Lifted Disjoint Paths with Application in Multiple Object Tracking
Appendix

Andrea Hornakova [*1] **Roberto Henschel** [*2] **Bodo Rosenhahn** [2] **Paul Swoboda** [1]

Abstract

This appendix supplements our work by presenting missing proofs regarding the solver and details about our tracker.

Sections 10.1 up to Section 10.4 provide proofs used in Sections 4 and 6.

Section 10.6 provides further information how the optimal assignments used in Section 7.5 were obtained. The impact of the employed post-processing used in our tracker is analyzed in Section 10.7. Details about the used fusion network are given in Section 10.8. Finally, evaluation metrics for all tracked sequences are provided in Section 10.9.

10. Appendix

10.1. Proofs for Section 4

Proposition 3. *Path inequalities* (8) *define a strictly tighter relaxation of the lifted disjoint path problem than the lifted multicut path inequalities*

$$\forall vw \in E' \; \forall P \in vw\text{-paths}(G) : \quad y'_{vw} \geq \sum_{ij \in P_E} (y_{ij} - 1) + 1 \,. \tag{13}$$

Proof. Let us define the following sets:

$$S_B = \{(y, y') \in [0,1]^E \times [0,1]^{E'} | (y, y') \text{ satisfy (8)}\} \,,$$
$$S_M = \{(y, y') \in [0,1]^E \times [0,1]^{E'} | (y, y') \text{ satisfy (13)}\} \,.$$

- Let us prove that $S_B \subset S_M$

Let us rewrite the right hand side of (8) for a path $P \in vw\text{-paths}(G)$:

$$\begin{aligned} y'_{vw} &\geq \sum_{vj : j \in P_V} y_{vj} - \sum_{i \in P_V \setminus \{v,w\}} \sum_{k \notin P_V} y_{ik} = \\ &= \sum_{vj : j \in P_V} y_{vj} - \sum_{i \in P_V \setminus \{v,w\}} (x_i - \sum_{j \in P_V} y_{ij}) = \\ &= \sum_{i \in P_V \setminus w} \sum_{j \in P_V} y_{ij} - \sum_{i \in P_V \setminus \{v,w\}} x_i \geq \\ &\geq \sum_{ij \in P_E} y_{ij} - \sum_{i \in P_V \setminus \{v,w\}} 1 = \\ &= \sum_{ij \in P_E} (y_{ij} - 1) + 1 \,. \end{aligned} \tag{14}$$

- Let us prove that $S_B \subsetneq S_M$
We prove that there exists $(y, y') \in [0,1]^E \times [0,1]^{E'}$ such that (y, y') satisfies (13) and does not satisfy (8). An example is given in Figure 2. There are four possible paths from v to w. If we use Constraints (13), the highest lower bound on y'_{vw} is given by path $P = (vv_2, v_2v_4, v_4w)$ and it is as follows:

$$y'_{vw} \geq (0.5 - 1) + (0.5 - 1) + (1 - 1) + 1 = 0 \,.$$

Let us apply Constraint (8) using path $P = (vv_1, v_1v_2, v_2v_3, v_3v_4, v_4w)$. We obtain the following threshold on y'_{vw}

$$y'_{vw} \geq 0.5 + 0.5 - 0 - 0 = 1 \,.$$

□

Proposition 1. *The lifted path inequalities* (10) *provide a strictly better relaxation than the path inequalities* (8)*.*

Proof. Let us define the following sets

$$S_B = \{(y, y') \in [0,1]^E \times [0,1]^{E'} | (y, y') \text{ satisfy (8)}\} \,,$$
$$S_L = \{(y, y') \in [0,1]^E \times [0,1]^{E'} | (y, y') \text{ satisfy (10)}\} \,.$$

- Let us prove that $S_L \subset S_B$:
Note that every path $P \in vw\text{-paths}(G)$ belongs to the set of $vw\text{-paths}(G \cup G')$ too. It just holds that $P_{E'} = \emptyset$. Let

[*]Equal contribution [1]Computer Vision and Machine Learning, Max Planck Institute for Informatics, Saarbrücken, Saarland, Germany [2]Institut for Image Processing, Leibniz University Hannover, Hannover, Niedersachsen, Germany. Correspondence to: Andrea Hornakova <andrea.hornakova@mpi-inf.mpg.de>, Roberto Henschel <henschel@tnt.uni-hannover.de>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

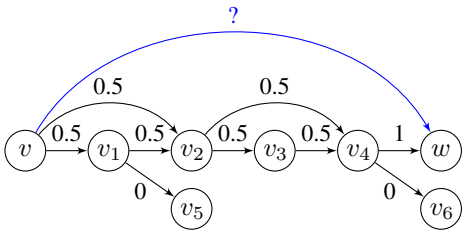

Figure 2. Failure case for lifted multicut path inequality (13). The path inequality (8) gives the correct lower bound for lifted edge y'_{vw} in this case. Example for Proposition 3.

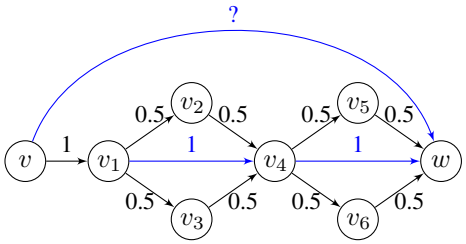

Figure 3. Exemplary case where the path inequalities (8) give a trivial lower bound on lifted edge y'_{vw}. The lifted path inequality (10) gives the correct lower bound. Example for Proposition 1.

us rewrite the right hands side of the inequality from (10) for such $P \in vw\text{-path}(G \cup G')$ where $P_{E'} = \emptyset$.

$$y'_{vw} \geq \sum_{vj:j\in P_V} y_{vj} - \sum_{i\in P_V\setminus\{v,w\}} \sum_{k\notin P_V} y_{ik} + \sum_{ij\in P_{E'}} y'_{ij} - \sum_{ij\in P_{E'}\cap E} y_{ij} = \\ = \sum_{vj:j\in P_V} y_{vj} - \sum_{i\in P_V\setminus\{v,w\}} \sum_{k\notin P_V} y_{ik}\,.$$

Which is exactly the right hand side of (8). Therefore, any pair of real vectors $(y, y') \in [0,1]^E \times [0,1]^{E'}$ that satisfies (10) must satisfy (8) as well.

- Let us prove that $S_L \subsetneq S_B$:
  We prove that there exists $(y, y') \in [0,1]^E \times [0,1]^{E'}$ such that (y, y') satisfies (8) and does not satisfy (10). See the graph in Figure 3. There are four possible paths from v to w in G. If we use Constraints (8), all the paths give us the same lower bound on y'_{vw}

$$y'_{vw} \geq 1 - 0.5 - 0.5 = 0\,.$$

If we use Constraints (10) with path $P = (vv_1, v_1v_4, v_4w)$ where $P_{E'} = \{v_1v_4, v_4w\}$, we obtain

$$y'_{vw} \geq 1 - 0.5 - 0.5 - 0.5 - 0.5 + 1 + 1 = 1\,.$$

□

Proposition 2. *The lifted path-induced cut inequalities* (11) *define a strictly tighter relaxation than the path-induced cut inequalities* (9).

Furthermore the lifted path-induced cut inequalities (11) *and* (12) *define a strictly better relaxation than* (11) *alone.*

Proof. Let us define the following sets

$$S_B = \{(y, y') \in [0,1]^E \times [0,1]^{E'} | (y, y') \text{ satisfy (9)}\}\,,$$
$$S_{L1} = \{(y, y') \in [0,1]^E \times [0,1]^{E'} | (y, y') \text{ satisfy (11)}\}\,,$$
$$S_{L2} = \{(y, y') \in [0,1]^E \times [0,1]^{E'} | (y, y') \text{ satisfy (12)}\}\,.$$

- First, we prove $S_{L1} \subset S_B$:
  We use the same argument as in the proof of Proposition 1. Every path $P \in vw\text{-paths}(G)$ belongs to the set of $vw\text{-paths}(G \cup G')$ and it holds that $P_{E'} = \emptyset$. Let us rewrite the right hands side of the inequality from (11) for such $P \in vw\text{-path}(G \cup G')$ where $P_{E'} = \emptyset$.

$$y'_{vw} \leq \sum_{i\in P_V} \sum_{\substack{k\notin P_V \\ kw\in\mathcal{R}}} y_{ik} - \sum_{ij\in P_{E'}} y'_{ij} + \sum_{ij\in P_{E'}\cap E} y_{ij} = \\ = \sum_{i\in P_V} \sum_{\substack{k\notin P_V \\ kw\in\mathcal{R}}} y_{ik}\,.$$

Which is exactly the right hand side of (9). Therefore, any pair of real vectors $(y, y') \in [0,1]^E \times [0,1]^{E'}$ that satisfies (11) must satisfy (9).

- Let us prove $S_{L1} \subsetneq S_B$:
  We prove that there exists $(y, y') \in [0,1]^E \times [0,1]^{E'}$ such that (y, y') satisfies (9) and does not satisfy (11). See the example in Figure 4. There are four possible paths in G from v to either u_1 or u_2. They are $P_1 = (vv_3, v_3u_1)$, $P_2 = (vv_2, v_2u_1)$, $P_3 = (vv_3, v_3u_2)$, $P_4 = (vv_2, v_2u_2)$. Using (11), all of them give us the same threshold on y'_{vw}:

$$y'_{vw} \leq 0.5 + 0.5 + 0 = 1\,.$$

If we use Constraint (11) with path $P = (vu_1)$, we obtain the following threshold:

$$y'_{vw} \leq 0.5 + 0.5 + 0 - 1 = 0\,.$$

- Let us prove that $S_{L1} \cap S_{L2} \subsetneq S_{L1}$
  It holds trivially that $S_{L1} \cap S_{L2} \subset S_{L1}$. Let us prove that there exists $(y, y') \in [0,1]^E \times [0,1]^{E'}$ such that $(y, y') \in S_{L1}$ and $(y, y') \notin S_{L1} \cap S_{L2}$.
  See the example graph in Figure 5. Similarly as in Figure 4, there are four possible paths from v to either u_1 or u_2 in G. There are no active lifted edges that would enable us to obtain a better upper bound on y'_{vw} using (11)

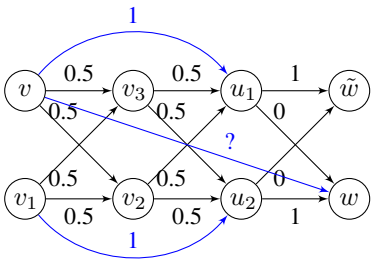

Figure 4. Exemplary case where the path-induced cut inequalities (9) fail to give non-trivial upper bounds for lifted edge y'_{vw}. The lifted path-induced cut-inequalities (11) give the correct upper bound in this case. Example for Proposition 2.

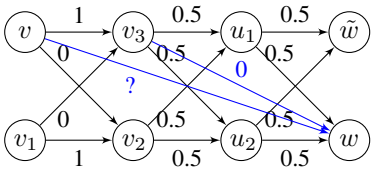

Figure 5. Exemplary failure case for the lifted path-induced cut inequalities (11). The lifted path-induced cut inequalities (12) give the correct upper bound for lifted edge y'_{vw}. Example for Proposition 2.

than the following:

$$y'_{vw} \leq 1\,.$$

However, if we use Constraints (12) with path $P = (vv_3)$ and $y'_{v_3w} = 0$, we obtain

$$y'_{vw} \leq 0\,.$$

□

10.2. Symmetric Form of Cut Inequalities

Inequalities symmetric to (9):

$$\forall vw \in E'\ \forall P \in uw\text{-paths}(G) \text{ s.t. } vu \in \mathcal{R} \wedge u \neq v: \quad y'_{vw} \leq \sum_{i \in P_V} \sum_{\substack{k \notin P_V,\\ vk \in \mathcal{R}}} y_{ki}\,. \tag{15}$$

Inequalities symmetric to (11)

$$\forall vw \in E'\ \forall P \in uw\text{-paths}(G \cup G') \text{ s.t. } vu \in \mathcal{R} \wedge u \neq v: \quad y'_{vw} \leq \sum_{i \in P_V} \sum_{\substack{k \notin P_V,\\ vk \in \mathcal{R}}} y_{ki} - \sum_{ij \in P_{E'}} y'_{ij} + \sum_{ij \in P_{E'} \cap E} y_{ij}\,. \tag{16}$$

Inequalities symmetric to (12)

$$\forall vw \in E'\ \forall P \in uw\text{-paths}\ (G \cup G') \text{ s.t. } vu \in E': \quad y'_{vw} \leq \sum_{i \in P_V \setminus u} \sum_{\substack{k \notin P_V,\\ vk \in \mathcal{R}}} y_{ki} - \sum_{ij \in P_{E'}} y'_{ij} + \sum_{ij \in P_{E'} \cap E} y_{ij} + y'_{vu}\,. \tag{17}$$

Proposition 4. *The lifted path-induced cut inequalities* (16) *define a strictly tighter relaxation than the path-induced cut inequalities* (15).
The lifted path-induced cut inequalities (16) *and* (17) *define a strictly better relaxation than* (16) *alone.*

Proof. Analogical to the proof of Proposition 2. See Figure 6 for example analogical to the one in Figure 4 and Figure 7 for example analogical to the one in Figure 5. □

Proposition 5. *1. The path-induced cut inequalities* (9) *together with their symmetric counterpart* (15) *define a strictly tighter relaxation than inequalities* (9) *alone.*

2. The path-induced cut inequalities (11) *together with their symmetric counterpart* (16) *define a strictly tighter relaxation than inequalities* (11) *alone.*

3. Using path-induced cut inequalities (17) *together with* (11), (12) *and* (16) *strictly improves the relaxation.*

Proof.
1. See the example in Figure 8.
Upper bound on y'_{vw} by (9): $y'_{vw} \leq 0.5 + 0.5 = 1$.
Upper bound on y'_{vw} by (15): $y'_{vw} \leq 0$.

2. See the example in Figure 6.
Upper bound on y'_{vw} by (11): $y'_{vw} \leq 0.5 + 0.5 = 1$.
Upper bound on y'_{vw} by (16) using path $P = (u_2w)$: $y'_{vw} \leq 0 + 0.5 + 0.5 - 1 = 0$.

3. See the example in Figure 7.
Upper bounds on y'_{vw} by (11), (12), (16): $y'_{vw} \leq 1$.
Upper bound on y'_{vw} by (17) using path $P = (uw)$ and $y'_{vu} = 0$: $y'_{vw} \leq 0$.

□

10.3. Other Valid Inequalities

Basic flow constraints (5) together with the advanced constrains on lifted edges (6)-(12) are sufficient for defining the set of feasible solutions of the lifted disjoint paths problem (3). Moreover, they define an efficient LP relaxation (Section 4) and enable efficient separation procedures (Section 5). Below, we present lifted flow inequalities specific to the lifted disjoint paths problem applied to MOT that help

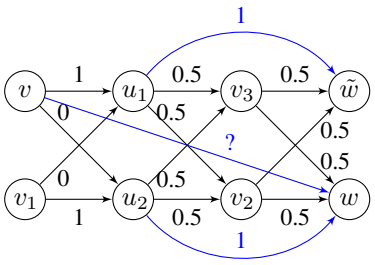

Figure 6. The best upper bound on y'_{vw} is provided by inequalities (16). Example for Proposition 4 and Proposition 5.

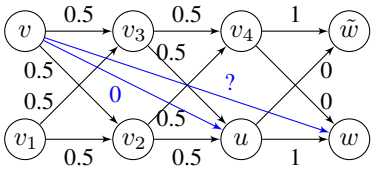

Figure 7. The best upper bound on y'_{vw} is provided by inequalities (17). Example for Proposition 4 and Proposition 5.

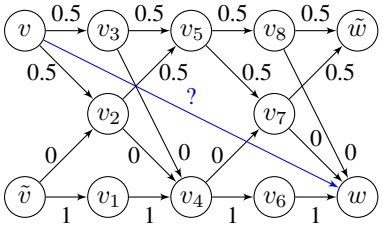

Figure 8. The best upper bound on y'_{vw} is provided by inequalities (15). Example for Proposition 5.

to improve the speed of our ILP solver. The inequalities depend on the fact that every node can be connected to maximally one node in each time frame. Therefore the number of lifted edges originating (or ending) in a given point and ending (resp. originating) in a specific time frame is at most one.

$$\forall k, l \in \{1, \dots, T\} : k > l,\ \forall v \in V_l : \sum_{vu \in E' : u \in V_k} y'_{vu} \leq x_v\,, \tag{18}$$

$$\forall k, l \in \{1, \dots, T\} : k < l,\ \forall w \in V_l : \sum_{uw \in E' : u \in V_k} y'_{uw} \leq x_w\,. \tag{19}$$

The number of constraints (18) and (19) is linear in the number of vertices. Therefore, we add them to our initial constraint set. This enables to reduce the search space for the branch and bound method in the early solver stages when only few constraints of type (8)-(12) have been added.

10.4. Proofs for Section 6 Complexity

We define $Y_{GG'}$ to be the set of all $(y, y') \in \{0,1\}^E \times \{0,1\}^{E'}$ such that (y, y') are feasible solutions of the lifted disjoint path problem (3).

Integer multicommodity flow. The integer multicommodity flow problem is defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with edge capacities $c \in \mathbb{N}^{\mathcal{E}}$ and source/sink pairs $s_i t_i$ and edge flows $f_i \in \mathbb{N}^{\mathcal{E}}$ and demands R_i, $i = 1, \dots, k$.

The aim is to send k flows from their sources to their sinks such that the flows obey the edge capacities. Formally,

$$\sum_{i=1}^{k} f_e^i \leq c_e \qquad \forall e \in E \tag{20}$$

$$\sum_{u : uv \in E} f_{uv}^i = \sum_{w : vw \in E} f_{vw}^i \qquad \forall i \in [k]\ \forall v \notin \{s_i, t_i\} \tag{21}$$

$$\sum_{v : s_i v \in E} f_{s_i v}^i \geq R_i \qquad \forall i \in [k] \tag{22}$$

where $[k]$ denotes the set $\{1, \dots, k\}$. Even has shown in (Even et al., 1976) that the integer multicommodity flow problem is NP-complete also in the case of unit capacity edges and two source sink pairs. Below we detail a construction that gives us a correspondence between edge-disjoint paths in $\mathcal{G}$ and node-disjoint paths in the transformed graph G. This construction is similar to transforming a graph into its line graph. The lifted edges in the transformed graph will count how many units of flow go from sources to sinks.

Lemma 1. *There exists a polynomial transformation from any graph $\mathcal{G}$ with source/sink pairs s_i, t_i, $i = 1, \dots, k$*

with demands R_i to a pair of graphs G and G' with edge costs c and c' respectively such that there exists a feasible integer multicommodity flow in $\mathcal{G}$ if and only if the lifted disjoint paths problem for G, G' has objective $\min_{(y,y')\in Y_{GG'}} \langle c, y\rangle + \langle c', y'\rangle \leq -\sum_{i=1}^{k} R_i$.

Proof. Without loss of generality, we consider these feasible flow sets $f_1, \dots, f_k$ where it holds $\forall i \in [k] : \sum_{s_i v \in E} f^i_{s_i v} = R_i$. Note that if the flow of commodity i is higher than its demand R_i, we can reduce it to R_i by removing the flow across one or more $s_i t_i$-paths in $\mathcal{G}$ without violating other constraints.
We first detail the graph transformation (see Figures 9 and 10).

- For all edges $ij \in \mathcal{E}$ add a vertex v_{ij} to V.
- For each pair of vertices $v_{ij}, v_{jk} \in V$ add an edge (v_{ij}, v_{jk}) to E.
- Add vertices s and t to V.
- Add to V vertices $s_i^1, s_i^2, \dots, s_i^{R_i}$ representing requirements of each commodity i.
- For each vertex s_i^r add an edge (s, s_i^r) to E.
- For each pair of vertices $s_i^r, v_{s_i j}$ add edge $(s_i^r, v_{s_i j})$ to E.
- For all $v_{kt_i} \in V$ (representing and edge from k to t_i in $\mathcal{G}$) add an edge (v_{kt_i}, t) to E.
- For all pairs of vertices $v_{s_i j}\ v_{kt_i} \in V$ add an edge $(v_{s_i j}, v_{kt_i})$ to E'. That is, the lifted edges connect all vertices representing edges from s_i in $\mathcal{G}$ with vertices representing the edges to t_i in $\mathcal{G}$.
- Cost function on base edges $\forall e \in E : c_e = 0$.
- Cost function on lifted edges $\forall e' \in E' : c'_{e'} = -1$.

An illustration of this construction can be seen in Figures 9 and 10. Note that the construction of $\mathcal{G}$ in (Even et al., 1976) allows $s_i = s_j$ for $i \neq j$. In this case, we still construct separate vertices for their incident edges in G.

Every path $\mathcal{P} = (s_i k_1, k_1 k_2, \dots, k_n t_i)$ in $\mathcal{G}$ can be assigned to a path $P = (s s_i^r, s_i^r v_{s_i k_1}, v_{s_i k_1} v_{k_1 k_2}, \dots, v_{k_n t_i} t)$ in G where $r \in [R_i]$ can be chosen arbitrarily and vice versa. Note that such a path P saturates exactly one lifted edge $(v_{s_i k_1}, v_{k_n t_i})$. Moreover, every feasible set of flow functions $f_1, \dots, f_k$ satisfying for all $i \in [k] : \sum_{s_i v \in E} f^i_{s_i v} = R_i$ defines a set of edge-disjoint paths from $s_1, \dots, s_k$ to $t_1, \dots, t_k$ in $\mathcal{G}$. This set corresponds to a set of $\sum_{i=1}^{k} R_i$ st-paths in G whose edges and vertices are disjoint and where every path saturates exactly one lifted edge $v_{s_i j} v_{kt_i}$. Every lifted edge contributes with -1 to the total cost. So, this set of disjoint st-paths has total cost $-\sum_{i=1}^{k} R_i$.
Reversely, let us have a set of vertex- and edge-disjoint st-paths in G of size $\sum_{i=1}^{k} R_i$ where every path contains some $v_{s_i j} v_{kt_i}$-path as its subpath and therefore its cost is $-\sum_{i=1}^{k} R_i$. This set defines uniquely a set of feasible flow functions $f_1, \dots, f_k$.
So, there exist feasible functions $f_1, \dots, f_k$ satisfying $f_i = R_i$ for all $i \in [k]$ iff $\min_{(y,y')\in Y_{GG'}} \gamma(y, y') \leq -\sum_{i=1}^{k} R_i$.

□

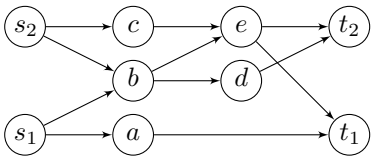

Figure 9. Integer multicommodity flow network transformation: Original graph.

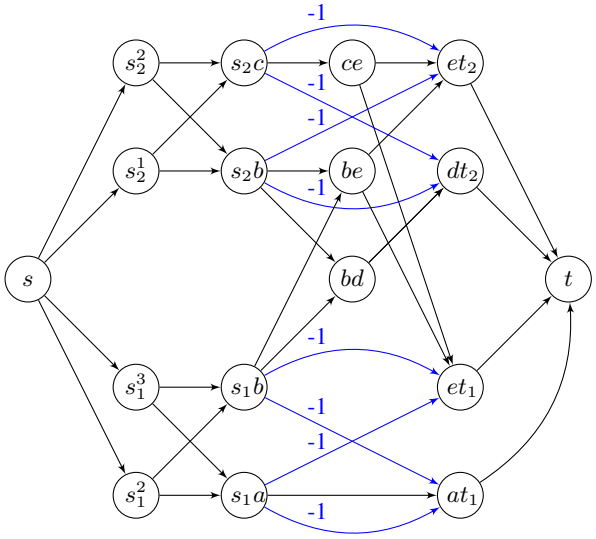

Figure 10. Integer multicommodity flow network transformation. Transformed graph from Figure 9 for flow demands $R_1 = 2, R_2 = 2$. Edges without label have cost 0.

Theorem 1. *Lifted disjoint paths problem (3) with negative lifted edges only is NP-hard.*

Proof. The NP-complete integer multicommodity flow problem with unit edge capacities can be reduced in polynomial time to the lifted disjoint paths problem (3) with negative lifted edges only. The transformation is described in Lemma 1.

□

3-SAT. The boolean satisfiability problem (SAT) is a classical NP-complete problem (Cook, 1971). A transformation

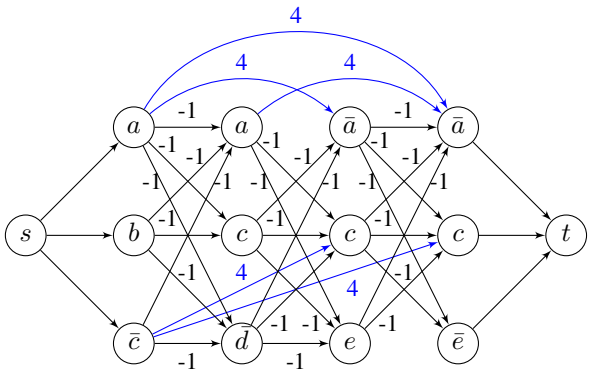

Figure 11. Reduction to lifted disjoint paths problem for 3-SAT formula $(a \vee b \vee \bar{c}) \wedge (a \vee c \vee \bar{d}) \wedge (\bar{a} \vee c \vee e) \wedge (\bar{a} \vee c \vee \bar{e})$.

from its NP-complete special version 3-SAT is commonly used for proving than a problem is NP-hard or NP-complete.

Theorem 2. *Lifted disjoint paths problem* (3) *with positive lifted edges only is NP-hard.*

Proof. Below, we detail a transformation from 3-SAT to the lifted disjoint paths problem with positive lifted edges only. For the transformation, it holds that a 3-SAT formula consisting of k clauses has a true assignment iff $\min_{(y,y') \in Y_{GG'}} \gamma(y, y') \leq -(k-1)$.

Let a 3-SAT problem containing k ordered clauses $C_1 \dots C_k$ be given. Each clause C_i consists of a conjunction of literals, which is either a variable a or its complement $\bar{a}$. We construct graphs $G = (V, E)$ and $G' = (V', E')$ as follows.

- The graph G has k layers. Every layer corresponds to one clause. Each layer contains 3 vertices labeled with the literals in the corresponding clause. Specifically, for a variable a in clause C_i we associate node v_{ia}, analoguously for a complemented variable $\bar{b}$ in clause C_i we associate node $v_{i\bar{b}}$.
- For every pair of vertices $v_{il_1} \in V$ and $v_{i+1l_2} \in V$ where $l_1 \neq \bar{l_2}$ add an edge (v_{il_1}, v_{i+1l_2}) to E and set $c_{(v_{il_1}, v_{i+1l_2})} = -1$.
- For every variable a and every pair of vertices $v_{ia}, v_{j\bar{a}} \in V$ where $j > i+1$ add an edge $(v_{ia}, v_{j\bar{a}})$ to E' and set $c'_{(v_{ia}, v_{j\bar{a}})} = k$. Do so analoguously for every pair of variables $v_{i\bar{a}}$ and v_{ja}.
- Add an edge from s to all vertices corresponding to the first clause. And an edge to t from all vertices corresponding to the last clause.

An illustration of this construction can be found in Figure 11.

Every path $P \in st\text{-paths}(G)$ that has cost $-(k-1)$ saturates vertices labelled by non-contradicting literals. We can obtain a 3-SAT solution from P as follows. If $v_{ia} \in P_V$, set variable $a := true$. If $v_{j\bar{b}} \in P$, set variable $b := false$. Variables not contained as labels of vertices in P_V can have arbitrary values.
Similarly, every solution of 3-SAT problem defines at least one path $P \in st\text{-paths}(G)$ that has cost $-(k-1)$.

□

10.5. Implementation Details on the Lifted Disjoint Paths Solver

The solver for the lifted disjoint paths problem is implemented in C++ and builds upon Gurobi 7.5. All experiments were conducted on a machine with a 6-Core Intel 2.00GHz CPU and 128 GB RAM.

10.6. Optimal data association

The experiment of Section 7.5 compares the assignments of our tracking system with the optimal assignments. We elaborate on the details to obtain the optimal assignments. We start with the pre-processed input detections, according to Section 7.2. For each frame, we compute the intersection over union between the detections and ground-truth boxes of the respective frame, which forms a weighted bipartite graph. Edges with a corresponding intersection over union below 0.5 are removed. Then, we use Hungarian matching to find a maximum-weight matching. Unmatched detections are considered as false positives, while matched detections are assigned the corresponding ground-truth label. Thus, we obtain the trajectories on the input detections using the optimal assignment. Finally, depending on the time threshold of Table 1, trajectories are synthetically splitted at skip-edges longer than the specified threshold.

10.7. Ablation study on post-processing methods.

Solving the proposed lifted disjoint paths problem establishes the assignment of input detections to object identities very close to the best possible assignment (Section 7.5).

To localize tracked objects also in the frames in which the object detector failed to detect them, some trackers apply an additional object detector on these frames based on the available input detections. This can be seen as performing interpolation and extrapolation, if viewed from the perspective of data association in a tracking-by-detection framework, e.g. see (Bergmann et al., 2019). As a result, improvements can be achieved from extending trajectories to image areas without input detections by applying of a very accurate object detector.

In order to make our tracking performance comparable with other trackers, we follow this strategy and employ an inter- and extrapolation based on (Bergmann et al., 2019).

During the inter- and extrapolation, output detections (coming from the lifted disjoint paths solver) are preserved. In particular, the detections are not rejected, reshaped, neither are their labels changed by Tracktor. Instead, we apply Tracktor to recover further locations of an object in the frames where detections of the object were missing. The procedure is based on its trajectory obtained from the lifted disjoint paths solver. Note that our adaption ignores additional, unassigned input detections, whereas the original implementation (Bergmann et al., 2019) of Tracktor fuses the detections coming from Tracktor's detector with detections provided by the dataset.

Method	MOTA	IDF1
Assignment	52.8	64.3
Assignment (optimal)	53.4	66.8
Assignment+SI	57.8	67.6
Assignment+SI*	59.5	68.9
Assignment+VI	59.6	68.5
Assignment+VI+VE	65.7	71.5
Assignment+VI+VE+SI	**67.0**	72.4

Table 3. Ablation study on inter- and extrapolation, evaluated on the MOT17 train set. SI = spatial interpolation only on sequences filmed from a static camera, SI* = spatial interpolation on all sequences, VI = visual interpolation, VE = visual extrapolation. Assignment and assignment (optimal) denote the results of the lifted disjoint paths problem and the optimal assignment, as reported in Section 7.5 given 2s time gap. Note that Tracktor's object detector is fine-tuned on MOT17Det. In our experiments, this resulted in bigger improvements on the MOT17 training set than on the test set, compare Table 2.

Table 3 reports the influence of employing inter- and extrapolation. The first two rows repeat values from Table 1 given the maximal 2s time gap. Since our solver produces nearly optimal data assignemt with respect to the used input detections, further improvements can only be achieved by applying interpolation and extrapolation on the tracks obtained by the solver.

We compare the visual interpolation (VI) as well as visual extrapolation (VE), both using the method of (Bergmann et al., 2019) with spatial interpolation (SI). For SI, we employ linear interpolation based solely on the geometric bounding box information.

The interpolation SI is applied only to sequences with a fixed camera in order to guarantee robust approximations. Still, the improvements by Assignment+SI over the baseline is evident. Especially the MOTA metric, which measures mainly the coverage of objects by detections, improves by about 10%. We also evaluate spatial interpolation for all sequences (SI*), which improves the tracker further to 59.5 MOTA and 68.9 IDF1. However, performing spatial interpolation on sequences with moving cameras can lead to error propagation. Thus, our final tracker Lif_T relies on the more robust visual interpolation and employs spatial interpolation only on sequences filmed from a static camera.

On the contrary, the visual interpolation based on (Bergmann et al., 2019) can be applied robustly to all sequences, but only in situations where the object is visible. Accordingly, the method Assignment+VI further improves over the baseline, as it is applied to more frames.

Recovering the position of tracked objects also outside of the time range of its computed trajectory (Assignment+VI+VE) further helps to improve the tracking accuracy, enhancing MOTA by about 20% and IDF1 by about 10% IDF1, as VE extends computed trajectory thereby achieving longer identity consistencies.

Finally, we employ spatial interpolation on the remaining cases where detections are missing and the objects are fully occluded (Assignment+VI+VE+SI) resulting in a slight improvement over Assignment+VI+VE.

Note that we use the method Assignment+VI+VE+SI to evaluate our tracker on the MOT15, MOT16 and MOT17 test set, as reported in Table 2. The impact of the post-processing on the training set using Tracktor seems to be very high. We conjectured this might be due to the fact that Tracktor's object detector is trained on MOT17Det (which are the detections of MOT17), leading to some degree of overfitting. Note that Tracktor is not trained the MOT17 tracking ground truth, so that it is still regarded as a meaningful validation procedure (Bergmann et al., 2019). Therefore, we created another tracker Lif_TsimInt that uses a simple interpolation, namely only linear interpolation between detections of a trajectory, for all sequences. The tracker thus corresponds to Assignment+SI*. Comparing Table 3 with Table 2, we see that indeed, the impact of the post-processing on the test set is significantly lower. We conclude that while the post-processing improves the tracking performance, the main performance of our tracker is due to our contributions.

Recall that most offline tracking systems obtain trajectories by solving a data association problem, e.g. (Henschel et al., 2018; Tang et al., 2017; Ristani & Tomasi, 2018). Our proposed tracker is able to achieve near-optimal results with respect to the input detections. Applying interpolation and extrapolation further improves the results, and makes it conceptually comparable to Tracktor. Still, with post-processing on our computed data-association, we improve over Tracktor by 25%. We argue that solving the data association accurately is important to obtain a final high-quality result after post-processing.

10.8. Further Details on the Feature Fusion Network.

We discuss in detail the neural network which fuses the input features, thereby extending Section 7.3.

Architecture of the fusion network. Considering one assignment hypothesis represented by an edge $e = vw$, the DeepMatching densities $\rho \in [0,1]^6$ as well the temporal distance t between the corresponding detections v and w serve as a confidence score for the remaining input features. They describe which of the input features is a reliable metric for a given assignment hypothesis, but they are not giving any information about the correctness of the assignment hypothesis. We transform the density features non-linearly and denote them together with the temporal distance as control features $\mathcal{C}(e) := (\log(\rho), t) \in \mathbb{R}^6 \times [0,2]$. The remaining features described in Section 7.3 are denoted as $\mathcal{F}(e) \in [0,1]^n$.

One plausible architecture is to use a convex combination of the input features, such that the coefficients depend on the control features. To this end, let $\alpha_i(\mathcal{C}(e), W_{\alpha_i})$ for $i = 1, \cdots, n$ denote a neural network with the control features as input and W_{α_i} as learnable weights. Further, let $\beta_i(\mathcal{F}(e)_i, W_{\beta_i})$ for $i = 1, \cdots, n$ be a neural network applied to i-th feature of $\mathcal{F}(e)$, with learnable weights W_{β_i}.

The input features and control features can then be fused via

$$\sum_{i=1}^{n} \alpha_i(\mathcal{C}(e), W_{\alpha_i}) \beta_i(\mathcal{F}(e)_i, W_{\beta_i}), \tag{23}$$

such that

$$\sum_{i=1}^{n} \alpha_i(\mathcal{C}(e), W_{\alpha_i}) = 1. \tag{24}$$

To ensure stable training, (23) should be applied to a sigmoid function and trained using binary cross-entropy loss.

Nonetheless, our tracker implementation employs neural network based mainly on a combination of relu units and fully connected layers, which performed slightly better, still sharing the idea of seperating the input into control features and input features. The detailed architecture is depicted in Figure 12.

Training details. Training of the neural network is performed directly on the (preprocessed) input detections. Labels are retrieved by assigning each detection to the best fitting ground-truth bounding box. Detections with ambiguous assignments are ignored within the training phase.

In order to train the edge classifier, special care has to be taken as the training set is highly imbalanced. The number of edges which correspond to true negatives (pairs of detections which do not belong to the same person) clearly dominates the number of true positive edges (pairs of detections belonging to the same person).

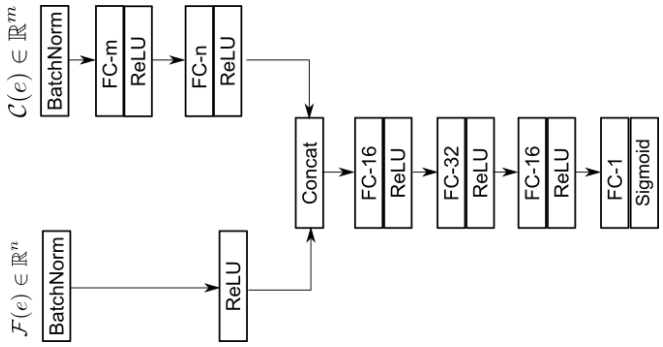

Figure 12. The architecture of the edge classifier used in Lif_T. FC-i denotes a fully-connected layer with i nodes in as outputs. Using a concatenation with subsequent fully connected layer, m control features and n input features are fused.

To address this issue, the network is trained on a randomly sampled subset of all possible edges, such that the ratio of true positive edges and true negative edges per time distance between the end nodes of the edges remains fixed. The maximal temporal distance of an edge is set to 2 seconds, allowing to recover persons even after long occlusions.

The weights of the fusion network are optimized according to the binary cross-entropy loss. We employ stochastic gradient descent with the learning rate set to 10^{-2} and Nesterov momentum set to 0.9, for a total of 10 epochs. Training and inference is performed using Pytorch 1.3 on a Nvidia RTX 2080 Ti.

Accuracy of the fusion network. The performance of a tracking system depends highly on the accuracy of the edge classifier (and the corresponding edge weights).

Therefore, we report our evaluation of the edge classifier on all training sequences of the filtered MOT17 train set in Table 4. Together with Table 5 and Table 2, it shows that improvements in the tracking features directly correlate to high quality tracking results thanks to the proposed solver. While Table 4 shows very good performance of the edge classifier, a powerful graph model and solver is still crucial to obtain high quality tracking results. Even small errors (we observed 5% maximal error) in the edge classifier can cause many errors in the tracking results if an unsuitable procedure is used. Also note that for training the edge classifier, detections with ambiguous assignment to the ground truth boxes were ignored. So, these potentially difficult cases are excluded int the evaluation of the edge classifier. Especially the interpolation and extrapolation is prone to error propagation, once a single identity switch has been created, which heavily affects, among others, the IDF1 score. Our lifted

Sequence	Acc ↑	Prec ↑	TPR ↑	TNR ↑
MOT17-02-DPM	1.00	0.99	1.00	1.00
MOT17-04-DPM	1.00	0.98	0.99	1.00
MOT17-05-DPM	0.95	0.95	1.00	0.99
MOT17-09-DPM	1.00	0.98	0.98	1.00
MOT17-10-DPM	1.00	0.99	0.99	1.00
MOT17-11-DPM	1.00	1.00	0.99	1.00
MOT17-13-DPM	0.99	0.97	0.96	1.00
MOT17-02-SDP	1.00	0.96	1.00	1.00
MOT17-04-SDP	1.00	0.98	0.98	1.00
MOT17-05-SDP	0.99	0.92	1.00	0.98
MOT17-09-SDP	0.97	0.81	0.99	0.97
MOT17-10-SDP	0.99	0.94	0.97	1.00
MOT17-11-SDP	1.00	0.99	0.99	1.00
MOT17-13-SDP	0.99	0.90	0.96	0.99
MOT17-02-FRCNN	1.00	0.98	1.00	1.00
MOT17-04-FRCNN	1.00	0.97	0.99	1.00
MOT17-05-FRCNN	0.99	0.94	1.00	1.00
MOT17-09-FRCNN	0.99	0.97	0.98	1.00
MOT17-10-FRCNN	0.99	0.95	0.98	1.00
MOT17-11-FRCNN	1.00	0.99	0.99	1.00
MOT17-13-FRCNN	0.99	0.90	0.95	0.99

Table 4. Performance metrics on the edge classifier. The performance is measured in terms of the accuracy (Acc), precision (Prec), true positive rate (TPR) and true negative rate (TNR). The arrows indicate that higher metric values are better.

disjoint paths formulation can be advantageous, since lifted edges aggregate multiple edge classifiers which can correct individual wrong classifications of single edges.

10.9. Extended Quantitative Results

We provide additional evaluations on our tracking system as well as on the lifted disjoint paths solver.

Detailed tracking evaluations. We provide the evaluations of the MOT15, MOT16, MOT17 test sets as well as the MOT17 train set per sequence in Table 5. In addition, the table contains the solver time (STime) in seconds, needed to solve the corresponding lifted disjoint paths problem.

	Sequence	MOTA↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDS↓	Frag↓	STime↓
MOT17-Train	MOT17-02-DPM	40.5	50.3	13	29	19	11017	26	23	127
	MOT17-04-DPM	69.9	73.9	41	22	298	13986	38	41	1521
	MOT17-05-DPM	58.2	67.0	31	40	40	2824	27	65	36
	MOT17-09-DPM	72.9	71.6	14	1	58	1370	15	7	59
	MOT17-10-DPM	67.4	70.2	26	8	106	4043	39	82	173
	MOT17-11-DPM	67.3	73.9	24	26	55	3017	11	28	115
	MOT17-13-DPM	63.6	67.2	45	36	64	4127	43	48	59
	MOT17-02-FRCNN	47.4	57.2	15	22	89	9656	26	27	229
	MOT17-04-FRCNN	67.5	74.1	38	21	98	15310	29	13	1535
	MOT17-05-FRCNN	60.2	68.9	35	36	73	2651	30	62	92
	MOT17-09-FRCNN	71.5	72.9	14	1	54	1451	10	7	51
	MOT17-10-FRCNN	73.2	76.2	33	2	270	3096	73	145	398
	MOT17-11-FRCNN	73.1	78.8	32	18	82	2436	18	27	133
	MOT17-13-FRCNN	77.1	75.8	68	10	203	2394	73	89	388
	MOT17-02-SDP	55.0	61.3	16	16	65	8236	52	50	586
	MOT17-04-SDP	77.7	81.8	46	13	243	10296	49	66	4133
	MOT17-05-SDP	64.0	69.5	41	22	105	2351	33	84	80
	MOT17-09-SDP	73.0	73.0	14	1	69	1356	12	12	127
	MOT17-10-SDP	75.0	78.6	35	2	349	2759	105	160	756
	MOT17-11-SDP	74.4	78.4	36	14	115	2277	27	36	198
	MOT17-13-SDP	70.8	71.4	62	24	200	3150	55	81	364
	MOT17-Train	67.0	72.4	679	364	2655	107803	791	1153	11430
MOT17-Test	MOT17-01-DPM	48.3	58.1	8	11	68	3258	10	19	38
	MOT17-03-DPM	73.3	70.1	82	17	3560	24276	160	256	24311
	MOT17-06-DPM	58.1	64.7	61	77	178	4728	28	155	113
	MOT17-07-DPM	44.4	52.3	7	21	155	9176	60	209	297
	MOT17-08-DPM	34.7	47.4	18	37	254	13507	32	44	146
	MOT17-12-DPM	48.3	62.3	18	41	35	4437	11	52	68
	MOT17-14-DPM	36.1	48.8	12	77	268	11449	91	239	323
	MOT17-01-FRCNN	47.7	58.1	8	10	246	3119	7	24	79
	MOT17-03-FRCNN	72.2	71.8	71	17	2664	26277	124	250	11678
	MOT17-06-FRCNN	60.4	63.7	68	61	279	4358	32	207	203
	MOT17-07-FRCNN	44.0	54.9	8	20	279	9110	63	227	281
	MOT17-08-FRCNN	31.9	43.3	17	37	383	13973	35	59	130
	MOT17-12-FRCNN	47.3	58.0	16	43	37	4521	11	34	84
	MOT17-14-FRCNN	36.2	49.0	16	72	629	11061	108	358	359
	MOT17-01-SDP	47.8	57.8	9	10	346	3008	10	31	95
	MOT17-03-SDP	78.2	77.3	92	13	3778	18879	132	323	16219
	MOT17-06-SDP	60.3	65.1	67	64	305	4345	33	217	144
	MOT17-07-SDP	45.8	55.0	8	18	285	8793	71	280	483
	MOT17-08-SDP	34.8	47.7	18	34	429	13288	48	69	202
	MOT17-12-SDP	47.3	60.7	18	42	158	4394	14	53	85
	MOT17-14-SDP	38.3	51.4	15	69	630	10662	109	370	376
MOT16	MOT16-01	48.3	58.2	8	10	78	3217	10	19	38
	MOT16-03	73.0	69.9	80	17	3732	24329	159	310	24311
	MOT16-06	58.2	64.7	62	77	249	4548	29	159	113
	MOT16-07	45.6	53.4	7	16	189	8637	57	212	297
	MOT16-08	43.4	55.7	18	24	284	9149	32	44	146
	MOT16-12	50.2	64.0	18	37	44	4072	11	51	68
	MOT16-14	36.1	48.8	12	77	268	11449	91	239	323
2D MOT15	ADL-Rundle-1	39.6	60.8	13	2	2277	3303	44	175	325
	ADL-Rundle-3	59.2	69.9	23	7	902	3217	29	42	153
	AVG-TownCentre	61.8	67.3	96	33	417	2217	99	213	20
	ETH-Crossing	57.6	69.3	7	9	35	387	3	18	2
	ETH-Jelmoli	51.4	67.1	18	14	520	701	12	44	20
	ETH-Linthescher	53.7	62.2	42	98	318	3795	21	95	11
	KITTI-16	36.2	32.7	5	1	456	521	108	60	57
	KITTI-19	43.3	49.4	11	17	467	2315	249	142	135
	PETS09-S2L2	56.9	43.6	9	2	476	3531	152	225	180
	TUD-Crossing	88.0	90.9	11	0	64	62	6	13	13
	Venice-1	45.8	62.1	9	3	905	1561	7	20	30

Table 5. We provide the results of our tracker Lif_T, evaluated per sequence. In addition, we provide the time necessary to solve the corresponding lifted disjoint path problem instance (STime), in seconds. Arrows indicate whether low or high metric values are better. Tracking results on the test sets were evaluated by the MOTChallenge server https://www.motchallenge.net