The Complexity of Object Association in Multiple Object Tracking

Robert Ganian[1], **Thekla Hamm**[1], **Sebastian Ordyniak**[2]
[1]Algorithms and Complexity Group, TU Wien, Vienna, Austria
[2]University of Leeds, School of Computing, Leeds, UK
{rganian,thekla.hamm,sordyniak}@gmail.com

Abstract

Object association, i.e., the identification of which observations correspond to the same object, is a central task for the area of multiple object tracking. Two prominent models capturing this task have been introduced in the literature: the Lifted Multicut model and the more recent Lifted Paths model. Here, we carry out a detailed complexity-theoretic study of the problems arising from these two models that is aimed at complementing previous empirical work on object association. We obtain a comprehensive complexity map for both models that takes into account natural restrictions to instances such as possible bounds on the number of frames, number of tracked objects and branching degree, as well as less explicit structural restrictions such as having bounded treewidth. Our results include new fixed-parameter and XP algorithms for the problems as well as hardness proofs which altogether indicate that the Lifted Paths problem exhibits a more favorable complexity behavior than Lifted Multicut.

Introduction

Multiple Object Tracking (MOT) is an important area of research in computer vision and artificial intelligence (Milan et al. 2017; Chu et al. 2020; Huang and Zhou 2019) dating back to 1988 (Pylyshyn and Storm 1988). The tracking-bydetection paradigm allows us to decompose MOT into two tasks. First, an object detector is used on each frame of a video sequence in order to find the putative locations of all objects appearing in the video. Then, in the object association task, we remove false positive detections and associate correct detections to the corresponding identities, resulting in trajectories for individual objects. The focus of this paper lies on the latter of the two tasks.

A popular approach for the object association task in MOT (also called the image decomposition problem (Keuper et al. 2015; Arbelaez et al. 2011)) is to model it as a multicut problem on a graph representation of the instance (Tang et al. 2015, 2016; Keuper et al. 2020; Kumar, Charpiat, and Thonnat 2014). There, one builds a graph G which contains a vertex for each detection in each time frame, and whenever two detections (in different time frames) could represent consecutive appearances of the same object, an edge is added connecting these two detections. The edges are equipped with weights that represent the likelihood that two detections are of the same object, and the aim is to partition G into connected components while minimizing the sum of the weights of removed edges; the intuition is that each connected component corresponds to one object. This multicut model was subsequently extended with special "lifted" edges in order to encode long-range interactions, i.e., auxiliary information about the (dis-)similarity of non-consecutive detections (Tang et al. 2017; Babaee, Li, and Rigoll 2019; Horňáková, Lange, and Andres 2017); the distinction between normal and lifted edges is that connected components are defined only using the former[1].

A closely related approach to object association models the problem not via multicut, but via disjoint paths (often considered in the context of network flows) (Zhang, Li, and Nevatia 2008; Berclaz et al. 2011; Chari et al. 2015; Hofmann, Wolf, and Rigoll 2013). In particular, the representation used here is a directed graph G' obtained in a very similar manner as the graph model for multicut, with the main difference being the use of arcs which are directed from earlier to later frames and the addition of two special vertices: a universal source s and a universal sink t. The problem one needs to solve on G' is to find a maximum-weight set of internally vertex-disjoint s-t paths, where each such path will correspond to the track of one object. Very recently, Horňáková et al. (2020) extended the disjoint paths model via lifted arcs which—as in the multicut model—encode auxiliary information about detections that are not consecutive. There, the authors also argued that the lifted disjoint paths model had advantages over the lifted multicut model; for instance, the lifted multicut model does not encode information about individual frames, and hence the fact that an object only appears once per frame needs to be enforced separately (Horňáková et al. 2020).

Aim. Unlike in previous works, here we initiate a deeper study of LIFTED MULTICUT and LIFTED PATHS from a complexity-theoretic point of view. It is known that both problems are NP-complete in their full generality, but what if we are dealing with instances where, e.g., the number of objects we are interested in is small? Or, does the problem become tractable when the number of frames is significantly smaller than the number of detections? Do the answers to

[1]Formal definitions are provided in the Preliminaries.

these questions depend on whether the input graphs have natural properties? The theoretical *parameterized complexity* paradigm (Downey and Fellows 2013; Cygan et al. 2015) offers exactly the tools needed to provide detailed answers to such questions, and yet almost nothing is known about the parameterized complexity of the two problems of interest—contrasting recent advances in our understanding of the parameterized complexity of fundamental problems in many other subfields of AI (Ganian et al. 2020; Bredereck et al. 2020; Pfandler et al. 2015; Ordyniak and Szeider 2013; Ganian and Ordyniak 2018). The primary aim of this paper is to remedy this situation.

The cornerstone of parameterized complexity is the idea of analyzing the difficulty of problems not only when measured in terms of the input size, but also under the assumption that one or several numerical *parameters* of instances are small: in particular, if we consider k to be the sum of all designated parameters and n to be the input size, then we distinguish whether an NP-complete problem P:

- can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some function f (in which case it is *fixed-parameter tractable* or FPT), or
- can be solved in time $n^{f(k)}$ for a function f but is believed not to be FPT (in which case it is XP i.e., solvable in polynomial-time for constant parameter values, but W[1]-hard), or
- remains NP-hard even when k is fixed to be a constant (in which case it is *para*NP-*hard*).

Contributions. Our main contribution is a comprehensive complexity classification of LIFTED PATHS and LIFTED MULTICUT with respect to all combinations of five natural kinds of restrictions on instances—each captured by a numerical value that can be either considered as a parameter, or as a fixed constant. The first three of these values are directly tied to the MOT instance:

1. the number of frames (r) for LIFTED PATHS[2],
2. the number of objects we are interested in, i.e., the number of paths (p), and
3. the maximum "branching" degree of an observation, which is the number of immediate predecessors and successors of an observation (d).[3]

The remaining two restrictions are tied to a graph parameter called *treewidth* (Robertson and Seymour 1986), which intuitively measures how tree-like a graph is. Aside from being an extremely well-studied and versatile parameter that has been shown to be bounded in various graph models (e.g., in control flow graphs (Thorup 1998)), it is also a measure of a graph's sparsity that has also been shown to have interesting algorithmic properties for numerous AI problems (Peters 2016; Eiben et al. 2020; Ganian and Ordyniak 2018; Ganian, Ordyniak, and Ramanujan 2017). In MOT, it is reasonable to expect that some instances will result in bounded-treewidth graphs at least as far as the "base graph" (i.e., the graph containing only the non-lifted edges) is concerned—for instance, this is true whenever the number of observations per frame and the number of time frames between two consecutive detections of any object are bounded. Hence our fourth parameter is:

4. the treewidth of the base graph (tw).

And as our fifth and final parameter, we consider:

5. the treewidth of the *combined graph*, i.e., the graph containing both kinds of edges (tw*).

Our results for LIFTED PATHS are summarized in Figure 1. Observe that while parameterizing by tw* is sufficient to achieve fixed-parameter tractability, this is a much stronger restriction than parameterizing by tw and, in a sense, "penalizes" users for adding auxiliary information via lifted edges. On the other hand, the runtime of all our other algorithms is virtually oblivious to the presence and structure of such lifted edges—in particular, there adding more information via lifted edges will never make a class of tractable instances intractable.

The highlights of our technical contributions to the complexity map for LIFTED PATHS are:

- a fixed-parameter algorithm w.r.t. p, r, d (Theorem 2, obtained via an algorithmic technique called color coding),
- an algorithm (Theorem 4) which not only establishes fixed-parameter tractability w.r.t. tw, r, d and XP-tractability w.r.t. tw, r, but also implies XP-tractability w.r.t. a well-studied graph parameter called *treedepth* (Nesetril and de Mendez 2012),
- an involved W[1]-hardness reduction for the parameterization by tw when r is fixed to a constant (Theorem 10), which provides a precise understanding of the complexity in the yellow bubble of Figure 1.

We then turn our attention to LIFTED MULTICUT. While the problem exhibits a similar behavior to LIFTED PATHS as

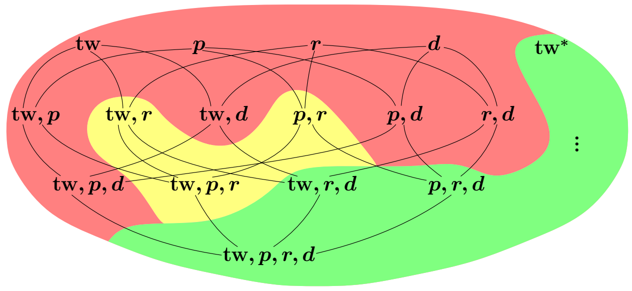


Figure 1: Overview of our complexity results for LIFTED PATHS with respect to all combinations of considered parameters. Red (top bubble) marks paraNP-hardness, yellow (middle bubble) marks XP-time solvability with lower bounds excluding fixed-parameter tractability, and green (bottom bubble) marks fixed-parameter tractability. Our results also include precise lower bounds for the yellow bubble that distinguish between parameters and constants.

[2]Frames do not immediately translate to a meaningful notion in the LIFTED MULTICUT model as it is used in the literature.

[3]A fourth natural option would be to consider the number o of detections per frame, but this is not a useful restriction. In particular, if r is a parameter then parameterizing by o trivializes the instance, and otherwise one can transform any instance into an equivalent one such that o is bounded by a constant.

far as tw* is concerned, we show that it is significantly more difficult to achieve tractability via restrictions to the base graph. In particular, we resolve the problem's complexity for all other parameterizations by developing a non-trivial reduction which establishes paraNP-completeness even on extremely simple base graphs and when the aim is to track only two objects.

Related Work. Other problems have also been used to model the object association task in MOT, such as CLIQUE (Zamir, Dehghan, and Shah 2012; Dehghan, Assari, and Shah 2015), INDEPENDENT SET (Brendel, Amer, and Todorovic 2011), MULTIGRAPH MATCHING (Hu et al. 2020) and INTEGER QUADRATIC PROGRAMMING (Henschel et al. 2018). In contrast to the two models considered here, the above examples often offer only limited options for integrating long-range interactions (Horňáková et al. 2020).

Preliminaries

For an integer i, we let $[i] = \{1, 2, \ldots, i\}$ and $[i]_0 = [i] \cup \{0\}$. We refer to the handbook by Diestel (2012) for standard graph terminology, and to the more recent books for a basic overview of parameterized complexity theory (Downey and Fellows 2013; Cygan et al. 2015). A *universal source* in a digraph is a vertex with no incoming arcs but outgoing arcs to all other vertices. Analogously, a *universal sink* in a digraph is a vertex with no outgoing arcs but incoming arcs from all other vertices.

Treewidth. A *nice tree-decomposition* $\mathcal{T}$ of a graph $G = (V, E)$ is a pair (T, χ), where T is a tree (whose vertices we call *nodes*) rooted at a node r and χ is a function that assigns each node t a set $\chi(t) \subseteq V$ such that the following holds:

- For every $uv \in E$ there is a node t such that $u, v \in \chi(t)$.
- For every vertex $v \in V$, the set of all nodes t satisfying $v \in \chi(t)$ forms a subtree of T.
- $|\chi(\ell)| = 1$ for every leaf ℓ of T and $|\chi(r)| = 0$.
- There are only three kinds of non-leaf nodes in T:

  Introduce node: a node t with exactly one child t' such that $\chi(t) = \chi(t') \cup \{v\}$ for some vertex $v \notin \chi(t')$.
  Forget node: a node t with exactly one child t' such that $\chi(t) = \chi(t') \setminus \{v\}$ for some vertex $v \in \chi(t')$.
  Join node: a node t with two children t_1, t_2 such that $\chi(t) = \chi(t_1) = \chi(t_2)$.

The *width* of a nice tree-decomposition (T, χ) is the size of a largest set $\chi(t)$ minus 1, and the *treewidth* of the graph G, denoted tw(G), is the minimum width of a nice tree-decomposition of G. Efficient fixed-parameter algorithms are known for computing a nice tree-decomposition of near-optimal width (Bodlaender et al. 2016; Kloks 1994). Whenever we speak of the treewidth of a directed graph, we mean the treewidth of its underlying undirected graph, which is the simple graph obtained by replacing each arc by an undirected edge.

We let T_t denote the subtree of T rooted at a node t, and we use $\chi(T_t)$ to denote the set $\bigcup_{t' \in V(T_t)} \chi(t')$.

Problem Definitions. The formal definitions of our problems follow those given in the literature (Horňáková et al. 2020; Tang et al. 2017; Babaee, Li, and Rigoll 2019; Horňáková, Lange, and Andres 2017). An instance of LIFTED PATHS is a tuple $\mathcal{I} = (G, H, w, \ell, p)$ where:

- $G = (V, E)$ is an n-vertex acyclic digraph with V partitioned into:
  - a universal source $V_0 = \{s\}$ and sink $V_{r+1} = \{t\}$, and
  - *(time) frames* $V_1, \ldots, V_r$,

  such that each arc $ab \in E$ where $a \in V_i$ and $b \in V_j$ satisfies $i < j$.
- $H = (V \setminus \{s, t\}, F)$ is an acyclic digraph,
- $w_G : V \cup E \to \mathbb{Z}$ and $w_H : F \to \mathbb{Z}$ are weight functions such that $w_G(s) = w_G(t) = 0$,
- $\ell \in \mathbb{Z}$ is a target lower bound, and
- $p \in \mathbb{N} \cup \{+\infty\}$ is an upper bound on the number of paths.

The *weight* $w(P)$ of a path P in G is defined as $(\sum_{x \in P} w_G(x)) + \sum_{ab | ab \in F \wedge \{a,b\} \subseteq P} w_H(ab)$, where P is understood as a set of vertices and edges. The task in LIFTED PATHS is to either find an internally vertex-disjoint set $\mathcal{P}$ of at most p-many s-t paths such that $\sum_{P \in \mathcal{P}} w(P) \geq \ell$, or correctly determine that no such set $\mathcal{P}$ exists.

Instances of the LIFTED MULTICUT problem share multiple similarities to those of LIFTED PATHS, but also have some differences (several of which are highlighted in bold). Each such instance is a tuple $\mathcal{I} = (G, H, w, \ell, p)$ where

- $G = (V, E)$ and $H = (V, F)$ are n-vertex **undirected** graphs,
- $w_G : \boldsymbol{E} \to \mathbb{Z}$ and $w_H : F \to \mathbb{Z}$ are weight functions,
- $u \in \mathbb{Z}$ is a target **upper bound**, and
- $p \in \mathbb{N} \cup \{+\infty\}$ is an upper bound on the number of **parts**.

The task in LIFTED MULTICUT is to decide whether G can be vertex-partitioned into at most p-many subgraphs $G_1, \ldots, G_j$, which we call parts, such that:

1. each G_i, $i \in [j]$ is connected, and
2. $(\sum_{e \in E'} w_G(e)) + (\sum_{e \in F'} w_H(e)) \leq u$, where $E' \subseteq E$ and $F' \subseteq F$ are the subsets of edges whose two endpoints lie in pairwise distinct parts.

Throughout this paper, we consider the following parameters for LIFTED MULTICUT and LIFTED PATHS:

- p: the number of parts or paths, respectively,
- r: the number of frames in LIFTED PATHS,
- d: the maximum degree in G (for LIFTED MULTICUT) or of the underlying undirected graph of $G - \{s, t\}$ (for LIFTED PATHS),
- tw: the treewidth of the base graph, i.e., G or the underlying undirected graph of G, respectively,
- tw*: the treewidth of the combined graph, i.e., $G^* = (V, E \cup F)$ or the underlying undirected graph of $G^* = (V, E \cup F)$, respectively.

Algorithms and Tractability Results

In this section we will provide all of the algorithms for LIFTED PATHS required to obtain the complexity map detailed in Figure 1, as well as establish the tractability of LIFTED MULTICUT parameterized by tw*. We divide our exposition into three subsections: one dedicated to algorithms that utilize some combination of "basic parameters" (i.e., r, p or d), one dedicated to algorithms that exploit the treewidth of the base graph (tw), and one establishing the tractability of both problems parameterized by tw*.

Basic Parameters. We begin with a simple observation that identifies a tractable fragment of LIFTED PATHS via an enumeration argument.

Observation 1. LIFTED PATHS *can be solved in time* $\mathcal{O}(n^{p\cdot r})$.

Our second task is to obtain a fixed-parameter algorithm for LIFTED PATHS parameterized by $p+r+d$. The algorithm relies on a technique called *color coding* (Cygan et al. 2015).

Theorem 2. LIFTED PATHS *can be solved in time* $e^{pr}(pr)^{\mathcal{O}(\log pr)}(p+1)^{pr}d^r n \log n$.

Proof Sketch. The idea behind color coding is to randomly color the input such that with high proability the solution is colored in a way that allows one to efficiently find it. In our case, we will use colorings of the vertices of G with p colors and we will look for solutions that are "colorful". That is, suppose we are given a coloring $\lambda : V(G)\setminus\{s,t\} \rightarrow \{1,\ldots,p\}$ of the vertices of G with p colors. A solution $\mathcal{P}$ is *colorful* if:

- every path $P \in \mathcal{P}$ is colored uniformly, i.e., every vertex in $P \setminus \{s,t\}$ is assigned the same color by λ and
- every two paths in $\mathcal{P}$ are assigned distinct colors by λ.

In order to employ color coding using this notion, we need to show that colorful solutions can be found efficiently and that a random coloring makes a solution colorful with sufficiently high probability. We start by observing the former, i.e., that COLORFUL LIFTED PATH, where one is additionally given a vertex-coloring λ of $G-\{s,t\}$ with p colors and asks whether G has a colorful solution, can be solved efficiently. This can be done by enumerating all paths of each color and choosing one of maximum weight for each color.

Claim 3. COLORFUL LIFTED PATHS *can be solved in time* $\mathcal{O}(d^r n)$.

Since the coloring is not provided to us but we assume it to be chosen uniformly and indenpendently at random for every vertex of $G - \{s,t\}$, we also need to argue that there is a sufficiently high probability that a solution $\mathcal{P}$ is colorful.

Because there are at least p colorful colorings of the paths in $\mathcal{P}$ (if there is only one path in $\mathcal{P}$) and the total number of colorings of the at most pr many vertices used by paths in $\mathcal{P}$ is at most p^{pr}, we obtain that the probability that $\mathcal{P}$ is colorful is at least $\frac{p}{p^{pr}} = p^{-(p-1)r}$. Therefore, the randomized algorithm that colors $V(G) \setminus \{s,t\}$ uniformly and independently at random and then uses Claim 3 to check whether there is a colorful solution, will find a solution in an instance (if one exists) with probability at least $p^{-(p-1)r}$. But this means that repeating the algorithm $p^{(p-1)r}$ times already gives a randomized algorithm with constant error probability. Note that the total runtime of this randomized algorithm is at most $\mathcal{O}(p^{(p-1)r}d^r n)$. Moreover, the use of random colorings can be avoided by applying well-known derandomization techniques (Cygan et al. 2015, Section 5.6.1), resulting in a deterministic algorithm with the stated runtime. $\square$

Treewidth of the Base Graph. Both of the tractability results involving tw required for the complexity map of Figure 1 follow from a single algorithm, obtained below. Observe that the number of directed paths going through any vertex in G can be upper-bounded by d^r, and hence by n^r.

Theorem 4. LIFTED PATHS *can be solved in time* $\Delta^{\mathcal{O}(\text{tw})}n^3$*, where* Δ *is the maximum number of directed paths going through any vertex in* G*.*

Proof Sketch. Let $\mathcal{I} = (G,H,w,\ell,p)$ be an instance of LIFTED PATHS and let (T,χ) be a nice tree-decomposition of $G - \{s,t\}$ with root r of width $\omega \in \mathcal{O}(\text{tw})$, which can be computed in linear time (Bodlaender et al. 2016). We provide a dynamic programming algorithm that computes a set of records for every node of the tree-decomposition in a bottom-up manner.

For a node $t \in V(T)$ a *record* is a triple $(\mathcal{P},\delta,W)$, where $\mathcal{P}$ is a set of (vertex-)disjoint paths in G each containing at least one vertex from $\chi(t)$, δ is a natural number, and W is a real number. A record $R = (\mathcal{P},\delta,W)$ is *valid* for t if W is the maximum weight of any solution for the instance $\mathcal{I}$ induced by the vertices in $V(R,t) = (\chi(T_t) \cup \{s,t\}) \setminus (V(\mathcal{P}) \cup \chi(t))$ that uses exactly δ paths. That is, W is the maximum total weight of any set $\mathcal{P}$ of exactly δ vertex-disjoint paths in $G[V(R,t)]$. We denote by $\mathcal{R}(t)$ the set of all valid records for t. Note that $\mathcal{I}$ has a solution if and only if $(\emptyset,\delta,W) \in \mathcal{R}(r)$ with $\delta \leq p$ and $W \geq \ell$. Moreover, $|\mathcal{R}(t)|$ is upper-bounded by $\Delta^\omega \cdot n$.

To complete the proof, it now suffices to show how to compute $\mathcal{R}(t)$ for every node t of T via leaf-to-root dynamic programming. We illustrate this procedure on the example of introduce and join nodes. For a vertex $v \in V(G)$, let $\mathcal{P}_G(v)$ be the set of directed paths in $G - \{s,t\}$ going through v.

Introduce nodes: Let t be an introduce node of T with child t' and $\chi(t) \setminus \chi(t') = \{v\}$ for some $v \in V(G) \setminus \{s,t\}$. Then, $\mathcal{R}(t)$ contains all records that are in $\mathcal{R}(t')$ and an additional record $(\mathcal{P} \cup \{P\},\delta,W)$ for every record $(\mathcal{P},\delta,W) \in \mathcal{R}(t')$ and $P \in \mathcal{P}_G(v)$ such that P contains no vertex in $V(\mathcal{P}) = \bigcup_{P\in\mathcal{P}} V(P)$ and no vertex in $\chi(t')$. More formally:

$$\begin{aligned}\mathcal{R}(t) \quad = \quad & \mathcal{R}(t') \cup \{\, (\mathcal{P} \cup \{P\},\delta,W) \mid (\mathcal{P},\delta,W) \in \mathcal{R}(t') \\ & \wedge P \in \mathcal{P}_G(v) \wedge P \cap (V(\mathcal{P}) \cup \chi(T_{t'})) = \emptyset \,\}\end{aligned}$$

Join nodes: Let t be a join node of T with children t_1 and t_2. To compute $\mathcal{R}(t)$ we proceed in two steps. First we compute an auxiliary set $\mathcal{R}'(t)$ from $\mathcal{R}(t_1)$ and $\mathcal{R}(t_2)$ as follows:

$$\begin{aligned}\mathcal{R}'(t) \quad = \quad & \{\, (\mathcal{P},\delta_1+\delta_2,W_1+W_2) \mid \\ & (\mathcal{P},\delta_1,W_1) \in \mathcal{R}(t_1) \wedge (\mathcal{P},\delta_2,W_2) \in \mathcal{R}(t_2) \,\}\end{aligned}$$

To obtain $\mathcal{R}(t)$ from $\mathcal{R}'(t)$, we then remove "duplicate" records, i.e., records that agree in $\mathcal{P}$ and δ. More formally:

$$\begin{aligned}\mathcal{R}(t) \quad = \quad & \{\, (\mathcal{P},\delta,W) \mid (\mathcal{P},\delta,W) \in \mathcal{R}'(t) \wedge \\ & W = \max\{W' | (\mathcal{P},\delta,W') \in \mathcal{R}'(t)\} \,\} \quad \square\end{aligned}$$

Treewidth of the Combined Graph. To complete the upper-bound component of our complexity analysis, we establish the fixed-parameter tractability of LIFTED PATHS and LIFTED MULTICUT parameterized by tw*. This is achieved via the use of Courcelle's famous theorem (Courcelle 1990), or more precisely its extension towards optimization problems by Arnborg, Lagergren, Seese (1991); see also the summary provided by, e.g., Langer et al. (2014).

Proposition 5 (Courcelle's Theorem for Optimization)**.** *Given* (1) *a labeled digraph $G = (V, E)$ whose underlying undirected graph has treewidth k,* (2) *a weight function $\omega : V \cup E \to \mathbb{Z}$, and* (3) *a formula $\phi(X_1, \dots, X_\ell)$ in* Monadic Second Order Logic *with ℓ free set variables, finding an interpretation $(S_1, \dots, S_\ell)$ such that $G \models \phi(S_1, \dots, S_\ell)$ and which maximizes $\sum_{a \in \bigcup_{i\in[\ell]} S_i} \omega(a)$ is fixed-parameter tractable parameterized by $|\phi| + k$.*

Theorem 6. LIFTED PATHS *and* LIFTED MULTICUT *are fixed-parameter tractable when parameterized by* tw*.

Proof Sketch. Let us first consider the LIFTED PATHS problem. Let G^* be the combined digraph $(V, E \cup F)$ where we label each arc as "base" (if it occurs in E), "lifted" (if it occurs in F), or both. Consider a weight function ω that is defined for each edge e as $w_G(e) + w_H(e)$ (where any undefined terms are replaced by 0) and for each vertex v as $\omega_G(v)$. To apply Courcelle's Theorem, we construct a Monadic Second Order Logic formula $\phi(X_1, X_2, X_3)$ that will be interpreted over (sets of) vertices and edges of G^* with three free set variables. Crucially, the formula will be true if and only if X_1 is interpreted as a set S_1 of arcs that form directed paths from s to t, X_2 as a set S_2 of arcs whose both endpoints lie on some path defined by S_1, and S_3 are all vertices occurring on paths in S_1. Assuming basic knowledge of Monadic Second Order Logic, constructing such a formula is a moderately tedious but ultimately simple task, and hence we leave this as an exercise for interested readers. Once we construct such a formula of size bounded by a function of tw*, then the fixed-parameter tractability of LIFTED PATHS parameterized by tw* follows from Proposition 5.

The same approach can be used for LIFTED MULTICUT; the main distinction is that instead of using the formula ϕ, we construct a formula whose set variables will capture the edges that are cut in a partitioning of G. $\square$

Lower Bounds

We proceed towards the second component of our complexity map, notably lower bounds. These are structured in three subsections: one providing two NP-hardness reductions for restrictions of LIFTED PATHS, one dedicated to identifying the precise boundaries of fixed-parameter tractability for LIFTED PATHS, and the last one establishing the intractability of LIFTED MULTICUT even under severe restrictions.

Excluding Polynomial Algorithms for Lifted Paths. The two NP-hardness proofs for LIFTED PATHS both start from a variant of the classical Satisfiability problem (SAT).

Theorem 7. LIFTED PATHS *is* NP*-hard even when restricted to instances such that* tw ≤ 2*,* $d \leq 3$ *and* $p = 1$*.*

Proof Sketch. We reduce from the decision version of MAX-2-SAT: given a 2-CNF formula φ over variables $x_1, \dots, x_n$ and clauses $C_1, \dots, C_m$ along with an integer a, does there exist an assignment of the variables in φ which satisfies at least a clauses? MAX-2-SAT is well-known to be NP-hard (Garey, Johnson, and Stockmeyer 1976).

The construction proceeds as follows. For each variable x_i, G contains the vertices $x_i^{start}, x_i^{true}, x_i^{false}$ and x_i^{end} and the following arcs: $x_i^{start}x_i^{true}, x_i^{start}x_i^{false}, x_i^{true}x_i^{end}, x_i^{false}x_i^{end}$. Similarly, for each clause C_i we construct the vertices $c_i^{start}, c_i^{first}, c_i^{second}, c_i^{end}$ and add the following arcs: $c_i^{start}c_i^{first}, c_i^{start}c_i^{second}, c_i^{first}c_i^{end}, c_i^{second}c_i^{end}$. We also add an arc from each x_i^{end} to x_{i+1}^{start} (for $i < n$), an arc from each c_i^{end} to c_{i+1}^{start} (for $i < m$), and the arc $x_n^{end}c_1^{start}$. Naturally, we also have s as a universal source and t as a universal sink; this completes the construction of G. Observe that there are multiple paths from x_1^{start} to c_m^{end}, but in each path we can choose either a "true" or "false" vertex for each variable and either a "first" or "second" vertex for each clause. All vertices and edges in G receive a weight of 0.

For the arc set F of H, we proceed as follows. For each literal that occurs in a clause, for example the literal $\neg x_i$ in clause $C_j = (\neg x_i, x_z)$, we create an arc from the vertex representing the same valuation of that variable to the vertex representing the position that literal holds in the clause; in the mentioned example, we would create the arc $x_i^{false}c_j^{first}$. This way, each vertex corresponding to a literal in a clause will have one incoming arc in F, and we let w_H assign a weight of 1 to each such arc. Moreover we introduce the arc $x_1^{start}c_m^{end}$ and let $w_H(x_1^{start}c_m^{end}) = 2m$. The aim of this is to ensure that we must include a path that contains both these vertices together to achieve an objective value of at least $2m$.

Finally, we set $\ell = a + 2m$. It can be shown that the initial MAX-2-SAT instance is a **Yes**-instance if and only if the constructed LIFTED PATHS instance is a **Yes**-instance for $p = 1$ and the instance has the desired properties. $\square$

Theorem 8. LIFTED PATHS *is* NP*-hard even when restricted to instances such that* $r \leq 5$*,* $d \leq 6$*, and all weights are set to zero or one.*

Proof Sketch. We reduce from 3-SAT-2, i.e., the variant of 3-SAT where each variable occurs at most twice positively and at most twice negatively, which is known to be NP-hard (Yannakakis 1978). Consider an instance of 3-SAT-2 with variables $x_1, \dots, x_n$ and clauses $C_1 = (l_1^1, l_1^2, l_1^3), \dots, C_m = (l_m^1, l_m^2, l_m^3)$.

We construct an instance of LIFTED PATHS by first letting $G = (V, E)$ have the vertex set $\{s, t\} \cup \bigcup_{i\in[5]} V_i$ where

- $V_1 = \{a_1, \dots a_n\}$ is the set of *variable decision vertices*,
- $V_2 = \{v_{x_i}^1, v_{\bar{x}_i}^1 \mid i \in [n]\}$ is the set of *first literal occurrence vertices*,
- $V_3 = \{v_{x_i}^2, v_{\bar{x}_i}^2 \mid i \in [n]\}$ is the set of *second literal occurrence vertices*,
- $V_4 = \{b_1, \dots b_n\}$ is the set of *variable decision partner vertices*, and
- $V_5 = \{c_1, \dots, c_m\}$ is the set of *clause vertices*.

The arc set of G is defined as $E = \{sv \mid v \in V \setminus \{s\}\} \cup \{vt \mid v \in V \setminus \{s,t\}\} \cup \{a_i v_{x_i}^1, a_i v_{\bar{x}_i}^1 \mid i \in [n]\} \cup \{v_l^1 v_l^2 \mid l \in \{x_1, \bar{x_1}, \dots, x_n, \bar{x_n}\}\} \cup \{v_{x_i}^2 b_i, v_{\bar{x}_i}^2 b_i \mid i \in [n]\} \cup \{v_{l_i^j}^h c_i \mid i \in [m], j \in [3], h \in [2]\}$. In this way each a_i or b_i has degree 2 in $G - \{s,t\}$, each v_l has degree at most 4 in $G - \{s,t\}$, and each c_j has degree at most 6 in $G - \{s,t\}$. We set the weights of all arcs and vertices in G to zero. Second, we set $F = \{a_i b_i \mid i \in [n]\} \cup \{v^{\ell}_{l^j_i} c_i \mid i \in [m], j \in [3], \ell \in [2]\}$ with all arcs of H having weight 1. To complete the proof, it suffices to show that the original 3-SAT-2 instance was satisfiable if and only if the constructed LIFTED PATHS instance with target lower bound $\ell = n+m$ is a **Yes**-instance (with $p = \infty$). To observe this, consider the set of paths constructed as follows:

- whenever a variable x_i is set to true and that variable is the "first" variable to satisfy the clause c_j with its first literal, use the path $(s, v^1_{x_i}, c_j, t)$ (where $v^1_{\bar{x}_i}$ would be used if x_i is set to false, and $v^1_{\bar{x}_i}$ for its second literal);
- whenever a variable is set to true, use the path $(s, a_i, v^1_{\bar{x}_i}, v^2_{\bar{x}_i}, b_i, t)$.

It is easy to verify that the constructed paths correspond to a solution for the original 3-SAT-2 instance. $\square$

Excluding Fixed-Parameter Algorithms for Lifted Paths. The first result in this section is a fairly straightforward W[1]-hardness reduction for LIFTED PATHS parameterized by $r + \text{tw} + p$; in particular, the reduction rules out fixed-parameter tractability for the problem parameterized by r even if the tw and p are fixed to very small constants. However, on its own the reduction still leaves one final gap in our understanding of the complexity of LIFTED PATHS. In particular, while Theorem 2 also provides a fixed-parameter algorithm parameterized by p when r is fixed to a constant, at this point it is not yet clear whether the problem is FPT or W[1]-hard when parameterized by tw in the case of r being fixed by a constant. The second result in this section then closes this gap via an intricate reduction. Both reductions start from the well-known W[1]-complete problem MULTICOLORED CLIQUE (Pietrzak 2003).

Theorem 9. LIFTED PATHS *is* W[1]*-hard parameterized by* r*, even if* $p = 1$ *and* $\text{tw} = 2$.

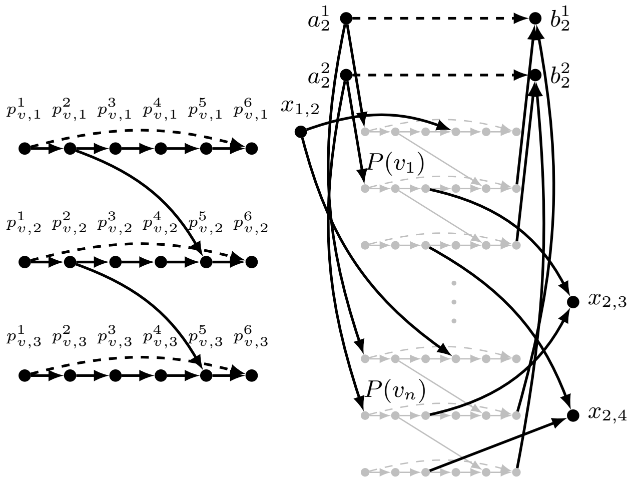
We now proceed to our second, significantly more involved W[1]-hardness reduction.

Theorem 10. LIFTED PATHS *is* W[1]*-hard parameterized by* tw *even if* $r = 8$.

Proof Sketch. Consider an instance (D,k) of MULTICOLORED CLIQUE with partition $V(D) = M_1 \cup \dots \cup M_k\}$. In the following we denote by $E_{i,j}$ the set of all edges in $E(D)$ with one endpoint in M_i and the other endpoint in M_j, for every i and j with $1 \le i < j \le k$. To show the theorem, we will construct an instance $\mathcal{I} = (G, H, w_G, w_H, \ell, \infty)$ of LIFTED PATHS where $r = 8$ and G has treewidth at most $\binom{k}{2} + 2k(k-2) + 3$ such that D has a k-clique if and only if $\mathcal{I}$ is a **Yes**-instance.

For every vertex $v \in V(D)$, we add a *vertex gadget* $P(v)$ to $\mathcal{I}$ that is illustrated in Figure 2 and consists of $k-1$ directed paths (denoted $P_1(v), ..., P_{k-1}(v)$) having 6 vertices (the vertices $p^j_{v,i}$) each. Moreover, H contains a *vertex arc* of weight 1 from the first vertex of each path to its last vertex and we denote by A_V the set of all vertex arcs added in this manner.

For every i and j with $1 \le i < j \le k$, we add the vertex $x_{i,j}$ together with the following arcs to G and H:



$p^1_{v,1}$ $p^2_{v,1}$ $p^3_{v,1}$ $p^4_{v,1}$ $p^5_{v,1}$ $p^6_{v,1}$
$p^1_{v,2}$ $p^2_{v,2}$ $p^3_{v,2}$ $p^4_{v,2}$ $p^5_{v,2}$ $p^6_{v,2}$
$p^1_{v,3}$ $p^2_{v,3}$ $p^3_{v,3}$ $p^4_{v,3}$ $p^5_{v,3}$ $p^6_{v,3}$
a^1_2
b^1_2
a^2_2
b^2_2
$x_{1,2}$
$P(v_1)$
$P(v_n)$
$x_{2,3}$
$x_{2,4}$

Figure 2: Illustrations for the construction used in the proof of Theorem 10 for the case that $k = 4$. Arcs in G and H are represented by solid lines and dashed lines, respectively. Left: The gadget $P(v)$. Right: The connections between the choice gadget and the vertices in X with $M_2 = \{v_1, \dots, v_n\}$. The paths $P(v)$ for $v \in M_2$ are given in gray.

- for every vertex $v \in M_i$, we add an arc from the third vertex $p^3_{v,j-1}$ of $P_{j-1}(v)$ to $x_{i,j}$ to G,
- for every vertex $v \in M_j$, we add an arc from $x_{i,j}$ to the fourth vertex $p^4_{v,i}$ of $P_i(v)$,
- for every edge $e = \{u,v\} \in E(D)$ of D with $u \in M_i$ and $v \in M_j$, we add an arc of weight 1 to H, which we refer to as an *edge arc*, that goes from the third vertex $p^3_{u,j-1}$ of $P_{j-1}(u)$ to the fourth vertex $p^4_{v,i}$ of $P_i(v)$. Let A_E be the set of all edge arcs added to H in this manner and let $X = \{ x_{i,j} \mid 1 \le i < j \le k \}$.

Moreover, for every i with $1 \le i \le k$, we add a *choice gadget* $C(i)$ having vertices $a^1_i, b^1_i, \dots, a^{k-2}_i, b^{k-2}_i$ and an arc in H of weight 3 from a^j_i to b^j_i for every j with $1 \le j \le k-2$; we refer to these as *choice arcs*. Let A_C be the set of all choice arcs added to H in this manner. Finally, we connect the choice gadgets to the vertex gadgets as follows. For every i with $1 \le i \le k$, we add the following arcs to G:

- an arc from a^j_i to the first vertex $p^1_{v,j}$ of $P_j(v)$ for every j with $1 \le j \le k-2$ and every $v \in M_i$.
- an arc from the last vertex $p^6_{v,j+1}$ of $P_{j+1}(v)$ to b^j_i for every j with $1 \le j \le k-2$ and every $v \in M_i$.

The choice gadget and the vertices in X together with their connections to the path gadgets are illustrated in Figure 2. Note that G (without s and t) can be partitioned into the following 8 layers (and therefore $r = 8$): The vertices a^j_i, followed by the first, second, and third vertex of each path in $P(v)$ for $v \in V(D)$, the vertices in X, followed by the fourth, fifth, and sixth vertex of each path in $P(v)$ for $v \in V(D)$, and the vertices b^j_i. Setting $\ell = (|V(D)| - k)(k-1) + 3(k-2)k + \binom{k}{2}$ completes the construction of $\mathcal{I}$.

Regarding the treewidth of G, we observe that deleting the set $Y = \{s,t\} \cup X \cup \{ a^j_i, b^j_i \mid 1 \le i \le k \wedge 1 \le j \le k-2 \}$ from G results in a tree; since $|Y| = \binom{k}{2} + 2k(k-2) + 2$, it

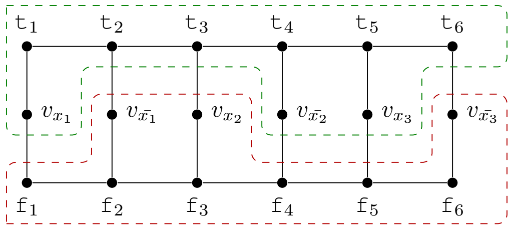


Figure 3: G of the constructed LIFTED MULTICUT instance in the proof of Theorem 11 for a NAE 3-SAT formula on variables x_1, x_2, x_3. The dashed lines indicate a partition which corresponds to assigning x_1, $\bar{x_2}$ and x_3 to true.

follows that $\mathrm{tw}(G) \leq \binom{k}{2} + 2k(k-2) + 3$, as required.

The main idea behind the construction is that every solution $\mathcal{P}$ for $\mathcal{I}$ has to connect the vertices $a_i^1, b_i^1, \ldots, a_i^{k-2}, b_i^{k-2}$ of the choice gadget for every i with $1 \leq i \leq k$ using only the $k-1$ paths of the path gadget $P(v)$ of a single vertex $v \in M_i$; since otherwise more than $k-1$ paths in $\{\, P(u) \mid u \in M_i \,\}$ are used up to connect the higher valued arcs in A_C. Moreover, to reach the total weight all other paths in $\{\, P(u) \mid u \in M_i \,\}$ need to be used to connect the vertex arcs in A_V. Finally, the maximum number of edge arcs in A_E can only be connected if the vertices $\{v_1, \ldots, v_k\}$ that have been chosen to connect the choice gadgets form a k-clique in D; since only the paths in $P(v_i)$ are free, i.e., not used to connect the vertex arcs and can therefore be used to connect the edge arcs. □

Lower Bounds for Lifted Multicut. As our final result, we show that compared to LIFTED PATHS, LIFTED MULTICUT remains highly intractable even when strongly restricting all considered aspects of the problem.

Theorem 11. LIFTED MULTICUT *is* NP*-hard for* $\mathrm{tw} = 2$, $p = 2$ *and* $d = 2$.

Proof Sketch.. We reduce from *Not All Equal 3-SAT (NAE 3-SAT)* in the following way: Consider an instance of NAE 3-SAT with variables $x_1, \ldots, x_n$ and clauses $C_1 = (l_1^1, l_1^2, l_1^3), \ldots, C_m = (l_m^1, l_m^2, l_m^3)$. W.l.o.g. no clause contains both the positive and negative literal of a variable. We construct an instance of LIFTED MULTICUT by firstly letting $G = (V, E)$ have the vertex set $V = \{\mathtt{t}_1, \mathtt{f}_1, \ldots, \mathtt{t}_{2n}, \mathtt{f}_{2n}\} \cup \{v_{x_i}, v_{\bar{x_i}} \mid i \in [n]\}$, and the edge set $E = \{\mathtt{t}_i\mathtt{t}_{i+1}, \mathtt{f}_i\mathtt{f}_{i+1} \mid i \in [2n-1]\} \cup \{\mathtt{t}_{2i-1}v_{x_i}, \mathtt{f}_{2i-1}v_{x_i} \mid i \in [n]\} \cup \{\mathtt{t}_{2i}v_{\bar{x_i}}, \mathtt{f}_{2i}v_{\bar{x_i}} \mid i \in [n]\}$. It is easy to verify that in this way $\mathrm{tw}(G) = 2$.

We set $w_G(e) = 2m$ for all edges $e \in E \setminus \{\mathtt{t}_i\mathtt{t}_{i+1}, \mathtt{f}_i\mathtt{f}_{i+1} \mid i \in [2n-1]\}$, and $w_G(\mathtt{t}_i\mathtt{t}_{i+1}) = 5mn$ and $w_G(\mathtt{f}_i\mathtt{f}_{i+1}) = 5mn$ for $i \in [2n-1]$. An example of the construction of G is given in Figure 3 (together with an illustration of the correspondance of an assignment for NAE 3-SAT to a partition of G).

Secondly we let H be defined on the vertex set V with the edge set given by $F = \{\mathtt{t}_1\mathtt{f}_1\} \cup \{v_{x_i}v_{\bar{x_i}} \mid i \in [n]\} \cup \{v_{l_i^{j_1}}v_{l_i^{j_2}} \mid i \in [m], 1 \leq j_1 < j_2 \leq 3\}$. We set $w_H(\mathtt{t}_1\mathtt{f}_1) = -5mn$, $w_H(v_{x_i}v_{\bar{x_i}}) = -1$ for $i \in [n]$, and $w_H(v_{l_i^{j_1}}v_{l_i^{j_2}}) = -1$ for $i \in [m]$ and $1 \leq j_1 < j_2 \leq 3$.

Now the original NAE 3-SAT instance is satisfiable if and only if the LIFTED MULTICUT instance with bound $u = 4mn - 5mn - 2m - n = -mn - 2m - n$ is a **Yes**-instance.

To observe this, note that the weights in G and H are chosen in a way that $\{\mathtt{t}_1, \ldots, \mathtt{t}_{2n}\}$ has to be a subset of one part of any partition of G with the desired objective value and $\{\mathtt{f}_1, \ldots, \mathtt{f}_{2n}\}$ has to be a subset in a different part of such a partition. This is because the edges connecting $\mathtt{t}_i$s and $\mathtt{f}_i$s repectively to each other have very high, and $\mathtt{t}_1\mathtt{f}_1$ has a very low (negative) weight. Similarly since separating any vertex v_{x_i} or $v_{\bar{x_i}}$ from both $\mathtt{t}_i$ and $\mathtt{f}_i$ in a partition of G is penalized more than separating it from *either* $\mathtt{t}_i$ *or* $\mathtt{f}_i$, one can argue that in any partition of G with the desired objective value there are only two parts, one containing all $\mathtt{t}_i$s and the other containing all $\mathtt{f}_i$s.

Moreover the weights of the edges of the form $v_{x_i}v_{\bar{x_i}}$ in H prevent v_{x_i} and $v_{\bar{x_i}}$ from being in the same part of a partition with the desired objective value, and the edges of the form $v_{l_i^{j_1}}v_{l_i^{j_2}}$ in H prevent all vertices corresponding to literals of any clause from being in the same part of a partition with the desired objective value. Thus a setting all variables to `true` which correspond to vertices in the same part as $\mathtt{t}_1$ in a partition with the desired objective value is equivalent to a not-all-equal satisfying assignment and vice versa. □

This hardness already holds under severe restrictions of all parameters considered in Theorem 11. We are actually able to draw the line between polynomial-time solvable and NP-hard cases exactly, even regarding the specific numerical bounds, by noting that the problem is trivial as soon as $p = 1$ or $d = 1$. As for the case of $\mathrm{tw} = 1$ we can show hardness also for this case—in fact, by a reduction from 1-IN-3-SAT we show that the problem remains NP-hard even when the base graph is a *star*.

Theorem 12. LIFTED MULTICUT *is* NP*-hard even when restricted to instances such that G is a star.*

Concluding Remarks

We carried out a detailed theoretical analysis of both the well-established LIFTED MULTICUT and the more recent but promising LIFTED PATHS model for object association in MOT. Our analysis identifies the precise boundaries of tractability for the problems under natural restrictions.

Somewhat unexpectedly, our findings suggest that the LIFTED PATHS model is by far better suited for exploiting the considered parameters than the LIFTED MULTICUT model, and our hardness proofs for LIFTED MULTICUT show intractability for extremely restricted instances, potentially signifying a substantial algorithmic advantage of considering LIFTED PATHS. Hence, it would be interesting to study if this is also reflected in practice, i.e., to design experiments that isolate and compare the effect on the running time of using LIFTED MULTICUT versus LIFTED PATHS.

Another interesting research direction would be to analyze the properties of real-life instances of LIFTED MULTICUT and LIFTED PATHS that arise during MOT, in particular with respect to how tw and tw* compare to each other and the identification of other algorithmically useful parameters.

Acknowledgements

Robert Ganian and Thekla Hamm acknowledge support by the Austrian Science Fund (FWF, projects P31336 and Y 1329). Sebastian Ordyniak acknowledges support from the Engineering and Physical Sciences Research Council (EPSRC, project EP/V00252X/1).

References

Arbelaez, P.; Maire, M.; Fowlkes, C. C.; and Malik, J. 2011. Contour Detection and Hierarchical Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 33(5): 898–916.

Arnborg, S.; Lagergren, J.; and Seese, D. 1991. Easy problems for tree-decomposable graphs. *J. Algorithms* 12(2): 308–340.

Babaee, M.; Li, Z.; and Rigoll, G. 2019. A dual CNN–RNN for multiple people tracking. *Neurocomputing* 368: 69–83.

Berclaz, J.; Fleuret, F.; Türetken, E.; and Fua, P. 2011. Multiple Object Tracking Using K-Shortest Paths Optimization. *IEEE Trans. Pattern Anal. Mach. Intell.* 33(9): 1806–1819.

Bodlaender, H. L.; Drange, P. G.; Dregi, M. S.; Fomin, F. V.; Lokshtanov, D.; and Pilipczuk, M. 2016. A $c^k n$ 5-Approximation Algorithm for Treewidth. *SIAM J. Comput.* 45(2): 317–378.

Bredereck, R.; Faliszewski, P.; Kaczmarczyk, A.; Knop, D.; and Niedermeier, R. 2020. Parameterized Algorithms for Finding a Collective Set of Items. In *AAAI*, 1838–1845. AAAI Press.

Brendel, W.; Amer, M. R.; and Todorovic, S. 2011. Multiobject tracking as maximum weight independent set. In *CVPR*, 1273–1280. IEEE Computer Society.

Chari, V.; Lacoste-Julien, S.; Laptev, I.; and Sivic, J. 2015. On pairwise costs for network flow multi-object tracking. In *CVPR*, 5537–5545. IEEE Computer Society.

Chu, Q.; Ouyang, W.; Liu, B.; Zhu, F.; and Yu, N. 2020. DASOT: A Unified Framework Integrating Data Association and Single Object Tracking for Online Multi-Object Tracking. In *AAAI*, 10672–10679. AAAI Press.

Courcelle, B. 1990. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Inf. Comput.* 85(1): 12–75.

Cygan, M.; Fomin, F. V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer. ISBN 978-3-319-21274-6. doi:10.1007/978-3-319-21275-3. URL https://doi.org/10.1007/978-3-319-21275-3.

Dehghan, A.; Assari, S. M.; and Shah, M. 2015. GMMCP tracker: Globally optimal Generalized Maximum Multi Clique problem for multiple object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 4091–4099. IEEE Computer Society.

Diestel, R. 2012. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer.

Downey, R. G.; and Fellows, M. R. 2013. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer.

Eiben, E.; Ganian, R.; Hamm, T.; and Ordyniak, S. 2020. Parameterized Complexity of Envy-Free Resource Allocation in Social Networks. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, 7135–7142. AAAI Press.

Ganian, R.; Kanj, I.; Ordyniak, S.; and Szeider, S. 2020. On the Parameterized Complexity of Clustering Incomplete Data into Subspaces of Small Rank. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, 3906–3913. AAAI Press.

Ganian, R.; and Ordyniak, S. 2018. The complexity landscape of decompositional parameters for ILP. *Artif. Intell.* 257: 61–71.

Ganian, R.; Ordyniak, S.; and Ramanujan, M. S. 2017. Going Beyond Primal Treewidth for (M)ILP. In Singh, S. P.; and Markovitch, S., eds., *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, 815–821. AAAI Press.

Garey, M. R.; Johnson, D. S.; and Stockmeyer, L. J. 1976. Some Simplified NP-Complete Graph Problems. *Theor. Comput. Sci.* 1(3): 237–267.

Henschel, R.; Leal-Taixé, L.; Cremers, D.; and Rosenhahn, B. 2018. Fusion of Head and Full-Body Detectors for Multi-Object Tracking. In *CVPR Workshops*, 1428–1437. IEEE Computer Society.

Hofmann, M.; Wolf, D.; and Rigoll, G. 2013. Hypergraphs for Joint Multi-view Reconstruction and Multi-object Tracking. In *CVPR*, 3650–3657. IEEE Computer Society.

Horňáková, A.; Henschel, R.; Rosenhahn, B.; and Swoboda, P. 2020. Lifted Disjoint Paths with Application in Multiple Object Tracking. In *ICML*, volume 121 of *Proceedings of Machine Learning Research*, 1–12. PMLR.

Horňáková, A.; Lange, J.-H.; and Andres, B. 2017. Analysis and Optimization of Graph Decompositions by Lifted Multicuts. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 1539–1548. International Convention Centre, Sydney, Australia: PMLR.

Hu, W.; Shi, X.; Zhou, Z.; Xing, J.; Ling, H.; and Maybank, S. J. 2020. Dual L_1-Normalized Context Aware Tensor Power Iteration and Its Applications to Multi-object Tracking and Multi-graph Matching. *Int. J. Comput. Vis.* 128(2): 360–392.

Huang, J.; and Zhou, W. 2019. Re^2EMA: Regularized and Reinitialized Exponential Moving Average for Target Model Update in Object Tracking. In *AAAI*, 8457–8464. AAAI Press.

Keuper, M.; Levinkov, E.; Bonneel, N.; Lavoué, G.; Brox, T.; and Andres, B. 2015. Efficient Decomposition of Image and Mesh Graphs by Lifted Multicuts. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 1751–1759.

Keuper, M.; Tang, S.; Andres, B.; Brox, T.; and Schiele, B. 2020. Motion Segmentation & Multiple Object Tracking by Correlation Co-Clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* 42(1): 140–153.

Kloks, T. 1994. *Treewidth: Computations and Approximations*. Berlin: Springer Verlag.

Kumar, R.; Charpiat, G.; and Thonnat, M. 2014. Multiple Object Tracking by Efficient Graph Partitioning. In Cremers, D.; Reid, I. D.; Saito, H.; and Yang, M., eds., *Computer Vision - ACCV 2014 - 12th Asian Conference on Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part IV*, volume 9006 of *Lecture Notes in Computer Science*, 445–460. Springer.

Langer, A.; Reidl, F.; Rossmanith, P.; and Sikdar, S. 2014. Practical algorithms for MSO model-checking on tree-decomposable graphs. *Comput. Sci. Rev.* 13-14: 39–74.

Milan, A.; Rezatofighi, S. H.; Dick, A. R.; Reid, I. D.; and Schindler, K. 2017. Online Multi-Target Tracking Using Recurrent Neural Networks. In *AAAI*, 4225–4232. AAAI Press.

Nesetril, J.; and de Mendez, P. O. 2012. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and Combinatorics*. Springer Verlag.

Ordyniak, S.; and Szeider, S. 2013. Parameterized Complexity Results for Exact Bayesian Network Structure Learning. *J. Artif. Intell. Res.* 46: 263–302.

Peters, D. 2016. Graphical Hedonic Games of Bounded Treewidth. In Schuurmans, D.; and Wellman, M. P., eds., *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, 586–593. AAAI Press.

Pfandler, A.; Rümmele, S.; Wallner, J. P.; and Woltran, S. 2015. On the Parameterized Complexity of Belief Revision. In Yang, Q.; and Wooldridge, M. J., eds., *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 3149–3155. AAAI Press.

Pietrzak, K. 2003. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. of Computer and System Sciences* 67(4): 757–771.

Pylyshyn, Z. W.; and Storm, R. 1988. Tracking multiple independent targets: evidence for a parallel tracking mechanism. *Spatial vision* 3: 179–197.

Robertson, N.; and Seymour, P. D. 1986. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms* 7(3): 309–322.

Tang, S.; Andres, B.; Andriluka, M.; and Schiele, B. 2015. Subgraph decomposition for multi-target tracking. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 5033–5041. IEEE Computer Society.

Tang, S.; Andres, B.; Andriluka, M.; and Schiele, B. 2016. Multi-person Tracking by Multicut and Deep Matching. In Hua, G.; and Jégou, H., eds., *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II*, volume 9914 of *Lecture Notes in Computer Science*, 100–111.

Tang, S.; Andriluka, M.; Andres, B.; and Schiele, B. 2017. Multiple People Tracking by Lifted Multicut and Person Re-identification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 3701–3710. IEEE Computer Society.

Thorup, M. 1998. All Structured Programs have Small Tree-Width and Good Register Allocation. *Inf. Comput.* 142(2): 159–181.

Yannakakis, M. 1978. Node- and Edge-Deletion NP-Complete Problems. In Lipton, R. J.; Burkhard, W. A.; Savitch, W. J.; Friedman, E. P.; and Aho, A. V., eds., *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, 253–264. ACM.

Zamir, A. R.; Dehghan, A.; and Shah, M. 2012. GMCP-Tracker: Global Multi-object Tracking Using Generalized Minimum Clique Graphs. In *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part II*, volume 7573 of *Lecture Notes in Computer Science*, 343–356. Springer.

Zhang, L.; Li, Y.; and Nevatia, R. 2008. Global data association for multi-object tracking using network flows. In *CVPR*. IEEE Computer Society.