Making Higher Order MOT Scalable: An Efficient Approximate Solver for Lifted Disjoint Paths

Andrea Hornakova*[1] Timo Kaiser*[2] Paul Swoboda[1] Michal Rolinek[3]
Bodo Rosenhahn[2] Roberto Henschel[2]

*Authors contributed equally, [1]Max Planck Institute for Informatics, Saarland Informatics Campus, [2]Institute for Information Processing, Leibniz University Hannover, [3]Max Planck Institute for Intelligent Systems, Tübingen

Abstract

We present an efficient approximate message passing solver for the lifted disjoint paths problem (LDP), a natural but NP-hard model for multiple object tracking (MOT). Our tracker scales to very large instances that come from long and crowded MOT sequences. Our approximate solver enables us to process the MOT15/16/17 benchmarks without sacrificing solution quality and allows for solving MOT20, which has been out of reach up to now for LDP solvers due to its size and complexity. On all these four standard MOT benchmarks we achieve performance comparable or better than current state-of-the-art methods including a tracker based on an optimal LDP solver.

1. Introduction

Deriving high-level understanding from a video is a desired task that has been studied in computer vision for a long time. Nevertheless, solving the problem is a long way off. A computer vision system able to extract the motions of objects appearing in a video in terms of trajectories is considered as a prerequisite for the goal. This task called multiple object tracking (MOT) has numerous applications, e.g. in the area of video surveillance [18], sports analysis [2, 42], urban planning [3], or autonomous driving [39, 19].

Yet, solving MOT is challenging, especially for long and crowded sequences. The predominant approach for MOT is the tracking-by-detection paradigm, which splits the problem into two subtasks. First, objects are detected in all video frames by an object detector. Then, the detections are linked across frames to form trajectories. While the performance of object detectors has improved considerably by latest advances of CNNs [47, 63, 46, 17], the latter task called the *data association* remains challenging. The data association reasons from *pairwise costs*, which indicate for each pair of detections the likelihood of belonging to the same object.

Appearance and spatio-temporal information are often ambiguous, especially in crowded scenes, so that pairwise costs can be misleading. Moreover, object detectors produce more errors in crowded scenes due to partial occlusions. To resolve these issues, it is crucial that the data association incorporates global context.

The disjoint paths problem (DP) [65, 35] is a natural model for MOT. Results are computed efficiently using a min-cost flow algorithm that delivers the global optimal solution. Unfortunately, the integration of long range temporal interactions is limited, as DP obeys the first-order Markov-chain assumption: for each trajectory, consistency is ensured only between directly linked detections, which is a strong simplification that ignores higher order consistencies among multiple linked detections.

To fix this deficiency, [28] generalizes DP to lifted disjoint paths (LDP) by using additional connectivity priors in terms of *lifted* edges. This makes the formulation much more expressive while it maintains the feasibility set of the DP (Sec. 3). The optimization problem enables to take into account pairwise costs between arbitrary detections belonging to one trajectory. It thus enables to incorporate long range temporal interactions effectively and leads to considerable improvement of recall and precision [28]. Similar extensions have been made for the multicut problem [56, 57].

While the integration of the global context by LDP is crucial to obtain high-quality tracking results, it makes the data association problem NP-hard. Still, [28] presented a global optimal LDP solver usable for semi-crowded sequences with reasonable computational effort. However, when applied to longer and crowded sequences, such approaches are not tractable anymore, due to too high demands on runtime and memory.

In order to close this gap, we present the first approximate solver for LDP. The resulting tracker scales to big problem instances and incorporates global context with similar accuracy as the global optimal LDP solver. Moreover, our solver outputs certificates in terms of primal/dual gaps.

In particular, our solver is based on a Lagrangean (dual) decomposition of the problem. This dual is iteratively optimized by dual block coordinate ascent (a.k.a. message passing) using techniques from [54], see Sec. 4.1. The decomposition relies on subproblems that are added in a cutting plane fashion. We obtain high-quality primal solutions by solving minimum cost flow problems with edge costs synthesizing information from both base and lifted edges from the dual task and improve them via a local search procedure.

We validate the quality of the solver on four standard MOT benchmarks (Sec. 5). We achieve comparable or better performance w.r.t. the current state-of-the-art trackers including the tracker based on the optimal LDP solver [28] on MOT15/16/17 [37, 45]. Furthermore, our proposed tracker performs on par with state-of-the-art on the more challenging MOT20 dataset [16] which is composed of long and crowded sequences. Therefore, lightweight features and a fast solver are crucial to perform tracking on such massive sequences. Our work thus extends the applicability of the successful LDP formulation to a wider range of instances.

Contribution of this work is in summary as follows:

- We make the LDP problem more accessible and applicable by introducing an approximate solver with better scalability properties than the global optimal LDP solver, while resulting in similar tracking performance, and being independent of Gurobi [22].
- We present an MOT system that is scalable to challenging sequences by using considerably less computationally demanding features than what is used in the state-of-the-art tracker [28]. Our system incorporates higher order consistencies in a scalable way, i.e. it uses an approximate solver and provides a gap to the optimum.

We make our LDP solver[1] and our MOT pipeline[2] available.

2. Related Work

(Lifted) disjoint paths. The disjoint paths problem is a natural model for multiple object tracking and is solvable with fast combinatorial solvers [35]. It has been used for the data association step of MOT in [6, 65]. Its extensions have been used for fusing different object detectors [12] or multi-camera MOT [27, 38]. Its main disadvantage is that it does not allow to integrate long range information because it evaluates only direct connections between object detections within a trajectory. The lifted disjoint paths problem introduced in [28] enhances DP by introducing lifted edges that enable to reward or penalize arbitrary connections between object detections. This incorporation of long range information leads to a significant improvement of the tracking performance yielding state-of-the-art results on main MOT benchmark but makes the problem NP-hard. The authors provide a globally optimal solver using Gurobi [22]. Despite a lot of efficient subroutines, the general time complexity of the provided solver remains exponential. Therefore, in order to extend LDP-based methods to highly dense MOT problems as in MOT20 it is crucial to reduce the complexity of the used LDP solver because the number of feasible connections between detections increases dramatically.

Multicut and lifted multicut. LDP is similar to (lifted) multicut [14, 29]. Multicut has been used for MOT in [26, 33, 36, 51, 55, 56], lifted multicut in [5, 57]. These trackers solve the underlying combinatorial problem approximately via heuristics without providing an estimation of the gap to optimality. Our approach, delivers an approximate solution together with a lower bound enabling to assess the quality of the solution. Additionally, LDP provides a strictly better relaxation than lifted multicut [28].

Other data association models for MOT. Several works employ greedy heuristics to obtain tracking results [9, 68, 7]. Such strategies normally suffer from occlusions or ambiguous situations, causing trajectory errors. Others use bipartite matchings [31, 52, 69, 62, 61, 60] to assign new detections with already computed trajectories of the past optimally. Since no global context is incorporated, they are prone to errors if individual edge costs are misleading.

Higher order MOT frameworks ensure consistencies within all detections of a trajectory. This can be done greedily, by computing one trajectory at a time via a generalized minimum clique problem [64], or globally using an extension to the maximum multi clique problem [15].

Several works employ continuous domain relaxation. When MOT is formulated as a binary quadratic program [23, 25, 58, 24], a modification of the Frank-Wolfe algorithm adapted to the non-convex case has been used [23]. Some approximations for binary linear programs use an LP-relaxation, optimize in the continuous domain and derive a binary solution from the continuous one [32, 12, 11]. They however do not provide the optimality gap, in contrast to our work. Higher order MOT can be considered as a classification problem using graph convolutions [10]. It allows to train features directly on the tracking task.

The multigraph-matching problem, a generalization of the graph matching problem, has been used for MOT [30]. Here, cycle consistency constraints of the multi-graph matching ensures higher order consistencies of the trajectories. Message passing for higher order matching in MOT has been used in [4] employing a variant of MPLP [20]. In contrast to our formulation, [4] does not model occlusions and does not allow for connectivity priors.

Probabilistic approaches to multiple-target tracking include multiple hypotheses tracking [34, 13], joint probabiblistic data association [48, 53] and others [53, 44].

3. Problem Formulation

The lifted disjoint paths problem (LDP) introduced in [28] is an optimization problem for finding a set of

[1] https://github.com/LPMP/LPMP
[2] https://github.com/TimoK93/ApLift

vertex-disjoint paths in a directed acyclic graph. The cost of each path is determined by the cost of edges in that path as in the basic disjoint paths problem (DP), but additionally there are higher order costs defined by lifted edges. A lifted edge contributes to the cost if its endpoints are part of the same path. This problem is a natural formulation for multiple object tracking (MOT), where lifted edges allow to re-identify the same objects over long distance.

While of greater expressivity, the LDP is NP-hard [28] in contrast to DP which is reducible to the minimum cost flow. Below, we recapitulate the formulation of LDP from [28].

3.1. Notation and Definitions.

Flow network: a directed acyclic graph $G = (V, E)$.
Start and terminal: nodes $s, t \in V$.
Lifted graph: a directed acyclic graph $G' = (V', E')$, where $V' = V \backslash \{s, t\}$.
The set of paths starting at v and ending in w is

$$vw\text{-paths}(G) = \left\{ (v_1 v_2, \dots, v_{l-1} v_l) : \begin{array}{c} v_i v_{i+1} \in E, \\ v_1 = v, v_l = w \end{array} \right\}. \tag{1}$$

For a vw-path P its edge set is P_E and its node set is P_V.
Reachability relation for two nodes $v, w \in V$ is defined as $vw \in \mathcal{R}_G \Leftrightarrow vw\text{-paths}(G) \neq \emptyset$. We assume that it is reflexive and $su \in \mathcal{R}_G, ut \in \mathcal{R}_G \; \forall u \in V$, i.e. all nodes can be reached from s and all nodes can reach the sink node t.
Flow variables: Variables $y \in \{0,1\}^E$ have value 1 if flow passes through the respective edges.
Node variables $z \in \{0,1\}^V$ denote flow passing through each node. Values 0/1 forces paths to be node-disjoint.
Variables of the lifted edges E' are denoted by $y' \in \{0,1\}^{E'}$. $y'_{vw} = 1$ signifies that nodes v and w are connected via the flow y in G. Formally,

$$y'_{vw} = 1 \Leftrightarrow \exists P \in vw\text{-paths}(G) : \forall ij \in P_E : y_{ij} = 1\,. \tag{2}$$

Lifted disjoint paths problem. Given edge costs $c \in \mathbb{R}^E$, node cost $d \in \mathbb{R}^V$ in flow network G and edge cost $c' \in \mathbb{R}^{E'}$ for the lifted graph G' the lifted disjoint paths problem is

$$\begin{aligned} \min_{\substack{y \in \{0,1\}^E, y' \in \{0,1\}^{E'}, \\ z \in \{0,1\}^V}} \quad & \langle c, y \rangle + \langle c', y' \rangle + \langle d, z \rangle \\ \text{s.t.} \quad & y \text{ node-disjoint } s,t\text{-flow in } G, \\ & z \text{ flow through nodes of } G \\ & y, y' \text{ feasible according to (2)} \end{aligned} \tag{3}$$

Set E' can be arbitrary. It makes sense to create a lifted edge vw only if $vw \in \mathcal{R}_G$ due to Formula (2) and only if v and w do not belong to neighboring frames. We describe our choice in Sec. 5.2.

Other notation and abbreviations are in Appendix 8.1.

4. Lagrange Decomposition Algorithm for LDP

Below we recapitulate Lagrange decomposition and the message passing primitive used in our algorithm (Sec. 4.1). Then, we propose a decomposition of the LDP problem (3) into smaller but tractable subproblems (Sec. 4.2-4.4). This decomposition is a dual task to an LP-relaxation of (3). Therefore, it provides a lower bound that is iteratively increased by the message passing. We solve Problem (3) in a simplified version of Lagrange decomposition framework developed in [54]. Our heuristic for obtaining primal solutions uses the dual costs from the subproblems (Sec. 4.6).

4.1. Lagrange Decomposition

We have an optimization problem $\min_{x \in \mathcal{X}} \langle c, x \rangle$ where $\mathcal{X} \subseteq \{0,1\}^n$ is a feasible set and $c \in \mathbb{R}^n$ is the objective vector. Its Lagrange decomposition is given by a set of *subproblems* $\mathcal{S}$ with associated *feasible sets* $\mathcal{X}^{\mathsf{s}} \subseteq \{0,1\}^{d_{\mathsf{s}}}$ for each $\mathsf{s} \in \mathcal{S}$. Each coordinate i of $\mathcal{X}^{\mathsf{s}}$ corresponds to one coordinate of $\mathcal{X}$ via an injection $\pi_{\mathsf{s}} : [d_{\mathsf{s}}] \rightarrow [n]$ alternatively represented by a matrix $A^{\mathsf{s}} \in \{0,1\}^{d_{\mathsf{s}}, n}$ where $(A^{\mathsf{s}})_{ij} = 1 \Leftrightarrow \pi_{\mathsf{s}}(i) = j$. For each pair of subproblems $\mathsf{s}, \mathsf{s}' \in \mathcal{S}$ that contain a pair of coordinates i, j such that $\pi_{\mathsf{s}}(i) = \pi_{\mathsf{s}'}(j)$, we have a *coupling constraint* $x_i^{\mathsf{s}} = x_j^{\mathsf{s}'}$ for each $x^{\mathsf{s}} \in \mathcal{X}^{\mathsf{s}}, x^{\mathsf{s}'} \in \mathcal{X}^{\mathsf{s}'}$.

We require that every feasible solution $x \in \mathcal{X}$ is feasible for the subproblems, i.e. $\forall x \in \mathcal{X}, \forall \mathsf{s} \in \mathcal{S} : A^{\mathsf{s}} x \in \mathcal{X}^{s}$.

We require that the objectives of subproblems are equivalent to the original objective, i.e. $\langle c, x \rangle = \sum_{s \in \mathcal{S}} \langle \theta^s, A^{\mathsf{s}} x \rangle$ $\forall x \in \mathcal{X}$. Here, $\theta^{\mathsf{s}} \in \mathbb{R}^{d_{\mathsf{s}}}$ defines the *objective* of subproblem s.

The *lower bound* of the Lagrange decomposition given the costs θ^{s} for each $\mathsf{s} \in \mathcal{S}$ is

$$\sum_{\mathsf{s} \in \mathcal{S}} \min_{x^{\mathsf{s}} \in \mathcal{X}^{\mathsf{s}}} \langle \theta^{\mathsf{s}}, x^{\mathsf{s}} \rangle\,. \tag{4}$$

Given coupling constraint $x_i^{\mathsf{s}} = x_j^{\mathsf{s}'}$ and $\gamma \in \mathbb{R}$, a sequence of operations of the form $\theta_i^{\mathsf{s}} \mathrel{+}= \gamma$, $\theta_j^{\mathsf{s}'} \mathrel{-}= \gamma$ is called a *reparametrization*.

Feasible primal solutions are invariant under reparametrizations but the lower bound (4) is not. The optimum of the dual lower bound equals to the optimum of a convex relaxation of the original problem, see [21].
Min-marginal message passing. Below, we describe reparametrization updates monotonically non-decreasing in the lower bound based on *min-marginals*. Given a variable x_i^{s} of a subproblem $\mathsf{s} \in S$, the associated *min-marginal* is

$$m_i^{\mathsf{s}} = \min_{x^{\mathsf{s}} \in \mathcal{X}^{\mathsf{s}} : x_i^{\mathsf{s}} = 1} \langle \theta^{\mathsf{s}}, x^{\mathsf{s}} \rangle - \min_{x^{\mathsf{s}} \in \mathcal{X}^{\mathsf{s}} : x_i^{\mathsf{s}} = 0} \langle \theta^{\mathsf{s}}, x^{\mathsf{s}} \rangle \tag{5}$$

i.e. the difference between the optimal solutions with the chosen variable set to 1 resp. 0.

Proposition 1 ([54]). *Given a coupling constraints $x_i^{\mathsf{s}} = x_j^{\mathsf{s}'}$ and $\omega \in [0,1]$ the following operation is non-decreasing w.r.t. the dual lower bound* (4)

$$\theta_i^{\mathsf{s}} \mathrel{-}= \omega \cdot m_i^{\mathsf{s}}, \qquad \theta_j^{\mathsf{s}'} \mathrel{+}= \omega \cdot m_i^{\mathsf{s}}. \tag{6}$$

The goal of reparametrization is two-fold. (i) Improving the objective lower bound to know how far our solution is from the optimum. (ii) Using reparametrized costs as the input for our primal heuristic yields high-quality primal solutions. The key components are efficient computations of (i) optima of subproblems for obtaining lower bound (4), (ii) constrained optima for obtaining min-marginals (5) and (iii) a primal heuristic using the reparametrized costs (Sec. 4.6). Lagrange decomposition has been used for other problems but the subproblem decomposition and minimization procedures are problem specific. Therefore, developing them for LDP is an important contribution for solving LDP in a scalable way while keeping a small gap to an optimum.

4.2. Inflow and Outflow Subproblems

For each node $v \in V$ of the flow graph, we introduce two subproblems: An inflow and an outflow subproblem. The subproblems contain all incoming resp. outgoing edges of node v together with the corresponding node. Formally, inflow resp. outflow subproblems contain the edges $\delta_E^-(v) \cup \delta_{E'}^-(v)$, resp. $\delta_E^+(w) \cup \delta_{E'}^+(w)$. Here, we adopt the standard notation where $\delta_E^-(v)$, resp. $\delta_E^+(v)$ denote all base edges incoming to v, resp. outgoing from v. Similarly, $\delta_{E'}^-(v), \delta_{E'}^+(v)$ denote lifted edges incoming to, resp. outgoing from v.

The feasible set $\mathcal{X}_v^{out}$ of the outflow subproblem for node v is defined as

$$\left\{ \begin{array}{l} z_v^{out} \in \{0,1\}, y^{out} \in \{0,1\}^{\delta_E^+(v)}, y'^{out} \in \{0,1\}^{\delta_{E'}^+(v)} : \\ (z_v^{out}, y^{out}, y'^{out}) = \mathbb{0} \vee \\ \exists P \in vt\text{-paths}(G) \text{ s.t.} \quad \begin{array}{r} z_v^{out} = 1 \\ y_{vw}^{out} = 1 \Leftrightarrow vw \in P_E \\ y'^{out}_{vu} = 1 \Leftrightarrow u \in P_V \end{array} \end{array} \right\}. \tag{7}$$

Consequently, either there is no flow going through vertex v and all base and lifted edges have label zero. Alternatively, there exists a vt-path P in G labeled by one. In this case the base edge adjacent to v corresponding to the first edge in P is one. All lifted edges connecting v with vertices of P also have value one. All other base and lifted edges are zero. Each feasible solution of the outflow subproblem can be represented by a path vt-path P. The feasible set of the inflow subproblem $\mathcal{X}_v^{in}$ is defined analogously. We sometimes omit the superscipts *out* for better readability.

Constraints between inflow and outflow subproblems. For node variables, we add the constraint $z_v^{in} = z_v^{out}$. For an edge $vw \in E \cup E'$ we require the shared edge in the outflow subproblem of v and in the inflow subproblem for w to agree, i.e. $y_{vw}^{out} = y_{vw}^{in}$ if $vw \in E$ and $y'^{out}_{vw} = y'^{in}_{vw}$ if $vw \in E'$.

Optimization of in- and outflow subproblems. Given costs θ^{out}, the optimal solution of an outflow problem for node v can be computed by depth-first search on the subgraph defined by the vertices reachable from v.

The algorithms rely on the following data structures:

- lifted_costs$[u]$ contains the minimum cost of all ut-paths w.r.t. to costs of all lifted edges connecting v with the vertices of the path.
- next$[u]$ contains the best neighbor of vertex u w.r.t. values in lifted_cost. That is, next$[u]$ = argmin$_{w:uw\in\delta_E^+(u)}$ lifted_cost$[w]$

Algorithm 1 Opt-Out-Cost

```
Input start vertex $v$, edge costs $\tilde\theta$
Output optimal value opt, lifted_cost $\forall w : vw \in \delta_E^+(v)$ optimal solution for $vw$ active $\alpha_{vw}$
1: for $u \in V : vu \in \mathcal{R}_G$ do
2:     lifted_cost$[u] = \infty$, next$[u] = \emptyset$
3: end for
4: lifted_cost$[t] = 0$, next$[t] = t$
5: Lifted-Cost-DFS-Out($v, v, \tilde\theta$, lifted_cost, next)
6: $\forall w : vw \in \delta_E^+(v) : \alpha_{vw} = \tilde\theta_v + \tilde\theta_{vw}$ + lifted_cost$[w]$
7: opt = $\min(\min_{vw\in\delta_E^+(v)} \alpha_{vw}, 0)$
```

Algorithm 2 Lifted-Cost-DFS-Out

```
Input $v, u, \tilde\theta$, lifted_cost, next
Output lifted_cost, next
1: $\alpha = 0$
2: for $uw \in \delta_E^+(u)$ do
3:     if next$[w] = \emptyset$ then Lifted-Cost-DFS-Out($v, w, \tilde\theta$)
4:     if lifted_cost$[w] < \alpha$ then
5:         $\alpha$ = lifted_cost$[w]$, next$[u] = w$
6:     end if
7: end for
8: if next$[u] = \emptyset$ then next$[u] = t$
9: lifted_cost$[u] = \alpha + \tilde\theta'_{vu}$
```

Alg. 1 and 2 give a general dept first search (DFS) procedure that, given a vertex v, computes optimal paths from all vertices reachable from v. Alg. 1 takes as input vertex v and edge costs $\tilde\theta$. Its subroutine Alg. 2 computes recursively for each vertex u reachable from v the value lifted_cost$[u]$. The overall optimal cost $\min_{(z,y,y')\in\mathcal{X}_v^{out}} \langle \tilde\theta, (z,y,y') \rangle$ of the subproblem is given by the minimum of node and base edge and lifted edges costs $\min_{vu\in\delta_E^+(v)} \tilde\theta_v^{out} + \tilde\theta_{vu}^{out}$ + lifted_cost$[u]$. We achieve linear complexity by exploiting that subpaths of minimum cost paths are minimal as well. The optimization for the inflow subproblem is analogous.

Message passing for in- and outflow subproblems. We could compute one min-marginal (5) by adapting Alg. 1 and forcing an edge to be taken or not. However, computing min-marginals one-by-one with performing operation (6) would be inefficient, since it would involve calling Alg. 1 $\mathcal{O}(|\delta^+_E(v)|+|\delta^+_{E'}(v))|$ times. Therefore, we present efficient algorithms for computing a sequence of min-marginals in Appendix 8.2. The procedures save computations by choosing the order of edges for computing min-marginals suitably and reuse previous calculations.

4.3. Path Subproblems

The subproblem contains a lifted edge vw and a path P from v to w consisting of both base and lifted edges. They reflect that (i) lifted edge vw must be labelled 1 if there exists an active path between v and w, and (ii) there cannot be exactly one inactive lifted edge within path P if vw is active. The reason is that the inactive lifted edge divides P into two segments that must be disconnected. This is contradictory to activating lifted edge vw because it indicates a connection between v and w. Path subproblems are similar to cycle inequalities for the multicut [14].

In order to distinguish between base and lifted edges of path P, we use notation $P_E = P \cap E$ and $P_{E'} = P \cap E'$. For the purpose of defining the feasible solutions of path subproblems, we define strong base edges $E_0 = \{vw \in E | vw\text{-paths}(G) = \{vw\}\}$. That is, base edge vw is strong iff there exists no other vw-path in graph G than vw itself.

The feasible set $\mathcal{X}^P$ of the path subproblem for vw-path P is defined as

$$y \in \{0,1\}^{P_E}, y' \in \{0,1\}^{P_{E'} \cup \{vw\}} :$$

$$\forall kl \in P_{E'} \cup \{vw\} : \tag{8}$$

$$\sum_{ij \in P_E} (1-y_{ij}) + \sum_{ij \in P_{E'} \cup \{vw\} \setminus \{kl\}} (1-y'_{ij}) \geq 1 - y'_{kl} \,,$$

$$\forall kl \in P_E \cap E_0 : \tag{9}$$

$$\sum_{ij \in P_E \setminus kl} (1-y_{ij}) + \sum_{ij \in P_{E'} \cup \{vw\}} (1-y_{ij}) \geq 1 - y_{kl} \,.$$

Equation (8) requires that a lifted edge in $P_{E'}$ or vw can be zero only if at least one other edge of the subproblem is zero. Equation (9) enforces the same for strong base edges.

The optimization of path subproblems is detailed in Alg. 12 in the Appendix. The principle is as follows. It checks whether there exists exactly one positive edge and whether it is either a lifted or a strong base edge. If so, the optimal solution is either (i) all edges except the two largest ones or (ii) all edges, whichever gives smaller objective value. If the above condition does not hold, the optimal solution can be chosen to contain all negative edges.

We use a variation of the path optimization algorithm with an edge fixed to 0 or 1 for computing min-marginals.

Cutting plane. Since there are exponentially many path subproblems, we add during the optimization only those that improve the relaxation. Details are in Appendix 8.4.

4.4. Cut Subproblems

The purpose of a cut subproblem is to reflect that a lifted edge uv must be labelled 0 if there exists a cut of base edges that separate u and v (uv-cut) all labelled 0.

The feasible set. A cut subproblem consists of a lifted edge uv and a uv-cut $C = \{ij \in E | i \in A, j \in B\}$ where $A, B \subset V$ with $A \cap B = \emptyset$. The space of feasible solutions $\mathcal{X}^C$ is defined as

$$y'_{uv} \in \{0,1\}, y \in \{0,1\}^C : \quad y'_{uv} \leq \sum_{ij \in C} y_{ij} \,,$$

$$\forall i \in A : \sum_{ij \in C} y_{ij} \leq 1 \,, \quad \forall j \in B : \sum_{ij \in C} y_{ij} \leq 1 \,,$$

$$uv \in C \Rightarrow y'_{uv} \geq y_{uv} \,. \tag{10}$$

The constraints stipulate that (i) the lifted edge uv is 0 if all the edges in the cut are 0, (ii) there exists at most one active outgoing resp. incoming edge for every vertex in A resp. B and (iii) if there is also base edge $uv \in C$ then whenever it is active, the lifted edge uv must be active.

Algorithm 3 Cut-Subproblem-Optimization

Input Edge costs θ^C
Output optimal value opt of subproblem.

```
1:  Define $\psi \in \mathbb{R}^{A \times B}$:
2:  $\psi_{ij} = \begin{cases} \theta^C_{uv} + \theta'^C_{uv}, & \text{if } ij = uv \wedge uv \in C \wedge \theta'^C_{uv} > 0 \\ \infty, & \text{if } ij \notin C \\ \theta^C_{ij}, & \text{otherwise} \end{cases}$
3:  $z^* \in \operatorname{argmin}_{z \in \{0,1\}^{A \times B}} \sum_{i \in A} \sum_{j \in B} \psi_{ij} z_{ij}$, s.t. $z\mathbb{1} \leq \mathbb{1}, z^\top \mathbb{1} \leq \mathbb{1}$
4:  opt $= \sum_{ij \in C} \psi_{ij} z^*_{ij}$
5:  if $\theta'^C_{uv} \geq 0$ then return opt
6:  if $\exists kl \in C : z_{kl} = 1$ then
7:      return opt $+ \theta'^C_{uv}$
8:  else
9:      $\alpha = \min_{ij \in C} \theta^C_{ij}$
10:     if $|\theta'^C_{uv}| > \alpha$ then return $\theta'^C_{uv} + \alpha$
11:     else return opt
12: end if
```

Optimization of a cut subproblem with respect to feasible set $\mathcal{X}^C$ is given by Alg. 3. Its key is to solve a linear assignment problem (LAP) [1] between vertex sets A and B. The assignment cost ψ_{ij} for $(i,j) \in A \times B$ is the cut edge cost θ^C_{ij} if edge ij belongs to C and ∞ otherwise. In the special case of uv-cut C containing base edge uv and the lifted edge cost θ'^C_{uv} being positive, the assignment cost ψ_{uv} is increased by θ'^C_{uv}.

A candidate optimal labeling of cut edges is given by values of LAP variables z_{ij}. If $\theta'^C_{uv} \geq 0$, the optimal value found by the LAP is the optimal value of the cut subproblem. If it is negative, we distinguish two cases: (i) If a cut edge kl labeled by one exists, the lifted edge cost θ'^{C}_{uv} is added to the optimal value of LAP. (ii) Otherwise, we inspect whether it is better to activate the smallest-cost cut edge and the lifted edge uv or keep all edges inactive.

We use a variation of Alg. 3 with an edge variable restricted to be either 0 or 1 for computing min-marginals.

Cutting plane. There are exponentially many cut subproblems. Therefore, we add only those that improve the lower bound. See Appendix 8.5 for details.

4.5. Message Passing

The overall algorithm for optimizing the Lagrange decomposition is Alg. 19 in the Appendix. First, inflow and outflow subproblems are initialized for every node. Then, for a number of iterations or until convergence, costs for each subproblems are adjusted iteratively by computing min-marginals and adjusting the reparametrization proportionally to the min-marginal's value. Additionally, every k-th iteration additional path and cut subproblems are separated and added to the Lagrange decomposition.

Solver complexity. We need $\mathcal{O}(|E^{inp}|)$ space where E^{inp} are all edges before graph sparsification. The most time consuming is computing lifted edges min-marginals for each in/outflow subproblem. Alg. 6 computes them for one outflow subproblem and it is linear in the number of detections per frame. This significantly improves the complexity of to the optimal LDP solver LifT, making LDP applicable to large problem instances. See Appendix 8.14 for details.

4.6. Primal Rounding

For computing primal solutions we solve a minimum cost flow (MCF) problem on the base edges and improve this initial solution with a local search heuristic.

Without lifted edges, the disjoint paths problem is an instance of MCF, which can be efficiently optimized via combinatorial solvers like the successive shortest path solver that we employ [1]. We enforce node disjoint paths via splitting each node $u \in V$ into two nodes $u^{in}, u^{out} \in V^{mcf}$ in the MCF graph $G^{mcf} = (V^{mcf}, E^{mcf})$, adding an additional edge $u^{in}u^{out}$ to E^{mcf} and setting capacity $[0, 1]$ on all edges E^{mcf}. Each node except s and t has demand 0. Alg. 4 calculates MCF edge costs from in/outflow subproblems using Alg. 1. We obtain the cost of each flow edge $u^{out}v^{in}$ from the inflow subproblem of v and the outflow subproblem of u using their minima where edge uv is active. This combines well the cost from base and lifted edges.

We describe the local search heuristic for improving the MCF solution in Alg. 25 in the Appendix. It works with sets of disjoint paths. First, paths are split if this leads to a decrease in the objective. Second, merges are explored. If a merge of two paths is not possible, we iteratively check whether cutting off one node from the first path's end or the second paths's beginning makes the connection possible. If yes and the connection is decreasing the objective, the nodes are cut off and the paths are connected.

Algorithm 4 Init-MCF

```
1: $\forall u \in V \backslash \{s, t\}$:
   $(o, lc, \alpha^{in})$=Opt-In-Cost$(u, \theta^{in}_{u})$
   $(o, lc, \alpha^{out})$=Opt-Out-Cost$(u, \theta^{out}_{u})$
2: $\forall u \in V \backslash \{s, t\} : \theta^{mcf}_{su^{in}} = \alpha^{in}_{su}, \theta^{mcf}_{u^{out}t} = \alpha^{out}_{ut}$
3: $\forall u \in \{uv \in E | u \neq s, v \neq t\} : \theta^{mcf}_{u^{out}v^{in}} = \alpha^{out}_{uv} + \alpha^{in}_{uv}$
```

5. Experiments

We integrate our LDP solver into an MOT system (Appendix, Fig. 1) and show on challenging datasets that higher order MOT is scalable to big problem instances. In the next sections, we describe our experimental setup and present results. We clarify the edge cost calculation and construction of the base and the lifted graph and their sparsification.

5.1. Pairwise Costs

We use multi layer perceptrons (MLP) to predict the likelihood that two detections belong to the same trajectory. We divide the maximal frame distance into 20 intervals of equal length and train one separate MLP for each set of frame distances. We transform the MLP output to the cost of the edge between the detections and use it in our objective (3). Negative cost indicates that two detections belong to the same trajectory. Positive cost reflects the opposite.

MLP architecture. Each MLP consists of a fully connected layer with the same number of neurons as the input size, followed by a LeakyReLU activation [43] and a fully connected single neuron output layer. We add sigmoid activation in the training. We describe our spatial and visual features used as the input in the paragraphs below.

Spatial feature uses bounding box information of two detections v and w. We align the boxes such that their centers overlap. The similarity feature $\sigma_{vw,\text{Spa}} \in [0, 1]$ is the intersection-over-union between two aligned boxes.

Appearance feature. We create an appearance feature F_v for each detection v by training the method [67] on the training set of the respective benchmark and additional data from [66, 59, 50]. The similarity feature $\sigma_{vw,\text{App}}$ between detection v and w given by $\sigma_{vw,\text{App}} := \max\{0, \langle F_v, F_w \rangle\}$ is used. A higher value indicates a higher similarity.

Global context normalization. The two features $\sigma_{vw,\text{Spa}}$, $\sigma_{vw,\text{App}}$ depend entirely on the nodes v and w. To include global context, we append several normalized versions of the two features to the edge feature vector, inspired by [28]. Both features $\sigma_{ij,*}$ of edge ij undergo a five-way normalization. In each case, the maximum feature value from a relevant set of edges is selected as the normalization value. The normalization is done by dividing the two features $\sigma_{ij,*}$ by each of their five normalization values. This yields 10 values. Another set of 10 values for edge ij is obtained by dividing $\sigma^2_{ij,*}$ by each of the five normalization values. Together with the two unnormalized features $\sigma_{ij,*}$, edge feature vectors have length 22. See Appendix 8.9 for details.

Training. We iteratively train our MLP on batches B containing sampled edges. To compensate the imbalance between true positive and true negative edges, we use an α-balanced focal loss [40] with $\gamma = 1$. We define the α-weight $\alpha^{(g,\Delta f)}$ to weight the correct classification of edge vw with ground truth flow value $g_{vw} \in \{0,1\}$, time distance Δf between v in frame f_v and w in frame f_w, and value $g \in \{0,1\}$ via $\alpha^{(g,\Delta f)} := 1/|\{vw \in E : |f_v - f_w| = \Delta f, g_{vw} = g\}|$. We optimize the classifier using Adam with $l_r = 0.1$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. To reduce complexity while maintaining variety during training, we introduce an extended sampling. Given a frame f, we create batches $B(f)$ by sampling detections from a fixed sequence of frame shifts starting at frame f ensuring that all temporal distances Δf are present in $B(f)$ (details in Appendix 8.11). We then subsample the k-nearest detections to a random generated image position with $k = 160$, which sensitizes training to crowded scenes. We train the MLP for 3 epochs with batches $B(f)$ for all frames f of the dataset.

5.2. Graph Construction

We create the base and the lifted graph edges between detections with time distance up to 2 seconds. We also add an edge from source s, and to sink t to each detection. In order to reduce computational complexity, we apply sparsification on both base and lifted graph as described later.

Costs. We obtain base and lifted costs c and c' from the same MLP classifier (Sec. 5.1). Due to decreasing classification accuracy with increasing frame distance Δf, we multiply the costs by a decay weight $\omega_{\Delta f} := (10 \cdot \Delta f + 0.1)^{-1}$, so that edges representing long temporal distances have lower weight. Edges from s and to t have costs zero.

Finally, we use simple heuristics to find pairs that are obviously matching or non-matching. We set the corresponding costs to be high in absolute value, negative for matching and positive for non-matching, thereby inducing soft constrains. An obvious match is given by a nearly maximal feature similarity. Detection pairs are obviously non-matching, if the displacement between their bounding boxes is too high. See Appendix 8.12 for details.

Sparsification. The base edges are an intersection of two edge sets. The first set contains for every $v \in V'$ edges to its 3 nearest (lowest-cost) neighbors from every subsequent time frame. The second set selects for every vertex the best edges to its preceding frames analogically. Moreover, edges longer than 6 frames must have costs lower than 3.0. To avoid double counting of edge costs, we subsequently set costs of all base edges between non-consecutive frames to zero, so that only lifted edges maintain the costs. If a lifted edge has cost around zero, it is not discriminative and we remove it, unless it overlaps with a (zero-valued) base edge.

5.3. Inference

For fair comparison to state of the art, we filter and refine detections using tracktor [7] as in [28]. Different to [28], we apply tracktor to recover missing detections before running the solver.

While we solve MOT15/16/17 on global graphs, we solve MOT20 in time intervals in order to decrease memory consumption and runtime. First, we solve the problem on non-overlapping adjacent intervals and fix the trajectories in the interval centers. Second, we solve the problem on a new set of intervals where each of them covers unassigned detections in two initial neighboring intervals and enables connections to the fixed trajectory fragments. We use the maximal edge length of 50 frames in MOT20. Therefore, 150 is the minimal interval length such that all edges from a detection are used when assigning the detection to a trajectory. This way, the solver has sufficient context for making each decision. Intervals longer than 200 frames increase the complexity significantly for MOT20, therefore we use interval length 150 in our experiments.

Post-processing. We use simple heuristics to check if base edges over long time gaps correspond to plausible motions, and split trajectories if necessary. Finally, we use linear interpolation to recover missing detections within a trajectory. Appendix 8.13 contains further details on inference.

5.4. Tracking Evaluation

We evaluate our method on four standard MOT benchmarks. The MOT15/16/17 benchmarks [37, 45] contain semi-crowded videos sequences filmed from a static or a moving camera. MOT20 [16] comprises crowded scenes with considerably higher number of frames and detections per frame, see Tab. 1. The challenge does not come only with the data size. Detectors make more errors in crowded scenes due to frequent occlusions and appearance features are less discriminative as the distance of people to the camera is high. Using higher order information helps in this context. However, the number of edges in our graphs grows quadratically with the number of detections per frame. Therefore, it is crucial to make the tracker scalable to these massive data. We use the following ingredients to solve the problems: (i) fast but accurate method for obtaining edge costs, (ii) approximate LDP solver delivering high-quality results fast, (iii) preprocessing heuristics, (iv) interval solution keeping sufficient context for each decision.

We use training data of the corresponding dataset for training and the public detections for training and test.

Table 1. Comparison of ApLift with the best performing solvers w.r.t. MOTA metric on the MOT challenge. ↑ higher is better, ↓ lower is better. The two rightmost columns: average number of frames per sequence and the average number of detections per frame for dataset.

	Method	MOTA↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDS↓	Frag↓	Frames	Density
MOT20	ApLift (ours)	**58.9**	56.5	**513**	**264**	17739	**192736**	2241	2112	1119.8	170.9
	MPNTrack [10]	57.6	**59.1**	474	279	16953	201384	**1210**	**1420**		
	Tracktor++v2 [7]	52.6	52.7	365	331	**6930**	236680	1648	4374		
MOT17	CTTrackPub [68]	**61.5**	59.6	621	752	**14076**	200672	2583	4965	845.6	31.8
	ApLift (ours)	60.5	**65.6**	**798**	**728**	30609	**190670**	1709	2672		
	Lif_T [28]	60.5	**65.6**	637	791	14966	206619	1189	3476		
	MPNTrack [10]	58.8	61.7	679	788	17416	213594	**1185**	**2265**		
MOT16	ApLift (ours)	**61.7**	**66.1**	**260**	**237**	9168	**60180**	495	802	845.6	30.8
	Lif_T [28]	61.3	64.7	205	258	4844	65401	389	1034		
	MPNTrack [10]	58.6	61.7	207	258	4949	70252	**354**	**684**		
	GSM [41]	57.0	55.0	167	262	**4332**	73573	475	859		
MOT15	Lif_T [28]	**52.5**	**60.0**	244	186	6837	21610	730	1047	525.7	10.8
	MPNTrack [10]	51.5	58.6	225	187	7260	21780	**375**	**872**		
	ApLift (ours)	51.1	59.0	**284**	**163**	10070	**19288**	677	1022		
	Tracktor15 [7]	44.1	46.7	130	189	**6477**	26577	1318	1790		

Table 2. Influence of lifted graph sparsification, message passing and using zero base costs on MOT17 train without postprocessing.

E'	MP steps	Base cost	IDF1↑	MOTA↑	FP↓	FN↓	IDS↓
Dense	82	Zero	**71.0**	**66.3**	2826	**109263**	1369
Dense	0	Zero	70.3	**66.3**	2832	109265	1354
Dense	82	Orig.	69.8	**66.3**	**2824**	109266	1355
Sparse	82	Orig.	69.1	**66.3**	2825	**109263**	**1316**

Table 3. Runtime and IDF1 comparison of LDP solvers: ApLift (ours) with 6, 11, 31 and 51 iterations and LifT[28] (two step procedure) on first n frames of sequence *MOT20-01* from MOT20.

n	Measure	LifT	Our6	Our11	Our31	Our51
50	IDF1↑	80.6	**83.3**	**83.3**	81.5	81.5
	time [s]	272	2	4	16	35
100	IDF1↑	80.4	**82.5**	**82.5**	81.6	81.6
	time [s]	484	14	24	97	218
150	IDF1↑	78.1	**81.0**	**81.0**	79.8	79.8
	time [s]	1058	25	46	192	431
200	IDF1↑	73.2	**75.4**	**75.4**	74.6	74.6
	time [s]	2807	36	66	277	616

We compare our method using standard MOT metrics. MOTA [8] and IDF1 [49] are considered the most representative as they incorporate other metrics (in particular recall and precision). IDF1 is more penalized by inconsistent trajectories. We also report mostly tracked (MT) and mostly lost trajectories (ML), false negatives (FN) and false positives (FP), ID switches (IDS) and fragmentations (Frag) as provided by the evaluation protocols [8] of the benchmarks.

Tab. 1 shows the comparison to the best (w.r.t. MOTA) peer-reviewed methods on test sets. Our approximate solver achieves almost the same results on MOT15/16/17 as the optimal LDP solver [28], while using simpler features. Overall, our method performs on par with state of the art on all evaluated benchmarks, especially in MOTA and IDF1. Our complete results and videos are publicly available[3]. The proposed method achieves overall low FN values but slightly high FP values. FP/FN are mostly affected by preprocessing the input detections and interpolation in the post-processing. The impact of post-processing (trajectory splits and interpolations) on MOT20, which causes FP but reduces FN and IDS, is analyzed in the Appendix (Tab. 4).

Tab. 2 shows the influence of various settings on the performance of MOT17 train. While we usually set the base edge costs to zero (Sec. 5.2), we need to keep them when using the sparsified lifted graph. Both, message passing and dense lifted edges improve IDF1 and IDS. However, MOTA, FN and FP remain almost unchanged.

Finally, we compare the runtime of our solver against the two step version of LifT for a sample sequence in Tab. 3. With increasing problem complexity, our solver outperforms LifT w.r.t. runtime while achieving similar IDF1. Counter-intuitively, as we progress towards increasingly better optimization objective values, the tracking metrics can slightly decrease due to imperfect edge costs. We compare our solver against optimal (one step) LifT on MOT17 train in Appendix 8.14.

[3]https://motchallenge.net/method/MOT=4031&chl=13

6. Conclusion

We demonstrated that the NP-hard LDP model is applicable for processing massive sequences of MOT20. The combination of an approximate LDP solver, efficiently computable costs and subdivision of data keeping sufficient context for each decision make this possible.

7. Acknowledgements

This work was supported by the Federal Ministry of Education and Research (BMBF), Germany, under the project LeibnizKILabor (grant no. 01DD20003), the Center for Digital Innovations (ZDIN) and the Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy within the Cluster of Excellence PhoenixD (EXC 2122).

References

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network flows*. Cambridge, Mass.: Alfred P. Sloan School of Management, Massachusetts, 1988. 5, 6

[2] Alexandre Alahi, Yannick Boursier, Laurent Jacques, and Pierre Vandergheynst. Sport players detection and tracking with a mixed network of planar and omnidirectional cameras. In *2009 Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–8. IEEE, 2009. 1

[3] Alexandre Alahi, Judson Wilson, Li Fei-Fei, and Silvio Savarese. Unsupervised camera localization in crowded spaces. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2666–2673. IEEE, 2017. 1

[4] Chetan Arora and Amir Globerson. Higher order matching for consistent multiple target tracking. pages 177–184, 12 2013. 2

[5] Maryam Babaee, Ali Athar, and Gerhard Rigoll. Multiple people tracking using hierarchical deep tracklet re-identification. *arXiv preprint arXiv:1811.04091*, 2018. 2

[6] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1806–1819, 2011. 2

[7] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixé. Tracking without bells and whistles. In *IEEE International Conference on Computer Vision*, pages 941–951, 2019. 2, 7, 8

[8] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008, 01 2008. 8

[9] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-speed tracking-by-detection without using image information. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2017. 2

[10] Guillem Brasó and Laura Leal-Taixé. Learning a neural solver for multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6247–6257, 2020. 2, 8

[11] William Brendel, Mohamed Amer, and Sinisa Todorovic. Multiobject tracking as maximum weight independent set. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1273–1280. IEEE, 2011. 2

[12] Visesh Chari, Simon Lacoste-Julien, Ivan Laptev, and Josef Sivic. On pairwise costs for network flow multi-object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5537–5545, 2015. 2

[13] Chee-Yee Chong, Shozo Mori, and Donald B Reid. Forty years of multiple hypothesis tracking-a review of key developments. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 452–459. IEEE, 2018. 2

[14] Sunil Chopra and Mendu R Rao. The partition problem. *Mathematical Programming*, 59(1):87–115, 1993. 2, 5

[15] Afshin Dehghan, Shayan Modiri Assari, and Mubarak Shah. GMMCP tracker: Globally optimal generalized maximum multi clique problem for multiple object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4091–4099, 2015. 2

[16] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stephan Roth, Konrad Schindler, and Laura Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes. *arXiv:2003.09003[cs]*, Mar. 2020. arXiv: 2003.09003. 2, 7, 24, 28, 30

[17] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6569–6578, 2019. 1

[18] Michele Fenzi, Jörn Ostermann, Nico Mentzer, Guillermo Payá-Vayá, Holger Blume, Tu Ngoc Nguyen, and Thomas Risse. Asev—automatic situation assessment for event-driven video analysis. In *2014 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 37–43. IEEE, 2014. 1

[19] Davi Frossard and Raquel Urtasun. End-to-end learning of multi-sensor 3d tracking by detection. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 635–642. IEEE, 2018. 1

[20] Amir Globerson and Tommi Jaakkola. Fixing max-product: Convergent message passing algorithms for map lp-relaxations. *Advances in neural information processing systems*, 20:553–560, 2007. 2

[21] Monique Guignard and Siwhan Kim. Lagrangean decomposition for integer programming: theory and applications. *RAIRO-Operations Research-Recherche Opérationnelle*, 21(4):307–323, 1987. 3

[22] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2019. 2

[23] Roberto Henschel, Laura Leal-Taixé, Daniel Cremers, and Bodo Rosenhahn. Fusion of head and full-body detectors for multi-object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, June 2018. 2

[24] Roberto Henschel, Timo von Marcard, and Bodo Rosenhahn. Simultaneous identification and tracking of multiple people using video and imus. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 2

[25] Roberto Henschel, Yunzhe Zou, and Bodo Rosenhahn. Multiple people tracking using body and joint detections. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 2

[26] Kalun Ho, Amirhossein Kardoost, Franz-Josef Pfreundt, Janis Keuper, and Margret Keuper. A two-stage minimum cost multicut approach to self-supervised multiple person tracking. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, November 2020. 2

[27] Martin Hofmann, Daniel Wolf, and Gerhard Rigoll. Hypergraphs for joint multi-view reconstruction and multi-object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3650–3657, 2013. 2

[28] Andrea Hornakova, Roberto Henschel, Bodo Rosenhahn, and Paul Swoboda. Lifted disjoint paths with application

in multiple object tracking. In *The 37th International Conference on Machine Learning (ICML)*, July 2020. 1, 2, 3, 6, 7, 8, 27, 28, 29

[29] Andrea Horňáková, Jan-Hendrik Lange, and Bjoern Andres. Analysis and optimization of graph decompositions by lifted multicuts. In *International Conference on Machine Learning*, 2017. 2

[30] Weiming Hu, Xinchu Shi, Zongwei Zhou, Junliang Xing, Haibin Ling, and Stephen Maybank. Dual L1-normalized context aware tensor power iteration and its applications to multi-object tracking and multi-graph matching. *International Journal of Computer Vision*, Oct 2019. 2

[31] Chang Huang, Bo Wu, and Ramakant Nevatia. Robust object tracking by hierarchical association of detection responses. In *European Conference on Computer Vision*, pages 788–801. Springer, 2008. 2

[32] Hao Jiang, Sidney Fels, and James J Little. A linear programming approach for multiple object tracking. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 2

[33] Margret Keuper, Siyu Tang, Bjoern Andres, Thomas Brox, and Bernt Schiele. Motion segmentation & multiple object tracking by correlation co-clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(1):140–153, 2018. 2

[34] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple hypothesis tracking revisited. In *Proceedings of the IEEE international conference on computer vision*, pages 4696–4704, 2015. 2

[35] Péter Kovács. Minimum-cost flow algorithms: an experimental evaluation. *Optimization Methods and Software*, 30(1):94–127, 2015. 1, 2

[36] Ratnesh Kumar, Guillaume Charpiat, and Monique Thonnat. Multiple object tracking by efficient graph partitioning. In *Asian Conference on Computer Vision*, pages 445–460. Springer, 2014. 2

[37] Laura Leal-Taixé, Anton Milan, Ian Reid, Stephan Roth, and Konrad Schindler. MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv:1504.01942 [cs]*, Apr. 2015. arXiv: 1504.01942. 2, 7

[38] Laura Leal-Taixé, Gerard Pons-Moll, and Bodo Rosenhahn. Branch-and-price global optimization for multi-view multi-target tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1987–1994. IEEE, 2012. 2

[39] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. Pnpnet: End-to-end perception and prediction with tracking in the loop. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11553–11562, 2020. 1

[40] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 07 2018. 7

[41] Qiankun Liu, Qi Chu, Bin Liu, and Nenghai Yu. Gsm: Graph similarity model for multi-object tracking. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 530–536. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Main track. 8

[42] Wei-Lwun Lu, Jo-Anne Ting, James J Little, and Kevin P Murphy. Learning to track and identify players from broadcast sports videos. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1704–1716, 2013. 1

[43] Andrew L. Maas, Awny Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the International Conference on Machine Learning*, Atlanta, Georgia, 2013. 6, 23, 24

[44] Florian Meyer, Thomas Kropfreiter, Jason L Williams, Roslyn Lau, Franz Hlawatsch, Paolo Braca, and Moe Z Win. Message passing algorithms for scalable multitarget tracking. *Proceedings of the IEEE*, 106(2):221–259, 2018. 2

[45] Anton Milan, Laura Leal-Taixé, Ian Reid, Stephan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*, Mar. 2016. arXiv: 1603.00831. 2, 7, 28, 30

[46] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1

[47] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 1

[48] Seyed Hamid Rezatofighi, Anton Milan, Zhen Zhang, Qinfeng Shi, Anthony Dick, and Ian Reid. Joint probabilistic data association revisited. In *Proceedings of the IEEE international conference on computer vision*, pages 3047–3055, 2015. 2

[49] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In Gang Hua and Hervé Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, pages 17–35, Cham, 2016. Springer International Publishing. 8

[50] Ergys Ristani, Francesco Solera, Roger S. Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision Workshop on Benchmarking Multi-Target Tracking*, 2016. 6

[51] Ergys Ristani and Carlo Tomasi. Tracking multiple people online and in real time. In *Asian Conference on Computer Vision*, pages 444–459. Springer, 2014. 2

[52] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *IEEE International Conference on Computer Vision*, pages 300–311, 2017. 2

[53] Julian Smith, Florian Particke, Markus Hiller, and Jörn Thielecke. Systematic analysis of the pmbm, phd, jpda and gnn multi-target tracking filters. In *2019 22th International Conference on Information Fusion (FUSION)*, pages 1–8, 2019. 2

[54] Paul Swoboda, Jan Kuske, and Bogdan Savchynskyy. A dual ascent framework for lagrangean decomposition of combinatorial problems. In *Proceedings of the IEEE Conference*

on Computer Vision and Pattern Recognition (CVPR), July 2017. 2, 3, 4

[55] Siyu Tang, Bjoern Andres, Miykhaylo Andriluka, and Bernt Schiele. Subgraph decomposition for multi-target tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5033–5041, 2015. 2

[56] Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Multi-person tracking by multicut and deep matching. In *European Conference on Computer Vision*, pages 100–111. Springer, 2016. 1, 2

[57] Siyu Tang, Mykhaylo Andriluka, Bjoern Andres, and Bernt Schiele. Multiple people tracking by lifted multicut and person re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2

[58] Timo von Marcard, Roberto Henschel, Michael J Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 601–617, 2018. 2

[59] Longhui Wei, Shiliang Zhang, Wen Gao, and Qi Tian. Person transfer gan to bridge domain gap for person re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 79–88, 2018. 6

[60] Nicolai Wojke and Alex Bewley. Deep cosine metric learning for person re-identification. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 748–756. IEEE, 2018. 2

[61] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE, 2017. 2

[62] Jiarui Xu, Yue Cao, Zheng Zhang, and Han Hu. Spatial-temporal relation networks for multi-object tracking. In *IEEE International Conference on Computer Vision*, pages 3988–3998, 2019. 2

[63] Fan Yang, Wongun Choi, and Yuanqing Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2129–2137, 2016. 1

[64] Amir Roshan Zamir, Afshin Dehghan, and Mubarak Shah. GMCP-tracker: Global multi-object tracking using generalized minimum clique graphs. In *European Conference on Computer Vision*, pages 343–356. Springer, 2012. 2

[65] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008. 1, 2

[66] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *IEEE International Conference on Computer Vision*, pages 1116–1124, 2015. 6

[67] Zhedong Zheng, Xiaodong Yang, Zhiding Yu, Liang Zheng, Yi Yang, and Jan Kautz. Joint discriminative and generative learning for person re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 6

[68] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer, 2020. 2, 8

[69] Ji Zhu, Hua Yang, Nian Liu, Minyoung Kim, Wenjun Zhang, and Ming-Hsuan Yang. Online multi-object tracking with dual matching attention networks. In *European Conference on Computer Vision*, pages 366–382, 2018. 2