Higher Order Multiple Object Tracking for Crowded Scenes

Andrea Hornakova [*] [†] Timo Kaiser [*] [‡] Bodo Rosenhahn[‡] Paul Swoboda[†]
Roberto Henschel[‡]

Abstract

The lifted disjoint paths formulation is a natural model for multiple object tracking. This model is able to obtain state-of-the-art results but is NP-hard. We present an efficient approximate message passing solver for LDP and integrate it into a multiple object tracker, which scales to very large instances that come from long and crowded scenes. We achieve comparable or better performance than state-of-the-art methods on MOT15/16/17 benchmarks and comparable results on the MOT20 benchmark. This has been out of reach up to now for known LDP-solvers due to the problem size and complexity of MOT20.

1. Introduction

In this work, we follow the tracking-by-detection paradigm. We solve the data association part using recently published Lifted Disjoint Paths (LDP) model [4], which achieved significant improvement over state-of-the-art trackers on MOT benchmarks MOT15/16/17.

LDP extends the standard disjoint paths (DP) problem, a special case of the network flow problem, which is a natural model for multiple object tracking. DP can be solved in polynomial time by minimum cost flow solvers. Unfortunately, the integration of long range temporal interactions is limited, since consistency is ensured only between directly linked detections. Consequently, this model ignores higher order consistencies among multiple linked detections.

To fix this, LDP generalizes DP by adding connectivity priors in terms of *lifted* edges. Lifted edges enable arbitrary pairs of detections to contribute to the objective value whenever they are connected via a trajectory. This model leads to more consistent trajectories than DP. However, the better expressiveness of LDP comes with the cost of making the problem NP-hard.

Consequently, the tracker based on an optimal LDP solver presented in [4] is not scalable to the new challenging dataset MOT20, which contains much more crowded and longer sequences. We present an approximate LDP solver and lightweight tracking framework, which achieves on par performance with the tracker from [4] on MOT15/16/17. Most importantly, our method extends the applicability of LDP, solving MOT20 [3] with comparable results to state-of-the-art.

[*]Equal contribution

[†]Computer Vision and Machine Learning, Max Planck Institute for Informatics, Saarbrücken, Saarland, Germany.

[‡]Institut for Image Processing, Leibniz University Hanover, Hannover, Niedersachsen, Germany.

2. Method

Following the LDP [4] formulation, the MOT problem is modelled by two directed acyclic graphs, a flow network (base graph) $G = (V, E)$ with start and terminal nodes $s, t \in V$ and lifted graph $G' = (V', E')$ where $V' = V \setminus \{s, t\}$. Trajectories are modelled by vertex-disjoint paths in graph G.

Each edge is assigned a cost reflecting the probability that its endpoints belong to the same trajectory. We denote the corresponding cost functions by $c \in \mathbb{R}^E$ for the base graph and $c' \in \mathbb{R}^{E'}$ for the lifted graph.

The goal is to find a $0/1$-labeling of all edges that minimizes the objective function in (2). Variables $y \in \{0,1\}^E$ have value 1 if flow passes through the respective edges. Feasible y must define node-disjoint flow in graph G. Variables on 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,

$$y'_{vw} = 1 \Leftrightarrow \exists P \in vw\text{-paths}(G) \text{ s.t. } \forall ij \in P_E : y_{ij} = 1\,. \tag{1}$$

Here, P_E denotes the edges of path P. Given all the above ingredients, the lifted disjoint paths problem is

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

Using the LDP formulation, our proposed method consists of (i) defining lightweight pairwise-costs, (ii) pruning the graphs without losing important information and (iii) solving the LDP problem with the approximate LDP solver proposed in this work. The overall framework is shown in Figure 1.

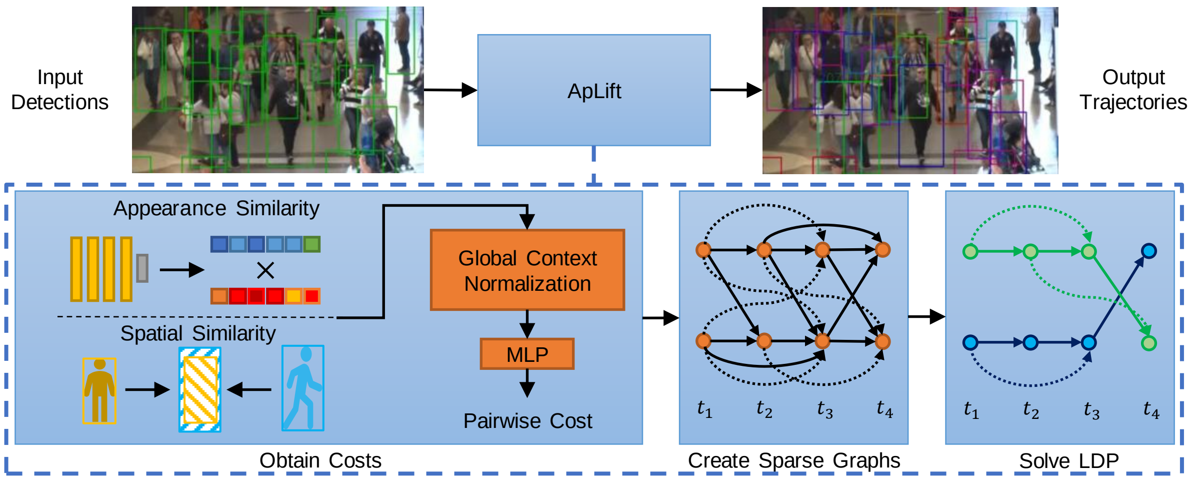

Figure 1. Overview of the ApLift framework. Input detections are used to obtain pairwise costs by an MLP with spatial and appearance features. Based on the costs, two sparse graphs are constructed and passed to our proposed approximate LDP solver. Dashed arrows represent lifted edges and solid arrows base edges. In figure *Solve LDP* equally colored nodes and edges belong to the same trajectory.

2.1. Pairwise Costs

Pairwise costs are obtained by an efficient multi layer perceptron (MLP). We decompose the temporal distances up to 2 seconds into 20 intervals and train an MLP for each, to incorporate temporal dependencies.

The MLP input is composed of a spatial and an appearance similarity feature and multiple normalization features. The appearance feature $s_{ij,\text{App}}$ is derived from a re-identification network [8] by measuring similarity between detections in the embedding space, and then projecting it to a non-negative value so that the subsequent normalization is applicable. The spatial feature $s_{ij,\text{Spa}}$ reflects height and width similarity of two detections b_i and b_j. A translation is performed such that b_i and b_j have the same box center position. Then, the intersection over union between b_i and b_j is computed. To incorporate global context, we append several normalized versions of the two features (w.r.t. to all other edges) to the edge feature vector, similarly as in [4]. Both features $s_{ij,*}$ of edge ij undergo a five-way normalization. In each case, the maximum feature value from a set of edges is selected as the normalization value. The first set contains all edges going from node i to the frame of node j. The second set contains all edges starting in i. The third set contains edges between all detections within a certain interval of video frames that comprise i and j. The fourth set contains all edges ending in j from the time frame of i. The fifth set contains all edges going to j. The normalization is done by dividing the two features $s_{ij,*}$ by each of their five normalization values. This yields 10 feature values. Another set of 10 feature values for edge ij is obtained by dividing $s^2_{ij,*}$ by each of the five normalization values. This yields another 10 feature values. Together with the two unnormalized features $s_{ij,*}$, we obtain feature vectors of length 22 for each edge. These 22 features are passed as input to the MLP, which consists of a fully connected layer containing 22 neurons, followed by a rectified linear activation function and a final fully connected layer with a single neuron. The output is the cost value.

The network is iteratively trained with sampled edges using a weighted focal loss and the Adam optimizer. The lightweight topology of the classifier needs only short training time and enables the training directly on a CPU.

2.2. Graph Definition

All nodes of the base graph $G = (V, E)$ are connected to start and terminal s and t with edge costs set to zero. We create base and lifted edges between nodes up to temporal distance 2 seconds. We obtain costs c and c' by the classifier described in Section 2.1. Edges with negative costs connect detections that probably belong to the same trajectory, while edges with positive costs connect those that do not.

To tackle decreasing prediction accuracy for edges with longer temporal distances, we add a weight decay on the cost values. Additionally, we use simple heuristics based on optical flow and spatial distance to determine obviously matching or not matching detection pairs. We set them to high positive resp. negative values to induce soft-constraints.

The initial costs for lifted and base edges are obtained from the same method. We further decide which detections should be connected by a base and/or a lifted edge and change some cost values as described below. For the base graph, we just keep the 3 nearest neighbours of each vertex in terms of lowest costs from every subsequent time frame. For time distances greater than 6 frames we keep only edges with cost lower than 3. To avoid double counting of costs, all base edge costs are finally set to zero if the detections are more than one time frame apart. Costs of these connections are assigned to overlapping lifted edges.

For the lifted graph, we remove all edges with costs close to zero unless they overlap with some base edge. The reason is that these lifted edges are not discriminative. This sparsification reduces the complexity dramatically without losing expressiveness.

2.3. Lagrange Decomposition

We solve Problem (2) in a Lagrange decomposition framework. That is, we propose a decomposition of the overall LDP problem into smaller but tractable subproblems. We use a simplified version of the framework developed in [7]. This decomposition is a dual task to an LP-relaxation of the LDP problem (2). So, it provides a lower bound that is iteratively increased by message passing (cost reparametrization). The dual costs from the subproblems are also used in our heuristic for obtaining primal solutions.

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* $\mathcal{X}^{\mathsf{s}}\subseteq\{0,1\}^{d_{\mathsf{s}}}$. Each coordinate i of $\mathcal{X}^{\mathsf{s}}$ corresponds to one coordinate of $\mathcal{X}$ via an injection $\pi_{\mathsf{s}}:[d_{\mathsf{s}}]\to[n]$ 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 $\mathsf{s},\mathsf{s}'\in\mathsf{S}$ that contain a pair of coordinates i,j such that $\pi_{\mathsf{s}}(i)=\pi_{\mathsf{s}'}(j)$, we have *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 in the subproblems, i.e. $\forall x\in\mathcal{X},\forall\mathsf{s}\in\mathcal{S}:A^{\mathsf{s}}x\in\mathcal{X}^{s}$.

For every subproblem an *objective* is given by $\theta^{\mathsf{s}}\in\mathbb{R}^{d_{\mathsf{s}}}$. We require that the objectives of subproblems are equivalent to the original objective, i.e. $\langle c,x\rangle=\sum_{s\in\mathcal{S}}\langle\theta^{s},A^{\mathsf{s}}x\rangle\ \forall x\in\mathcal{X}$. The *lower bound* of the Lagrange decomposition given the costs θ^{s} for each $\mathsf{s}\in S$ is

$$\sum_{\mathsf{s}\in\mathcal{S}}\min_{x^{\mathsf{s}}\in\mathcal{X}^{\mathsf{s}}}\langle\theta^{\mathsf{s}},x^{\mathsf{s}}\rangle\,. \tag{3}$$

A sequence of operations of the form $\theta_i^{\mathsf{s}}\mathrel{+}=\gamma$, $\theta_j^{\mathsf{s}'}\mathrel{-}=\gamma$ for coupling constraint $x_i^{\mathsf{s}}=x_j^{\mathsf{s}'}$ and $\gamma\in\mathbb{R}$ is called a *reparametrization*. Instead of optimizing the primal problem (2), we will optimize its dual problem, i.e. the lower bound (3) w.r.t. a decomposition described below.

Optimization. Any feasible primal solution is invariant under reparametrizations but the dual lower bound (3) is not. We apply a sequence of reparametrization updates that are monotonically non-decreasing in the lower bound. For that purpose, updates are realised via min marginals. That is, $\theta_i^{\mathsf{s}}\mathrel{+}=\omega m_i^{\mathsf{s}}$, $\theta_j^{\mathsf{s}'}\mathrel{-}=\omega m_i^{\mathsf{s}}$, where $\omega\in[0,1]$ is a damping factor and

$$m_i^{\mathsf{s}}=\min_{x^{\mathsf{s}}\in\mathcal{X}^{\mathsf{s}}:x_i^{\mathsf{s}}=1}\langle\theta^{\mathsf{s}},x^{\mathsf{s}}\rangle-\min_{x^{\mathsf{s}}\in\mathcal{X}^{\mathsf{s}}:x_i^{\mathsf{s}}=0}\langle\theta^{\mathsf{s}},x^{\mathsf{s}}\rangle\,. \tag{4}$$

So, the key components of the method are efficient computations of (i) optima of each subproblem for obtaining (3) and (ii) constrained optima for obtaining (4).

Inflow and outflow subproblems contain all incoming resp. all outgoing edges of a node v. We use them for each node $v\in V$. A feasible solution of an outflow subproblem of v either assignes value zero to all variables or it can be represented by a path $P=(v_0=v,\dots,v_n=t)$ in G such that (i) $y_{vw}=1\Leftrightarrow w=v_1$ and (ii) $y'_{vw'}=1\Leftrightarrow w'\in P$. Analogically for the inflow.

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 the vw is active. The reason is that the inactive lifted edge divides P into two segments that must be disconnected. This is contradicting to activating lifted edge vw because it indicates connection of v and w.

Cut subproblems reflect that a lifted edge uv must be labelled 0 if there exists a cut in the base graph that separate u and v where all its edges all labelled 0. 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 $A\cap B=\emptyset$.

Primal solutions are obtained by solving a minimum cost flow problem. Here, cost of each base edge $vw\in E$ is obtained from the inflow subproblem of w and the outflow subproblem of v using their minima where edge vw is active

$$c_{vw}=\min_{x\in\mathcal{X}^{i(w)}:x_{vw}=1}\langle\theta^{i(w)},x\rangle+\min_{x\in\mathcal{X}^{o(v)}:x_{vw}=1}\langle\theta^{o(v)},x\rangle\,.$$

The obtained solutions are further improved by a local search heuristic.

3. Experiments

We evaluate our method on four standard MOT benchmarks. The MOT15/16/17 benchmarks [5, 6] are composed of semi-crowded videos sequences filmed from a static or moving camera. MOT20 [3] comprises crowded scenes with considerably higher number of frames and detections per frame, see Table 1.

Consequently, MOT20 is more challenging, as crowded scenes frequently lead to full or partial occlusions which disturb detectors and diminish the significance of appearance features for re-identification. Incorporating higher order information into the graph model is therefore crucial to tackle this problem. The number of edges grows quadratically with the number of detections per frame.

	Method	MOTA↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDS↓	Frag↓	Frames	Density
MOT20	ApLift (ours)	**58.9**	56.5	**41.3**	**21.3**	17739	**192736**	2241	2112	1119.8	170.9
	MPNTrack [2]	57.6	**59.1**	38.2	22.5	16953	201384	**1210**	**1420**		
	Tracktor++v2 [1]	52.6	52.7	29.4	26.7	**6930**	236680	1648	4374		
MOT17	CTTrackPub [9]	**61.5**	59.6	26.4	31.9	**14076**	200672	2583	4965	845.6	31.8
	ApLift (ours)	60.5	**65.6**	**33.9**	**30.9**	30609	**190670**	1709	**2672**		
	Lif_T [4]	60.5	**65.6**	27.0	33.6	14966	206619	**1189**	3476		
MOT16	ApLift (ours)	**61.7**	**66.1**	**34.3**	**31.2**	9168	**60180**	495	802	845.6	30.8
	Lif_T [4]	61.3	64.7	27.0	34.0	**4844**	65401	389	1034		
	MPNTrack [2]	58.6	61.7	27.3	34.0	4949	70252	**354**	**684**		
MOT15	Lif_T [4]	**52.5**	**60.0**	33.8	25.8	**6837**	21610	730	1047	525.7	10.8
	MPNTrack [2]	51.5	58.6	31.2	25.9	7260	21780	**375**	**872**		
	ApLift (ours)	51.1	59.0	**39.4**	**22.6**	10070	**19288**	677	1022		

Table 1. Comparison of ApLift with the best performing solvers w.r.t. MOT metrics on the MOT challenge. ↑ higher is better, ↓ lower is better. Rightmost columns: average number of frames per sequence and the average number of bounding boxes per frame for given dataset.

n	Measure	Lift [4]	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

Table 2. Runtime and IDF1 comparison between our solver with 6, 11, 31 and 51 iterations against globally optimal Lif_T[4] solver on first n frames of sequence *MOT20-01* from MOT20.

We train our tracker with the training data provided by the respective benchmarks. During inference, the provided public detections are used. For fair comparison to state-of-the-art, we filter and refine the detections as in [4] before applying our method. To reduce memory consumption and runtime, we divide the sequences of MOT20 into intervals. First, we solve the problem on non-overlapping adjacent intervals and fix the trajectories in 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. This way, the solver has sufficient context for making each decision.

Finally, we use simple heuristics to split trajectories that represent implausible motions and we recover missing detections within a trajectory using linear interpolation.

Table 1 compares our tracker with the top peer-reviewed methods w.r.t. MOTA, including a method using an optimal LDP solver [4]. We achieve similar results on MOT15/16/17 as [4] and comparable results on MOT20 to current state-of-the-art [2].

Further experiments comparing runtime and IDF1 of the optimal LDP-based solver [4] are given in Table 2. We see that our approximative solver outperforms [4] w.r.t. tracking metrics after few iterations. Counter-intuitively, as we progress towards increasingly better optimization objective values, the tracking metrics can slightly decrease due to imperfect edge costs.

4. Conclusion

This work extends the applicability of the LDP model to massive sequences as MOT20 using efficiently computable edge costs, an approximate solver and a subdivision of the data providing sufficient context for each decision, resulting in state-of-the-art performance.

References

[1] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixé. Tracking without bells and whistles. In *CVPR*, 2019. 4

[2] Guillem Brasó and Laura Leal-Taixé. Learning a neural solver for multiple object tracking. In *CVPR*, 2020. 4

[3] 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. 1, 3

[4] Andrea Hornakova, Roberto Henschel, Bodo Rosenhahn, and Paul Swoboda. Lifted disjoint paths with application in multiple object tracking. In *ICML*, July 2020. 1, 2, 4

[5] 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. 3

[6] 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. 3

[7] Paul Swoboda, Jan Kuske, and Bogdan Savchynskyy. A dual ascent framework for lagrangean decomposition of combinatorial problems. In *CVPR*, July 2017. 3

[8] Zhedong Zheng, Xiaodong Yang, Zhiding Yu, Liang Zheng, Yi Yang, and Jan Kautz. Joint discriminative and generative learning for person re-identification. In *CVPR*, 2019. 2

[9] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *ECCV*. Springer, 2020. 4