP. Further, temporal distance at common space points is integral. Thus, as the original departure times of commodities in a block had the same fractional part they all have integral departure time if they depart this amount of time earlier.

(ii) the block be combined with another block, that is, the safety distance of two commodities, one in each of those two blocks, is exactly 1 and would get smaller by letting the block under consideration start any earlier.

Figure 3 illustrates the special case where (i) and (ii) occur simultaneously.




(a) Original solution: Commodities $1, 2$ and 3 are in one block

(b) Transformed solution

Figure 3: Transformation for commodities in a block with non-integral departure times

In both cases there is one block less with non integral departure times. The result follows by induction. □

The problem is still $\mathcal{NP}$-complete, if all commodities have the same source and destination on the line, but individual temporal safety distances. This setting is also of great importance for trajectory planning in the aircraft landing process where safety distances often depend on the weight-classes of aircraft [12].

**Proposition 4.4.** *Deciding feasibility for IIR-$\epsilon_i|\mathbb{Q}, *|$feas-NDTP is $\mathcal{NP}$-complete if all connection requests have common source and destination, but individual temporal safety distances $\epsilon_i \in \mathbb{Q}^+$ for each request $1 \leq i \leq k$.*

*Proof.* That the problem is in $\mathcal{NP}$ follows directly from the proof given for Theorem 4.3.

Hardness is shown by a reduction to the scheduling problem on one machine, with $k$ jobs with processing times $p_i$, release dates $\gamma_i$, and due dates $\delta_i$ for each $1 \leq i \leq k$. Minimizing the maximal lateness for this problem is $\mathcal{NP}$-complete, see [17].

Any instance of such a scheduling problem can be transferred to an IIR-$\epsilon_i|\mathbb{R}, *|$feas-NDTP instance as follows: For each job $1 \leq i \leq k$ introduce one commodity with $s_i = d_i = 0$, $\ell_i = \gamma_i + \frac{1}{2}p_i$, $u_i = \delta_i - \frac{1}{2}p_i$ and $\epsilon_i = \frac{1}{2}p_i$.

The scheduling problem can be solved with a maximal lateness of zero if and only if the corresponding IIR-$\epsilon_i|\mathbb{Q}, *|$feas-NDTP instance is feasible. □

## 5. The NDTP with Common Destination on the Grid and Mesh

In this section we study the two dimensional grid with four-neighborhood, *grid* for short, and the grid with eight-neighborhood, *mesh* for short, as base networks. Thereby, while both structures are frequently encountered, the mesh adds more flexibility for the trajectories. Especially in the case of trajectory planning for physical commodities this allows for a more realistic description.

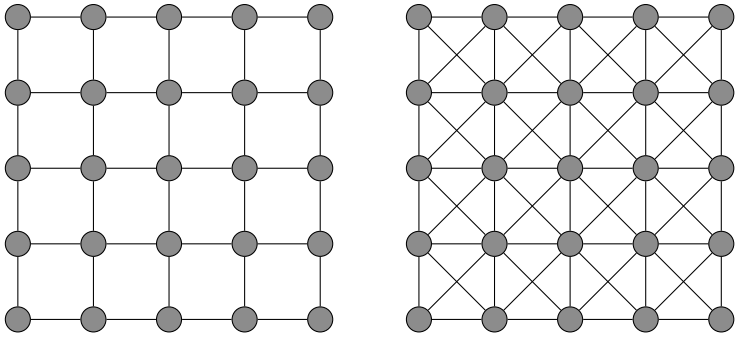
For both base networks we distinguish between the bounded and the unbounded version. For the bounded grid and the mesh, we are given two integers $a, b \in \mathbb{N}$. The grid $\mathsf{G}_{a,b} = (V_{a,b}, E^{\mathsf{G}}_{a,b})$

and the mesh $\mathsf{M}_{a,b} = (V_{a,b}, E^{\mathsf{M}}_{a,b})$ are defined as:

$$V_{a,b} = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid 1 \leq x \leq a,\ 1 \leq y \leq b\} \tag{2a}$$

$$E^{\mathsf{G}}_{a,b} = \{\{v, w\} \mid v, w \in V_{a,b},\ \|v - w\|_1 = 1\} \tag{2b}$$

$$E^{\mathsf{M}}_{a,b} = \{\{v, w\} \mid v, w \in V_{a,b},\ 1 \leq \|v - w\|_1 \leq 2,\ \|v - w\|_\infty = 1\} \tag{2c}$$



(a) $\mathsf{G}_{5,5}$ (b) $\mathsf{M}_{5,5}$

Figure 4: Example grid and mesh structure

The unbounded grid $\mathsf{G}_\infty$ and the unbounded mesh $\mathsf{M}_\infty$ have vertex set $V_\infty = \mathbb{Z} \times \mathbb{Z}$. Their edge sets $E^{\mathsf{G}}_\infty$ and $E^{\mathsf{M}}_\infty$ are defined analogously to (2b) and (2c), respectively.

We assume that the $x$-axis is horizontal and the $y$-axis vertical. In this context, we use the terms left, right, up, and down with their standard meaning.

In this section, we examine the three settings CU, CR, IR on the grid and the mesh as base networks where all connection requests have a common destination $d \in V_{a,b}$. A common sink can be found, e.g., when all information packages need to be gathered at one vertex or when commodities approach a common depot, such as an airport. First, we study the unbounded grid and mesh for settings CU and CR. It turns out that in both settings for the grid and the mesh, each instance is feasible. Besides the question of feasibility we also study the introduced optimization objectives. Therefore, we present Algorithm 4 which solves conflicts at the common destination. For all these optimization objectives, this provides lower bounds for general base graphs. We will use this for the mesh to provide optimal trajectories with respect to the three introduced optimization objectives in the settings CU and CR. The obtained objective values coincide with the values of the lower bounds given by Algorithm 4.

We want to stress, that for a finite base network the results for settings CU and CR do not carry over in general as instances can be infeasible. Minimal examples are given by $\mathsf{G}_{3,1} = \mathsf{M}_{3,1}$, $k = 2$, $r_1 = r_2 = 0$, $s_1 = (1, 1)$, $s_2 = (3, 1)$, and $d_1 = d_2 = (2, 1)$.

For setting IR we identify a class of instances that is feasible and give optimal trajectories.

*5.1. Common release dates on the unbounded grid and mesh*

In this subsection we study the NDTP on the grid and the mesh when requests have common release dates. Before considering optimization, we first show, that for the unbounded grid and a common destination each instance of the NDTP in settings CU and CR is feasible, which directly implies that this also holds for the unbounded mesh.

**Definition 5.1** (Frame)**.** *Let a vertex set $V \in \{V_{a,b}, V_\infty\}$ be given. For a vertex $d \in V$ we say that two vertices $v, w \in V$ are on the same frame if $\|v - d\|_\infty = \|w - d\|_\infty$, see Figure 5.*

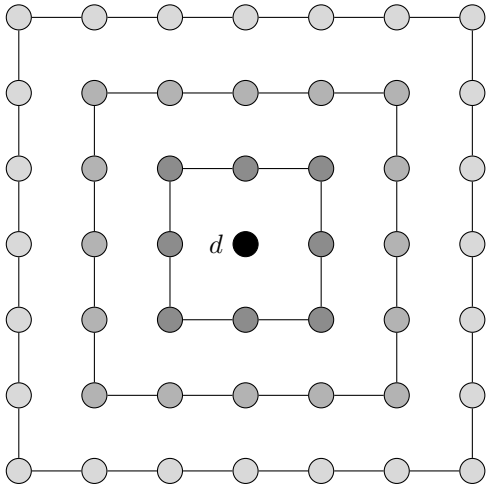



Figure 5: Example structure of frames around the common destination vertex. Different shades of gray illustrate correspond to different frames. The printed edges illustrate a Hamiltonian cycle for all vertices of a frame.

**Proposition 5.2.** *Problems $CU|\mathsf{G}_\infty, *|\mathsf{feas}$-NDTP and $CR|\mathsf{G}_\infty, 1, *|\mathsf{feas}$-NDTP with a common destination $d$ for all connection requests are feasible.*

*Proof.* We fix an ordering of the sources that are on each frame. Until not stated differently, in each time step we move each commodity one vertex clockwise within its frame following the Hamiltonian cycle as depicted in Figure 5. While doing this trajectories are disjoint. We start with the frame closest to the destination and *activate* the first commodity. That is, for this commodity and its current vertex we determine a shortest path to the destination that, if necessary, moves clockwise along its original frame. We move the commodity according to this shortest path to the destination while all others keep moving clockwise along their corresponding frame. As soon as the first commodity has arrived, we activate the second commodity, and so on. If there are no commodities left on the current frame, we continue with the next non-empty frame closest to the destination. The resulting trajectories never use an edge of $\mathsf{G}$ in opposite directions in consecutive time steps. Hence, they are feasible for both settings CR and CU. □

**Corollary 5.3.** *Every $CU|\mathsf{M}_\infty, *|\mathsf{feas}$-NDTP and $CR|\mathsf{M}_\infty, *|\mathsf{feas}$-NDTP instance is feasible.*

We now introduce a greedy algorithm that uses the earliest possible arrival times at the destination and returns a conflict free assignment of arrival times at the destination. This will yield lower bounds to all considered optimization objectives regardless of the base network, if unit traversal time and unit edge costs are considered, see Theorem 5.5. However, in Theorem 5.8 we will show that the bounds are indeed sharp for the $C * |\mathsf{M}_\infty, 1|*$-NDTP.

The main step of Algorithm 4 is to assign feasible integral arrival times to the commodities following the order in which they can earliest arrive at the destination. Thereby, it ensures that each time is assigned to at most one commodity and assigns the times such that each commodity

receives a time as early as possible. This is done by considering the commodities ordered by non-decreasing earliest arrival time. Iteratively, the next commodity is assigned, if possible, its earliest arrival time. Otherwise, it is assigned the time step directly after the latest so far assigned time. Algorithm 4 also computes the overall as well as individual additional time consumption that we call *delay*. We note that the obtained arrival times do not necessarily have to correspond to feasible trajectories.

**Algorithm 4** Greedy Delay Ordering

**Input:** Ordered sequence of earliest arrival times $t_1^d \leq \cdots \leq t_k^d$ in $d$.
**Output:** A total increase in time consumption $\Delta T$. For each commodity $i$, an individual increase in time consumption $\Delta_i$ and a new arrival time $\bar{t}_i^d$.

1: **Initialize** $\Delta T = 0, \Delta_1 = 0,\ \bar{t}_1^d = t_1^d$.
2: **for** $i = 2, ..., k$ **do**
3: $\quad \Delta_i = \max\{\bar{t}_{i-1}^d + 1 - t_i^d, 0\}$
4: $\quad \bar{t}_i^d = t_i^d + \Delta_i$
5: $\quad \Delta T = \Delta T + \Delta_i$
6: **end for**

**Theorem 5.5.** *Let a $*|G,1|*$-NDTP-instance $I$ with $k$ connection requests with common destination $d \in V$ be given. Assume commodities are numbered increasingly according to non-decreasing earliest possible arrival times $t_i^d$ at the common destination $d$. If $I$ is feasible, the following bounds are valid:*

1. $\mathsf{minsum}(I) \geq \sum_i \left(t_i^d - r_i\right) + \Delta T$
2. $\mathsf{minmax}(I) \geq \max_i\{\Delta_i\}$
3. $\mathsf{makespan}(I) \geq \bar{t}_k^d$

*where $\bar{t}_k^d, \Delta T, \Delta_i$ are computed by Algorithm 4.*

*Proof.* Ad minsum objective: As the considered costs coincide with the time each commodity spends in the network the given lower bound equals the objective if all commodities arrive as proposed by Algorithm 4. Assume an assignment of arrival times $\hat{t}_i^d$, $1 \leq i \leq k$ is given that is feasible for the common destination however different from $\bar{t}_k^d$ . Thus, also in this assignment no arrival time is assigned to more than one commodity and it respects the restriction that no commodity is assigned a time earlier than its earliest possible arrival time. The following exchange arguments reassign arrival times so that they coincide with the output provided by Algorithm 4, without increasing the value of the objective.

We start with commodity with index 1 and assign its arrival time to the time Algorithm 4 computes. As $\bar{t}_1^d = t_1^d$ is the earliest possible arrival time of any commodity, $\hat{t}_1^d \geq \bar{t}_1^d$ holds. Thus, this reassignment alone lowers the cost of the given assignment by $\delta_1 = \hat{t}_1^d - \bar{t}_1^d \geq 0$. If time $\bar{t}_1^d$ was occupied by a commodity in the given assignment of arrival times, there are at most $\bar{\delta}_1 \leq \delta_1$ many commodities with arrival times assigned between $\bar{t}_1^d$ and $\hat{t}_1^d$. We let each of them arrive one time unit later. This results in an overall cost decrease of $\Delta_1^{\text{cost}} = \delta_1 - \hat{\delta}_1 \geq 0$. Thus, it is not worse, if the given assignment has commodity with index 1 at time $\bar{t}_1^d$, while keeping the rest of the solution unchanged.

For each $1 \leq i \leq k$ rename $\hat{t}_i^d$ to the adapted arrival times as explained above.

Now, the same argument applies to commodity with index 2 and so on. In each exchange step, the cost implied by the given solution does not increase, when compared to the cost computed by Algorithm 4. The claim thus follows by induction.

Ad minmax objective: *Observation 1:* The arrival times $\bar{t}_i^d$ computed by Algorithm 4 have the following property. A time $\hat{t}$ is not assigned to any commodity if and only if for all commodities with $t_i^d \leq \hat{t}$ it holds that $\bar{t}_i^d < \hat{t}$.

Assume to be given an alternative assignment of arrival times $\hat{t}_i^d$, $1 \leq i \leq k$ with a lower minmax objective value than $\Delta = \max\{\Delta_i\}$. This implies that there is a largest index $K \in \operatorname{argmax}_i\{\Delta_i\}$ that is delayed by less than $\max\{\Delta_i\}$. Hence, $\hat{t}_K^d < \bar{t}_K^d$. Due to Observation 1 there is an index $j < K$ for the greedy assignment of arrival times such that there is an empty time slot before $\bar{t}_j^d = t_j^d$ and all times up to $\bar{t}_K^d$ are used according to the ordering of Algorithm 4. Thus, the commodity initially assigned to time $\hat{t}_K^d (\geq t_j^d)$ by Algorithm 4 has to be assigned some other time by the alternative solution. For the new slot for this commodity the same considerations apply. Thus, after a finite number of steps, there is some commodity $i'$ such that $j \leq i' < K$ with $\hat{t}_{i'}^d \geq \bar{t}_K^d$. As $i' < K$, it holds that $t_{i'}^d \leq t_K^d$ and thus, the delay of $i'$ is at least $\Delta_K = \max\{\Delta_i\}$ which is a contradiction.

Ad makespan objective: This is a direct consequence of Observation 1: If commodity $k$ arrives earlier at the destination in a different assignment of arrival times, some other commodity must arrive at $\bar{t}_k^d$ or later. □

**Remark 5.6.** *We note that in Theorem 5.5 the arrival time assignments do not necessarily need to correspond to feasible trajectories, as the length (or cost, respectively) with respect to all considered objectives can be computed simply with information on the sources, the common destination, the release dates, and the arrival times. In Theorem 5.8 we will show that there exist feasible trajectories for each CR|$\mathsf{M}_\infty$, 1|∗-NDTP and CU|$\mathsf{M}_\infty$, 1|∗-NDTP instance that take the given lower bounds.*

**Remark 5.7** (Adaptation of Algorithm 4 for the grid )**.** *For the grid we can tighten the bounds implied by Theorem 5.5 and hence Algorithm 4 by considering the following characteristic of a feasible walk in the grid: Each walk connecting any source $s$ and destination $d$ must have length $\|s - d\|_1 + 2q$ for some $q \in \mathbb{N}$. This is clear, as $\|s - d\|_1$ is the length of a shortest path, and every horizontal or vertical movement that does not get closer to the destination must be undone. This implies that each commodity in a grid can only be delayed by an even number of time steps.*

*Algorithm 4 can be adapted for the grid to make use of the above described property and hence can possibly provide better bounds. An adapted algorithm will work exactly the same as Algorithm 4, but distinguish between even and odd earliest arrival times and even and odd time steps. By assigning only even delays, it is assured that for no commodity the parity of arrival time changes. The bounds of Theorem 5.5 carry over by also distinguishing between even and odd time steps.*

For the mesh $\|v - d\|_\infty$ is equal to the length of a shortest path from $v$ to $d$. This implies, that for an instance of the NDTP on the mesh and common sink $d$ all commodities that have their source on the same frame have the same earliest possible arrival time $t^d$. This observation allows us to state the following Theorem 5.8.

**Theorem 5.8.** *Let a CU|$\mathsf{M}_\infty$, 1|∗-NDTP instance or a CR|$\mathsf{M}_\infty$, 1|∗-NDTP instance with $k$ connection requests with common destination $d \in V_\infty$ be given. The optimal cost for all considered optimization objectives are equal to the lower bounds given by Theorem 5.5.*

*Proof.* Let the commodities be indexed according to Algorithm 4. For each commodity $1 \leq i \leq k$, we compute $\Delta_i$ and $\bar{t}_i^d$ accordingly.

Let each commodity $1 \leq i \leq k$ move clockwise along its corresponding frame for exactly $\Delta_i$ time steps. Then start a shortest path from the current position to the destination. As each vertex of a frame has the same distance to the common destination, the arrival time of each commodity $1 \leq i \leq k$ is now exactly $\bar{t}_i^d$. As following a shortest path to $d$ in each step brings a commodity one frame closer to $d$ the resulting trajectories follow the restrictions of CR. Thus, the objective values will equal the lower bounds of Theorem 5.5. It remains to show that the trajectories are disjoint.

If a commodity with source on a frame with distance $\mathcal{D}$ to the destination starts its shortest path to the sink, vertices of frames that are closer to the destination are only used by commodities that are already following a shortest path to $d$. Thus, in each time step there is at most one vertex used of each frame that is closer to $d$. Hence the resulting trajectories are disjoint. □

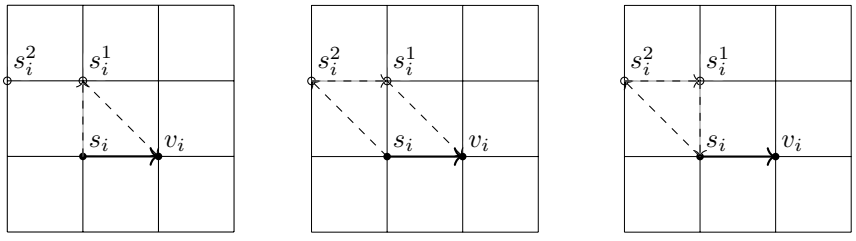
**Remark 5.9.** *It can be shown that the computation of trajectories as in the proofs of Proposition 5.2 and Theorem 5.8 can be performed in time growing polynomially in the encoding size of the connection requests and the unbounded grid and mesh. As* $\mathsf{G}_\infty$ *and* $\mathsf{M}_\infty$ *have to be given implicitly, and hence with constant encoding size, this requires an implicit representation of trajectories. As this is out of scope of this work we skip the technical details for this.*

**Remark 5.10.** *The construction of trajectories in the proofs of Proposition 5.2 and Theorem 5.8 do not need the fact that the grid is unbounded. Thus, the claims still hold true if the given grid or mesh completely contains all frames on which source vertices are placed.*

*5.2. Restricted movements and individual release dates*

In contrast to the previous subsection we now consider arbitrary release dates $r_i$ for each $1 \leq i \leq k$. Unfortunately, the so far developed construction rules for trajectories for instances with common release dates cannot easily be transferred. Further, not all instances are feasible. This can be seen, for example, by considering an instance where some commodity $i$ has release date $r_i = 0$ and at $t = 1$ there are commodities released from all neighbors of $s_i$, showing that an instance is not necessarily feasible.

Although the complexity of deciding feasibility for setting IR is open, we will give some sufficient conditions on the connection requests for an IR$|\mathsf{M}_{a,b}, 1|*$-NDTP instance to be feasible. Moreover, for instances satisfying these conditions we can construct optimal trajectories for all optimization objectives of Definition 2.3. The main idea is to assure the possibility for the commodities to simulate waiting by following small circles, such that they afterwards can follow an individual shortest path in the base network. Figure 6 shows an example setting of how a 3-cycle involving the source vertex $s_i$ can be used to achieve this waiting.




(a) One unit of delay: $s_i \to s_i^1 \to v_i$

(b) Two units of delay: $s_i \to s_i^2 \to s_i^1 \to v_i$

(c) Three units of delay: $s_i \to s_i^2 \to s_i^1 \to s_i \to v_i$

Figure 6: Delay patterns if a shortest path from $s_i$ to $d$ uses $s_i \to v_i$

If each commodity can use such a 3-cycle in order to be delayed, without causing conflicts with other commodities, we can construct optimal trajectories. In the Euclidean plane the convex hull of sources will provide a sufficient criterion to detect whether this can be done. We denote the convex hull of vectors $v_1, ..., v_\ell$ in the Euclidean plane by

$$\mathbf{conv}(v_1, ...., v_\ell) := \left\{ \sum_{j=1}^{\ell} \lambda_j v_j \mid \sum_{j=1}^{\ell} \lambda_j = 1, \lambda_j \in \mathbb{R}_{\geq 0} \right\}.$$

For any subset $S \subset \mathbb{R} \times \mathbb{R}$ of the Euclidean plane and any point $p$ in the Euclidean plane $\mathcal{D}(p, S) = \inf\{\|p - q\|_2 \mid q \in S\}$ denotes the distance between $p$ and $S$.

**Definition 5.11** (Non dominated requests)**.** *For a given IR$|\mathsf{M}_{a,b}, 1, 1|*$-NDTP instance with $k$ connection requests and common destination $d$ let the commodities be sorted increasingly according to non-decreasing earliest possible arrival times $t_i^d$ at $d$. In case of ties, we order them increasingly according to non-decreasing shortest path length between $s_i$ and $d$, for the mesh that is $\|s_i - d\|_\infty$. If ties persist, we break them arbitrarily.*

*We call the connection requests* non dominated *if*

1. *for each pair of requests $1 \leq i, j \leq k$ with $i \neq j$ it holds that $\|s_i - s_j\|_\infty \geq 3$,*
2. *for each $1 \leq i \leq k$, if $-\infty < j = \sup\{j < i \mid t_j^d < t_i^d\}$ it holds that*

$$3 \leq \mathcal{D}\left(s_i, \mathbf{conv}(s_1, s_2, \ldots, s_j, d)\right),$$

3. *every source has eight neighbors in $V_{a,b}$.*

**Remark 5.12.** *Let an IR$|\mathsf{M}_{a,b}, 1, 1|*$-NDTP instance with $k$ connection requests with common destination $d$ be given. The conditions for non dominated requests can be checked in polynomial time. This is clear for conditions 1. and 3.. For the check of condition 2. for each $1 \leq i \leq k$ the check whether or not $s_i$ is element of the convex hull can be done by finding a separating hyperplane. This can be done in polynomial time. If $s_i$ is not in the convex hull of the given points, one has to determine the convex hull of at most $k$ point in the Euclidean plane. This is described by at most $k$ points and $k$ segments and can be determined in time $\mathcal{O}(k \log(k))$. The squared distance to each of those points and segments can be computed in polynomial time.*

The conditions for non dominated requests ensure that no source is on the boundary. Thus, for each source we can select a 3-cycle as shown in Figure 6. As the distance between any two sources is at least three, no two of those 3-cycles will share a vertex. Condition 2. ensures that, if a commodity $j$ arrives before $i$, commodity $j$ never uses a vertex that is part of the delay pattern of commodity $i$ . This is detailed in the proof of Theorem 5.13.

**Theorem 5.13.** *Let an IR$|\mathsf{M}_{a,b}, 1|*$-NDTP instance with $k$ connection requests and common destination $d$ be given. If the requests are non dominated, there are feasible trajectories with objective values matching the lower bounds of Theorem 5.5. Thus, those trajectories are optimal for the considered optimization objectives.*

*Proof.* Assume that the commodities are sorted according to the definition of non dominated requests. For all commodities $1 \leq i \leq k$, we let Algorithm 4 compute $\Delta_i$.

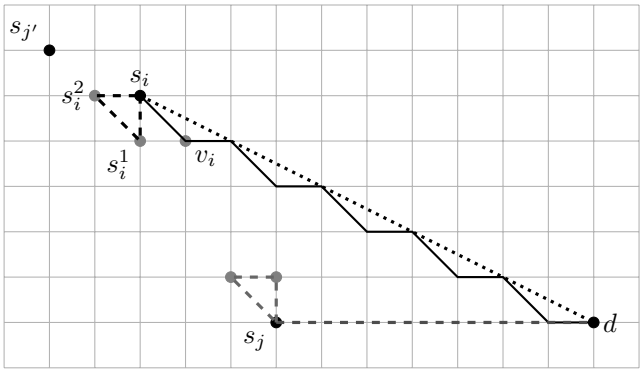



Figure 7: Shortest path selection according to Bresenhams algorithm and example layout for delay patterns

We now construct the trajectories in reverse order for $i = k, ..., 1$ as follows. For request $i$, we fix a shortest path from $s_i$ to $d$ in $\mathsf{M}_{a,b}$ such that for no vertex of this path the Euclidean distance to the segment $\overline{s_i d}$ is larger than one. Such a path can be computed by Bresenhams algorithm for the line segment between two integral points, see [1]. As this path only uses diagonal and *either* horizontal *or* vertical edges, it is indeed a shortest path from $s_i$ to $d$. We now delay commodity $i$ by $\Delta_i$ time units using appropriate $s_i^1, s_i^2$ vertices and delay patterns as given in Figure 6. To this end, let $v_i$ be the successor of $s_i$ in the path. We fix a possible 3-cycle $s_i, s_i^1, s_i^2$ for the delay patterns, such that $s_i^1$ is a neighbor of $v_i$ and $s_i^2$ is not a neighbor of $v_i$. Condition 3. of the definition for non dominated requests ensures that this is possible. For an illustration how the shortest path for $s_i$ can be selected, see Figure 7.

If a delay of more than three time units is assigned to a commodity, then we use a 3-cycle as often as needed. If the delay is not a multiple of three, then we use a cycle in the appropriate direction to continue with the remaining delay of one or two units. Afterwards, we follow the above selected shortest path to $d$. Thus, by denoting with $t_i^d$ the earliest possible arrival time for $i$ at $d$, the new arrival time is $\bar{t}_i^d = t_i^d + \Delta_i$. Thus, all commodities have arrival times that match the times computed by Algorithm 4. Hence, the objective values equal the lower bounds given in Theorem 5.5.

We now prove that these trajectories are disjoint. As arrival times are pairwise distinct and a shortest path to the common destination is used after the delaying is done, no conflicts can occur between commodities that already follow their shortest paths (otherwise the commodities would arrive at the same time).

It remains to show that by delaying the commodities, no conflict arises. Assume there is a conflict between two commodities $i < j$ and assume it occurs while $j$ is still following its delay pattern. We will show that this cannot happen by proving that the trajectory of commodity $i$ never uses any vertex in the base graph that is part of the delay pattern for $j$. All vertices in the delay pattern for any commodity have a shortest path length of exactly 1 to its source. As $\|s_i - s_j\|_\infty$ is at least three, no pair of delay patterns can share a vertex. It remains to show that a shortest path from $i$ to the destination does not use any vertex of the delay pattern of commodity $j$ . We consider two cases:

Case 1: Let $t_i^d < t_j^d$. Due to condition 2. of the definition for non dominated requests and the selected shortest path for $i$ the Euclidean distance between $s_j$ and any vertex of the shortest path

for $i$ is at least two. As vertices of the delay pattern have Euclidean distance to $s_j$ at most $\sqrt{2}$, which is smaller than two, the shortest path for $i$ and the delay pattern of $j$ cannot have common vertices in the base graph. Figure 7 illustrates this for some choice of $s_i$ and $s_j$.

Case 2: $t_i^d = t_j^d$. Due to the ordering given by the definition for non dominated requests $\|s_i - d\|_\infty \leq \|s_j - d\|_\infty$ holds. If the shortest path from $s_i$ to $d$ uses any neighbor $v$ of $s_j$ condition 1. in the definition for non dominated requests implies that

$$\|s_i - d\|_\infty \geq 2 + \|v - d\|_\infty \geq 1 + \|s_j - d\|_\infty > \|s_i - d\|_\infty$$

This is a contradiction and hence no shortest path for commodity $i$ can use any vertex of the delay pattern of $j$.

Thus, the conflict can only happen if commodity $i$ is still following a delay pattern and $j$ is already on a shortest path to the common destination $d$. In Figure 7 $s_i$ and $s_{j'}$ illustrate a possible layout. Due to the definition of disjointness for trajectories there are two possibilities for the conflict:

Case 1: Condition 1, that is, vertex disjointness in the time-expanded network, is violated. If the conflict occurs at $s_i$ there is no possibility that $j$ arrives after $i$ while using a shortest path, which is a contradiction. Thus, the conflict has to be at either $s_i^1$ or $s_i^2$ of the base network. Now, as $s_i^2$ is a neighbor of $s_i$, if there is a conflict at $s_i^2$, $j$ can only have a remaining distance to $d$ of at most $\|s_i - d\|_\infty + 1$. Further, $i$ is still using its delay pattern and from $s_i^2$ there are at least $\|s_i - d\|_\infty + 1$ many steps to go. This is a contradiction to $\bar{t}_i^d < \bar{t}_j^d$. If the conflict is at $s_i^1$, which is a neighbor of $v_i$, an analogous argument applies. After all, trajectories are vertex disjoint in the time-expanded network.

Case 2: Condition 2, that is using an edge of the base network in different directions is violated. Assume this happens for the edge between $s_i$ and $s_i^2$ at time $t$. Assume the case $i$ is at $s_i$ at time $t+1$ and $j$ at $s_i^2$. As at least one shortest path from $s_i$ to $d$ uses $v_i$ as its second vertex in the mesh one coordinate of $v_i$ is one unit closer to the coordinates of $d$ and the other at least not further away. Thus, it cannot be that any shortest path from $s_i$ to $d$ uses a vertex with Euclidean distance greater than one from $v_i$ as second vertex. As $s_i^2$ is not a neighbor of $v_i$ but $j$ following a shortest path that uses $s_i$ this leads to a contradiction. In the other case, that is $j$ is at $s_i$ in time $t+1$ and $i$ at $s_i^2$ the earliest possible arrival time for $i$ is after the arrival time for $j$ which is again a contradiction.

For the other edges similar arguments apply: Either $j$ cannot have used the edge in that direction while following a shortest path, or the earliest possible arrival time for $i$ is not before the latest possible arrival time of $j$. □

**Remark 5.14.** *The considerations so far are made for the mesh. They can be transferred to grids by changing condition 3. in the definition for non dominated requests to four neighbors for each source. Delay patterns have to be appropriate 4-cycles including both $s_i$ and the second vertex of a shortest path $v_i$ for each request $1 \leq i \leq k$. Such 4-cycles will allow for any even delay. Thus, the arrival times given by the adapted greedy algorithm for the grid can be met. The selection of a shortest path from any source to the common sink such that no vertex has distance greater than one from the segment can also be done by Bresenhams Algorithm. Thereby one has to make sure to replace any diagonal connection implied by Bresenhams Algorithm with an appropriate sequence of two orthogonal ones.*

## 6. The IR|$\mathsf{M}_{a,b}, c_e$|minsum-NDTP

Finally, we study the IR||$\mathsf{M}_{a,b}, c_e$|minsum-NDTP, where individual edge costs allow to depict environmental influences on the costs of trajectories, such as energy cost for a connection, terrain or wind for example. As the problem will turn out to be hard, a question that arises is whether we can find algorithms that have running times in $\mathcal{O}(f(k)|G|^{\mathcal{O}(1)})$ for some function $f : \mathbb{N} \to \mathbb{R}^+$, a base graph $G$ and $k$ connection requests. A problem is called fixed-parameter tractable if such an algorithm exists [8] . We will show that not even approximation for the IR|$\mathsf{M}_{a,b}, c_e$|minsum-NDTP is fixed-parameter tractable (unless $\mathcal{P} = \mathcal{NP}$).

In the following, we show that the IR|$\mathsf{M}_{a,b}, c_e$|minsum-NDTP is $\mathcal{W}[1]$-complete. $\mathcal{W}[1]$-hardness relies on a reduction to $k-$ CLIQUE which is widely believed not to be fixed-parameter tractable, see [8]. This will imply that approximation of IR|$\mathsf{M}_{a,b}, c_e$|minsum-NDTP is not fixed-parameter tractable and that IR|$\mathsf{M}_{a,b}, c_e$|minsum-NDTP is hence $\mathcal{NP}$-complete.

Our proof relies on a result by Slivkins [25] who shows that both the edge and vertex disjoint paths problem on general directed acyclic graphs are $\mathcal{W}[1]$-hard.

The remainder of this section is organized as follows: First, for any given directed acyclic graph $D = (N, A)$ we detail the construction of a corresponding mesh and non-negative edge costs. We further explain how to translate a connection request in $D$ to one in the mesh. Second we prove a one to one correspondence of disjoint trajectories in the time-expanded network (with bounded cost) to vertex disjoint paths in $D$. This implies $\mathcal{W}[1]$-hardness for optimization of IR|$\mathsf{M}_{a,b}, c_e$|minsum-NDTP (Theorem 6.5). Last we prove that approximation for IR|$\mathsf{M}_{a,b}, c_e$|minsum-NDTP is not fixed-parameter tractable (Theorem 6.6).

*Construction of the mesh:*

Let a directed acyclic graph $D = (N, A)$, with $n = |N|$ vertices be given together with a topological ordering of its vertices $v_1, .., v_n$ and non negative arc costs $c_{pq}$ for each arc $a = (v_p, v_q) \in A$.

We construct a polynomially sized (in the size of $D$) mesh where the vertices of $D$ are represented by the rows and information of the arcs can be deduced from the columns. Therefore, we first define the horizontal and vertical dimension of the mesh together with some notation and afterwards detail how an arc of $D$ is represented through the assignment of edge costs in the resulting mesh.

*Rows:* We use $2n$ rows labeled $y_1^+, y_1^-, ..., y_n^+, y_n^-$. Every two rows have a correspondence to one vertex of $D$.

*Columns:* Following the topological ordering of the vertices of $D$, for each vertex $v_p$ there is a so called gadget of $2n$ columns labeled $x_{p,n}^+, x_{p,n}^-, x_{p,n-1}^+, x_{p,n-1}^-, \ldots, x_{p,1}^+ . x_{p,1}^-$. The two columns labeled $x_{p,q}^+, x_{p,q}^-$ will be used to indicate whether there is an outgoing arc from $v_p$ to $v_q$.

*Size of the mesh:* Overall the number of vertices in the mesh we consider is

$$2n \cdot n \cdot 2n = \mathcal{O}(n^3).$$

As each vertex in a mesh has at most eight neighbors also the number of edges in the base network is polynomial in the size of the input graph.

**Remark 6.1.** *For ease of exposition we use an arrow notation to refer to edges in the above constructed mesh. That is, for a sequence of edges $\{v, w\}, \{w, z\}$ we write $v \to w \to z$ for short.*

*Representation of arcs:* A column labeled $x_{p,q}^+$ will carry the information whether there is an arc in $D$ connecting vertices $(v_p, v_q)$. Therefore, each arc in $D$ is represented by a sequence of edges which in total will have the same cost as $c_{pq}$. Unspecified edges will be assigned a high constant

cost $C$ ($> \sum_{a \in A} c_a$) such that it is immediately clear from the cost of a trajectory on the mesh, whether or not it can be translated to a path in $D$. For an illustrative example to keep track of the described sequences see Figure 8. Dashed lines represent the outgoing arcs from a considered vertex, while the line consisting of small triangles represents a vertex that could have possibly been reached by a lower indexed vertex.

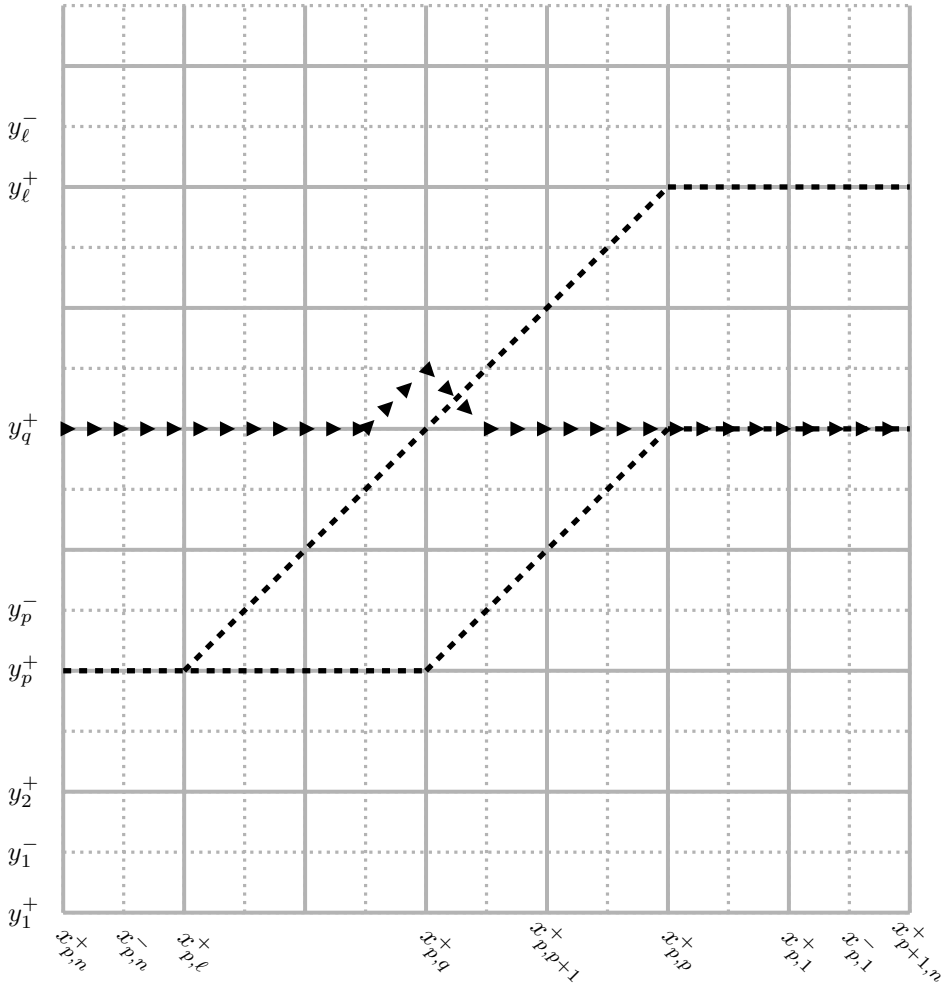



Figure 8: Example gadget for vertex $v_p$, where the original directed acyclic graph had arcs $(v_p, v_\ell), (v_p, v_q)$ and only vertex $v_q$ has an in going arc from any lower indexed vertex.

For $p = 1, ..., n$ consider $v_p \in N$. An arc $(v_p, v_q) \in A$ is represented in the following way:

Consider the gadget corresponding to $v_p$, that is, the columns labeled $x^*_{p,*}$. Set edge cost to zero (where not already done) for:

$$(x^+_{p,n}, y^+_p) \to (x^-_{p,n}, y^+_p) \to (x^+_{p,n-1}, y^+_p) \to \cdots \to (x^-_{p,q+1}, y^+_p) \to (x^+_{p,q}, y^+_p).$$

Now go 'diagonal right up' in the following manner:

$$(x^+_{p,q}, y^+_p) \to (x^-_{p,q}, y^-_p) \to (x^+_{p,q-1}, y^+_{p+1}) \to \cdots \to (x^-_{p,p+1}, y^-_{q-1}).$$

with cost zero. Finally set the edge cost of $(x^-_{p,p+1}, y^-_{q-1}) \to (x^+_{p,p}, y^+_q)$ to $c_{pq}$.

Now, starting from vertex $(x^+_{p,p}, y^+_q)$ to the right set edge costs to 0 in row $y_q$ until the beginning of gadget $p+1$, i.e column $x^+_{p+1,n}$ is reached. For the gadgets between $v_p$ and $v_q$ establish zero cost sequences on row $y^+_q$, deviating to row $y^-_q$ if necessary. How this has to be done in general is detailed for one gadget in the following.

Consider the gadget corresponding to a vertex $v_p$. The sequences that were introduced to represent outgoing arcs of $v_p$ in $D$, cross rows which correspond to other vertices of $D$. It has to be assured that, if for a vertex of $D$ with lower index than $p$ that has an arc to some vertex $v_q$ with $q > p$ no zero cost path on rows $y^+_q$, resp. $y^-_q$ can result in conflicts for trajectories.

For all vertices $v_q$ with $q > p$ that can possibly be reached from vertices with an index smaller than $p$ start a sequence of zero cost edges on row $y^+_q$ at column $x^+_{p,n}$. Continue those zero cost edges on row $y^+_q$ for the whole gadget corresponding to $v_p$, ending in columns $x^+_{p+1,n}$. This zero cost sequence only deviates from row $y^+_q$ and uses a vertex of row $y^-_q$ if possible 'conflicts' have to be avoided. That is, conflicts with diagonal connections in the mesh that represent arcs $(v_p, v_\ell)$ in $D$ with $\ell > q$ have to be avoided. This is done by a sequence of two edges each with cost zero as follows: $(x^-_{p,(\ell-(q-p)+1}, v^+_q) \to (x^+_{p,(\ell-(q-p)}, v^-_q) \to (x^-_{p,(\ell-(q-p)}, v^+_q)$.

This zig-zag sequence is the reason why we need two rows and columns per vertex.

Note: If a vertex $v_q$ can be reached from a lower indexed vertex and $v_p$, no conflict avoidance is necessary. If $v_p$ can be reached from a lower indexed vertex, there is a zero cost path in gadget $v_{p-1}$ that ends at $(x^+_{p,n}, y^+_p)$ that is continued as described above for the outgoing arcs of $v_p$.

All edges that are not specified in this mesh by now are assigned some high constant cost value $C$.

As direct returns are not allowed in setting IR this assures that if costs lower than $C$, for $C > \sum_{a \in A} c_a$, are assigned to a trajectory, the corresponding commodity in each time step moves one column to the right.

*Representation of vertices:* We secure that the usage of vertex labeled $(x^+_{p,n}, y^+_p)$ in the mesh represents usage of $v_p$ in $D$. That is, a trajectory in the mesh that corresponds to a path in $D$ uses $(x^+_{p,n}, y^+_p)$ if $v_p$ is used in $D$.

**Definition 6.2.** *For a given directed acyclic graph $D$ with arc costs $c_{pq}$ and a value $C \in \mathbb{N}$ we denote the mesh as constructed above as:*

$$\mathsf{M}(D, c_{pq}, C).$$

Connection requests: *A connection request between a terminal pair $\bar{s}_i = v_p, \bar{d}_i = v_q$ in $D$ is translated into a connection request $s_i, d_i$ in $\mathsf{M}(D, c_{pq}, L)$ with release date $r_i$ as follows:*

$$s_i = \left(x^+_{p,n}, y^+_p\right), \qquad d_i = \left(x^+_{q,n}, y^+_q\right), \qquad r_i = (p-1)(2n).$$

We note that when enumerating the columns of the mesh starting with 0 for column labeled $x^+_{1,n}$, then $r_i$ coincides with the number of the column labeled $x^+_{p,n}$. If all commodities move one column to the right at each time step, then all of them are in the same column at the same time.

**Lemma 6.3.** *Let a directed acyclic graph $D = (N, A)$ with $k$ connection requests between source sink pairs $(\bar{s}_i, \bar{d}_i)$ be given. For each $1 \le i \le k$ there exists a path in $D$ fulfilling request $i$ with cost*

*$C_i^*$ such that the $k$ paths are vertex disjoint if and only if for $\mathsf{M}(D, c_{pq}, C)$ with $C > \sum_{a \in A} c_a$, and connection requests according to the definition above, there are feasible trajectories that follow the restrictions of setting IR and for each $1 \leq i \leq k$ have costs $C_i^* < C$.*

*Proof.* ($\Rightarrow$): First we show how to transform a path $\mathcal{P}$ in $D$ into a trajectory in the mesh with equal costs. We illustrate this by a minimal example path $\mathcal{P} = (v_p, v_q)$ in $D$. The transformation will imply that the resulting trajectories fulfill the corresponding requests in the mesh and are disjoint, if the corresponding paths $\mathcal{P}_1, ..., \mathcal{P}_k$ in $D$ are vertex disjoint and fulfill the requests.

*Transformation of $\mathcal{P} = (v_p, v_q)$:* We start at vertex $\left(x^+_{p,n}, y^+_p\right)$ at time $(p-1)(2n)$. Now we move horizontally right until arriving at the column labeled $x^+_{p,q}$. This has cost zero as there is an arc $(v_p, v_q)$. We follow the path right diagonal up till reaching row $y_q$. Again this segment has cost zero, except for one edge with cost $c(v_p, v_q)$. Now, we follow the zero cost path to the right until the end of the gadget. Next, for the gadgets between $v_p$ and $v_q$ we follow the zero cost path on row $y_q$ ($y'_q$ where necessary) until the first column of gadget $v_q$ is reached. That is at vertex $(x^+_{q,n}, y_q)$. In this way, the trajectory cost in $\mathsf{M}(D, c_{pq}, C)^T$ corresponding to path $\mathcal{P}$ is equal to the cost in $D$.

*Disjointness:* According to the transformation above, each step the trajectories proceed in the mesh moves one column to the right. Hence, all commodities are always in the column number equal to the time step. Thus, if trajectories use vertices of the same column of the mesh they do this in the same time step. If a vertex of a column in the base mesh would be used by two different walks, it is implied by the cost of the corresponding trajectories, that they also use a common edge (the zig-zag sequences as depicted by the line of triangles in Figure 8 avoid common vertices otherwise). This cannot occur, as it implies that the paths in $D$ have common vertices. Hence, in the base mesh the trajectories are vertex disjoint and hence the trajectories are disjoint.

($\Leftarrow$): If such disjoint trajectories exist, we know from their cost, that only edges with cost lower than $C$ in $\mathsf{M}(D, c_{pq}, C)^T$ are used. As no direct return is allowed and the destination is right of the source for each request, this implies that each commodity moves one column to the right in each time step.

This allows us to deduce vertex disjoint paths in $D$ by following the trajectories from commodity $i$ in $\mathsf{M}(D, c_{pq}, C)^T$. We can identify a path connecting $\bar{s}_i, \bar{d}_i$ in $D$ in the following way. The first vertex of $D$ is $\bar{s}_i$ according to the definition of $s_i$. After the source vertex, the trajectory of commodity $i$ implies to use vertex $v_\ell$ of $D$ if the trajectory uses $(x^+_{\ell,n}, y^+_\ell)$. This can only be at time $(\ell-1)(2n)$ as each commodity always moves one step right in each time step and release dates are set accordingly. The resulting sequence of vertices in $D$ are a path, as two subsequent vertices $v_q, v_\ell$ of the path can only be implied by a trajectory if it went from row $y_q$ to row $y_\ell$. This is only possible with cost smaller than $L$ if there is an arc $(v_q, v_\ell)$ in $D$. This also implies that the cost of the trajectories are equal to the corresponding path in $D$.

Vertex disjointness of the paths in $D$ follows as the trajectories are vertex disjoint in the time-expanded mesh and hence at most one trajectory could have used a vertex $(x^+_{\ell,n}, y^+_\ell)$ of the mesh. □

**Remark 6.4.** *Lemma 6.3 does not apply if a* makespan*-objective is considered, as for the construction of $\mathsf{M}(D, c_{pq}, C)$ non-unitary edge costs are necessary. Thus, the costs do not represent the time consumption which would be necessary for the* makespan*-objective.*

**Theorem 6.5.** *IR$|\mathsf{M}_{a,b}, c_e|$minsum-NDTP and IR$|\mathsf{M}_{a,b}, c_e|$minmax-NDTP are $\mathcal{W}[1]$-hard.*

*Proof.* This is a direct consequence of Lemma 6.3: For every given instance of the vertex disjoint paths problem on a directed acyclic graph $D{=}\ (N, A)$ the construction of $\mathsf{M}(D, c_{pq}, \sum_{a \in A} c_a)$ for is polynomial in the size of $D$. Thus, as determining existence of disjoint paths is $\mathcal{W}[1]$-hard

for directed acyclic graphs according to [25] if follows from Lemma 6.3 that also optimizing for IR|$\mathsf{M}_{a,b}, c_e$|minsum-NDTP and IR|$\mathsf{M}_{a,b}, c_e$|minmax-NDTP is also $\mathcal{W}[1]$-hard: If trajectories with costs smaller than $\sum_{a \in A} c_a$ are found, existence of vertex disjoint paths in $D$ is implied, and non-existence otherwise. □

**Theorem 6.6.** *Under the assumption that $\mathcal{W}[1]$-hardness implies non fixed-parameter tractability [8]: $\alpha$-approximation for the IR|$\mathsf{M}_{a,b}, c_e$|minsum-NDTP and IR|$\mathsf{M}_{a,b}, c_e$|minmax-NDTP are not fixed-parameter tractable for any $\alpha \geq 1$.*

*Proof.* We prove the statement for IR|$\mathsf{M}_{a,b}, c_e$|minsum - NDTP. The statement for IR|$\mathsf{M}_{a,b}, c_e$|minmax-NDTP follows analogously.

Assume a fixed-parameter tractable approximation algorithm for the IR|$\mathsf{M}_{a,b}, c_e$|minsum-NDTP exists. This implies that for any directed acyclic graph $D$ we can use this algorithm and the construction of an polynomial sized (in $|D|$) mesh corresponding to $D$ as given by Definition 6.2 to determine existence of vertex disjoint paths in $D$ according to Lemma 6.3.

Therefore, assume arc costs in $D$ to be zero, and hence for the corresponding IR|$\mathsf{M}_{a,b}, c_e$|minsum - NDTP instance an optimal solution also has cost zero (if the vertex disjoint path exist in $D$). In order to be an $\alpha$ approximation algorithm the algorithm thus has to yield cost zero. If in the mesh construction $C > 0$ holds, no edge of cost $C$ can be used and thus the approximation algorithm leads to an optimal solution with cost zero. In this case existence of vertex disjoint paths in $D$ follows. Further if the algorithm returns a solution with cost at least $C$ we know that no vertex disjoint paths between the desired pairs exist in $D$. This contradicts the result of $\mathcal{W}[1]$-hardness for the existence of vertex disjoint paths by [25]. □

## 7. Conclusion

In this work we studied the non-stop disjoint trajectories problem for the path, the grid and the mesh as networks. We established two characteristics on the turning behavior of commodities pointing towards practical applications such as air traffic management. For this new and wide problem class we presented different polynomial time algorithms providing feasible solutions in all studied settings with common release dates for the path. For the mesh and connection requests with common destination, we detailed how to construct optimal trajectories for different objectives. Key ingredient of the proof is a greedy assignment of arrival times. Further, we proved two problem versions, to be $\mathcal{NP}$-complete. Both include individual release information and restricted turning abilities of commodities. First, for a path as base network where requests have release intervals we showed hardness by a reduction to 3-SAT. Second, on the mesh with arbitrary edge cost we showed that even approximation is not fixed-parameter tractable.

Future research efforts shall tackle the complexity of deciding feasibility and optimization of general instances on the unit grid with individual release dates for connection requests. This remains open although we can identify classes of feasible instances which we can even solve optimally under the considered objectives. Also extensions to grids and meshes with more than two dimensions and non unit time consumption are of interest. Depending on the application for practical use different disjointness measures as well as further restrictions on movements may be of interest.

## ACKNOWLEDGMENTS

Funding: This research has been conducted in the framework of the research project HOTRUN, financed by the German Federal Ministry of Economic Affairs and Energy (BMWi) under grant

20E1720B. This research was partly funded by the Deutsche Akademische Austauschdienst (DAAD) with means from the Bundesministerium für Bildung und Forschung (BMBF) through a Gastdozentur at the Chair of EDOM in the Department of Mathematics of the Friedrich-Alexander Universität Erlangen-Nürnberg (Projekt-ID 57425212), while one of the authors was enjoying a sabbatical leave from Universidad Autónoma Metropolitana Azcapotzalco. This author also acknowledges the support of the National Council of Science and Technology (Conacyt) and its National System of Researchers (SNI).

The authors thank Lukas Glomb and Florian Rösel for fruitful discussions on the topic and for proof-reading early parts of this work.

## References

[1] Jack E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965. doi: 10.1147/sj.41.0025. URL `https://10.1147/sj.41.0025`.

[2] Costas Busch, Malik Magdon-Ismail, Marios Mavronicolas, and Paul Spirakis. Direct routing: algorithms and complexity. *Algorithmica*, 45(1):45–68, 2006. ISSN 0178-4617. doi: 10.1007/s00453-005-1189-3. URL `https://doi.org/10.1007/s00453-005-1189-3`.

[3] Andreas Bärmann, Thorsten Gellermann, Maximilian Merkert, and Oskar Schneider. Staircase compatibility and its applications in scheduling and piecewise linearization. *Discrete Optimization*, 29:111–132, 2018. ISSN 1572-5286. doi: https://doi.org/10.1016/j.disopt.2018.04.001. URL `https://www.sciencedirect.com/science/article/pii/S1572528618300306`.

[4] Andreas Bärmann, Patrick Gemander, and Maximilian Merkert. The clique problem with multiple-choice constraints under a cycle-free dependency graph. *Discrete Applied Mathematics*, 283:59–77, 2020. ISSN 0166-218X. doi: https://doi.org/10.1016/j.dam.2019.12.015. URL `https://www.sciencedirect.com/science/article/pii/S0166218X1930558X`.

[5] Valentina Cacchiani, Alberto Caprara, and Paolo Toth. Non-cyclic train timetabling and comparability graphs. *Operations Research Letters*, 38(3):179–184, 2010. ISSN 0167-6377. doi: https://doi.org/10.1016/j.orl.2010.01.007. URL `https://www.sciencedirect.com/science/article/pii/S0167637710000088`.

[6] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158, New York, NY, USA, 1971. Association for Computing Machinery. ISBN 9781450374644. doi: 10.1145/800157.805047. URL `https://doi.org/10.1145/800157.805047`.

[7] Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Henk Meijer, and Christian Scheffer. Coordinated motion planning: reconfiguring a swarm of labeled robots with bounded stretch. *SIAM J. Comput.*, 48(6):1727–1762, 2019. ISSN 0097-5397. doi: 10.1137/18M1194341. URL `https://doi.org/10.1137/18M1194341`.

[8] Rodney. G. Downey and Michael R. Fellows. *Parameterized complexity*. Monographs in Computer Science. Springer-Verlag, New York, 1999. ISBN 0-387-94883-X. doi: 10.1007/978-1-4612-0515-9. URL `https://doi.org/10.1007/978-1-4612-0515-9`.

[9] Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoret. Comput. Sci.*, 10(2):111–121, 1980. ISSN 0304-3975. doi: 10.1016/0304-3975(80)90009-2. URL https://doi.org/10.1016/0304-3975(80)90009-2.

[10] Corinna Gottschalk, Arie M. C. A. Koster, Frauke Liers, Britta Peis, Daniel Schmand, and Andreas Wierz. Robust flows over time: models and complexity results. *Math. Program.*, 171(1-2, Ser. A):55–85, 2018. ISSN 0025-5610. doi: 10.1007/s10107-017-1170-3. URL https://doi.org/10.1007/s10107-017-1170-3.

[11] Alex Hall, Steffen Hippler, and Martin Skutella. Multicommodity flows over time: efficient algorithms and complexity. *Theoret. Comput. Sci.*, 379(3):387–404, 2007. ISSN 0304-3975. doi: 10.1016/j.tcs.2007.02.046. URL https://doi.org/10.1016/j.tcs.2007.02.046.

[12] Andreas Heidt, Hartmut Helmke, Manu Kapolke, Frauke Liers, and Alexander Martin. Robust runway scheduling under uncertain conditions. *Journal of Air Transport Management*, 56, 03 2016. doi: 10.1016/j.jairtraman.2016.02.009. URL https://doi.org/10.1016/j.jairtraman.2016.02.009.

[13] Bruce Hoppe. Efficient dynamic network flow algorithms. Technical report, Cornell University, 1995.

[14] R. M. Karp. On the computational complexity of combinatorial problems. *Networks*, 5(1):45–68, 1975. doi: https://doi.org/10.1002/net.1975.5.1.45. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/net.1975.5.1.45.

[15] Yusuke Kobayashi and Christian Sommer. On shortest disjoint paths in planar graphs. *Discrete Optim.*, 7(4):234–245, 2010. ISSN 1572-5286. doi: 10.1016/j.disopt.2010.05.002. URL https://doi.org/10.1016/j.disopt.2010.05.002.

[16] Richard E. Korf, Michael D. Moffitt, and Martha E. Pollack. Optimal rectangle packing. *Ann. Oper. Res.*, 179:261–295, 2010. ISSN 0254-5330. doi: 10.1007/s10479-008-0463-6. URL https://doi.org/10.1007/s10479-008-0463-6.

[17] J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. In *Studies in integer programming (Proc. Workshop, Bonn, 1975)*, pages 343–362. Ann. of Discrete Math., Vol. 1, 1977.

[18] Frauke Liers and Maximilian Merkert. Structural investigation of piecewise linearized network flow problems. *SIAM J. Optim.*, 26(4):2863–2886, 2016. ISSN 1052-6234. doi: 10.1137/15M1006751. URL https://doi.org/10.1137/15M1006751.

[19] James F. Lynch. The equivalence of theorem proving and the interconnection problem. *SIGDA Newsl.*, 5(3):31–36, sep 1975. ISSN 0163-5743. doi: 10.1145/1061425.1061430. URL https://doi.org/10.1145/1061425.1061430.

[20] Thomas Moscibroda and Onur Mutlu. A case for bufferless routing in on-chip networks. *SIGARCH Comput. Archit. News*, 37(3):196–207, jun 2009. ISSN 0163-5964. doi: 10.1145/1555815.1555781. URL https://doi.org/10.1145/1555815.1555781.

[21] Britta Peis, Martin Skutella, and Andreas Wiese. Packet routing on the grid. In *Latin American Symposium on Theoretical Informatics*, pages 120–130. Springer, 2010. ISBN 978-3-642-12200-2.

[22] Neil Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Combin. Theory Ser. B*, 63(1):65–110, 1995. ISSN 0095-8956. doi: 10.1006/jctb.1995.1006. URL `https://doi.org/10.1006/jctb.1995.1006`.

[23] Hans Röck. Some new results in flow shop scheduling. *Z. Oper. Res. Ser. A-B*, 28(1):1–16, 1984. ISSN 0340-9422. doi: 10.1007/BF01919082. URL `https://doi.org/10.1007/BF01919082`.

[24] Martin Skutella. An introduction to network flows over time. In *Research trends in combinatorial optimization*, pages 451–482. Springer, Berlin, 2009. doi: 10.1007/978-3-540-76796-1_21. URL `https://doi.org/10.1007/978-3-540-76796-1_21`.

[25] Aleksandrs Slivkins. Parameterized tractability of edge-disjoint paths on directed acyclic graphs. *SIAM J. Discrete Math.*, 24(1):146–157, 2010. ISSN 0895-4801. doi: 10.1137/070697781. URL `https://doi.org/10.1137/070697781`.

[26] Chelliah Sriskandarajah and Pierre Ladet. Some no-wait shops scheduling problems: complexity aspect. *European J. Oper. Res.*, 24(3):424–438, 1986. ISSN 0377-2217. doi: 10.1016/0377-2217(86)90036-6. URL `https://doi.org/10.1016/0377-2217(86)90036-6`.

[27] Jur van den Berg, Jack Snoeyink, Ming Lin, and Dinesh Manocha. Centralized path planning for multiple robots: Optimal decoupling into sequential plans. In *Robotics: Science and Systems V, Washington*, 06 2009. doi: 10.15607/RSS.2009.V.018. URL `https://doi.org/10.15607/RSS.2009.V.018`.

[28] Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1–24, 2015. ISSN 0004-3702. doi: 10.1016/j.artint.2014.11.001. URL `https://doi.org/10.1016/j.artint.2014.11.001`.