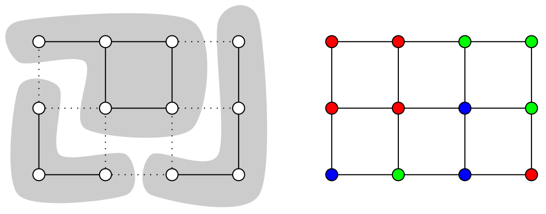
Joint Graph Decomposition & Node Labeling: Problem, Algorithms, Applications

Evgeny Levinkov[1], Jonas Uhrig[2,3], Siyu Tang[1], Eldar Insafutdinov[1], Mohamed Omran[1], Alexander Kirillov[4], Carsten Rother[4], Thomas Brox[2], Bernt Schiele[1] and Bjoern Andres[1]

[1]Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
[2]Computer Vision Lab, University of Freiburg, Germany [3]Daimler AG R&D, Sindelfingen, Germany
[4]Computer Vision Lab, Technische Universität Dresden, Germany

Abstract

We state a combinatorial optimization problem whose feasible solutions define both a decomposition and a node labeling of a given graph. This problem offers a common mathematical abstraction of seemingly unrelated computer vision tasks, including instance-separating semantic segmentation, articulated human body pose estimation and multiple object tracking. Conceptually, the problem we state generalizes the unconstrained integer quadratic program and the minimum cost lifted multicut problem, both of which are NP*-hard. In order to find feasible solutions efficiently, we define two local search algorithms that converge monotonously to a local optimum, offering a feasible solution at any time. To demonstrate their effectiveness in tackling computer vision tasks, we apply these algorithms to instances of the problem that we construct from published data, using published algorithms. We report state-of-the-art application-specific accuracy for the three above-mentioned applications.*



(a) Decomposition (b) Node Labeling

Figure 1: This article studies an optimization problem whose feasible solutions define both a decomposition (a) and a node labeling (b) of a given graph $G = (V, E)$. A decomposition of G is a partition Π of the node set V such that, for every $V' \in \Pi$, the subgraph of G induced by V' is connected. A node labeling of G is a map $f : V \to L$ from its node set V to a finite, non-empty set L of labels.

1. Introduction and Related Work

Graphs are a ubiquitous structure in the field of computer vision. In this article, we state an optimization problem whose feasible solutions define both a decomposition and a node labeling of a given graph (Fig. 1). We define and implement two local search algorithms for this problem that converge monotonously to a local optimum. The problem that we state is abstract enough to specialize to seemingly unrelated computer vision tasks. This abstraction allows us to apply the algorithms we define, without changes, to three distinct computer vision tasks: multiple object tracking, instance-separating semantic segmentation and articulated human body pose estimation. We report state-of-the-art application-specific accuracy for these three applications.

Multiple object tracking [3, 4, 8, 11, 19, 26, 28, 40, 41] can be seen as a task requiring two classes of decisions: For every point in an image, we need to decide whether this point depicts an object or background. For every pair of points that depict objects, we need to decide if the object is the same. Tang et al. [35, 36] abstract this task as a graph decomposition and node labeling problem w.r.t. a finite graph whose nodes are bounding boxes, and w.r.t. 01-labels indicating that a bounding box depicts an object. We generalize their problem to more labels and more complex objective functions. By applying this generalization to the data of Tang et al. [36], we obtain more accurate tracks for the multiple object tracking benchmark [25] than any published work.

Instance-separating semantic segmentation [9, 10, 22, 31, 32, 33, 42, 43] can be seen as a task requiring two classes of decisions: To every point in an image, we need to assign a label that identifies a class of objects (e.g., human, car, bicycle). For every pair of points of the same class, we need to decide if the object is the same. Kroeger et al. [21] state this problem as a multi-terminal cut problem w.r.t. a (super)pixel adjacency graph of the image. We generalize their problem to larger feasible sets. While Kroeger et al. [21] show qualitative results, we apply our algorithms to instances of the problem from the KITTI [13] and Cityscapes [9] benchmarks, obtaining more accurate results for Cityscapes than any published work.

Articulated human body pose estimation can be seen as a task requiring two classes of decisions: For every point in an image, we need to decide whether it depicts a part of the human body. For every pair of points that depict body parts, we need to decide if they belong to the same body. Pishchulin et al. [29] and Insafutdinov et al. [15] abstract this problem as a graph decomposition and node labeling problem w.r.t. a finite graph whose nodes are putative detections of body parts and w.r.t. labels that idenfity body part classes (head, wrist, etc.) and background. We generalize their problem to more complex objective functions. By reducing the running time for this task compared to their branch-and-cut algorithm (that computes also lower bounds), we can tackle instances of the problem with more nodes. This allows us to obtain more accurate pose estimates for the MPII Human Pose Dataset [2] than any published work.

Formally, the problem we propose and refer to as the minimum cost node labeling lifted multicut problem, NL-LMP, generalizes the NP-hard unconstrained integer quadratic program, UIQP, that has been studied intensively in the context of graphical models [16], and also generalizes the NP-hard minimum cost lifted multicut problem, LMP [18]. Unlike in pure node labeling problems such as the UIQP, neighboring nodes with the same label can be assigned to distinct components, and neighboring nodes with distinct labels can be assigned to the same component. Unlike in pure decomposition problems such as the LMP, the cost of assigning nodes to the same component or distinct components can depend on node labels. Also unlike in the LMP, constraining nodes with the same label to the same component constrains the feasible decompositions to be k-colorable, with $k \in \mathbb{N}$ the number of labels. For $k = 2$ in particular, this constrained NL-LMP specializes to the well-known MAX-CUT problem.

In order to find feasible solutions of the NL-LMP efficiently, we define and implement two local search algorithms that converge monotonously to a local optimum, offering a feasible solution at any time. These algorithms do not compute lower bounds. They output feasible solutions without approximation certificates. Hence, they belong to the class of primal feasible heuristics for the NL-LMP. The first algorithm we define and refer to as alternating Kernighan-Lin search with joins and node relabeling, KLj/r, is a generalization of the algorithm KLj of Keuper et al. [18] and of Iterated Conditional Modes (ICM). The second algorithm we define and refer to as joint Kernighan-Lin search with joins and node relabeling, KLj$*$r, is a generalization of KLj that transforms a decomposition and a node labeling jointly, in a novel manner. Both algorithms build on seminal work of Kernighan and Lin [17].

2. Problem

In this section, we define the minimum cost node labeling lifted multicut problem, NL-LMP. Sections 2.1–2.3 offer an intuition for its parameters, feasible solutions and cost function. Section 2.4 offers a concise and rigorous definition. Section 2.5 discusses special cases.

2.1. Parameters

Any instance of the NL-LMP is defined with respect to the following parameters:

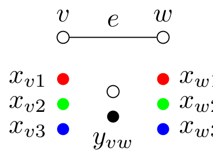
- A connected graph $G = (V, E)$ whose decompositions we care about, e.g., the pixel grid graph of an image.
- A graph $G' = (V, E')$ with $E \subseteq E'$. This graph can contain as edges pairs of nodes that are not neighbors in G. It defines the structure of the cost function.
- A digraph $H = (V, A)$ that fixes an arbitrary orientation of the edges E'. That is, for every edge $\{v, w\}$ of G', the graph H contains either the edge (v, w) or the edge (w, v). Formally, H is such that for all $v, w \in V$:

$$\{v, w\} \in E' \Leftrightarrow (v, w) \in A \vee (w, v) \in A \quad (1)$$

$$(v, w) \notin A \vee (w, v) \notin A \quad (2)$$

- A finite, non-empty set L called the set of *(node) labels*
- The following functions whose values are called *costs*:
  - $c : V \times L \to \mathbb{R}$. For any node $v \in V$ and any label $l \in L$, the cost c_{vl} is payed iff v is labeled l.
  - $c^{\sim} : A \times L^2 \to \mathbb{R}$. For any edge $vw \in A$ and any labels $ll' \in L^2$, the cost $c^{\sim}_{vw,ll'}$ is payed iff v is labeled l and w is labeled l' and v and w are in the same component.
  - $c^{\not\sim} : A \times L^2 \to \mathbb{R}$. For any edge $vw \in A$ and any labels $ll' \in L^2$, the cost $c^{\not\sim}_{vw,ll'}$ is payed iff v is labeled l and w is labeled l' and v and w are in distinct components.

2.2. Feasible Set



Every feasible solution of the NL-LMP is a pair (x, y) of 01-vectors $x \in \{0,1\}^{V \times L}$ and $y \in \{0,1\}^{E'}$. More specifically, x is constrained such that, for every node $v \in V$, there is precisely one label $l \in L$ with $x_{vl} = 1$. y is constrained so as to well-define a decomposition of G by the set $\{e \in E \,|\, y_e = 1\}$ of those edges that straddle distinct components. Formally, $(x, y) \in X_{VL} \times Y_{GG'}$ with X_{VL} and $Y_{GG'}$ defined below.

- $X_{VL} \subseteq \{0,1\}^{V \times L}$, the set of all characteristic functions of maps from V to L, i.e., the set of all $x \in \{0,1\}^{V \times L}$ such that

$$\forall v \in V : \quad \sum_{l \in L} x_{vl} = 1 \ . \quad (3)$$

For any $x \in X$, any $v \in V$ and any $l \in L$ with $x_{vl} = 1$, we say that node v is *labeled* l by x.

- $Y_{GG'} \subseteq \{0,1\}^{E'}$, the set of all characteristic functions of multicuts of G' lifted from G [1]. For any $y \in Y_{GG'}$ and any $e = \{v,w\} \in E'$, $y_e = 1$ indicates that v and w are in distinct components of the decomposition of G defined by the multicut $\{e' \in E \,|\, y_{e'} = 1\}$ of G. Formally, $Y_{GG'}$ is the set of all $y \in \{0,1\}^{E'}$ that satisfy the following system of linear inequalities:

$$\forall C \in \text{cycles}(G)\, \forall e \in C : \; y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \tag{4}$$

$$\forall \{v,w\} \in E' \setminus E\, \forall P \in vw\text{-paths}(G) :$$

$$y_{\{v,w\}} \leq \sum_{e \in P} y_e \tag{5}$$

$$\forall \{v,w\} \in E' \setminus E\, \forall C \in vw\text{-cuts}(G) :$$

$$1 - y_{\{v,w\}} \leq \sum_{e \in C} (1 - y_e) \quad . \tag{6}$$

2.3. Cost Function

For every $x \in \{0,1\}^{V \times L}$ and every $y \in \{0,1\}^{A \times L^2}$, a cost $\varphi(x,y) \in \mathbb{R}$ is defined by the form

$$\begin{aligned}\varphi(x,y) = &\sum_{v \in V} \sum_{l \in L} c_{vl}\, x_{vl} \\ &+ \sum_{vw \in A} \sum_{ll' \in L^2} c^{\sim}_{vw,ll'}\, x_{vl}\, x_{wl'}\, (1 - y_{\{v,w\}}) \\ &+ \sum_{vw \in A} \sum_{ll' \in L^2} c^{\not\sim}_{vw,ll'}\, x_{vl}\, x_{wl'}\, y_{\{v,w\}} \quad .\end{aligned} \tag{7}$$

2.4. Definition

We define the NL-LMP rigorously and concisely in the form of a linearly constrained binary cubic program.

Definition 1 For any connected graph $G = (V,E)$, any graph $G' = (V,E')$ with $E \subseteq E'$, any orientation $H = (V,A)$ of G', any finite, non-empty set L, any function $c : V \times L \to \mathbb{R}$ and any functions $c^{\sim}, c^{\not\sim} : A \times L^2 \to \mathbb{R}$, the instance of the *minimum cost node-labeling lifted multicut problem* (NL-LMP) with respect to $(G, G', H, L, c, c^{\sim}, c^{\not\sim})$ has the form

$$\min_{(x,y) \in X_{VL} \times Y_{GG'}} \varphi(x,y) \quad . \tag{8}$$

2.5. Special Cases

Below, we show that the NL-LMP generalizes the UIQP. This connects the NL-LMP to work on graphical models with second-order functions and finitely many labels. In addition, we show that NL-LMP generalizes the LMP, connecting the NL-LMP to recent work on lifted multicuts. Finally, we show that the NL-LMP is general enough to express subgraph selection, connectedness and disconnectedness constraints.

2.5.1 Unconstrained Integer Quadratic Program

Definition 2 For any graph $G' = (V,E')$, any orientation $H = (V,A)$ of G', any finite, non-empty set L, any $c : V \times L \to \mathbb{R}$ and any $c' : A \times L^2 \to \mathbb{R}$, the instance of the UIQP with respect to (G', H, L, c, c') has the form

$$\min_{x \in X_{VL}} \sum_{v \in V} \sum_{l \in L} c_{vl}\, x_{vl} + \sum_{vw \in A} \sum_{ll' \in L^2} c'_{vw,ll'}\, x_{vl}\, x_{wl'} \quad . \tag{9}$$

Lemma 1 *For any graph $G' = (V,E')$, any instance (G', H, L, c, c') of the* UIQP *and any $x \in X_{VL}$, x is a solution of this instance of the* UIQP *iff $(x, 1_{E'})$ is a solution of the instance $(G', G', H, L, c, c', c')$ of the* NL-LMP.

PROOF Without loss of generality, we can assume that G' is connected. (Otherwise, we add edges between nodes $v, w \in V$ as necessary and set $c'_{vw,ll'} = 0$ for any $l, l' \in L$.)

For any $x \in X_{GL}$, the pair $(x, 1_{E'})$ is a feasible solution of the instance of the NL-LMP because the map $1_{E'} : E' \to \{0,1\} : e \mapsto 1$ is such that $1_{E'} \in Y_{G'G'}$.

Moreover, $(x, 1_{E'})$ is a solution of the instance of the NL-LMP iff x is a solution of the instance of the UIQP because, for $c^{\not\sim} = c^{\sim}$, the form (7) of the cost function of the NL-LMP specializes to the form (9) of the cost function of the UIQP.

2.5.2 Minimum Cost Lifted Multicut Problem

Definition 3 [1] For any connected graph $G = (V,E)$, any graph $G' = (V,E')$ with $E \subseteq E'$ and any $c' : E' \to \mathbb{R}$, the instance of the minimum cost lifted multicut problem (LMP) with respect to (G, G', c') has the form

$$\min_{y \in Y_{GG'}} \sum_{e \in E'} c'_e y_e \quad . \tag{10}$$

Lemma 2 *Let (G, G', c') be any instance of the* LMP. *Let $(G, G', H, L, c, c^{\sim}, c^{\not\sim})$ be the instance of the* NL-LMP *with the same graphs and such that*

$$L = \{1\} \qquad c = 0 \qquad c^{\sim} = 0 \tag{11}$$

$$\forall (v,w) \in A : \quad c^{\not\sim}_{vw,11} = c'_{\{v,w\}} \quad . \tag{12}$$

Then, for any $y \in \{0,1\}^{E'}$, y is a solution of the instance of the LMP *iff $(1_{V \times L}, y)$ is a solution of the instance of the* NL-LMP.

PROOF Trivially, y is a feasible solution of the instance of the LMP iff $(1_{V \times L}, y)$ is a feasible solution of the instance of the NL-LMP. More specifically, y is a solution of the instance of the LMP iff $(1_{V \times L}, y)$ is a solution of the instance of the NL-LMP because, for any $x \in X_{VL}$, the cost function (7) of the NL-LMP assumes the special form below which is identical with the form in (10).

$$\varphi(x,y) \overset{(3),(11)}{=} \sum_{vw \in A} c^{\not\sim}_{vw,11} y_{\{v,w\}} \overset{(12)}{=} \sum_{e \in E'} c'_e y_e \quad . \tag{13}$$

2.5.3 Subgraph Selection

Applications such as [15, 29, 35, 36] require us to not only decompose a graph and label its nodes but to also select a subgraph. The NL-LMP is general enough to model subgraph selection. To achieve this, one proceeds in two steps: Firstly, one introduces a special label $\epsilon \in L$ to indicate that a node is not an element of the subgraph. We call these nodes *inactive*. All other nodes are called *active*. Secondly, one chooses a large enough $c^* \in \mathbb{N}$, a $c^\dagger \in \mathbb{N}_0$ and $c^\sim, c^\nsim$ such that

$$\forall vw \in A \, \forall l \in L \setminus \{\epsilon\}: \quad c^\sim_{vw,l\epsilon} = c^\sim_{vw,\epsilon l} = c^* \qquad (14)$$

$$c^\nsim_{vw,l\epsilon} = c^\nsim_{vw,\epsilon l} = 0 \qquad (15)$$

$$\forall vw \in A: \quad c^\sim_{vw,\epsilon\epsilon} = c^\dagger \ . \qquad (16)$$

By (14), inactive nodes are not joined with active nodes in the same component. By (15), cutting an inactive node from an active node has zero cost. By (16), joining inactive nodes has cost $c^\dagger$, possibly zero. Choosing $c^\dagger$ large enough implements an additional constraint proposed in [35] that inactive nodes are necessarily isolated. It is by this constraint and by a two-elementary label set that [35] is a specialization of the NL-LMP.

2.5.4 (Dis-)Connectedness Constraints

Some applications require us to constrain certain nodes to be in distinct components. One example is instance-separating semantic segmentation where nodes with distinct labels necessarily belong to distinct segments [21]. Other applications require us to constrain certain nodes to be in the same component. One example is articulated human body pose estimation for a single human in the optimization framework of [29] where every pair of active nodes necessarily belongs to the same human. Another example is connected foreground segmentation [27, 30, 34, 38] in which every pair of distinct foreground pixels necessarily belongs to the same segment.

The NL-LMP is general enough to model a combination of connectedness constraints and disconnectedness constraints by sufficiently large costs: In order to constrain distinct nodes $v, w \in V$ with labels $l, l' \in L$ to be in *the same component*, one introduces an edge $(v, w) \in A$, a large enough $c^* \in \mathbb{N}$ and costs $c^\nsim$ such that $c^\nsim_{vw,ll'} = c^\nsim_{vw,l'l} = c^*$. In order to constrain distinct nodes $v, w \in V$ with labels $l, l' \in L$ to be in *distinct components*, one introduces an edge $(v, w) \in A$, a large enough $c^* \in \mathbb{N}$ and costs $c^\sim$ such that $c^\sim_{vw,ll'} = c^\sim_{vw,l'l} = c^*$.

3. Algorithms

In this section, we define two local search algorithms that compute feasible solutions of the NL-LMP efficiently. Both algorithms attempt to improve a current feasible solution recursively by *transformations*. One class of transformations alters the node labeling of the graph by replacing a single node label. A second class of transformations alters the decomposition of the graph by moving a single node from one component to another. A third class of transformations alters the decomposition of the graph by joining two components.

As proposed by Kernighan and Lin [17] and generalized to the LMP by Keuper et al. [18], a local search is carried out not over the set of individual transformations of the current feasible solution but over a set of sequences of transformations. Complementary to this idea, we define and implement two schemes of combining transformations of the decomposition of the graph with transformations of the node labeling of the graph. This leads us to define two local search algorithms for the NL-LMP.

3.1. Encoding Feasible Solutions

To encode feasible solutions $(x, y) \in X_{VL} \times Y_{GG'}$ of the NL-LMP, we consider two maps: A *node labeling* $\lambda : V \to L$ that defines the $x^\lambda \in X_{VL}$ such that

$$\forall v \in V \, \forall l \in L: \quad x^\lambda_{vl} = 1 \Leftrightarrow \lambda(v) = l \ , \qquad (17)$$

and a so-called *component labeling* $\mu : V \to \mathbb{N}$ that defines the $y^\mu \in \{0,1\}^{E'}$ such that

$$\forall \{v, w\} \in E': \quad y^\mu_{\{v,w\}} = 0 \Leftrightarrow \mu(v) = \mu(w) \ . \qquad (18)$$

3.2. Transforming Feasible Solutions

To improve feasible solutions of the NL-LMP recursively, we consider three transformations of the encodings λ and μ:

For any node $v \in V$ and any label $l \in L$, the transformation $T_{vl} : L^V \to L^V : \lambda \mapsto \lambda'$ changes the label of the node v to l, i.e.

$$\forall w \in V: \quad \lambda'(w) := \begin{cases} l & \text{if } w = v \\ \lambda(w) & \text{otherwise} \end{cases} . \qquad (19)$$

For any node $v \in V$ and any component index $m \in \mathbb{N}$, the transformation $T'_{vm} : \mathbb{N}^V \to \mathbb{N}^V : \mu \mapsto \mu'$ changes the component index of the node v to m, i.e.

$$\forall w \in V: \quad \mu'(w) := \begin{cases} m & \text{if } w = v \\ \mu(w) & \text{otherwise} \end{cases} . \qquad (20)$$

For any component indices $m, m' \in \mathbb{N}$, the transformation $T'_{mm'} : \mathbb{N}^V \to \mathbb{N}^V : \mu \mapsto \mu'$ puts all nodes currently in the component indexed by m into the component indexed by m', i.e.

$$\forall w \in V: \quad \mu'(w) := \begin{cases} m' & \text{if } \mu(w) = m \\ \mu(w) & \text{otherwise} \end{cases} . \qquad (21)$$

Not every component labeling μ is such that $y^\mu \in Y_{GG'}$. In fact, y^μ is feasible if and only if, for every $m \in \mu(V)$, the node set $\mu^{-1}(m)$ is connected in G. For efficiency, we allow for transformations (20) whose output μ' violates this condition, as proposed in [18]. This happens when an *articulation node* of a component is moved to a different component. In order to *repair* any μ' for which y^μ is infeasible, we consider a map $R : \mathbb{N}^V \to \mathbb{N}^V : \mu' \mapsto \mu$ such that, for any $\mu' : V \to \mathbb{N}$ and any distinct $v, w \in V$, we have $\mu(v) = \mu(w)$ if and only if the exists a vw-path in G along which all nodes have the label $\mu'(v)$. We implement R as connected component labeling by breadth-first-search.

3.3. Searching Feasible Solutions

We now define two local search algorithms that attempt to improve an initial feasible solution recursively, by applying the transformation defined above. Initial feasible solutions are given, for instance, by the finest decomposition of the graph G that puts every node in a distinct component, or by the coarsest decomposition of the graph G that puts every node in the same component, each together with any node labeling. We find an initial feasible solution for our local search algorithm by first fixing an optimal label for every node independently and by then solving the resulting LMP, i.e., (8) for the fixed labels $x \in X_{VL}$, by means of greedy agglomerative edge contraction [18].

KLj/r Algorithm. The first local search algorithm we define, alternating Kernighan-Lin search with joins and node relabeling, KLj/r, alternates between transformations of the node labeling and transformations of the decomposition. For a fixed decomposition, the labeling is transformed by Func. 1 which greedily updates labels of nodes independently. For a fixed labeling, the decomposition is transformed by Func. 2, *without those parts of the function that are written in green*, i.e., precisely the algorithm KLj of [18]. (All symbols that appear in the pseudo-code are defined above, except the iteration counter t, cost differences δ, Δ, and 01-vectors α used for bookkeeping, to avoid redundant operations.)

KLj$*$r Algorithm. The second local search algorithm we define, joint Kernighan-Lin search with joins and node relabeling, KLj$*$r, transforms the decomposition and the node labeling jointly, by combining the transformations (19)–(21) in a novel manner. It is given by Func. 2, *with those parts of the function that are written in green.*

Like the alternating algorithm KLj/r, the joint algorithm KLj$*$r updates the labeling for a fixed decomposition (calls of Func. 1 from Func. 2). Unlike the alternating algorithm KLj/r, the joint algorithm KLj$*$r updates the decomposition and the labeling also jointly. This happens in Func. 3 that is called from KLj$*$r, *with the part that is written in green.*

Func. 3 looks at two components $V := \mu^{-1}(m)$ and $W := \mu^{-1}('m)$ of the current decomposition. It attempts to improve the decomposition as well as the labeling by moving a node from V to W or from W to V *and by simultaneously changing its label.* As proposed by Kernighan and Lin [17],

Function 1: $(\Delta, \lambda') =$ update-labeling(μ, λ)

```
$\lambda_0 := \lambda$    $\Delta := 0$    $t := 0$
***repeat***
    ***choose*** $(\hat{v}, \hat{l}) \in \operatorname{argmin}_{(v,l) \in V \times L} \varphi(x^{T_{vl}(\lambda_t)}, y^{\mu_t}) - \varphi(x^{\lambda_t}, y^{\mu_t})$
    $\delta := \varphi(x^{T_{\hat{v}\hat{l}}(\lambda_t)}, y^{\mu_t}) - \varphi(x^{\lambda_t}, y^{\mu_t})$
    ***if*** $\delta < 0$
        $\lambda_{t+1} := T_{\hat{v}\hat{l}}(\lambda_t)$
        $\Delta := \Delta + \delta$
        $t := t + 1$
    ***else***
        ***return*** $(\Delta, \lambda_t)$
```

Func. 3 does not make such transformations greedily but first constructs a sequence of such transformations greedily and then executes the first k with k chosen so as to decrease the objective value maximally. KLj/r constructs a sequence of transformations analogously, but the node labeling remains fixed throughout every transformation of the decomposition. Thus, KLj$*$r is a local search algorithm whose local neighborhood is strictly larger than that of KLj/r.

Our C++ implementation computes cost differences incrementally and solves the optimization problem over trans-

Function 2: $(\Delta', \mu', \lambda') =$ update-lifted-multicut(μ, λ)

```
$\mu_0 := \mu$    $t := 0$
$(\delta, \lambda_0) :=$ update-labeling$(\mu_0, \lambda)$
***let*** $\alpha_0 : \mathbb{N} \to \{0,1\}$ such that $\alpha_0(\mathbb{N}) = 1$
***repeat***
    $\Delta := 0$    $\mu_{t+1} := \mu_t$    $\lambda_{t+1} := \lambda_t$
    ***let*** $\alpha_{t+1} : \mathbb{N} \to \{0,1\}$ such that $\alpha_{t+1}(\mathbb{N}) = 0$
    ***for each*** $\{m, m'\} \in \binom{\mu(V)}{2}$
        ***if*** $\alpha_t(m) = 0 \wedge \alpha_t(m') = 0$
            ***continue***
        $(\delta, \mu_{t+1}, \lambda_{t+1}) :=$ update-2-cut$(\mu_{t+1}, \lambda_{t+1}, m, m')$
        ***if*** $\delta < 0$
            $\alpha_{t+1}(m) := 1$   $\alpha_{t+1}(m') := 1$   $\Delta := \Delta + \delta$
    ***for each*** $m \in \mu(V)$
        ***if*** $\alpha_t(m) = 0$
            ***continue***
        $m' := 1 + \max \mu(V)$                (new component)
        $(\delta, \mu_{t+1}, \lambda_{t+1}) :=$ update-2-cut$(\mu_{t+1}, \lambda_{t+1}, m, m')$
        ***if*** $\delta < 0$
            $\alpha_{t+1}(m) := 1$   $\alpha_{t+1}(m') := 1$   $\Delta := \Delta + \delta$
    $(\delta, \lambda_{t+1}) :=$ update-labeling$(\mu_{t+1}, \lambda_{t+1})$
    $\Delta := \Delta + \delta$
    ***if*** $y^{\mu_{t+1}} \notin Y_{GG'}$
        $\mu_{t+1} := R(\mu_{t+1})$                (repair heuristic)
        $\Delta := \varphi(x^{\lambda_{t+1}}, y^{\mu_{t+1}}) - \varphi(x^{\lambda_0}, y^{\mu_0})$
    $t := t + 1$
***while*** $\Delta < 0$
```

formations by means of a priority queue, as described in detail in Appendix A. The time and space complexities are identical to those of KLj and are established in [18], as transformations that take linear time in the number of labels take constant time in the size of the graph.

Function 3: $(\Delta', \mu', \lambda') =$ update-2-cut(μ, λ, m, m')

```
$\mu_0 := \mu$    $\lambda_0 := \lambda$    $t := 0$
*if* $\mu^{-1}(m') = \emptyset$
    $V_0 := \mu^{-1}(m)$
*else*
    $V_0 := \{v \in \mu^{-1}(m) \,|\, \exists w \in \mu^{-1}(m') : \{v, w\} \in E\}$
*if* $\mu^{-1}(m) = \emptyset$
    $W_0 := \mu^{-1}(m')$
*else*
    $W_0 := \{w \in \mu^{-1}(m') \,|\, \exists v \in \mu^{-1}(m) : \{v, w\} \in E\}$
*let* $\alpha : \mathbb{N} \to \{0, 1\}$ such that $\alpha(\mathbb{N}) = 1$
*while* $V_t \cup W_t \neq \emptyset$
    $\delta := \delta' := \infty$
    *if* $V_t \neq \emptyset$
        *choose* $(\hat{v}, \hat{l}) \in \operatorname{argmin}_{(v,l) \in V_t \times L} \varphi(x^{T_{vl}(\lambda_t)}, y^{T'_{vm'}(\mu_t)}) - \varphi(x^{\lambda_t}, y^{\mu_t})$
        $\delta := \varphi(x^{T_{\hat{v}\hat{l}}(\lambda_t)}, y^{T'_{\hat{v}m'}(\mu_t)}) - \varphi(x^{\lambda_t}, y^{\mu_t})$
    *if* $W_t \neq \emptyset$
        *choose* $(\hat{w}, \hat{l}) \in \operatorname{argmin}_{(w,l) \in W_t \times L} \varphi(x^{T_{wl}(\lambda_t)}, y^{T'_{wm}(\mu_t)}) - \varphi(x^{\lambda_t}, y^{\mu_t})$
        $\delta' := \varphi(x^{T_{\hat{w}\hat{l}}(\lambda_t)}, y^{T'_{\hat{w}m}(\mu_t)}) - \varphi(x^{\lambda_t}, y^{\mu_t})$
    *if* $\delta \leq \delta'$
        $\mu_{t+1} := T'_{\hat{v}m'}(\mu_t)$    (move node $\hat{v}$ to component $m'$)
        $\lambda_{t+1} := T_{\hat{v}\hat{l}}(\lambda_t)$    (label node $\hat{v}$ with label $\hat{\lambda}$)
        $\alpha(\hat{v}) := 0$    (mark $\hat{v}$ as inactive)
    *else*
        $\mu_{t+1} := T'_{\hat{w}m}(\mu_t)$    (move node $\hat{w}$ to component $m$)
        $\lambda_{t+1} := T_{\hat{w}\hat{l}}(\lambda_t)$    (label node $\hat{w}$ with label $\hat{\lambda}$)
        $\alpha(\hat{w}) := 0$    (mark $\hat{w}$ as inactive)
    $V_{t+1} := \{v \in V \,|\, \mu_{t+1}(v) = m \wedge \alpha(v) = 1 \wedge$
                        $\exists \{v, w\} \in E : \mu_{t+1}(w) = m'\}$
    $W_{t+1} := \{w \in V \,|\, \mu_{t+1}(w) = m' \wedge \alpha(w) = 1 \wedge$
                        $\exists \{v, w\} \in E : \mu_{t+1}(v) = m\}$
    $t := t + 1$
$\hat{t} := \min \operatorname{argmin}_{t' \in \{0,\dots,t\}} \varphi(x^{\lambda_{t'}}, y^{\mu_{t'}}) - \varphi(x^{\lambda_0}, y^{\mu_0})$
$\Delta_1 := \varphi(x^{\lambda_{\hat{t}}}, y^{\mu_{\hat{t}}}) - \varphi(x^{\lambda_0}, y^{\mu_0})$
$\Delta_2 := \varphi(x^{\lambda_0}, y^{T'_{mm'}(\mu)}) - \varphi(x^{\lambda_0}, y^{\mu_0})$    (join $m$ and $m'$)
*if* $\min\{\Delta_1, \Delta_2\} \geq 0$
    *return* $(0, \mu, \lambda)$
*else if* $\Delta_1 < \Delta_2$
    *return* $(\Delta_1, \mu_{\hat{t}}, \lambda_{\hat{t}})$
*else*
    *return* $(\Delta_2, T_{mm'}(\mu), \lambda)$
```

4. Applications

We show applications of the proposed problem and algorithms to three distinct computer vision tasks: articulated human body pose estimation, multiple object tracking, and instance-separating semantic segmentation. For each task, we set up instances of the NL-LMP from published data, using published algorithms.

4.1. Articulated Human Body Pose Estimation

We turn toward applications of the NL-LMP and the algorithms KLj/r and KLj∗r to the task of estimating the articulated poses of all humans visible in an image. Pishchulin et al. [29] and Insafutdinov et al. [15] approach this problem via a graph decomposition and node labeling problem that we identify as a special case of the NL-LMP with $c^{\not\sim} = 0$ and with subgraph selection (Section 2.5.3). We relate their notation to ours in Appendix B. Nodes in their graph are putative detections of body parts. Labels define body part classes (head, wrist, etc.). In our notation, $x_{vl} = 1$ indicates that the putative detection v is a body part of class l, and $y_{vw} = 1$ indicates that the body parts v and w belong to distinct humans. The test set of [15] consists of 1758 such instances of the NL-LMP.

To tackle these instances, Insafutdinov et al. define and implement a branch-and-cut algorithm in the integer linear programming software framework Gurobi. We refer to their published C++ implementation as B&C.

Cost and time. In Fig. 2, we compare the convergence of B&C (feasible solutions and lower bounds) with the convergence of our algorithms, KLj/r and KLj∗r (feasible solutions only). Shown in this figure is the average objective value over the test set w.r.t. the absolute running time. Thanks to the lower bounds obtained by B&C, it can be seen from this figure that KLj/r and KL+r arrive at near optimal feasible solutions after 10^{-1} seconds, five orders of magnitude faster than B&C. This result shows that primal feasible heuristics for the NL-LMP, such as KLj/r and KLj∗r, are practically useful in the context of this application.

Application-specific accuracy. In Tab. 1, we compare feasible solutions output by KLj/r and KLj∗r after convergence with those obtained by B&C after at most three hours. It can be seen from this table that the feasible solutions output by KLj/r and KLj∗r have lower cost and higher application-specific accuracy (Acc) on average. KLj∗r yields a lower average cost than KLj/r with slightly higher running time. The fact that lower cost does not mean higher application-specific accuracy is explained by the application-specific accuracy measure that does not penalize false positives.

The shorter absolute running time of KLj/r and KLj∗r allows us to increase the number of nodes from 150, as in [15], to 420. It can be seen from the last two rows of Tab. 1 that this increases the application-specific accuracy by 4%.

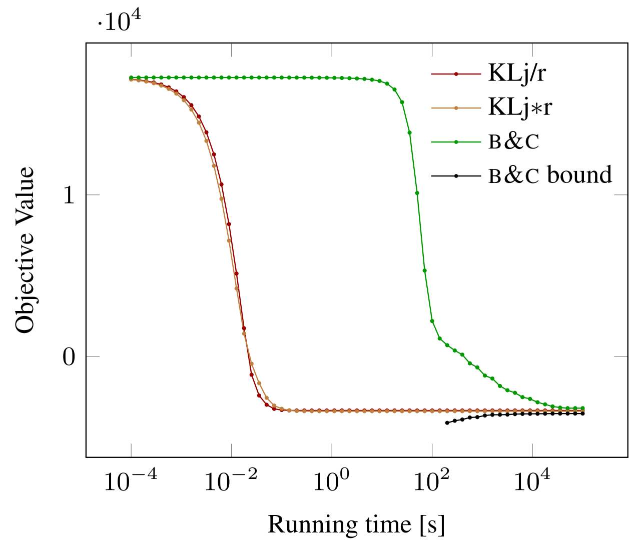


Figure 2: Convergence of B&C, KLj/r and KLj∗r in an application to the task of articulated human body pose estimation.

$\lvert V\rvert$	Alg.	AP	Mean cost	Mean time [s]	Median time [s]
150	[15]	65.5	-3013.30	9519.26	308.28
	KLj/r	66.5	-3352.74	**0.033**	**0.031**
	KLj∗r	**66.6**	**-3419.07**	0.119	0.100
420	KLj/r	**70.6**	-6184.36	**0.098**	**0.053**
	KLj∗r	**70.6**	**-6608.53**	0.534	0.254

Table 1: Comparison of B&C [15], KLj/r and KLj∗r in an application to the task of human body pose estimation.

4.2. Instance-Separating Semantic Segmentation

We turn toward applications of the NL-LMP and the algorithms KLj/r and KLj∗r to the task of instance-separating semantic image segmentation. We state this problem here as an NL-LMP whose nodes correspond to pixels in a given image, and whose labels define classes of objects (human, car, bicycle, etc.). In our notation, $x_{vl} = 1$ indicates that the pixel v shows an object of class l, and $y_{vw} = 1$ indicates that the pixels v and w belong to distinct objects.

Specifically, we apply the algorithms KLj/r and KLj∗r to instances of the NL-LMP for the task of instance-separating semantic segmentation posed by the KITTI [13] and Cityscapes [9] benchmarks. For KITTI, we construct instances of the NL-LMP from data published by Uhrig et al. [37] as described in detail in Appendix C. For Cityscapes, we construct instances of the NL-LMP as follows. For costs $c^{\sim}$, we again use data of Uhrig et al. [37]. For costs c, we use a ResNet-50 [14] network with dilated convolutions [7]. We train the network in a fully convolutional manner with image crops (768 px·512 px) subjected to minimal data augmentation (horizontal flips). More details are in Appendix C.

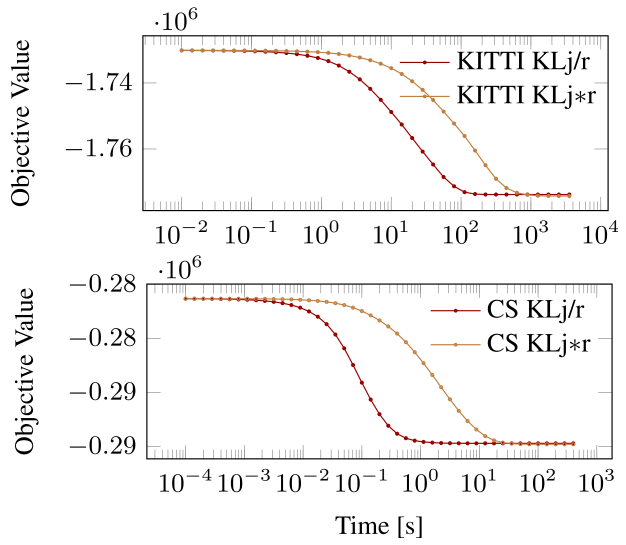


Figure 3: Convergence of KLj/r and KLj∗r in an application to the task of instance-separating semantic segmentation.

Data	Algorithm	AP	$\text{AP}^{50\%}$
KITTI validation [13]	KLj/r	**50.5**	**82.9**
	KLj∗r	50.3	82.4
KITTI test [13]	[37]	41.6	69.1
	KLj∗r	**43.6**	**71.4**
Cityscapes validation [9]	KLj/r	11.3	**26.8**
	KLj∗r	**11.4**	26.1
Cityscapes test [9]	MCG+R-CNN [9]	4.6	12.9
	[37]	8.9	21.1
	KLj∗r	**9.8**	**23.2**

Table 2: Comparison of KLj/r and KLj∗r in an application to the task of instance-separating semantic segmentation.

Cost and time. In Fig. 3, we compare the convergence of KLj/r and KLj∗r. Shown in this figure w.r.t. the absolute running time are the average objective values over the KITTI and Cityscapes validation sets, respectively. It can be seen from this figure that KLj/r converges faster than KLj∗r. Both algorithms are practical for this application but not efficient enough for video processing in real-time.

Application-specific accuracy. In Tab. 2, we compare feasible solutions output by KLj/r and KLj∗r after convergence with the output of the algorithm of Uhrig et al [37]. It can be seen from this table that the application of KLj/r and KLj∗r improves the application-specific average precision, AP and $\text{AP}^{50\%}$. The AP of feasible solutions output by KLj∗r for the Cityscapes test set is higher than that of any published algorithm. A higher AP is reported by Kirillov et al. [20], who use the model and algorithms proposed in this paper with improved pairwise $c^{\sim}$ and unary c costs.

Method	MOTA ↑	MOTP ↑	FAF ↓	MT ↑	ML ↓	FP ↓	FN ↓	ID Sw ↓	Frag↓	Hz ↑	Detector
[12]	41.0	74.8	1.3	11.6%	51.3%	7896	99224	430	963	1.1	Public
[19]	42.9	76.6	1.0	13.6%	46.9%	**5668**	97919	499	659	0.8	Public
[8]	46.4	76.6	1.6	**18.3%**	41.4%	9753	**87565**	**359**	**504**	2.6	Public
[36]	46.3	75.7	1.1	15.5%	**39.7%**	6373	90914	657	1114	0.8	Public
KLj/r	**47.6**	**78.5**	1.0	17.0%	40.4%	5844	89093	629	768	**8.3**	Public
KLj∗r	**47.6**	**78.5**	**0.98**	17.0%	40.4%	5783	89160	627	761	0.7	Public

Table 3: Comparison of the algorithms KLj/r and KLj∗r in an application to the task of multiple object tracking.

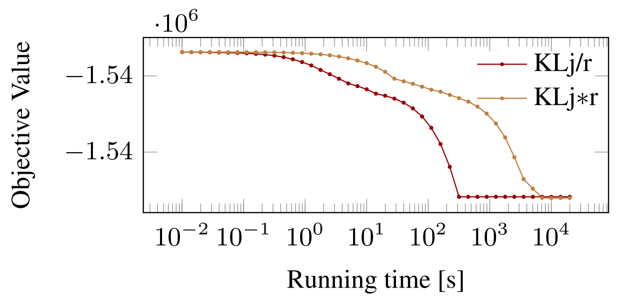


Figure 4: Convergence of the algorithms KLj/r and KLj∗r in an application to the task of multiple object tracking.

4.3. Multiple Object Tracking

We turn toward applications of the NL-LMP and the algorithms KLj/r and KLj∗r to the task of multiple object tracking. Tang et al. [35] approach this problem via a graph decomposition and node labeling problem that we identify as a special case of the NL-LMP with two labels and subgraph selection (Sec. 2.5.3). We relate their notation to ours rigorously in Appendix D. Nodes in their graph are putative detections of persons. In our notation, $x_{vl} = 1$ indicates that the putative detection v is active, and $y_{vw} = 1$ indicates that the putative detections v and w are of distinct persons. For the test set of the multiple object tracking benchmark [25], Tang et al. construct seven such instances of the NL-LMP.

To tackle these large instances, in [36] Tang et al. solve the subgraph suppression problem first and independently, by thresholding on the detections scores, and then solve the minimum cost multicut problem for the remaining subgraph by means of the algorithm KLj of [18], without re-iterating. Here, we apply to the joint NL-LMP the algorithms KLj/r and KLj∗r and compare their output to that of [36] and of other top-performing algorithms [8, 12, 19]. We use the same data as in [36], therefore the performance gain is due to our algorithms that solve the full problem [35].

Cost and time. The convergence of the algorithms KLj/r and KLj∗r is shown in Fig. 4. It can be seen from this figure that KLj/r converges faster than KLj∗r.

Application-specific accuracy. We compare the feasible solutions output by KLj/r and KLj∗r to the state-of-the-art for the benchmark [25]. To this end, we report in Tab. 3 the standard CLEAR MOT metric, including: multiple object tracking accuracy (MOTA), multiple object tracking precision (MOTP), mostly tracked object (MT), mostly lost (ML) and tracking fragmentation (FM). MOTA combines identity switches (ID Sw), false positives (FP) and false negatives (FN) and is most widely used. Our feasible solutions are published also at the benchmark website unser the names NLLMP (KLj/r) and NLLMPj (KLj∗r). It is can be seen from Tab. 3 that the feasible solutions obtained by KLj/r and KLj∗r rank first in MOTA and MOTP. Compared to [36], KLj/r and KLj∗r reduce the number of false positives and false negatives. The average inverse running time per frame of a video sequence (column "Hz" in the table) is better for KLj/r by a margin than for any other algorithm. Overall, these results show the practicality of the NL-LMP in conjunction with the local search algorithms KLj/r and KLj∗r for applications in multiple object tracking.

5. Conclusion

We have stated the minimum cost node labeling lifted multicut problem, NL-LMP, an NP-hard combinatorial optimization problem whose feasible solutions define both a decomposition and a node labeling of a given graph. We have defined and implemented two local search algorithms, KLj/r and KLj∗r, that converge monotonously to a local optimum, offering a feasible solution at any time. We have shown applications of these algorithms to the tasks of articulated human body pose estimation, multiple object tracking and instance-separating semantic segmentation, obtaining state-of-the-art application-specific accuracy. We conclude that the NL-LMP is a useful mathematical abstraction in the field of computer vision that allows researchers to apply the same optimization algorithm to diverse computer vision tasks. To foster collaboration between the fields of computer vision and combinatorial optimization, we make our code publicly available at https://github.com/bjoern-andres/graph

References

[1] B. Andres, A. Fuksová, and J.-H. Lange. Lifting of multicuts. *CoRR*, abs/1503.03791, 2016. 3

[2] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2D human pose estimation: New benchmark and state of the art analysis. In *CVPR*, 2014. 2

[3] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *TPAMI*, 33(9):1806–1819, 2011. 1

[4] W. Brendel, M. Amer, and S. Todorovic. Multiobject tracking as maximum weight independent set. In *CVPR*, 2011. 1

[5] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016. 13

[6] L. C. Chen, S. Fidler, and R. Urtasun. Beat the MTurkers: Automatic image labeling from weak 3d supervision. In *CVPR*, 2014. 12, 13, 14

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv:1606.00915*, 2016. 7, 13

[8] W. Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *ICCV*, 2015. 1, 8

[9] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for semantic urban scene understanding. In *CVPR*, June 2016. 1, 7, 12, 13, 14, 15

[10] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, 2016. 1

[11] L. Fagot-Bouquet, R. Audigier, Y. Dhome, and F. Lerasle. Improving multi-frame data association with sparse representations for robust near-online multi-object tracking. In *ECCV*, 2016. 1

[12] L. Fagot-Bouquet, R. Audigier, Y. Dhome, and F. Lerasle. Improving multi-frame data association with sparse representations for robust near-online multi-object tracking. In *ECCV*, 2016. 8

[13] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *CVPR*, 2012. 1, 7, 12, 13

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 7, 13

[15] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele. DeeperCut: A deeper, stronger, and faster multi-person pose estimation model. In *ECCV*, 2016. 2, 4, 6, 7, 11

[16] J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, T. Kröger, J. Lellmann, N. Komodakis, B. Savchynskyy, and C. Rother. A comparative study of modern inference techniques for structured discrete energy minimization problems. *International Journal of Computer Vision*, 115(2):155–184, 2015. 2

[17] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, 49:291–307, 1970. 2, 4, 5, 10, 11

[18] M. Keuper, E. Levinkov, N. Bonneel, G. Lavoué, T. Brox, and B. Andres. Efficient decomposition of image and mesh graphs by lifted multicuts. In *ICCV*, 2015. 2, 4, 5, 6, 8, 10, 11

[19] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg. Multiple hypothesis tracking revisited. In *ICCV*, 2015. 1, 8

[20] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother. Instancecut: from edges to instances with multicut. *CoRR*, abs/1611.08272, 2016. 7

[21] T. Kroeger, J. H. Kappes, T. Beier, U. Koethe, and F. A. Hamprecht. Asymmetric cuts: Joint image labeling and partitioning. In *GCPR*, 2014. 1, 4

[22] X. Liang, Y. Wei, X. Shen, J. Yang, L. Lin, and S. Yan. Proposal-free network for instance-level object segmentation. *CoRR*, abs/1509.02636, 2015. 1

[23] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *CoRR*, abs/1506.04579, 2015. 13

[24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 11

[25] A. Milan, L. Leal-Taixé, I. D. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831*, 2016. 1, 8

[26] A. Milan, S. Roth, and K. Schindler. Continuous energy minimization for multitarget tracking. *TPAMI*, 36(1):58–72, 2014. 1

[27] S. Nowozin and C. H. Lampert. Global interactions in random field models: A potential function ensuring connectedness. *SIAM Journal on Imaging Sciences*, 3(4):1048–1074, 2010. 4

[28] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, 2011. 1

[29] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. Gehler, and B. Schiele. DeepCut: Joint subset partition and labeling for multi person pose estimation. In *CVPR*, 2016. 2, 4, 6, 11

[30] M. Rempfler, B. Andres, and B. Menze. The minimum cost connected subgraph problem in medical image analysis. In *MICCAI*, 2016. 4

[31] M. Ren and R. Zemel. End-to-end instance segmentation and counting with recurrent attention. In *arXiv:1605.09410 [cs.CV]*, 2016. 1, 14

[32] B. Romera-Paredes and P. H. S. Torr. Recurrent instance segmentation. In *ECCV*, 2016. 1

[33] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1

[34] J. Stühmer and D. Cremers. A fast projection method for connectivity constraints in image segmentation. In *EMMCVPR*, 2015. 4

[35] S. Tang, B. Andres, M. Andriluka, and B. Schiele. Subgraph decomposition for multi-target tracking. In *CVPR*, 2015. 1, 4, 8, 14

[36] S. Tang, B. Andres, M. Andriluka, and B. Schiele. Multi-person tracking by multicut and deep matching. In *ECCV Workshops*, 2016. 1, 4, 8

[37] J. Uhrig, M. Cordts, U. Franke, and T. Brox. Pixel-level encoding and depth layering for instance-level semantic labeling. In *GCPR*, 2016. 7, 11, 13, 14

[38] S. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. In *CVPR*, 2008. 4

[39] Z. Wu, C. Shen, and A. van den Hengel. Bridging category-level and instance-level semantic image segmentation. *CoRR*, abs/1605.06885, 2016. 13

[40] Y. Xiang, A. Alahi, and S. Savarese. Learning to track: Online multi-object tracking by decision making. In *ICCV*, 2015. 1

[41] A. R. Zamir, A. Dehghan, and M. Shah. GMCP-Tracker: Global multi-object tracking using generalized minimum clique graphs. In *ECCV*, 2012. 1

[42] Z. Zhang, S. Fidler, and R. Urtasun. Instance-level segmentation with deep densely connected MRFs. In *CVPR*, 2016. 1, 14

[43] Z. Zhang, A. G. Schwing, S. Fidler, and R. Urtasun. Monocular object instance segmentation and depth ordering with CNNs. In *ICCV*, 2015. 1, 14

A. Implementation Details

Func. 1 and 3 choose local transformations that decrease the cost optimally. Our implementation computes cost differences incrementally, as proposed by Kernighan and Lin [17]. The exact computations are described below.

Transforming the Labeling. Func. 1 repeatedly chooses a node $\hat{v}$ and a label $\hat{l}$ such that labeling $\hat{v}$ with $\hat{l}$ decreases the cost maximally. I.e., Func. 1 repeatedly solves the optimization problem

$$(\hat{v}, \hat{l}) \in \operatorname*{argmin}_{(v,l)\in V\times L} \varphi(x^{T_{vl}(\lambda_t)}, y^{\mu_t}) - \varphi(x^{\lambda_t}, y^{\mu_t}) \quad . \tag{22}$$

While $\varphi(x^{\lambda_t}, y^{\mu_t})$ is constant, it is more efficient to minimize the difference $\varphi(x^{T_{vl}(\lambda_t)}, y^{\mu_t}) - \varphi(x^{\lambda_t}, y^{\mu_t})$ than to minimize $\varphi(x^{T_{vl}(\lambda_t)}, y^{\mu_t})$, as the difference can be computed locally, considering only the neighbors w of v in G':

$$\begin{aligned}
&\varphi(x^{T_{vl}(\lambda_t)}, y^{\mu_t}) - \varphi(x^{\lambda_t}, y^{\mu_t}) \\
&= c_{vl} - c_{v\lambda_t(v)} \\
&\quad + \sum_{vw\in A} (1 - y_{\{v,w\}}) \left(c^{\sim}_{vw,l\lambda_t(w)} - c^{\sim}_{vw,\lambda_t(v)\lambda_t(w)} \right) \\
&\quad + \sum_{wv\in A} (1 - y_{\{v,w\}}) \left(c^{\sim}_{wv,\lambda_t(w)l} - c^{\sim}_{wv,\lambda_t(w)\lambda_t(v)} \right) \\
&\quad + \sum_{vw\in A} y_{\{v,w\}} \left(c^{\nsim}_{vw,l\lambda_t(w)} - c^{\nsim}_{vw,\lambda_t(v)\lambda_t(w)} \right) \\
&\quad + \sum_{wv\in A} y_{\{v,w\}} \left(c^{\nsim}_{wv,\lambda_t(w)l} - c^{\nsim}_{wv,\lambda_t(w)\lambda_t(v)} \right) \\
&=: \Delta_{t,vl} \quad .
\end{aligned} \tag{23}$$

Initially, i.e., for $t = 0$, we compute $\Delta_{0,vl}$ for every node v and every label l. In subsequent iterations, i.e., for $t \in \mathbb{N}$ and the minimizer $(\hat{v}, \hat{l})$ of (22) chosen in this iteration, we update cost differences for all neighbors w of $\hat{v}$ in G' and all labels $l \in L$. The update rule is written below for an edge $(w, \hat{v}) \in A$. The update for an edge in the opposite direction is analogous. Below, (24) subtracts the costs due to $\hat{v}$ being labeled $\lambda_t(\hat{v})$ (which is possibly outdated), while (25) adds the costs due to $\hat{v}$ having obtained a new and possibly different label $\hat{l}$.

$$\begin{aligned}
\Delta_{t+1,wl} =\, &\Delta_{t,wl} \\
&- (1 - y_{\{w,\hat{v}\}}) \left(c^{\sim}_{w\hat{v},l\lambda_t(\hat{v})} - c^{\sim}_{w\hat{v},\lambda_t(w)\lambda_t(\hat{v})} \right) \\
&- y_{\{w,\hat{v}\}} \left(c^{\nsim}_{w\hat{v},l\lambda_t(\hat{v})} - c^{\nsim}_{w\hat{v},\lambda_t(w)\lambda_t(\hat{v})} \right) \qquad (24) \\
&+ (1 - y_{\{w,\hat{v}\}}) \left(c^{\sim}_{w\hat{v},l\hat{l}} - c^{\sim}_{w\hat{v},\lambda_t(w)\hat{l}} \right) \\
&+ y_{\{w,\hat{v}\}} \left(c^{\nsim}_{w\hat{v},l\hat{l}} - c^{\nsim}_{w\hat{v},\lambda_t(w)\hat{l}} \right) \quad . \qquad (25)
\end{aligned}$$

We solve (22) by means of a priority queue in time complexity $\mathcal{O}(|V|\log|V| + |V|(|L| + \log|V|)\deg G')$ with $\deg G'$ the node degree of G'. For sparse graphs and constant number $|L|$ of labels, this is $\mathcal{O}(|V|\log|V|)$.

Transformation of Labeling and Decomposition. The algorithm KLj of [18] for the minimum cost lifted multicut problem generalizes the Kernighan-Lin-Algorithm [17] for the minimum cost multicut problem. The algorithms KLj/r and KLj∗r we define further generalize KLj to the NL-LMP. The critical part is Func. 3 that solves the optimization problem

$$(\hat{v}, \hat{l}) \in \operatorname*{argmax}_{(v,l)\in V_t\times L} \varphi(x^{T_{vl}(\lambda_t)}, y^{T'_{vm'}(\mu_t)}) - \varphi(x_t^{\lambda}, y^{\mu_t}) \tag{26}$$

Let us consider w.l.o.g. two sets of vertices A and B representing two neighboring components of the graph G. Then we compute $\forall v \in A \cup B, \forall l \in L$:

$$\Delta_{vl} = c_{v\lambda_t(v)} - c_{vl} + \tag{27}$$

$$\begin{cases}
\sum_{w\in A\setminus\{v\}} & c^{\sim}_{vw,\lambda_t(v)\lambda_t(w)} - c^{\nsim}_{vw,l\lambda_t(w)} \\
\sum_{w\in B} & c^{\nsim}_{vw,\lambda_t(v)\lambda_t(w)} - c^{\sim}_{vw,l\lambda_t(w)} \\
\sum_{w\notin A\cup B} & c^{\nsim}_{vw,\lambda_t(v)\lambda_t(w)} - c^{\nsim}_{vw,l\lambda_t(w)}
\end{cases} , \tag{28}$$

where $w \in \mathcal{N}_{G'}(v)$. In eq. (28) the first two cases are exactly the same as given in [17] for the edges *between* partitions A and B. But in our case changing vertex's class label may affect the cut costs of edges between A and B and any other partition. Also, we have join and cut costs.

Let us assume w.l.o.g. that vertex $\hat{v}$ was chosen to be moved from set A to set B, i.e. $A = A \setminus \{\hat{v}\}$. Now we can update the expected gains of $\forall w \in \mathcal{N}_{G'}(\hat{v}), \forall l \in L$:

$$\begin{aligned}
\Delta_{wl} = \Delta_{wl} &- \left(c^{\sim}_{\hat{v}w,\lambda_t(\hat{v})\lambda_t(w)} - c^{\nsim}_{\hat{v}w,\lambda_t(\hat{v})l} \right) \\
&+ c^{\nsim}_{\hat{v}w,\hat{l}\lambda_t(w)} - c^{\sim}_{\hat{v}w,\hat{l}l} \quad , \quad \text{if } w \in A \quad ,
\end{aligned} \tag{29}$$

$$\begin{aligned}
\Delta_{wl} = \Delta_{wl} &- \left(c^{\nsim}_{\hat{v}w,\lambda_t(\hat{v})\lambda_t(w)} - c^{\sim}_{\hat{v}w,\lambda_t(\hat{v})l} \right) \\
&+ c^{\sim}_{\hat{v}w,\hat{l}\lambda_t(w)} - c^{\nsim}_{\hat{v}w,\hat{l}l} \quad , \quad \text{if } w \in B \quad .
\end{aligned} \tag{30}$$

After that $B = B \cup \{\hat{v}\}$. In the above equations, the expression in parenthesis cancels the current contribution for vertex w, that assumed $\hat{v}$ was labeled $\lambda_t(v)$ and belonged to partition A. For the case when $|L| = 1$ and $c^{\not\sim} = c^{\sim}$ the above equations simplify to exactly the ones as in [17], but multiplied by 2, because in our objective we have two terms that operate on the edges simultaneously.

As we generalize [18] by an additional loop over the set L of labels, the analysis of the time complexity carries over from [18] with an additional multiplicative factor $|L|$.

B. Articulated Human Body Pose Estimation

B.1. Problem Statement

Pishchulin et al. [29] introduce a binary cubic problem w.r.t. a set C of body joint classes and a set D of putative detections of body joints. Every feasible solution is a pair (x, y) with $x : D \times C \to \{0,1\}$ and $y : \binom{D}{2} \to \{0,1\}$, constrained by the following system of linear inequalities:

$$\forall d \in D \forall cc' \in \tbinom{C}{2} : \quad x_{dc} + x_{dc'} \leq 1 \tag{31}$$

$$\forall dd' \in \tbinom{D}{2} : \quad y_{dd'} \leq \sum_{c \in C} x_{dc}$$

$$y_{dd'} \leq \sum_{c \in C} x_{d'c} \tag{32}$$

$$\forall dd'd'' \in \tbinom{D}{3} : \quad y_{dd'} + y_{d'd''} - 1 \leq y_{dd''} \tag{33}$$

The objective function has the form below with coefficients α and β.

$$\sum_{d \in D} \sum_{c \in C} \alpha_{dc} \, x_{dc} + \sum_{dd' \in \binom{D}{2}} \sum_{c,c' \in C} \beta_{dd'cc'} x_{dc} x_{d'c'} y_{dd'} \tag{34}$$

We identify the solutions of this problem with the solutions of the NL-LMP w.r.t. the complete graphs $G = G' = (D, \binom{D}{2})$, the label set $L = C \cup \{\epsilon\}$ and the costs $c^{\not\sim} = 0$ and

$$c_{vl} := \begin{cases} \alpha_{vl} & \text{if } l \in C \\ 0 & \text{if } l = \epsilon \end{cases} \tag{35}$$

$$c^{\sim}_{vw,ll'} := \begin{cases} \beta_{vwll'} & \text{if } l \in C \wedge l' \in C \\ 0 & \text{if } l = \epsilon \text{ xor } l' = \epsilon \\ \infty & \text{if } l = l' = \epsilon \end{cases} . \tag{36}$$

Note that in [29], $y_{dd'} = 1$ indicates a join. In our NL-LMP, $y_{dd'} = 1$ indicates a cut.

B.2. Further Results

Quantitative results for each body joint are shown in Tab. 4. Qualitative results for the MPII Human Pose dataset are shown in Fig. 5.

$\lvert V \rvert$	Alg.	Head	Sho	Elb	Wri	Hip	Knee	Ank	AP
150	[15]	84.9	79.2	66.4	52.3	65.5	59.2	51.2	65.5
150	KLj/r	**87.1**	80.0	**66.8**	**53.6**	66.1	60.0	51.8	66.5
150	KLj∗r	86.8	**80.2**	67.5	53.5	**66.3**	**60.3**	**51.9**	**66.6**
420	KLj/r	**90.2**	**85.2**	71.5	59.5	**71.3**	**63.1**	53.1	**70.6**
420	KLj∗r	89.8	**85.2**	**71.8**	**59.6**	71.1	63.0	**53.5**	**70.6**

Table 4: Comparison of B&C [15], KLj/r and KLj∗r in an application to the task of articulated human body pose estimation.

C. Instance-Separating Semantic Segmentation

We tackle the problem of instance-separating semantic segmentation by adapting the approach of Uhrig et al. [37]. They propose three complementary representations, which are learned jointly by a fully convolutional network (FCN) [24], that facilitate the problem of separating individual objects: Semantics, depth, and directions towards object centers. To extract object instances, a template matching approach was proposed, followed by a proposal fusion.

Instead of template matching and clustering, we rely on a generic graphical formulation of the problem using only the three predicted output scores from the network of Uhrig et al. [37], together with a suitable formulation of unary c and pairwise terms $c^{\sim}$ and $c^{\not\sim}$. As there might be up to two million nodes for a direct mapping of pixel scores to the graph, we report performance on different down-sampled versions to reduce overall computation time and reduce the impact of noise in high resolutions. Results on KITTI were achieved on half of the input resolution, for Cityscapes we down-sample the FCN scores by a factor of eight before the graph optimization.

C.1. Cut Costs Details

To define cut costs between connected pixels in the graph, we use an equally weighted sum of the three following components:

The probability of fusing two pixels v and w of different **semantic classes** is $1 - p(\lambda(v) = a, \lambda(w) = b)$, the probability of confusing label class a and b, which was computed from the training set.

To incorporate the depth and center direction channels, we neither use scores nor argmax predictions directly. Instead, we weight the predicted softmax scores for all non-background classes with their corresponding class to recover a continuous center direction and depth map. As objects at different **depth values** should be separated, we generate higher cut probabilities for those pixels. From training data, we found the probability of splitting two neighboring pixels to be one when the predicted depth values differ by more than 1.6 units.

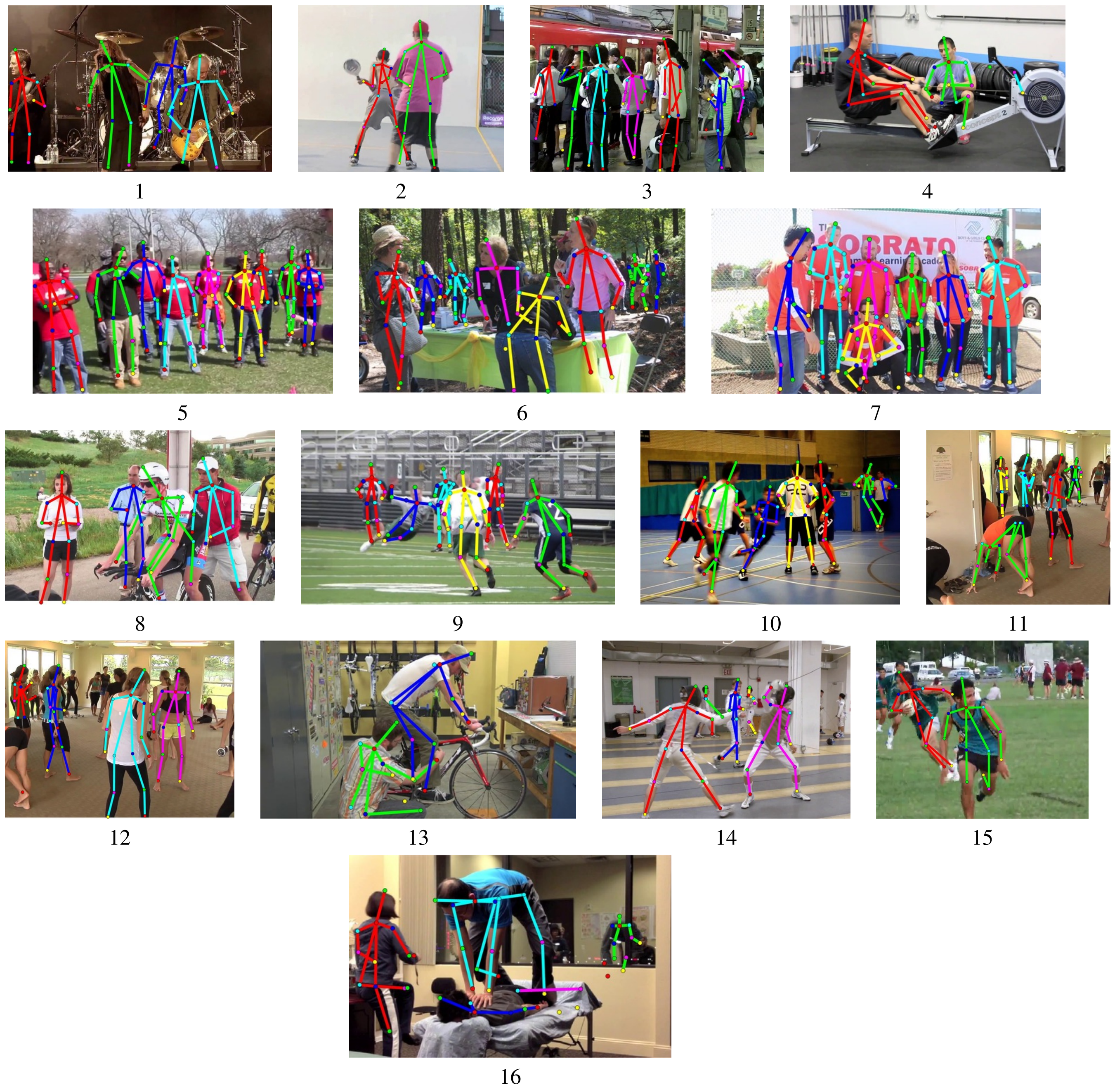


Figure 5: Pose estimation results on the MPII Human Pose dataset.

If **center directions** have opposite orientations, there should be a high probability for splitting the two pixels. However, opposite directions also appear at the center of an object. Therefore, we define the cut probability as the minimum of an angular inconsistency, which punishes two pixels that point at different directions, as well as a center inconsistency, which punishes if two pixels do not point at each other, *c.f*. Fig. 6. This induces high cut probabilities at the borders of objects, as directions of pixels should have opposite center direction. The probability of splitting two neighbors due to direction inconsistency was found to be one at 90 degrees.

C.2. Dataset Specifics

For the KITTI dataset [6, 13], the only pixel-level annotated object class is *car*. For Cityscapes [9] however, there

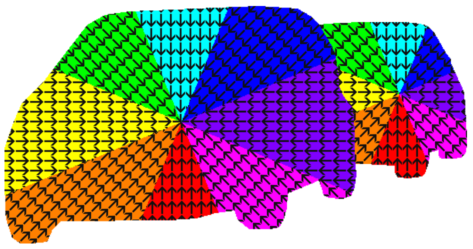


Figure 6: Instance center directions with color coding from [37]. Near object centers, directions point towards each other (center consistency). Within colored regions, directions have similar angles (angular consistency). Along object borders, directions are inconsistent in both ways.

are 8 different object classes (*person, rider, car, truck, bus, train, motorcycle, bicycle*), together with 11 background classes versus 1 background class for KITTI. We found that the network model used by Uhrig et al. [37] performs close to optimum for semantic labeling on KITTI data, however has some flaws on Cityscapes.

Therefore we chose a more sophisticated network structure, which performs much better on the many different classes on Cityscapes. We use a ResNet [14] with dilated convolutions [7] for cut costs c, namely the unary terms consisting of scores for the problem of semantic labeling, which was trained independently on the Cityscapes dataset [9].

To obtain the unaries for Cityscapes, we use a slightly modified ResNet-50 network. We introduce dilated convolutions in the conv4_x and conv5_x layers to increase the output resolution from $^1/_{32}$ to $^1/_8$ of the input resolution. We then remove the final average pooling layer and for classification use a convolutional layer with 5×5 dilated kernels with a dilation size of 12. This is identical to the best performing basic ResNet-50 variant reported in ([39], Table 1).

Due to GPU memory constraints, we train with $512px \times 768px$ crops randomly sampled from the full-resolution training set images. We apply minimal data augmentation, i.e. random horizontal flips, and train with a batch size of 5. We train the network for 60000 iterations using the Adam solver with an initial learning rate of 0.000025, weight decay of 0.0005 and momentum of 0.9. We use the "poly" learning rate policy to gradually reduce the learning rate during training with the power parameter set to 0.9, which as reported in both [23] and [5] yields better results than the commonly used "step" reduction policy.

At test-time we apply the network to overlapping $1024px \times 768px$ crops of the full-resolution test set images and stitch the results to obtain the final predictions.

For KITTI however, we stick with the original semantic scores. The only adaptation for our definition of the semantic cut costs c is an additional weighting of the semantic scores: As depth and center directions are only estimated for objects, all three channels contain knowledge of the objectness of a certain pixel. We therefore use the semantic scores weighted by the depth and direction scores for objects as unaries. This increases robustness of the semantics as all three channels must agree to achieve high scores.

Algorithm	Dataset	AP	$AP^{50\%}$
Ours KLj/r (raw)	KITTI val	43.0	72.5
Ours KLj∗r (raw)	KITTI val	43.5	72.6
Ours KLj/r (fused)	KITTI val	**50.5**	**82.9**
Ours KLj∗r (fused)	KITTI val	50.3	82.4
Pixel Encoding [37]	KITTI test	41.6	69.1
Ours KLj∗r (fused)	KITTI test	**43.6**	**71.4**

Table 5: Comparison of algorithms for instance segmentation on the KITTI [6] datasets using the mean average precision metrics introduced in [9].

C.3. Post Processing

Using the unary and pairwise terms defined above, we solve the graph for labels and components with our proposed algorithms KLj/r and KLj∗r. As the center direction representation inherently cannot handle cases of full occlusions, e.g. if a bicycle is split into two connected components by a pedestrian in front of it, we apply a similar component fusion technique as proposed in [37]: We accumulate direction predictions within each component and fuse it with another suitable component when direction predictions are clearly overshooting into a certain direction. We compare performance of the raw graph output as well as the fused instances in Tab. 5 (top).

C.4. Detailed Results

As there are different metrics used by related approaches, we report performance on the Cityscapes [9] and KITTI [6, 13] dataset using both proposed metrics. The instance score required for the evaluation on Cityscapes was chosen as the size of the instance in pixels multiplied by its mean depth - this score achieved slightly better results compared to a constant score.

For KITTI, we outperform all existing approaches using the Cityscapes metric (without adapting the semantic scores of Uhrig et al. [37]), which averages precision and recall performance for multiple overlaps, *c.f.* Tab. 5 (bottom). We evaluate the performance using KLj/r or KLj∗r and raw graph output (raw) or the post-processed results using above described fusion (fused) in Tab. 5 (top). Using the KITTI metrics, we perform among the best results while having a slight preference of Recall over Precision, *c.f.* Tab. 6.

For Cityscapes, we report evaluation metrics using both the raw scores of Uhrig et al. [37] as well as our final proposed model using the semantic scores of a ResNet [14]

Alg.	IoU	AvgFP	AvgFN	InsPr	InsRe	InsF1
[43]	77.4	0.479	0.840	48.9	43.8	46.2
[42]	77.0	0.375	1.139	65.3	50.0	56.6
[37]	84.1	0.201	0.159	**86.3**	74.1	**79.7**
[31]	**87.4**	**0.118**	0.278	-	-	-
Ours	83.9	0.555	**0.111**	69.2	**76.5**	72.7

Table 6: Comparison of algorithms for instance segmentation on the KITTI test dataset [6] using metrics proposed in [42]. Ours describes the performance of our KLj∗r variant.

	person	rider	car	truck	bus	train	motorcycle	bicycle
[9]	5.6	3.9	26.0	13.8	**26.3**	15.8	8.6	3.1
[37]	**31.8**	**33.8**	37.8	7.6	12.0	8.5	20.5	**17.2**
Ours	18.4	29.5	**38.3**	**16.1**	21.5	**24.5**	**21.4**	16.0

Table 7: Comparison of performance on Cityscapes test using the mean average precision metric $\text{AP}^{50\%}$ [9]. Ours describes the performance of our KLj∗r (ResNet) variant.

together with the center direction and depth scores of Uhrig et al. [37], *c.f.* Tab. 8 (top). Using our adapted ResNet version, we outperform the currently published state-of-the art, *c.f.* Tab. 8 (bottom). Note that we report significantly better performance for the large vehicle classes truck, bus, and trains despite starting from the same FCN output, *c.f.* Tab. 7. This comes from incorporating confusion probabilities between unreliable classes as well as optimizing jointly for semantics and instances.

C.5. Qualitative Results

See Fig. 7 for some qualitative results for our instance-separating semantic segmentation on the Cityscapes validation dataset [9]. It can be seen that we perform equally well for large and small objects, we only tend to fuse pedestrians too often, which explains the worse performance on pedestrians - *c.f.* the mother with her child on the right in the last row of Fig. 7. Also, the impact of the proposed post-processing based on the fusion algorithm proposed by Uhrig et al. [37] can be seen clearly: Due to noisy predictions, the raw graph output is often highly over-segmented. However, after applying the fusion step, most objects are correctly fused.

C.6. Outlook

The reason for the varying performance for objects of different semantic classes certainly comes from their very different typical forms, which we do not incorporate in our general approach. Uhrig et al. [37] use different aspect

Algorithm	Dataset	AP	$\text{AP}^{50\%}$
Pixel Encoding [37]	CS val	9.9	22.5
Ours ([37] scores)	CS val	9.4	22.1
Ours KLj/r (ResNet)	CS val	11.3	**26.8**
Ours KLj∗r (ResNet)	CS val	**11.4**	26.1
MCG+R-CNN [9]	CS test	4.6	12.9
Pixel Encoding [37]	CS test	8.9	21.1
Ours KLj∗r (ResNet)	CS test	**9.8**	**23.2**

Table 8: Comparison of algorithms for instance segmentation on the Cityscapes (CS) dataset [9] using the mean average precision metrics introduced in [9].

ratios for their sliding object templates to cope for these changes. In future work, we would like to combine multiple graphs for different semantic classes to boost individual class performance. Also, the predicted FCN representation and scores will be adjusted for better suiting the requirements of our graph optimization.

D. Multiple Object Tracking

D.1. Problem Statement

Tang et al. [35] introduce a binary linear program w.r.t. a graph $G = (V, E)$ whose nodes are candidate detections of humans visible in an image. Every feasible solution is a pair (x, y) with $x \in \{0, 1\}^V$ and $y \in \{0, 1\}^E$, constrained such that

$$\forall \{v, w\} \in E : \quad y_{vw} \leq x_v \tag{37}$$

$$y_{vw} \leq x_w \tag{38}$$

$$\forall C \in \text{cycles}(G)\ \forall e \in C : \quad 1 - y_e \leq \sum_{f \in C \setminus \{e\}} (1 - y_f) \tag{39}$$

The objective function has the form below with coefficients α and β.

$$\sum_{v \in V} \alpha_v x_v + \sum_{e \in E} \beta_e y_e \tag{40}$$

We identify the solutions of this problem with the solutions of the NL-LMP w.r.t. the graphs $G' = G$, the label set $L = \{\epsilon, 1\}$ and the costs $c^{\not\sim} = 0$ and

$$c_{vl} := \begin{cases} \alpha_v & \text{if } l = 1 \\ 0 & \text{if } l = \epsilon \end{cases} \tag{41}$$

$$c^{\sim}_{vw,ll'} := \begin{cases} \beta_{vw} & \text{if } l = 1 \wedge l' = 1 \\ 0 & \text{if } l = 1 \text{ xor } l' = 1 \\ \infty & \text{if } l = l' = \epsilon \end{cases} . \tag{42}$$

Note that in [35], $y_{dd'} = 1$ indicates a join. In our NL-LMP, $y_{dd'} = 1$ indicates a cut.

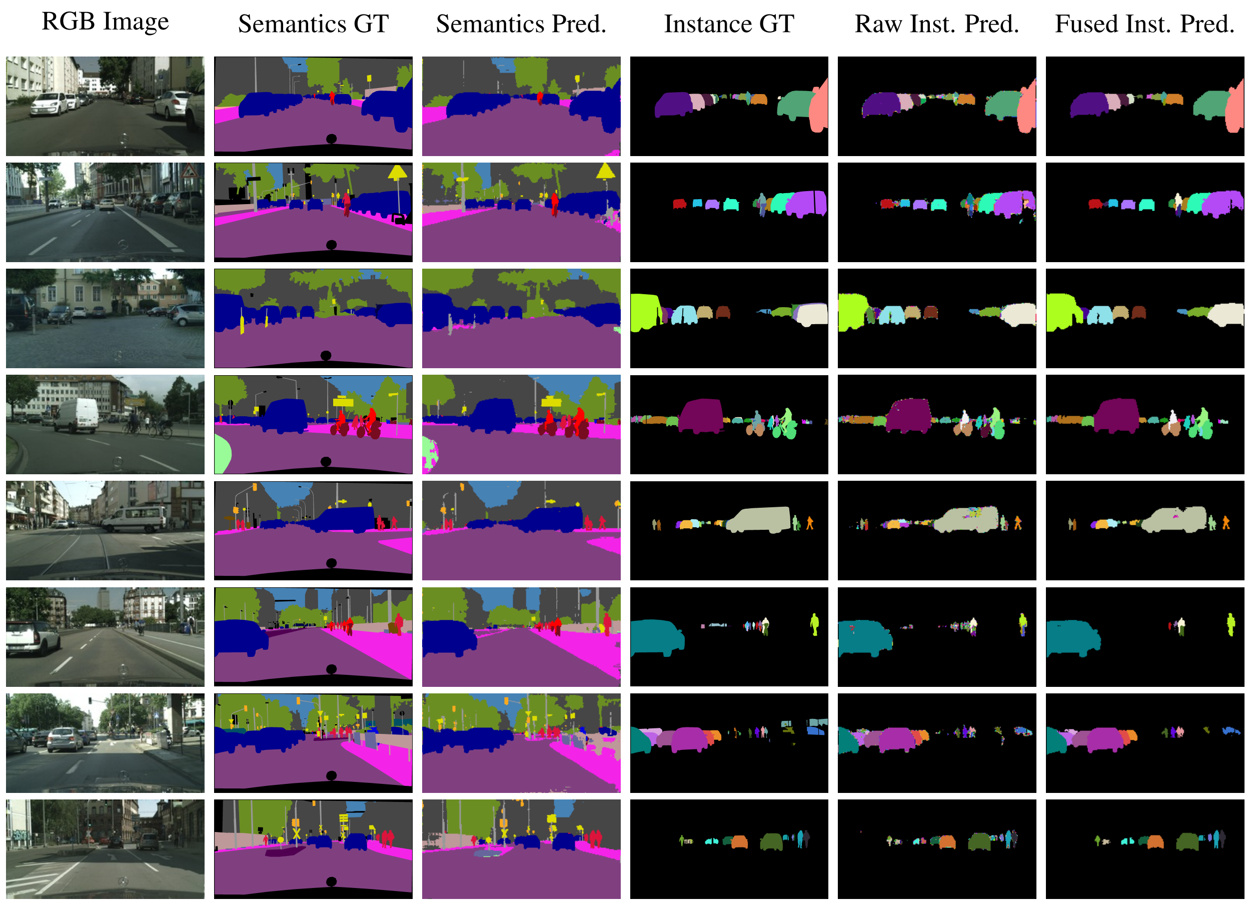


Figure 7: Visualization of our predictions on the Cityscapes validation dataset [9], where we can compare with corresponding ground truth (GT) and show respective RGB images.

D.2. Further Results

A complete evaluation of our experimental results in terms of the Multiple Object Tracking Challenge 2016 can be found at http://motchallenge.net/tracker/NLLMPa.