Generalizations of the Multicut Problem for Computer Vision

A dissertation submitted towards the degree of
Doctor of Engineering (Dr.-Ing.)
of the Faculty of Mathematics and Computer Science
of Saarland University

by

Evgeny Levinkov, M.Sc.

Saarbrücken
January 2019

Date of defense	8th of April, 2019
Dean of the faculty	Univ.-Prof. Dr. Sebastian Hack

Examination Committee

Chair	Prof. Dr. Markus Bläser
Reviewer, advisor	Prof. Dr. Björn Andres
Reviewer	Prof. Dr. Victor Lempitsky
Reviewer	Prof. Dr. Carsten Rother
Reviewer	Prof. Dr. Bernt Schiele
Academic assistant	Dr. Paul Swoboda

ABSTRACT

Graph decomposition has always been a very important concept in machine learning and computer vision. Many tasks like image and mesh segmentation, community detection in social networks, as well as object tracking and human pose estimation can be formulated as a graph decomposition problem. The *multicut* problem in particular is a popular model to optimize for a decomposition of a given graph. Its main advantage is that no prior knowledge about the number of components or their sizes is required. However, it has several limitations, which we address in this thesis:

Firstly, the multicut problem allows to specify only cost or reward for putting two direct neighbours into distinct components. This limits the expressibility of the cost function. We introduce special edges into the graph that allow to define cost or reward for putting any two vertices into distinct components, while preserving the original set of feasible solutions. We show that this considerably improves the quality of image and mesh segmentations.

Second, multicut is notorious to be NP-hard for general graphs, that limits its applications to small super-pixel graphs. We define and implement two primal feasible heuristics to solve the problem. They do not provide any guarantees on the runtime or quality of solutions, but in practice show good convergence behaviour. We perform an extensive comparison on multiple graphs of different sizes and properties.

Third, we extend the multicut framework by introducing node labels, so that we can jointly optimize for graph decomposition and nodes classification by means of exactly the same optimization algorithm, thus eliminating the need to hand-tune optimizers for a particular task. To prove its universality we applied it to diverse computer vision tasks, including human pose estimation, multiple object tracking, and instance-aware semantic segmentation. We show that we can improve the results over the prior art using exactly the same data as in the original works.

Finally, we use employ multicuts in two applications: 1) a client-server tool for interactive video segmentation: After the pre-processing of the video a user draws strokes on several frames and a time-coherent segmentation of the entire video is performed on-the-fly. 2) we formulate a method for simultaneous segmentation and tracking of living cells in microscopy data. This task is challenging as cells split and our algorithm accounts for this, creating parental hierarchies.

We also present results on multiple model fitting. We find models in data heavily corrupted by noise by finding components defining these models using higher order multicuts. We introduce an interesting extension that allows our optimization to pick better hyperparameters for each discovered model.

In summary, this thesis extends the multicut problem in different directions, proposes algorithms for optimization, and applies it to novel data and settings.

ZUSAMMENFASSUNG

Die Zerlegung von Graphen ist ein sehr wichtiges Konzept im maschinellen Lernen und maschinellen Sehen. Viele Aufgaben wie Bild- und Gittersegmentierung, Kommunitätserkennung in sozialen Netzwerken, sowie Objektverfolgung und Schätzung von menschlichen Posen können als Graphzerlegungsproblem formuliert werden. Der Mehrfachschnitt-Ansatz ist ein populäres Mittel um über die Zerlegungen eines gegebenen Graphen zu optimieren. Sein größter Vorteil ist, dass kein Vorwissen über die Anzahl an Komponenten und deren Größen benötigt wird. Dennoch hat er mehrere ernsthafte Limitierungen, welche wir in dieser Arbeit behandeln:

Erstens erlaubt der klassische Mehrfachschnitt nur die Spezifikation von Kosten oder Belohnungen für die Trennung von zwei Nachbarn in verschiedene Komponenten. Dies schränkt die Ausdrucksfähigkeit der Kostenfunktion ein und führt zu suboptimalen Ergebnissen. Wir fügen dem Graphen spezielle Kanten hinzu, welche es erlauben, Kosten oder Belohnungen für die Trennung von beliebigen Paaren von Knoten in verschiedene Komponenten zu definieren, ohne die Menge an zulässigen Lösungen zu verändern. Wir zeigen, dass dies die Qualität von Bild- und Gittersegmentierungen deutlich verbessert.

Zweitens ist das Mehrfachschnittproblem berüchtigt dafür NP-schwer für allgemeine Graphen zu sein, was die Anwendungen auf kleine superpixel-basierte Graphen einschränkt. Wir definieren und implementieren zwei primal-zulässige Heuristiken um das Problem zu lösen. Diese geben keine Garantien bezüglich der Laufzeit oder der Qualität der Lösungen, zeigen in der Praxis jedoch gutes Konvergenzverhalten. Wir führen einen ausführlichen Vergleich auf vielen Graphen verschiedener Größen und Eigenschaften durch.

Drittens erweitern wir den Mehrfachschnitt-Ansatz um Knoten-Kennzeichnungen, sodass wir gemeinsam über Zerlegungen und Knoten-Klassifikationen mit dem gleichen Optimierungs-Algorithmus optimieren können. Dadurch wird der Bedarf der Feinabstimmung einzelner aufgabenspezifischer Löser aus dem Weg geräumt. Um die Allgemeingültigkeit dieses Ansatzes zu überprüfen, haben wir ihn auf verschiedenen Aufgaben des maschinellen Sehens, einschließlich menschliche Posenschätzung, Mehrobjektverfolgung und instanz-bewusste semantische Segmentierung, angewandt. Wir zeigen, dass wir Resultate von vorherigen Arbeiten mit exakt den gleichen Daten verbessern können.

Abschließend benutzen wir Mehrfachschnitte in zwei Anwendungen: 1) Ein Nutzer-Server-Werkzeug für interaktive Video Segmentierung: Nach der Vorbearbeitung eines Videos zeichnet der Nutzer Striche auf mehrere Einzelbilder und eine zeit-kohärente Segmentierung des gesamten Videos wird in Echtzeit berechnet. 2) Wir formulieren eine Methode für simultane Segmentierung und Verfolgung von lebenden Zellen in Mikroskopie-Aufnahmen. Diese Aufgabe ist anspruchsvoll, da Zellen sich aufteilen und unser Algorithmus dies in der Erstellung von Eltern-

Hierarchien mitberücksichtigen muss.

Wir präsentieren außerdem Resultate zur Mehrmodellanpassung. Wir berechnen Modelle in stark verrauschten Daten indem wir mithilfe von Mehrfachschnitten höherer Ordnung Komponenten finden, die diesen Modellen entsprechen. Wir führen eine interessante Erweiterung ein, die es unserer Optimierung erlaubt, bessere Hyperparameter für jedes entdeckte Modell auszuwählen.

Zusammenfassend erweitert diese Arbeit den Mehrfachschnitt-Ansatz in unterschiedlichen Richtungen, schlägt Algorithmen zur Inferenz in den resultierenden Modellen vor und wendet ihn auf neuartigen Daten und Umgebungen an.

CONTENTS

1 SYNOPSIS

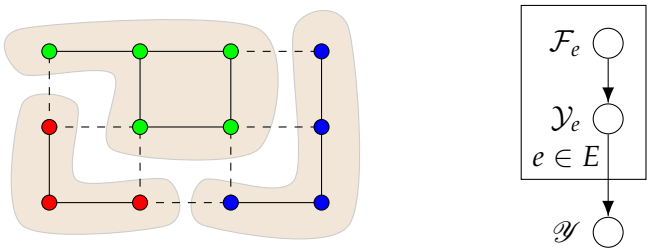
Graphs, as a data structure, are a convenient and abstract way of expressing problems consisting of entities and relationships between them. Being a rather abstract structure, graphs have found applications in a variety of subfields of mathematics and computer science, including computer vision. The latter was facilitated by the fact that a large pool of efficient graph algorithms has been developed, that can be applied straightforwardly to multiple applications. This is important in practice, because it allows users to solve the tasks they have at hand without fine-tuning or even modifying the algorithms. Some examples of important and widely used graph problems are: shortest path, graph matching, and graph decomposition. In this thesis, we study the latter problem in detail and apply it to computer vision tasks.

Graph decomposition aims at splitting the node set of a given graph into connected components, according to a certain objective; an example is given in Fig. 1.1 (a). Turning to computer vision, probably, the first example of a graph that comes to mind is a regular grid graph of an image. Another example is a graph of vertices or faces of a 3D mesh. In both cases we can perform image or mesh segmentation by means of graph decomposition.

Other important examples include:

- multiple object tracking, when a graph has putative detections as vertices connected inside and across frames of a video
- multiple person pose estimation, when a graph contains possible detections of body joints that are connected in an image
- community detection in social network graphs, that have people as vertices and edges represent mutual trust or distrust between users
- segmentation and tracking of living cells in a sequence of microscopy images
- multiple geometric model fitting

Being such an important concept, graph decompositions (sometimes also referred to as *clusterings*) have gained a lot of attention over the last decades in computer vision and, more generally, machine learning communities. Notable approaches include graph spectral clustering, k-means, (multiway) min-cut and others. What unites them is the necessary a priori knowledge of the number of clusters (or assumptions on their sizes) and non-negative edge costs. In this thesis we focus on a conceptually different framework that we introduce further.



(a) **(b)**

Figure 1.1: **(a)** An example of a graph decomposition and its encodings. In a vertex labeling switching, for example, green and blue labels will encode the same decomposition. Dashed lines, in turn, constitute a multicut of the graph and uniquely define its decomposition. **(b)** Depicted is a Bayesian Network defining a set of probability measures on multicuts (Andres *et al.*, 2011).

1.1 MULTICUT PROBLEM

One way of representing a decomposition of a graph is to assign to each vertex an identifier of a component it belongs to, i.e., a *vertex labeling* (Fig. 1.1 (a)). The drawback of this encoding is that a permutation of components' identifiers will result in a different vertex labeling, while encoding the same decomposition. This ambiguity hampers the optimization, because the search space of the feasible solutions can be factorially large and one needs to know in advance (or somehow estimate) the required number of such identifiers.

A different encoding, that uniquely represents every decomposition of a graph, is called a *multicut* of a graph (dashed lines in Fig. 1.1 (a)).

Definition 1 A multicut M of a graph $G = (V, E)$ is a subset of edges E such that no cycle of the graph intersects with this subset precisely once

$$\text{multicuts}(G) = \{M \subseteq E \mid \forall C \in \text{cycles}(G) : |M \cap C| \neq 1\} \tag{1.1}$$

If we assign either 0 or 1 to each edge such that edges labeled 1 straddle distinct components, then the set of all multicuts Y_G of a graph can be expressed in terms of linear, so called *cycle*, inequalities as follows:

$$Y_G = \left\{ y : E \mapsto \{0,1\} \mid \forall C \in \text{cycles}(G), \forall e \in C : y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \right\}. \tag{1.2}$$

Chopra and Rao (1993) showed that it is enough to consider only chordless cycles.

Conventionally, 1 is used for the edges in the multicut, however, it is of course possible to switch the meaning of 0 and 1. Indeed, we shall do so in Chapter 7, with a straightforward change of (1.2).

Definition 2 Given a graph G and a cost function $c : E \mapsto \mathbb{R}$ that assigns a cost or reward for putting two vertices into different components, the following integer linear program is called the *minimum cost multicut problem*

$$\min_{y \in Y_G} \sum_{e \in E} c_e y_e \tag{1.3}$$

It is important to note that unless c contains negative values, zero is a trivial solution, i.e., all vertices are put into the same component. This fact shows one of the key difference to other graph decomposition approaches, where costs are required to be non-negative. The drawback is that (1.3) is NP-hard to solve (Bansal *et al.*, 2004).

Another distinctive feature of (1.3) is the fact that there are no further constraints that would specify the number of components, expected sizes, etc. Instead, the optimal number of components is deduced automatically from the solution. This is especially useful in settings where the number of components is not known a priori and to be estimated from data.

Probabilistic model Multicuts enjoy a probabilistic model. Andres *et al.* (2011) define, with respect to a graph $G = (V, E)$ and with respect to the Bayesian Network depicted in Fig. 1.1 (b), a measure of the conditional probability of a $y \in \{0, 1\}^E$, given the feasible set Y_G of the characteristic functions of all multicuts of G and given, for every edge $e \in E$, a vector $f_e \in \mathbb{R}^d$ of $d \in \mathbb{N}$ edge features. Specifically,

$$p_{\mathcal{Y}|\mathcal{F},\mathscr{Y}} \propto p_{\mathscr{Y}|\mathcal{Y}} \cdot \prod_{e \in E} p_{\mathcal{Y}_e|\mathcal{F}_e} = p_{\mathscr{Y}|\mathcal{Y}} \cdot \prod_{e \in E} p_{\mathcal{Y}_e|\mathcal{F}_e}(1, f_e)^{y_e} \, p_{\mathcal{Y}_e|\mathcal{F}_e}(0, f_e)^{1-y_e} \tag{1.4}$$

$$\text{with} \qquad p_{\mathscr{Y}|\mathcal{Y}}(Y_G, y) \propto \begin{cases} 1 & \text{if } y \in Y_G \\ 0 & \text{otherwise} \end{cases} . \tag{1.5}$$

Now, one can perform a classical maximum a posteriori probability (MAP) inference (Andres *et al.*, 2011; Kappes *et al.*, 2011) to find the most probable multicut:

$$\arg\max_{y \in Y_G} \prod_{e \in E} p_{\mathcal{Y}_e|\mathcal{F}_e}(1, f_e)^{y_e} \, p_{\mathcal{Y}_e|\mathcal{F}_e}(0, f_e)^{1-y_e} = \tag{1.6}$$

$$= \arg\max_{y \in Y_G} \sum_{e \in E} y_e \, \log p_{\mathcal{Y}_e|\mathcal{F}_e}(1, f_e) + (1 - y_e) \log p_{\mathcal{Y}_e|\mathcal{F}_e}(0, f_e) = \tag{1.7}$$

$$= \arg\max_{y \in Y_G} \sum_{e \in E} y_e \log \frac{p_{\mathcal{Y}_e|\mathcal{F}_e}(1, f_e)}{p_{\mathcal{Y}_e|\mathcal{F}_e}(0, f_e)} + |E| \, \log p_{\mathcal{Y}_e|\mathcal{F}_e}(0, f_e) = \tag{1.8}$$

$$= \arg\min_{y \in Y_G} \sum_{e \in E} y_e \log \frac{p_{\mathcal{Y}_e|\mathcal{F}_e}(0, f_e)}{p_{\mathcal{Y}_e|\mathcal{F}_e}(1, f_e)} . \tag{1.9}$$

Hence, if one sets the edge costs $c_e = \log \frac{p_{\mathcal{Y}_e|\mathcal{F}_e}(0,f_e)}{p_{\mathcal{Y}_e|\mathcal{F}_e}(1,f_e)}$ in (1.3), then minimizers of (1.3) and (1.9) coincide. This facilitates interpretation of solutions of (1.3) as MAP estimates.

1.2 RELATED WORK

Theoretical analysis Grötschel and Wakabayashi (1989) cast a complete graph clustering problem as clique partitioning problem, which can be regarded as a special case of the multicut problem for a complete graph. They give an integer linear program for it and devise a cutting plane algorithm to solve it. In a follow up work, Grötschel and Wakabayashi (1990) provide theoretical description of the induced polytope, defining its facets, and use these results to further improve their cutting plane algorithm, so that it can be applied to large real-world instances. Deza *et al.* (1991) and Deza *et al.* (1992) proceeded with studying the geometry of the multicut polytope of a complete graph. Bansal *et al.* (2004) stated a binary linear program known as *correlation clustering* problem, that was shown by Demaine *et al.* (2006) to be analogous to the multicut problem for general weighted graphs.

The seminal work by Chopra and Rao (1993) formulates the multicut problem also for general graphs and studies their facets. The most important observation, that has practical implications, is that only chordless cycles in (1.2) are facet defining. This dramatically decreases the number of inequalities that have to be separated in each node of branch-and-cut solving algorithms. In Chapter 2 we introduce an extension called *lifted multicut* and show its practical importance. Horňáková *et al.* (2017) study the lifted multicut polytope and show that it is fully-dimensional and its facets are highly non-trivial: they describe several classes of facet-defining inequalities. Lange and Andres (2017) analyse facets of lifted multicut polytope defined w.r.t. paths and trees.

Classical multicuts allow only for pairwise potentials operating on pair of nodes, but in certain applications higher order terms are also required. A straightforward generalization to higher order potentials that operate on sets of mutually connected nodes is called a higher order multicut problem, as has been proposed by Kappes *et al.* (2016) and Kim *et al.* (2011). Later Keuper (2017) proposed a higher order multicut model defining not only costs but also graph connectivity in terms of edges with arbitrary order.

We contribute to the theoretical development by introducing node labels into the multicut objective. This allows us to jointly optimize for node labeling (node classification) and graph decomposition, that has several important applications.

Solving algorithms One principled approach to solving instances of the multicut problem pursued, e.g., in (Kappes *et al.*, 2011; Kim *et al.*, 2011, 2014; Yarkony *et al.*, 2012; Yarkony, 2014), is by solving the linear program (LP) obtained by an outer relaxation (Yarkony *et al.*, 2012) of the multicut polytope. In conjunction with suitable rounding, this yields an $\mathcal{O}(\log n)$ approximation that was established independently by Charikar *et al.* (2005) and Demaine *et al.* (2006). Lately, Fukunaga (2018) has come up with an LP-based pivoting algorithm that results in an $\mathcal{O}(k \log n)$ approximation for higher-order multicut problem, where k is the order of the problem. Although solving an LP relaxation is normally quick, separating violated inequalities in a cutting-plane algorithm takes most of the computation time.

In an attempt to avoid this separation problem, Andres *et al.* (2012b) and Kappes *et al.* (2015a) separate only integral points by cycle inequalities (by breadth first search) and resort to general classes of cuts for fractional points. For some instances of the problem, exact solutions are found in less than the time required to solve the canonical LP relaxation. For planar graphs, the minimum cost multicut problem remains NP-hard (Bachrach *et al.*, 2013; Voice *et al.*, 2012), but admits a polynomial time approximation scheme (PTAS) (Klein *et al.*, 2015). Due to a doubly exponential constant factor, this PTAS has mostly been of theoretical interest.

Partial optimality results for the multicut problem, established in (Alush and Goldberger, 2012, 2015), imply for some instances, a decomposition into independent sub-problems that can in principle be exploited by any algorithm. Swoboda and Andres (2017) proposed a message passing algorithms to compute lower bounds and re-parametrization of the multicut problem, but applications to large-scale problems still pose computational challenges. Recent work by Lange *et al.* (2018) further establish partial optimality conditions that allow to solve problems for series-parallel graphs to optimality, in linear time. They also compute faster lower bounds, than by a canonical LP relaxation, and a re-weighting, that allows local search algorithms to obtain better solutions.

A separate line of research is concentrated on devising local search algorithms for the multicut problem, that relate to the pioneering work by Kernighan and Lin (1970) that propose an efficient algorithm for set partitioning problem; the latter being essentially a multicut defined w.r.t. a complete graph. The first algorithm that allowed applications of multicuts to large graphs based on α-expansion was proposed by Bagon and Galun (2011). Beier *et al.* (2014) proposed a method that iteratively updates boundary between neighbouring components by solving MAX-CUT problem. Beier *et al.* (2015) proposed a scheme that contracts a graph, solves the multicut problem on a smaller graph and then expands the solution to the original graph. In principle, any solver may benefit from this scheme.

We contribute by defining two efficient local search algorithms, that do not provide any guarantees on the quality of the solution or runtime. In practice, they converge to good solution in affordable time, that allowed us to apply multicut-based methods to a variety of novel and large-scale applications.

Applications Image segmentation (Andres *et al.*, 2011; Yarkony *et al.*, 2012) is the most straightforward application of multicuts in computer vision. However, due to runtime issues previously it was possible to solve it only on graphs defined w.r.t. super pixels. In this thesis, we applied it directly to pixel grid graphs of images. Multicuts have also been shown handy in applications to 3d volume segmentation (Andres *et al.*, 2012b) and motion segmentation (Keuper *et al.*, 2015; Keuper, 2017).

Apart from classical problems, multicuts have been applied to less obviously suitable tasks as well. Insafutdinov *et al.* (2016) formulate multiple person pose estimation as multicut problem with simultaneous node labeling. Each node represents a possible body joint detection with scores for each possible body join type and the

task is to classify the body joints and split them between multiple people. Tang *et al.* (2015) formulate multiple object tracking as multicut problem, where each node is a possible object detection and nodes are connected both inside and across frames of a video. Interestingly, both these tasks can be expressed in the same node labeling multicut framework that we propose in this thesis.

Additionally, in this thesis we apply multicuts to communities detection in large social networks and to multiple model fitting for the first time, to the best of our knowledge.

1.3 CONTRIBUTIONS

Our contributions mainly stem from limitations of the multicut problem (Def. 2. One such limitation is the fact that a cost or reward can be assigned only to the nodes that are direct neighbours in the graph. However, in practice, it is normally beneficial to let each node know more about its local neighbourhood. We propose to mitigate this limitation by introducing additional edges between non-direct neighbours, that do not change the connectivity. This allows to create more plausible segmentations.

Another limitation is the fact that multicuts can optimize only a decomposition. However, in many applications it is also necessary to perform node labeling, i.e., for each node select exactly one class label. We propose a novel formulation that jointly optimizes both the node labeling and graph decomposition. We applied this formulation to three distinct computer vision tasks without any modifications.

One such important application is instance-aware semantic segmentation, that has attracted significant attention recently. We train a general object boundary detector and combine it with the output scores of a semantic segmentation network in a joint optimization framework. This allows us to reason simultaneously about the classes and objects in an image.

The third limitation rises from the fact the multicut-based approaches are in general NP-hard to solve. This limited their applicability to only relatively small graphs, e.g., super-pixel graphs of images. In this thesis, we propose two efficient local search algorithms (heuristics), that allow to obtain good solutions in affordable time. They do not provide any guarantees on the runtime or quality of the solutions, but our extensive experiments show that they can be very useful in practice.

Thanks to our efficient heuristics we were able to apply multicut, for the first time, to grid-graphs of images and come up with several other applications: First, we developed an online tool for interactive video segmentation. After pre-processing of a video (computing super-voxels and features) a user draws strokes on the video frames and a temporally coherent segmentation is computed on-the-fly.

Second, we apply higher-order multicuts (costs are defined on subsets of vertices of arbitrary cardinality) to the task of multiple model fitting. We fit elementary hypotheses into subsets of data points and compute their residuals under certain model assumption. We take then the probability of a point belonging to a model being inversely proportional to the residuals. Then we cluster data points so that

the overall probability for all the models is maximized. We also present results on multiple model fitting with hyperparameter selection. In real data all models might have a unique noise level, and most conventional algorithms will fit a model with and average hyperparameter. This might lead to under- and over-assignment of the data points. Our approach, that is based on node labeling higher order multicuts, is able to choose a model hyperparameter for each model individually.

Finally, we develop a model for simultaneous segmentation and tracking of living cells. Challenges in this case include low resolution microscopy images, significant level of noise, low temporal resolution. Also, cells can divide and the algorithm should handle these events and not lose the association.

1.4 OUTLINE

Here, we summarize each chapter of the thesis. We also mention the respective publications, that some of the chapters are based on, and specify explicitly contributions of the authors.

Chapter 2: Lifted Multicut Problem One of the fundamental limitations of the classical multicut formulation is that the cost or reward can be assigned to only direct neighbours in a graph. In practice, for example, in case of image segmentation local estimates between neighbouring pixels are prone to noise. Here, we proposed an approach that adds additional cost edges (preserving the original feasible set of decompositions), whose values are computed automatically using geodesic probabilistic lifting. As solving the multicut problem is NP-hard, we developed two efficient heuristics. Our results show, that using the same data we can improve results considerably with a bit of extra computations.

The content of this chapter corresponds to the ICCV 2015 publication "Efficient Decomposition of Image and Mesh Graphs by Lifted Multicuts". Margret Keuper was the leading author and contributed probabilistic geodesic lifting and experiments on images. Evgeny Levinkov contributed two efficient heuristics and their comparison to other solvers. Nicolas Bonnel contributed experiments on meshes.

Chapter 3: Empirical Comparison of Local Search Algorithms We carried out an extensive comparison of our proposed heuristics, namely GAEC and KLj, and existing algorithms on data sets with varying size and properties. Where it was possible, we computed globally optimal solutions using branch-and-cut and lower bounds by solving the canonical LP relaxation. We also proposed a generalization of GAEC, called Greedy Fixation, that fixes variables not only to 0 (join), but to 1 (cut) as well.

Our results show, that although heuristics do not provide any guarantees on the quality of solutions, nor any bounds on the runtime, in practice, they converge to solutions that are not far from the optimal ones in terms of both objective value and metrics, but do so orders of magnitude faster, than the exact solver.

The content of this chapter corresponds to the GCPR 2017 publication "A Com-

parative Study of Local Search Algorithms for Correlation Clustering". Evgeny Levinkov was the leading author of this paper. Alexander Kirillov contributed by training a siamese CNN for the MNIST experiment.

Chapter 4: Interactive Multicut Video Segmentation Chapter 4 shows, that our multicut heuristics are fast and provide good quality solutions. In this chapter we applied them to the task of interactive video segmentation. The pipeline starts with preprocessing of the video into super-voxels and computing their features. Then, user can perform two kinds of operations: 1) draw continuous strokes that specify super-voxels that belong to the same object, 2) by clicking on distinct super-voxels user can specify that they belong to different objects. User may also split super-voxels to refine the segmentation. We learn a logistic regression on-the-fly and compute the multicut solution in real time.

The content of this chapter corresponds to the Pacific Graphics 2016 short paper publication "Interactive Multicut Video Segmentation". Evgeny Levinkov was the leading author of this paper. James Tompkin and Nicolas Bonnel contributed developing the client-server application and carried out the user study.

Chapter 5: Moral Lineage Tracing One of the core problems in developmental biology is lineage tracing, i.e., the problem of establishing parental relationship of cells as they live, move around, and divide. The task is complicated by the fact that it is also necessary to segment individual cells in each video frame and track them throughout the whole video sequence. We proposed a joint formulation for segmentation and tracking, that is based on the multicut problem, but more constrained. For example, the cells are constrained to never merge, so that morality is preserved.

To solve the resulting problem we offered a branch-and-cut algorithm together with the cutting-plane algorithm for the canonical LP relaxation. The solutions we obtained compare favourably to others. The runtime of our solver was prohibitive, but a follow up work by Rempfler *et al.* (2017) greatly improved on this issue.

The content of this chapter corresponds to the CVPR 2016 publication "Moral Lineage Tracing". Evgeny Levinkov was the shared leading author of this paper. Florian Jug contributed with experimental evaluation of the proposed method.

Chapter 6: Node Labeling Lifted Multicut Problem Multicuts allow to optimize only a decomposition of graph, however, several important applications require to also optimize for a node labeling as well. Here, we proposed a joint formulation, that allows to perform both tasks simultaneously. We applied this formulation to such important computer vision tasks as multi-person pose estimation, multi-person tracking, and instance segmentation. We give two heuristics that can be applied to the above mentioned tasks without any changes. Without additional learning we improved considerably over the base lines.

The content of this chapter corresponds to the CVPR 2017 publication "Joint Graph Decomposition & Node Labeling: Problem, Algorithms, Applications".

Evgeny Levinkov was the leading author of this paper. Jonas Uhrig contributed experiments on instance segmentation. Eldar Insafutdinov provided us with the data for multi-person pose estimation experiments. Siyu Tang provided us with the data for multi-person tracking.

Sec. 6.4.3 presents our results on instance-aware semantic segmentation and corresponds to the CVPR 2017 publication "InstanceCut: from Edges to Instances with MultiCut". Alexander Kirillov was the leading author of this paper. Evgeny Levinkov contributed the node labeling multicut formulation and inference code.

Additionally, our inference code has been used as well in the CVPR 2017 publication "ArtTrack: Articulated Multi-person Tracking in the Wild" (accepted as an oral presentation, 2.65% acceptance rate, Evgeny Levinkov is a co-author).

Chapter 7: Geometric Multiple Model Fitting with Model Selection Geometric multiple model fitting is the task of explaining data points by models under certain assumption, e.g., line fitting. We formulate it as a higher order multicut problem defined over cliques of data points, that allows us to group points, that are likely to belong to the same model, together.

Usually, model fitting algorithms fit models with all the hyperparameters fixed and require an inspection of results. By introducing node labels into the formulation we can work with a set of possible hyperparameters and let the optimization choose a better one for each model, i.e., perform model selection. We think that this is much closer to the real world data.

Chapter 8: Conclusions In this chapter we summarize the thesis and elaborate on existing limitations and discuss future directions of research. We also elaborate on possibility of end-to-end learning of decompositions.

Part I

Lifted Multicuts for Computer Vision

LIFTED MULTICUT PROBLEM 2

Contents

Formulations of the image decomposition problem (Arbeláez *et al.*, 2011) as a Multicut Problem (MP) w.r.t. a superpixel graph have received considerable attention. In contrast, instances of the MP w.r.t. a pixel grid graph have received little attention, firstly, because the MP is NP-hard and instances w.r.t. a pixel grid graph are hard to solve in practice, and, secondly, due to the lack of long-range terms in the objective function of the MP. We propose a generalization of the MP with long-range terms (LMP). We design and implement two efficient algorithms (primal feasible heuristics) for the MP and LMP which allow us to study instances of both problems w.r.t. the pixel grid graphs of the images in the BSDS-500 benchmark (Arbeláez *et al.*, 2011). The decompositions we obtain do not differ significantly from competitors, suggesting that the LMP provides a useful formulation of the image decomposition problem. To demonstrate the generality of the LMP, we apply it also to the mesh decomposition problem posed by the Princeton benchmark (Chen *et al.*, 2009), obtaining competitive decompositions.

2.1 INTRODUCTION

Formulations of the Image Decomposition Problem (Arbeláez *et al.*, 2011) as a Minimum Cost Multicut Problem (MP) (Chopra and Rao, 1993; Deza and Laurent, 1997) have received considerable attention (Alush and Goldberger, 2012; Andres *et al.*, 2011, 2012b, 2013; Bagon and Galun, 2011; Beier *et al.*, 2015, 2014; Kappes *et al.*, 2011, 2016, 2015b; Kim *et al.*, 2011, 2014; Nowozin and Jegelka, 2009; Yarkony *et al.*, 2012, 2015). Advantages of this formulation are in order: Firstly, the feasible solutions of the MP relate one-to-one to the decompositions of a graph. In particular, the number of components is not fixed in advance but is determined by the solution. Secondly, the MP, unlike balanced cut problems (Shi and Malik, 2000), does not favor one decomposition over another by definition. Thirdly, multicut algorithms are easy to use; they take as input a graph, *e.g.* the pixel grid graph of an image, and, for every edge, a real-valued cost (reward) of the incident nodes being in distinct components, *e.g.* $\log \frac{1-p_e}{p_e} + \log \frac{1-p^*}{p^*}$, for an estimated probability p_e of boundary (Arbeláez *et al.*, 2011) at the edge e, and a prior probability $p^* \in (0, 1)$ of cuts. The output is a 01-labeling of the edges that well-defines a decomposition of the graph by 0 indicating "join" and 1 indicating "cut".

One disadvantage is the NP-hardness of the MP (Bansal *et al.*, 2004; Demaine *et al.*, 2006). Despite significant progress in the design of efficient heuristics (Bagon and Galun, 2011; Beier *et al.*, 2015, 2014; Kernighan and Lin, 1970), instances of the MP for image segmentation have so far only been solved w.r.t. superpixel adjacency graphs and not w.r.t. pixel grid graphs, with the sole and notable exception of (Bagon and Galun, 2011). A second disadvantage results from the fact that a multicut makes explicit only for edges whether the incident nodes are in distinct components. It does not make explicit for pairs of nodes that are not neighbors whether these are in distinct components. Hence, the linear objective function of the MP w.r.t. a pixel grid graph cannot assign a cost specifically to all decompositions for which a pair of pixels that are not neighbors are in distinct components. This limitation, noted *e.g.* in (Andres *et al.*, 2013), hampers applications as it is often hard to estimate, for an image and a pair of neighboring pixels, whether the image is to be cut precisely between these pixels (only these estimates are used in the MP), and as it is sometimes easy to estimate for pixels at larger distance whether these are in distinct components (these estimates are not used in the MP).

An optimization problem whose feasible solutions relate one-to-one to the decompositions of a graph and whose objective function can assign, for any pair of nodes, a cost to all decompositions for which these nodes are in distinct components, although desirable, has not been proposed before.

Contribution We propose the Minimum Cost Lifted Multicut Problem (LMP), a generalization of the MP whose feasible solutions relate one-to-one to the decompositions of a graph and whose objective function can assign, for any pair of nodes, a real-valued cost (reward) to all decompositions for which these nodes are

in distinct components. We design and implement two efficient algorighms for both the MP and the LMP and evaluate both problem formulations in conjunction with both algorithms for the Image Decomposition Problem in terms of the BSDS-500 benchmark (Arbeláez *et al.*, 2011) and for the Mesh Decomposition Problem in terms of the Princeton Mesh Segmentation benchmark (Chen *et al.*, 2009).

2.2 RELATED WORK

The MP is known as Correlation Clustering in machine learning and theoretical computer science (Bansal *et al.*, 2004; Demaine *et al.*, 2006). For complete graphs, which are of special interest in machine learning, the well-known MP and the proposed LMP coincide.

A generalization of the MP by a higher-order objective function, called the Higher-Order Multicut Problem (HMP), was proposed in (Kim *et al.*, 2011) and is studied in detail in (Kappes *et al.*, 2016; Kim *et al.*, 2014). In principle, the HMP subsumes all optimization problems whose feasible solutions coincide with the multicuts of a graph, including the LMP we propose. In fact, the HMP is strictly more general than the LMP; its objective function can assign an objective value to all decompositions for which any set of edges is cut, unlike the objective function of the LMP which is limited to single edges. However, the instances of the HMP that are equivalent to the instances of the LMP we propose have an objective function whose order is equal to the number of edges in the graph and are hence impractical. Thus, the HMP and LMP are complementary in practice.

Efficient algorithms (primal feasible heuristics) for the MP are proposed and analyzed in (Bagon and Galun, 2011; Beier *et al.*, 2014, 2015; Kernighan and Lin, 1970). The algorithms we design and implement are compared here to CGC (Beier *et al.*, 2014). Our implementation of (an extension of) the Kernighan-Lin Algorithm (KL) (Kernighan and Lin, 1970) is compared here, in addition, to the implementation of KL in (Andres *et al.*, 2012a; Kappes *et al.*, 2015a).

Toward image decomposition (Arbeláez *et al.*, 2011), the state of the art in boundary detection, that even exceeds human performance, are UberNet (Kokkinos, 2017), DeepBoundary (Kokkinos, 2016), and CED (Wang *et al.*, 2017), followed closely by RDS (Liu and Lew, 2016) and HED (Xie and Tu, 2015). Our experiments are based on (Dollár and Zitnick, 2015), which is publicly available and only slightly outperformed by (Bertasius *et al.*, 2015a; Hallman and Fowlkes, 2015). The state of the art in image decomposition is MCG (Arbeláez *et al.*, 2014), followed closely by (Arbeláez *et al.*, 2011; Isola *et al.*, 2014). Our results are compared quantitatively to MCG (Arbeláez *et al.*, 2014).

Toward mesh decomposition (Theologou *et al.*, 2015), the state of the art is SyncSpecCNN (Yi *et al.*, 2017), followed closely by ACNN (Boscaini *et al.*, 2016) and Yi *et al.* (2016). Our experiments are based on (Kalogerakis and Hertzmann, 2010; Zhang *et al.*, 2012; Kin-Chung Au *et al.*, 2011). In prior work, methods based on learning mostly rely on a unary term which requires components to be labeled

semantically (Kalogerakis and Hertzmann, 2010; Xie *et al.*, 2014). One method based on edge probabilities was introduced previously (Benhabiles *et al.*, 2011). It applies a complex post-process (contour thinning and completion, snake movement) to obtain a decomposition. We show the first mesh decompositions based on multicuts.

2.3 PROBLEM

2.3.1 Minimum Cost Lifted Multicut Problem

We now define an optimization problem, the Minimum Cost Lifted Multicut Problem, whose feasible solutions relate one-to-one to the decompositions of a graph and whose objective function can assign, for any pair of nodes, a cost to all decompositions for which these nodes are in distinct components. Here, a component of a graph is any non-empty subgraph that is node-induced and connected. A decomposition of a graph is any partition Π of the node set such that, for every $V' \in \Pi$, the subgraph induced by V' is connected (and hence a component of the graph). An instance of the problem is defined w.r.t.:

- A simple, undirected graph $G = (V, E)$, *e.g.*, the pixel grid graph of an image or the triangle adjacency graph of a mesh.
- Additional edges $F \subseteq \binom{V}{2} \setminus E$ connecting nodes that are not neighbors in G. In practice, we choose F so as to connect any two nodes $v, w \in V$ whose distance d_{vw} in the graph holds $1 < d_{vw} \leq d^*$ for a maximum distance $d^* \in \mathbb{R}_0^+$, fixed for the experiments in Sec. 2.5.
- For every edge $vw \in E \cup F$, a cost $c_{vw} \in \mathbb{R}$ assigned to all feasible solutions for which v and w are in distinct components. The estimation of c_{vw} from image and mesh data is discussed in Sections 2.3.3 and 2.5.

With respect to the above, we define a feasible set $Y_{EF} \subseteq \{0,1\}^{E \cup F}$ whose elements $y \in Y_{EF}$ are 01-labelings of all edges $E \cup F$. The feasible set is defined such that two conditions hold: Firstly, the feasible solutions $y \in Y_{EF}$ relate one-to-one to the decompositions of the graph G. Secondly, for every edge $vw \in E \cup F$, $y_{vw} = 1$ if and only if v and w are in distinct components of G. This is expressed rigorously by two classes of constraints: The linear inequalities (2.2) below constrain y such that $\{e \in E \mid y_e = 1\}$ is a multicut of the graph G (Chopra and Rao, 1993). For any decomposition of a graph, the multicut related to the decomposition is the subset of those edges that straddle distinct components. In addition, the linear inequalities (2.3) and (2.4) constrain y such that, for any $vw \in F$, $y_{vw} = 0$ if and only if there exists a path in G from v to w, along which all edges are labeled 0.

Definition 3 For any simple, undirected graph $G = (V, E)$, any $F \subseteq \binom{V}{2} \setminus E$ and any $c : E \cup F \to \mathbb{R}$, the 01 linear program written below is called an instance of the

Minimum Cost Lifted Multicut Problem (LMP) w.r.t. G, F and c.

$$\min_{y \in Y_{EF}} \sum_{e \in E \cup F} c_e y_e \tag{2.1}$$

with $Y_{EF} \subseteq \{0,1\}^{E \cup F}$ the set of all $y \in \{0,1\}^{E \cup F}$ with

$$\forall C \in \text{cycles}(G)\ \forall e \in C:\ y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \tag{2.2}$$

$$\forall vw \in F\ \forall P \in vw\text{-paths}(G):\ y_{vw} \leq \sum_{e \in P} y_e \tag{2.3}$$

$$\forall vw \in F\ \forall C \in vw\text{-cuts}(G): 1 - y_{vw} \leq \sum_{e \in C} (1 - y_e) \tag{2.4}$$

2.3.2 Properties

We now discuss properties of the LMP (Def. 3):

For $F = \emptyset$, the LMP specializes to the MP (Chopra and Rao, 1993; Deza and Laurent, 1997). Its feasible set $Y_{E\emptyset}$ consists of the characteristic functions of all multicuts of G (which relate one-to-one to the decompositions of G). Its linear objective function can be chosen so as to assign, for any edge $vw \in E$, a cost $c_{vw} \in \mathbb{R}$ to all decompositions of G for which the nodes v and w are in distinct components. It cannot be chosen so as to assign, for distinct nodes v and w that are not neighbors in G, a cost precisely to all decompositions of G for which v and w are in distinct components.

For $F \neq \emptyset$, the LMP is not a MP. Its feasible solutions still relate one-to-one to the decompositions of the graph G (because $\varphi : Y_{EF} \to Y_{E\emptyset} : y \mapsto y_E$ is a bijection). Its objective function can be chosen so as to assign, for any $vw \in E \cup F$, a cost to all decompositions for which the nodes v and w are in distinct components. Thus, the LMP generalizes the MP. The feasible solutions $y \in Y_{EF}$ are called *lifted multicuts* from (V, E) to $(V, E \cup F)$ and are studied in (Horňáková *et al.*, 2017).

For some instances of the LMP, notably if $c_F < 0$ (Andres *et al.*, 2013), its solutions can be identified with the solutions of the instance of the MP w.r.t. the larger graph $G' := (V, E \cup F)$ and c. For the general LMP, this is not true. The feasible solutions of the MP with respect to G' and c do not relate one-to-one to the decompositions of G, unlike the feasible solutions of the LMP which are the characteristic functions of *some* multicuts of G', namely those that are *lifted* from G (Horňáková *et al.*, 2017).

A cutting plane algorithm for the LMP, based on the canonical LP-relaxation of the ILP in Def. 3, is impractical for the instances we consider in Sec. 2.5: Although the inequalities (2.2)–(2.4) can be separated efficiently, the number of to-be-separated inequalities (2.4) is prohibitive, and the facet-defining subset of (2.4) is unknown (Horňáková *et al.*, 2017). Thus, we propose in Sec. 2.4 two primal feasible heuristics for the LMP.

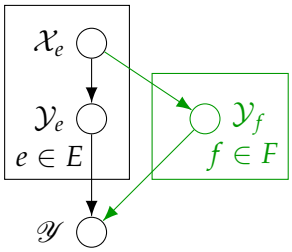


Figure 2.1: Depicted in black is a Bayesian Network defining a set of probability measures on multicuts (Andres *et al.*, 2011). Depicted in green is our extension defining a set of probability measures on lifted multicuts.

2.3.3 Probabilistic Model

We now define a familiy of probability measures on lifted multicuts for which the maximally probable lifted multicuts are the solutions the LMP (Def. 3). This relates the coefficients c of the LMP to image and mesh data.

Probability measures on multicuts Andres *et al.* (2011) define, with respect to a graph $G = (V, E)$ and with respect to the Bayesian Network depicted in Fig. 2.1 (in black), a measure of the conditional probability of a $y \in \{0,1\}^E$, given the feasible set $Y_{E\emptyset}$ of the characteristic functions of all multicuts of G and given, for every edge $e \in E$, a vector $x_e \in \mathbb{R}^n$ of $n \in \mathbb{N}$ edge features. Specifically,

$$p_{\mathcal{Y}|\mathcal{X},\mathscr{Y}} \propto p_{\mathscr{Y}|\mathcal{Y}} \cdot \prod_{e \in E} p_{\mathcal{Y}_e|\mathcal{X}_e} \tag{2.5}$$

$$\text{with} \qquad p_{\mathscr{Y}|\mathcal{Y}}(Y_{E\emptyset}, y) \propto \begin{cases} 1 & \text{if } y \in Y_{E\emptyset} \\ 0 & \text{otherwise} \end{cases} . \tag{2.6}$$

They show that y maximizes $p_{\mathcal{Y}|\mathcal{X},\mathscr{Y}}$ if and only if it is a solution of the instance of the MP with respect to G and $c \in \mathbb{R}^E$ such that

$$\forall e \in E : \quad c_e = \log \frac{p_{\mathcal{Y}_e|\mathcal{X}_e}(0, x_e)}{p_{\mathcal{Y}_e|\mathcal{X}_e}(1, x_e)} . \tag{2.7}$$

Probability measures on lifted multicuts We extend the Bayesian Network of (Andres *et al.*, 2011) in order to incorporate estimated probabilities not only for edges but also for pairs of nodes that are not neighbors.

The extension is depicted in Fig. 2.1 (in green). It contains one additional random variable $\mathcal{Y}_f$ for every $f \in F$. The conditional probability measures $p_{\mathcal{Y}|\mathcal{X},\mathscr{Y}}$ consistent with the extended Bayesian Network have the form

$$p_{\mathcal{Y}|\mathcal{X},\mathscr{Y}} \propto p_{\mathscr{Y}|\mathcal{Y}} \cdot \prod_{e \in E} p_{\mathcal{Y}_e|\mathcal{X}_e} \cdot \prod_{f \in F} p_{\mathcal{Y}_f|\mathcal{X}_E} . \tag{2.8}$$

A realization of all random variables $\mathcal{Y}$ is a 01-labeling $y \in \{0,1\}^{E \cup F}$ of all edges $vw \in E \cup F$. In order to constrain it to the characteristic functions of lifted multicuts, we consider (2.6) with Y_{EF} instead of $Y_{E\emptyset}$.

Probabilistic Geodesic Lifting Estimating, for edges $vw = e \in E$, the probability $p_{\mathcal{Y}_e|\mathcal{X}_e}$ of the nodes v and w being in distinct components, given features x_e defined by image and mesh data, is the classical problem of boundary estimation (Arbeláez *et al.*, 2011). In our experiments described in Sec. 2.5, we build on recent work (Dollár and Zitnick, 2015; Kalogerakis and Hertzmann, 2010; Kin-Chung Au *et al.*, 2011; Zhang *et al.*, 2012) in this field.

Estimating, for pairs $vw = f \in F$ of nodes v and w that are not neighbors, the probability $p_{\mathcal{Y}_f|\mathcal{X}_E}$ of v and w being in distinct components is a much harder problem: As these nodes could be connected by any path in G, this probability depends on the features x_E of all edges. In our experiments, we define, for all $vw = f \in F$:

$$p_{\mathcal{Y}_f|\mathcal{X}_E}(0, x_E) := \max_{P \in vw\text{-paths}(G)} \prod_{e \in P} p_{\mathcal{Y}_e|\mathcal{X}_e}(0, x_e) \ . \tag{2.9}$$

On the one hand, this under-estimates the probability as only one path is considered. On the other hand, it is the largerst such under-estimate as a maximally probable such path is considerd. Note also that $-\log p_{\mathcal{Y}_f|\mathcal{X}_E}(0, x_E)$ can be computed efficiently using, *e.g.*, Dijkstra's algorithm.

2.4 LOCAL SEARCH ALGORITHMS

We now introduce two efficient algorithms (primal feasible heuristics) which are applicable to the LMP (Def. 3) and the MP (the special case of the LMP for $F = \emptyset$).

Alg. 1 is an adaptation of greedy agglomeration, more specifically, greedy additive edge contraction. It takes as input an instance of the LMP defined by $G = (V, E)$, F and c (Def. 3) and constructs as output a decomposition of the graph G. Alg. 2 is an extension of the Kernighan-Lin Algorithm (Kernighan and Lin, 1970). It takes as input an instance of the LMP and an initial decomposition of G and constructs as output a decomposition of G whose lifted multicut has an objective value lower than or equal to that of the initial decomposition. Both algorithms maintain a decomposition of G, represented by graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ whose nodes $a \in \mathcal{V}$ are components of G and whose edges $ab \in \mathcal{E}$ connect any components a and b of G which are neighbors in G. Objective values are computed w.r.t. the larger graph $G' = (V, E \cup F)$ and c. We perform an intensive empirical comparison on various data sets of our proposed algorithms and competitors in Chapter 3.

2.4.1 Greedy Additive Edge Contraction (GAEC)

Overview Alg. 1 starts from the decomposition into single nodes. In every iteration, a pair of neighboring components is joined for which the join decreases

Algorithm 1: Greedy Additive Edge Contraction (GAEC)

```
while E ≠ ∅ do
    ab := argmax_{a'b' ∈ E} χ_{a'b'}
    if χ_ab < 0 then
        break
    contract ab in G and G'
    foreach ab ≠ ab' ∈ E' do
        χ_{ab'} := χ_{ab'} + χ_{bb'}
```

the objective value maximally. If no join strictly decreases the objective value, the algorithm terminates.

Implementation Our implementation (Andres) uses *ordered adjacency lists* for the graph $\mathcal{G}$ and for a graph $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$ whose edges $ab \in \mathcal{E}'$ connect any components a and b of G for which there is an edge $vw \in E \cup F$ with $v \in a$ and $w \in b$. It uses a *disjoint set data structure* for the partition of V and a *priority queue* for an ordered sequence of costs $\chi : \mathcal{E} \to \mathbb{R}$ of feasible joins. Its worst-case time complexity $O(|V|^2 \log |V|)$ is due to a sequence of at most $|V|$ contractions, in each of which at most $\deg \mathcal{G}' \leq |V|$ edges are removed, each in time $O(\log \deg \mathcal{G}') \in O(\log |V|)$.

2.4.2 Kernighan-Lin Algorithm with Joins (KLj)

Overview Alg. 2 starts from an initial decomposition provided as input. In each iteration, an attempt is made to improve the current decomposition by one of the following transformations: 1. moving nodes between two neighboring components, 2. moving nodes from one component to an additional, newly introduced component, 3. joining two neighboring components. The main operation "update_bipartition" is described below. It takes as input the current decomposition and a pair $ab \in \mathcal{E}$ of neighboring components of G and assesses Transformations 1 and 3 for this pair. Transformations 2 are assessed by executing "update_bipartition" for each component and $\emptyset$.

The operation "update_bipartition" constructs a sequence of elementary transformations of the components a and b and a $k \in \mathbb{N}_0$ such that the first k elementary transformations in the sequence, carried out in order, descrease the objective value maximally. Each elementary transformation consists in either moving a node currently in the component a which currently has a neighbor in the component b from a to b, or in moving a node currently in the component b which currently has a neighbor in the component a from b to a. The sequence of elementary transformations is constructed greedily, always choosing one elementary transformation that decreases the objective function maximally. If either the first k elementary transformations together or a complete join of the components a and b strictly decreases the objective

Algorithm 2: Kernighan-Lin Algorithm with Joins (KLj)

```
repeat
    foreach ab ∈ ℰ do
        if has_changed(a) or has_changed(b) then
            update_bipartition(𝒢, a, b)
    foreach a ∈ 𝒱 do
        if has_changed(a) then
            repeat
                update_bipartition(𝒢, a, ∅)
            until no changes
until no changes
```

value, an optimal among these operations is carried out.

Implementation Our implementation (Andres) of Alg. 2 tags components that are updated in order to avoid that a pair of components, that is fixed under "update_bipartition", is processed more than once. In the operation "update_bipartition", we maintain the set $\Omega \subseteq E$ of edges of G that straddle the components a and b. A substantial complication in the case of an LMP ($F \neq \emptyset$) arises from the fact that moving a node $v \in V$ from a component $a \subseteq V$ to a neighboring component $b \subseteq V$ might leave the set $a \setminus \{v\}$ disconnected. Keeping track of these cut-vertices by the Hopcroft-Tarjan Algorithm (Hopcroft and Tarjan, 1973) turned out to be impractical due to excessive absolute runtime. Our implementation allows for elementary transformations that leave components disconnected; we even compute the difference to the objective value incorrectly in such a case while constructing the sequence of elementary transformations. However, the first k elementary transformations are carried out only if the correct difference to the objective value (computed after the construction of the entire sequence) is optimal. Our implementation of "update_bipartition" has the worst-case time complexity $O(|a \cup b|(|\Omega| + \deg G'))$. The number of outer iterations of Alg. 2 is not bounded here by a polynomial but is typically small (less than 20 for all experiments in Sec. 2.5).

2.5 EFFECTIVENESS FOR IMAGE AND MESH DECOMPOSITION

2.5.1 Image Decomposition

We now apply both formulations of the graph decomposition problem, the Minimum Cost Multicut Problem (MP) and the Minimum Cost Lifted Multicut Problem (LMP) defined in Sec. 2.3, in conjunction with both algorithms defined in Sec. 2.4, GAEC and KLj, to the Image Decomposition Problem posed by the BSDS-500 benchmark (Arbeláez *et al.*, 2011).

	Boundary	Volume		
	F-measure	Covering	RI	VI
gPb-owt-ucm (Arbeláez *et al.*, 2011)	0.73	0.59	0.83	1.69
SE+MS+SH (Dollár and Zitnick, 2015)+ucm	0.73	0.59	0.83	1.71
MCG (Arbeláez *et al.*, 2014)	0.75	0.61	0.83	1.57
SE+**MP** GAEC	0.71	0.50	0.80	2.36
SE+**MP** GAEC-KLj	0.71	0.50	0.80	2.36
SE+**MP** 1-KLj	0.71	0.49	0.80	2.41
SE+**MP** GAEC-CGC	0.71	0.50	0.80	2.23
SE+**LMP**10 GAEC	0.71	0.51	0.80	2.33
SE+**LMP**10 GAEC-KLj	0.73	0.58	0.82	1.76
SE+**LMP**10 1-KLj	0.73	0.58	0.82	1.76
SE+**LMP**20 GAEC	0.71	0.52	0.80	2.22
SE+**LMP**20 GAEC-KLj	0.73	0.58	0.82	1.74
SE+**LMP**20 1-KLj	0.73	0.57	0.82	1.75

Table 2.1: Written above are boundary and volume metrics measuring the distance between the man-made decompositions of the BSDS-500 benchmark (Arbeláez *et al.*, 2011) and the decompositions defined by multicuts (MP), lifted multicuts (LMP) and top-performing competing methods (Arbeláez *et al.*, 2011, 2014; Dollár and Zitnick, 2015). Parameters are fixed for the entire data set (ODS).

For every test image, we define instances of the MP and the LMP as descibed in Sec. 2.3. For each of these, we compute a feasible solution, firstly, by greedy additive edge contraction (GAEC, Alg. 1) and, secondly, by applying the extended Kernighan-Lin Algorithm (KLj, Alg. 2) to the output of GAEC. All decompositions obtained in this way are compared to the man-made decompositions in the BSDS-500 benchmark in terms of boundary precision and recall (BPR) (Arbeláez *et al.*, 2011) and variation of information (VI) (Meilă, 2007). The VI is split into a distance due to false joins, plus a distances due to false cuts, as in (Keuper *et al.*, 2015). Statistics for the entire BSDS-500 test set are shown in Tab. 2.1 and Fig. 2.2 and are discussed below, after a specification of the experimental setup.

Setup For every image, instances of the MP are defined w.r.t.: 1. the pixel grid graph of the image, 2. for every edge in this graph, *i.e.*, for every pair of pixels that are 4-neighbors, the probability estimated in (Dollár and Zitnick, 2015) of these pixels being in distinct components, 3. a prior probability p^* of neighboring pixels being in distinct components. We vary $p^* \in \{0.05, 0.10, \dots, 0.95\}$, constructing one instance of the MP for every image and every p^*. For each of these instance of the MP, three instances of the LMP are defined by Probabilistic Geodesic Lifting (Sec. 2.3.3), one for each $d^* \in \{5, 10, 20\}$. Each experiment described in this section is conducted using one Intel Xeon CPU E5-2680 operating at 2.70 GHz (no parallelization).

Results It can also be seen from Fig. 2.2 that feasible solutions of the MP found by GAEC are not improved significantly by either of the local search heuristics KLj or CGC (Beier *et al.*, 2014). Compared to the man-made decompositions in the benchmark in terms of BPR and VI, feasbile solutions of the MP found by GAEC, improved by either CGC or KLj, are significantly worse than MCG (Arbeláez *et al.*, 2014) for this benchmark (Tab. 2.1).

In contrast, feasible solutions of the LMP found by GAEC are improved effectively and efficiently by KLj. CGC is not practical for the larger, non-planar graphs of the instances of the LMP we define; the absolute runtime exceeds 48 hours for every image and $p^* = 0.5$. Compared to the man-made decompositions in the benchmark, feasible solutions of the LMP found by GAEC and improved by KLj are not significantly worse than MCG (Arbeláez *et al.*, 2014) for this benchmark. The effect of changing p^* is shown for the average over all test images in Fig. 2.2 and for one image in particular in Fig. 2.5. The best decompositions for this image as well as for all images on average are obtained for $p^* = 0.5$. The effect of changing d^* is shown for the average over all test images in Fig. 2.3. It can be seen from this figure that increasing d^* from 5 to 10 improves results while further increasing d^* to 20 does not change results noticeably

2.5.2 Mesh Decomposition

We now apply our formulations and algorithms without any changes to the Mesh Segmentation Problem (Chen *et al.*, 2009).

Setup The Princeton Segmentation Benchmark (Chen *et al.*, 2009) consists of 19 classes, ranging from humans to man-made objects, each containing 20 meshes. Manual segmentations of these meshes provide us with a ground truth for evaluation and supervised learning. We compute informative features known to provide good results in previous work (Kalogerakis and Hertzmann, 2010; Benhabiles *et al.*, 2011; Xie *et al.*, 2014): curvatures (minimum, maximum, Gaussian and mean) computed at two different scales, shape diameter (Shapira *et al.*, 2008) and dihedral angle. Except the dihedral angle, which is computed for each edge, the curvatures and shape diameter are computed for each vertex of the mesh. To derive each of these criteria for an edge, we consider (1) its mean value over the two vertices of the edge, (2) its difference between the two vertices at each side of the edge and (3) the difference between its mean values computed on 1-ring neighborhoods at each side of the edge. This last combination provides additional robustness and multi-resolution behavior. This way, we obtain a 28-dimensional vector, which is more compact and efficient than the hundreds of features used in prior work (Kalogerakis and Hertzmann, 2010; Xie *et al.*, 2014). Probabilities of edges being cut are learned from the ground truth using a Random Forest classifier, through a leave-one-out experiment similarly to previous work (Kalogerakis and Hertzmann, 2010; Xie *et al.*, 2014). We apply Alg. 2 on the dual graph of the mesh (one node per triangle), varying the prior probabilities p^* of neighboring triangles being in distinct components, and $d^* \in \{60, 70, 80, 100\}$.

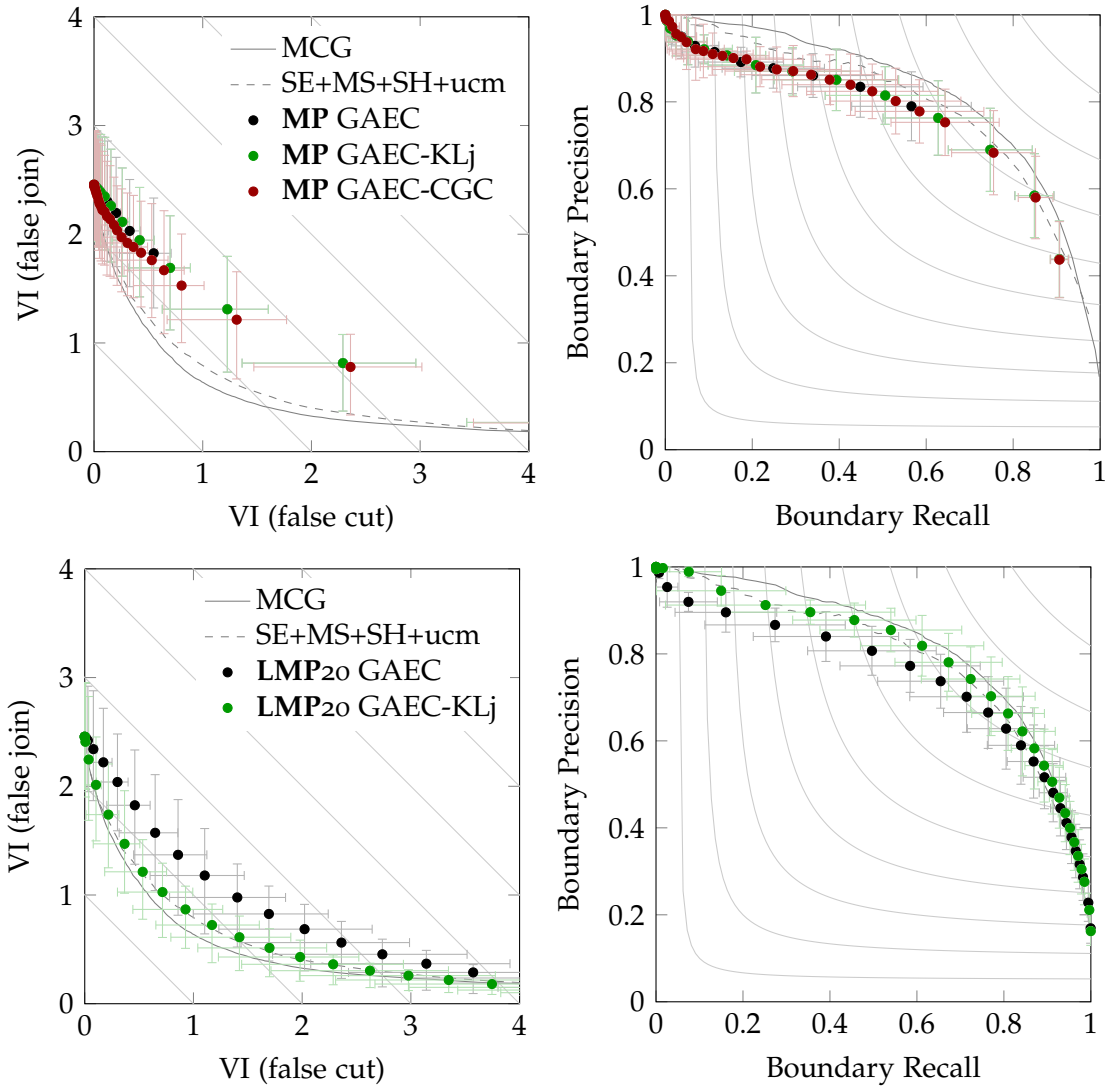


Figure 2.2: Depicted above is an assessment of the Multicut Problem (MP) and the Lifted Multicut Problem with $d^* = 20$ (LMP20) in conjunction with Alg. 1 (GAEC) and Alg. 2 (KLj), in an application to the image decomposition problem posed by the BSDS-500 benchmark (Arbeláez *et al.*, 2011). Every point in the figures above shows, for one problem and algorithm, the average over all test images in the benchmark. Depicted are, on the **left**, the variation of information (VI), split additively into a distance due to false cuts and a distance due to false joins, on the **right**, the accuracy of boundary detection, split into recall and precision. MCG (Arbeláez *et al.*, 2014) and SE+MS+SH (Dollár and Zitnick, 2015)+ucm are depicted as solid/dashed gray lines. Error bars depict the 0.25 and 0.75-quantile.

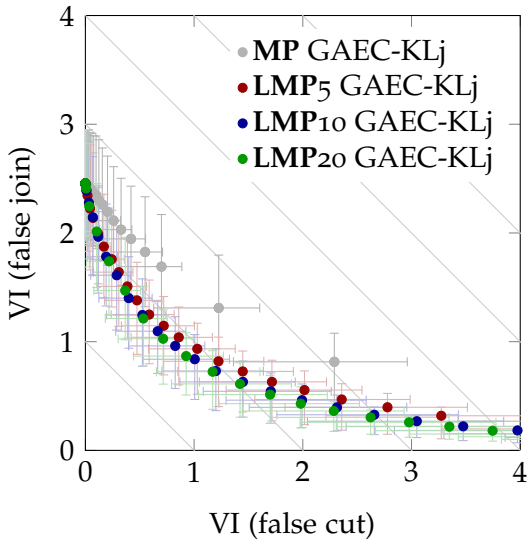


Figure 2.3: Depicted is the effect of the lifting distance d^*. It can be seen that increasing d^* from 5 to 10 considerably improves the quality of image decompositions as measured by the VI; further increasing d^* to 20 does not result in a measurable improvement.

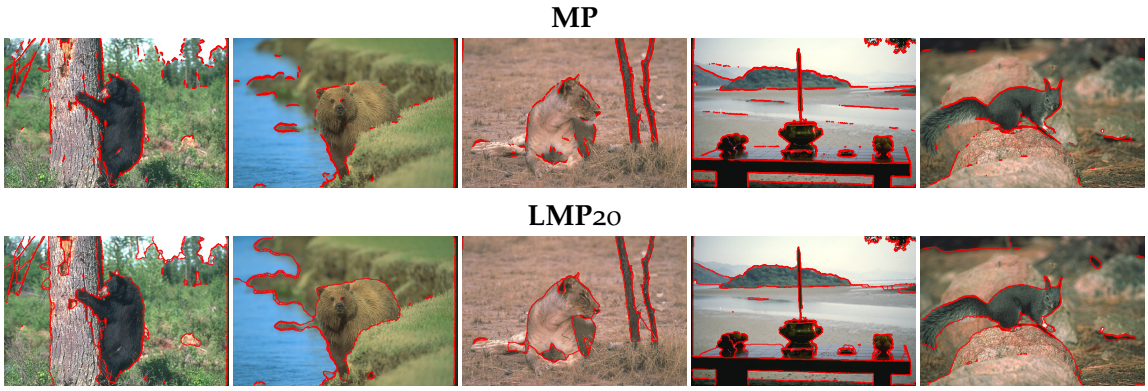


Figure 2.4: Depicted above is a comparison of decompositions defined by feasible solutions of the MP (**top** row for, the optimal $p^* = 0.8$), with decompositions defined by feasible solutions of the LMP (**bottom** row, for the optimal $p^* = 0.5$ and $d^* = 20$). All solutions are found by Alg. 2 (KLj), initialized with the output of Alg. 1 (GAEC). The decompositions defined by feasible solutions of the MP have closed contours but consist of tiny components on or near the boundary of desired components. This problem is overcome by feasible solutions of the LMP due to the long-range terms in the objective function.

$p^* = 0.3$ $p^* = 0.4$ $p^* = 0.5$ $p^* = 0.6$ $p^* = 0.7$

Figure 2.5: Depicted is the effect of the prior probability p^* which establishes a trade-off between over- and under-segmentation, reasonable in particular at $p^* = 0.5$.

	ZH	KC	KH		**MP**		**LMP**70		**LMP**opt	
	RI	RI	RI	VI	RI	VI	RI	VI	RI	VI
Human	0.89	0.88	0.88	1.43	0.21	2.93	0.86	1.62	0.87	1.79
Cup	0.80	0.79	0.90	0.35	0.74	0.68	0.89	0.39	0.90	0.39
Glasses	0.91	0.90	0.86	0.68	0.35	1.83	0.84	0.76	0.90	0.68
Airplane	0.89	0.87	0.92	0.67	0.69	1.39	0.92	0.82	0.92	0.83
Ant	0.98	0.96	0.98	0.37	0.93	0.67	0.98	0.42	0.98	0.42
Chair	0.89	0.88	0.95	0.43	0.79	0.98	0.93	0.55	0.93	0.55
Octopus	0.98	0.96	0.98	0.29	0.86	0.80	0.98	0.35	0.98	0.33
Table	0.90	0.94	0.94	0.28	0.76	0.81	0.94	0.28	0.94	0.29
Teddy	0.97	0.95	0.97	0.37	0.69	1.37	0.96	0.51	0.96	0.50
Hand	0.92	0.89	0.90	0.85	0.29	2.36	0.83	1.26	0.85	1.32
Plier	0.91	0.93	0.95	0.57	0.25	2.14	0.91	0.88	0.93	0.84
Fish	0.70	0.76	0.87	0.70	0.64	1.27	0.80	1.09	0.80	1.09
Bird	0.91	0.90	0.91	0.73	0.67	1.36	0.93	0.88	0.93	0.99
Armadillo	0.91	0.89	0.93	1.11	0.21	3.27	0.92	1.60	0.92	1.48
Bust	0.75	0.76	0.76	1.35	0.42	1.67	0.69	2.25	0.69	2.25
Mech	0.87	0.88	0.89	0.46	0.78	0.69	0.84	0.59	0.84	0.59
Bearing	0.83	0.82	0.91	0.45	0.87	0.60	0.84	0.69	0.84	0.69
Vase	0.88	0.83	0.85	0.75	0.55	1.34	0.83	0.90	0.84	0.87
FourLeg	0.86	0.82	0.86	1.34	0.30	2.58	0.84	1.84	0.84	1.72
Average	0.88	0.87	0.91	0.69	0.58	1.51	0.88	0.93	0.89	0.93

Table 2.2: Written above are boundary and volume metrics measuring the distance between the man-made decompositions of meshes in the Princeton Benchmark and the decompositions defined by multicuts (MP), lifted multicuts (LMP) and top-performing competing methods ZH=(Zhang *et al.*, 2012), KC=(Kin-Chung Au *et al.*, 2011), and KH=(Kalogerakis and Hertzmann, 2010) (statistics obtained from the respective papers when available). The evaluation is for a fixed parameter set for the entire database ($p^* = 0.55$, $d^* = 70$: LMP70), as well as for the best parameter set we found for each class of meshes (LMPopt). Results for the MP are for the optimal $p^* = 0.9$.

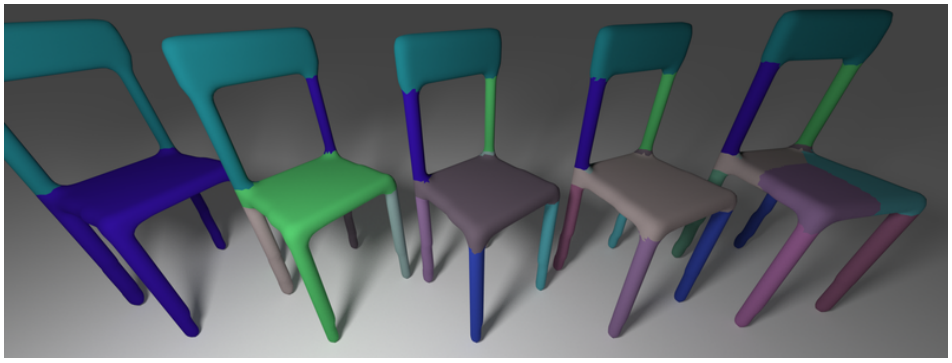


Figure 2.6: Depicted above is the effect of varying the prior probability p^* of adjacent triangles being in distinct components. Here, $p^* \in \{0.5, 0.55, 0.58, 0.6, 0.62\}$, from left to right.

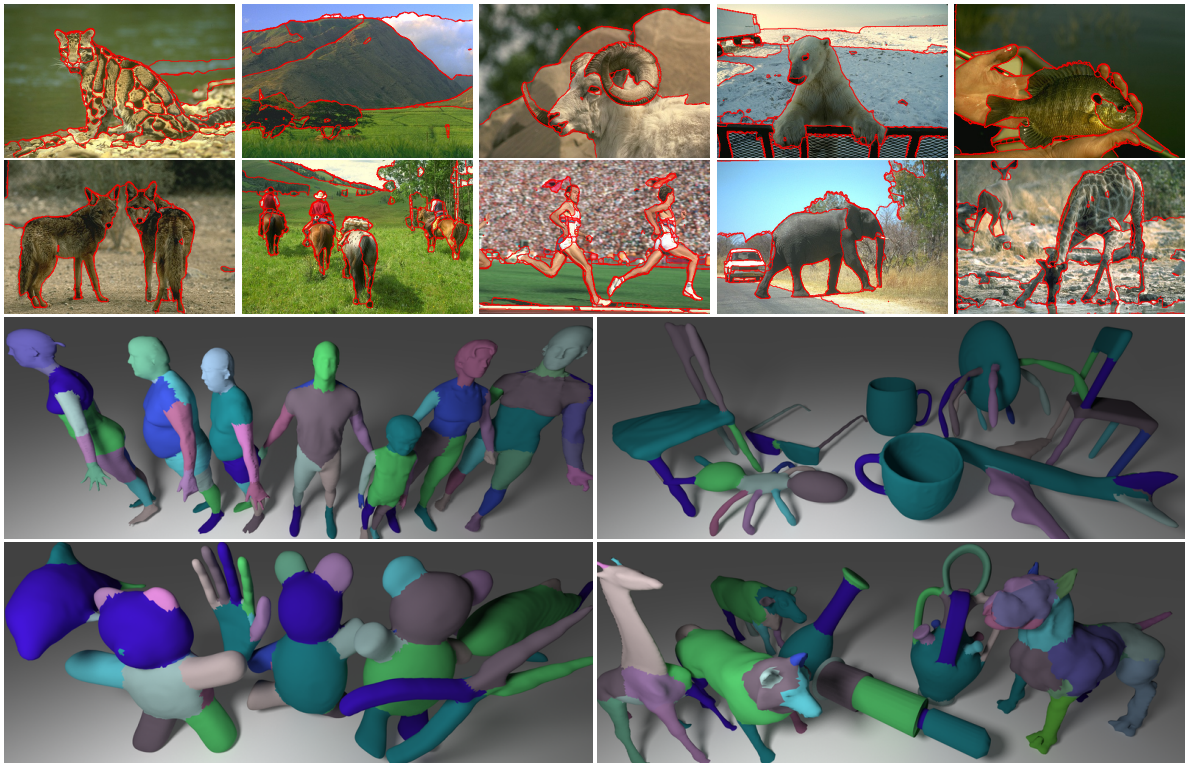


Figure 2.7: Depicted above is a sample of decompositions of images and meshes found by solving an instance of the Lifted Multicut Problem using Alg. 2. Included are some cases where this approach fails.

Results An evaluation in terms of Rand's index (RI) and the VI is shown in Tab. 2.2. The results are slightly better than (Zhang *et al.*, 2012; Kin-Chung Au *et al.*, 2011) and close to those of (Kalogerakis and Hertzmann, 2010). However, (Kalogerakis and Hertzmann, 2010) require a semantic labeling of the ground-truth, while our multicut formulation only requires boundary information. The median computation time per model is resp. 51 seconds for the lifting and 59 seconds for Alg. 2 on an Intel i7 Pentium laptop computer operating at 2.20 GHz. Graphs have a median of 18000 nodes and 27000 edges. A sample of our results is shown in Fig. 2.7. Varying the prior probability p^* of cuts allows for controlling the amount of over or under-segmentation, as shown in Fig. 2.6.

2.6 CONCLUSION

We have introduced a generalization of the Minimum Cost Multicut Problem, the Minimum Cost Lifted Multicut Problem, which overcomes limitations of the MP in applications to image and mesh segmentation. We have defined and implemented two efficient algortihms (primal feasible heuristics) applicable to the MP and the LMP. We have assessed both algorithms in conjunction with both optimization problems in applications to image decomposition (BSDS-500 benchmark (Arbeláez *et al.*, 2011)) and mesh decomposition (Princeton benchmark (Chen *et al.*, 2009)). In both applications, we have found solutions that do not differ significantly from competing methods. This suggests that the LMP is a useful formulation of graph decomposition problems in vision.

3 EMPIRICAL COMPARISON OF LOCAL SEARCH ALGORITHMS

Contents

In previous chapter we introduced two heuristics for the lifted multicut problem and noted that multicut problem is a special case of the latter. Here, we empirically compare four local search algorithms for the multicut problem by applying them to a variety of instances of the multicut problem for the tasks of image segmentation, hand-written digit classification, and social network analysis. Although the local search algorithms establish neither lower bounds nor approximation certificates, they converge monotonously to a fixed point, offering a feasible solution at any time. For some algorithms, the time of convergence is affordable for all instances we consider. This finding encourages a broader application of the multicut problem, especially in settings where the number of clusters is not known and needs to be estimated from data.

3.1 INTRODUCTION

Given a finite set and, for any pair of distinct elements, a real-valued cost to be paid if these elements are put in distinct subsets, partitioning the set optimally, so as to minimize the sum of costs, is an NP-hard problem (Bansal *et al.*, 2004; Grötschel and Wakabayashi, 1989). Given a graph and, for any of its edges, a cost to be paid if the incident nodes are put in distinct components, decomposing the graph optimally, so as to minimize the sum of costs, is a generalization that specializes to the former problem for complete graphs (Chopra and Rao, 1993; Demaine *et al.*, 2006). The problem is known as *correlation clustering* from (Bansal *et al.*, 2004; Demaine *et al.*, 2006) and is known as the multicut problem stated in (Chopra and Rao, 1993; Horňáková *et al.*, 2017) in the form of an integer program whose feasible solutions are binary labelings $x \in \{0,1\}^E$ of the edges E of a graph. For any edge $e = \{v, w\}$, the label $x_e = 1$ indicates that this edge e is cut, i.e., that the nodes v and w are in distinct components:

Let us recall Def. 2 that states, that for any graph $G = (V, E)$ and any $c \in \mathbb{R}^E$, the instance of the multicut problem w.r.t. G and c is the binary linear problem

$$\min_{y \in \{0,1\}^E} \quad \sum_{e \in E} c_e \, y_e \tag{3.1}$$

$$\text{subject to} \quad \forall C \in \text{chordless-cycles}(G) \; \forall e \in C : \; y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \tag{3.2}$$

Correlation clustering differs from clustering based on non-negative distances such as *multi-terminal cut* (Dahlhaus *et al.*, 1994), *k-cut* (Goldschmidt and Hochbaum, 1994), and *balanced cut* problems. In correlation clustering, no costs or constraints are put explicitly on the number or sizes of clusters. These properties need not be known when stating an instance of the problem. Instead, they are defined by any feasible solution. This is appealing in applications where the number and size of clusters is to be estimated from data, e.g., image segmentation and social network analysis.

Contribution We empirically compare local search algorithms for some instances of the multicut problem. The algorithms we compare are primal feasible heuristics. They output a sequence of feasible solutions that converges strictly to a fixpoint. They do not offer bounds or optimality certificates for these fixpoints. We compute such bounds by polyhedral algorithms where possible. We show that local search algorithms are practical for relatively large instances of the multicut problem, including one for clustering of the MNIST test set of images of hand-written digits (LeCun *et al.*, 1998) and two for clustering social networks (Leskovec *et al.*, 2010).

3.2 RELATED WORK

One principled approach to solving instances of the multicut problem, pursued, e.g., in (Kappes *et al.*, 2011; Kim *et al.*, 2011, 2014; Yarkony *et al.*, 2012; Yarkony, 2014),

is by solving the linear program (LP) obtained by an outer relaxation (Chopra and Rao, 1993; Grötschel and Wakabayashi, 1989; Yarkony *et al.*, 2012) of the multicut polytope (Chopra and Rao, 1993). In conjunction with suitable rounding, this yields a logarithmic approximation that was established independently in (Charikar *et al.*, 2005) and (Demaine *et al.*, 2006). Solving an LP relaxation and rounding the solution can be efficient in general and practical for small instances. For large and complex instances, even the LP relaxation consisting of only the cycle inequalities (3.2) can be prohibitive.

In an attempt to avoid this separation problem,Andres *et al.* (2012b); Kappes *et al.* (2015a) separate only integral points by cycle inequalities (by breadth first search) and resort to general classes of cuts for fractional points. For some instances of the problem, exact solutions are found in less than the time required to solve the canonical LP relaxation. Yet, the absolute running time of both these algorithms is prohibitive for the large instances we study in Section 3.4.

For planar graphs, the minimum cost multicut problem remains NP-hard (Bachrach *et al.*, 2013; Voice *et al.*, 2012), but admits a polynomial time approximation scheme (PTAS) (Klein *et al.*, 2015). Due to a doubly exponential constant factor, this PTAS has mostly been of theoretical interest.

Partial optimality results for the multicut problem, established in (Alush and Goldberger, 2012, 2015), imply for some instances, a decomposition into independent sub-problems that can in principle be exploited by any algorithm. Swoboda and Andres (2017) proposed a message passing algorithms to compute lower bounds and re-parametrization of the multicut problem, but applications to large-scale problems still pose computational challenges. Recent work by Lange *et al.* (2018) further establish partial optimality conditions that allow to solve problems for series-parallel graphs to optimality, in linear time. They also compute faster lower bounds, than by a canonical LP relaxation, and a re-weighting, that allows local search algorithms to obtain better solutions.

Local search algorithms for the multicut problem, that relate to pioneering work by Kernighan and Lin (1970) are studied by Beier *et al.* (2014); Pan *et al.* (2015). While these algorithms do not offer lower bounds or approximation certificates, they output a sequence of feasible solutions that converges strictly to a fixpoint.

3.3 ALGORITHMS

This section describes the algorithms for solving instances of multicut problems used in the comparison. GAEC and KLj were presented in Sec. 2.4. Branch-and-Cut algorithm converges to a globally optimal solution, while local search algorithms output a sequence of feasible solutions that converges strictly to a fixpoint. The latter do not offer any bounds or approximation certificates for these feasible solutions.

3.3.1 Branch-and-Cut

Solving (3.1) directly is infeasible for any real dataset, because the number of constraints (3.2) is exponential for general graphs. For complete graphs this number is exactly $\binom{|V|}{3}$, which is still too large for practical applications.

In order to find the solution of (3.1) we implemented an efficient Branch-and-Cut (B&C) algorithm similar to way proposed in (Andres *et al.*, 2012b; Kappes *et al.*, 2011). It iteratively solves a less constrained problem, using Gurobi, and then adds the constraints (3.2) violated by the current solution. It goes on until no any constraints are violated. The search of violated constraints can be efficiently implemented by means of connected components labeling and breadth-first search.

In the experiments (when feasible) we also compute lower bounds (LB) given by the canonical relaxation of (3.1) that allows $y \in [0,1]^{|E|}$, which is solved using a cutting plane approach (Kappes *et al.*, 2015a).

3.3.2 Cut, Glue and Cut (CGC)

Cut, Glue and Cut, defined in (Beier *et al.*, 2014), is a local search algorithm that consists of two phases. During the *cut phase*, the graph is recursively bipartitioned. During the *glue and cut phase*, pairs of neighboring clusters are visited and, for any such pair, a two-colorable cut of their union is found by solving a max-cut problem. If the graph is planar, the max-cut problem is solved exactly and efficiently by the Blossom Algorithm (Schraudolph and Kamenetsky, 2009). If the graph is non-planar, the max-cut problem is solved approximately by the QPBO-I Algorithm (Rother *et al.*, 2007).

We examine here the C++ implementation of (Beier *et al.*, 2014), using the Blossom Algorithm for planar graphs. The implementation is shown in (Beier *et al.*, 2014) to converge faster than Expand-and-Explore (Bagon and Galun, 2011) and PlanarCC (Yarkony *et al.*, 2012).

3.3.3 Greedy Fixation (GF)

Greedy Fixation, a simple algorithm we propose, is similar in spirit to GAEC. Unlike GAEC, the treatment of joins (0) and cuts (1) is symmetric. GF starts from the partial labeling $y \in \{0,1,*\}^E$ of edges according to which all edges are undecided ($*$). It proceeds by fixing edge labels y_e to 0 (join) or 1 (cut) in the order of their absolute cost. Whenever an edge is labeled 0, other edges are labeled 0 if this is implied by transitivity.

We examine here our C++ implementation of GF that is analogous to that of GAEC. The time complexity of GF is equal to that of GAEC. The absolute running time is longer by a constant factor.

Data set	#	$\|V\|$	$\|E\|$	Properties
seg-2d (Andres *et al.*, 2011)	100	156 – 3764	439 – 10970	planar
seg-3d-300 (Andres *et al.*, 2012b)	8	3846 – 5896	23763 – 36221	non planar
seg-3d-450 (Andres *et al.*, 2012b)	8	15150 – 17074	94121 – 107060	non planar
slashdot (Leskovec *et al.*, 2010)	1	82144	500481	unit costs
epinions (Leskovec *et al.*, 2010)	1	131828	711210	unit costs
mnist	1	10000	49995000	complete

Table 3.1: Characteristics of data sets used for evaluation with number of instances varying from 1 to 100. The sizes of instances vary from hundreds to thousands of nodes to millions of edges. We also consider graphs of different topologies.

3.4 EXPERIMENTS

We apply the algorithms described in Section 3.3 and GAEC and KLj (Sec. 2.4) to instances of the multicut problem from the six data sets characterized in Tab. 3.1. Five of these data sets have been used as a benchmark for multicut algorithms in (Beier *et al.*, 2015, 2014; Kappes *et al.*, 2015a). The sixth (mnist) is our contribution, a multicut problem whose feasible solutions define a partition of the MNIST test set of hand-written digits (LeCun *et al.*, 1998). As initial clusterings (input), we consider the output of GAEC, the output and GF, the clustering in which all nodes are in one cluster ($y = 0$), as well as the clustering in which every node forms a separate cluster ($y = 1$).

The data sets of image segmentation ("seg" in Tab. 3.1) come with an estimated cut probability $p_e \in (0, 1)$ for every edge $e \in E$. In order to obtain instances of the multicut problem, we define costs w.r.t. these probabilities according to $c_e := \log \frac{1-p_e}{p_e}$ and refer to (Kappes *et al.*, 2015b) for a discussion of the probabilistic model. For the clustering of the MNIST test set of hand-written digits, we estimate the cut probabilities as described in Section 3.4.3. For the clustering of social networks, we transform signed directed graphs encoding (dis)trust as described in Section 3.4.4.

We report for each combination of an algorithm and a data set the running time until termination, the objective value of the output feasible solution, and, where available, a distance of the output feasible solution form a known true clustering. This distance is measured by the variation of information (VI) (Meilă, 2007), split additively (Keuper *et al.*, 2015) into the contribution due to false cuts (VI_p) and false joins (VI_r), and is measured also by Rand's Index (RI). VI is a metric that is lower-bounded from below by 0. RI is a measure of similarity that is bounded from above by 1. To put the costs in perspective, we report the lower bounds (LB) on the minimum cost obtained from the LP relaxation of (Kappes *et al.*, 2015a), and we also report feasible solutions found by the branch-and-cut (B&C) algorithm.

All C++ implementations are compiled with the same parameters and are executed without multi-threading, on the same machine, an Intel® Core™i3-2100 CPU,

Algorithm	T [s]	Value	VI	VI_p	VI_r	RI
zeros	0.000	0.00	1.68	0.00	1.68	0.44
ones	0.000	2201.12	5.60	5.39	0.20	0.83
Branch-and-Cut	1.498	-2966.85	2.27	1.64	0.63	0.86
Lower Bound	3.182	-2967.57	–	–	–	–
GAEC	0.019	-2954.40	2.26	1.62	0.65	0.85
GF	**0.013**	-2950.90	2.28	1.66	0.63	0.85
zeros + KL (Kappes *et al.*, 2015a)	0.328	-2771.95	2.42	1.51	0.90	0.72
zeros + CGC (Beier *et al.*, 2014)	0.427	-2963.77	2.26	1.62	0.64	0.85
zeros + KLj	0.054	-2944.14	2.27	1.65	0.62	0.86
ones + KL (Kappes *et al.*, 2015a)	0.131	-2808.10	2.88	2.39	0.49	0.86
ones + CGC (Beier *et al.*, 2014)	0.990	-2963.94	2.28	1.64	0.65	0.85
ones + KLj	0.057	-2947.37	2.29	1.66	0.63	0.85
GAEC + KL (Nowozin and Jegelka, 2009)	49.948	-2957.84	2.26	1.62	0.64	0.85
GAEC + KL (Kappes *et al.*, 2015a)	0.430	-2957.79	2.26	1.62	0.64	0.85
GAEC + CGC (Beier *et al.*, 2014)	0.393	**-2964.20**	2.27	1.64	0.63	0.85
GAEC + KLj	0.046	-2959.08	2.27	1.62	0.65	0.85
GF + KL (Nowozin and Jegelka, 2009)	49.380	-2955.39	2.28	1.65	0.63	0.85
GF + KL (Kappes *et al.*, 2015a)	0.406	-2955.34	2.28	1.65	0.63	0.85
GF + CGC (Beier *et al.*, 2014)	0.391	-2963.86	2.27	1.63	0.64	0.85
GF + KLj	0.042	-2958.96	2.27	1.63	0.64	0.85

Table 3.2: Empirical comparison of the algorithms on the problem of segmenting planar images (*seg-2d*).

operating at 3.10 GHz and equipped with 8 GB of RAM.

3.4.1 Image Segmentation: *seg-2d*

Toward the problem of segmenting each of the 100 test images of the Berkeley Segmentation Data Set (Martin *et al.*, 2001), we consider the instances of the multicut problem defined in (Andres *et al.*, 2011). These instances are publicly available as part of the OpenGM benchmark (Kappes *et al.*, 2015a). Results are shown in Fig. 3.1 and Tab. 3.2.

It can be seen form these results that local search algorithms find feasible solutions which are close in terms of objective value and accuracy to the optimal solutions found by B&C for these instances. Moreover, local search algorithms find such solutions one to two orders of magnitude faster than B&C. GF is slightly faster than GAEC, terminating with feasible solutions that are slightly worse. KLj is an order of magnitude faster than CGC (Beier *et al.*, 2014) and KL (Kappes *et al.*, 2015a) and is more stable under with regard to initialization. The implementation of KL from (Nowozin and Jegelka, 2009) which is not optimized for performance does not process the data set within 24 hours for "zeros" and "ones" initialization. This

Algorithm	T [s]	Value	VI	VI_p	VI_r	RI
zeros	0.000	0.00	4.09	0.00	4.09	0.12
ones	0.000	40478.83	6.77	6.69	0.07	0.88
Branch-and-Cut	68.088	-27302.76	1.64	0.82	0.82	0.88
Lower Bound	631.769	-27304.81	–	–	–	–
GAEC	0.070	-27162.81	1.76	0.85	0.91	0.87
GF	**0.054**	-27142.13	1.75	0.86	0.89	0.87
zeros + KL (Kappes *et al.*, 2015a)	2.020	-25570.35	3.30	1.13	2.17	0.72
zeros + CGC (Beier *et al.*, 2014)	5.481	-27253.30	1.80	0.83	0.96	0.86
zeros + KLj	0.781	-27275.89	1.68	0.83	0.85	0.88
ones + KL (Kappes *et al.*, 2015a)	45.608	-25217.20	2.22	1.44	0.77	0.87
ones + CGC (Beier *et al.*, 2014)	25.144	-27268.02	1.74	0.84	0.91	0.87
ones + KLj	0.647	-27266.11	1.64	0.80	0.84	0.88
GAEC + KL (Kappes *et al.*, 2015a)	8.543	-27201.47	1.76	0.85	0.91	0.87
GAEC + CGC (Beier *et al.*, 2014)	2.398	-27277.15	1.71	0.84	0.87	0.87
GAEC + KLj	0.406	-27258.25	1.69	0.84	0.85	0.88
GF + KL (Kappes *et al.*, 2015a)	8.628	-27193.11	1.74	0.86	0.88	0.87
GF + CGC (Beier *et al.*, 2014)	2.298	**-27277.64**	1.69	0.83	0.86	0.88
GF + KLj	0.437	-27251.91	1.70	0.84	0.85	0.88

Table 3.3: Empirical comparison of the algorithms on the problem of segmenting volume images (*seg-3d-300*).

shows that attention to constant factors is important at this scale of the problem.

3.4.2 Volume Image Segmentation: *seg-3d-300* and *seg-3d-450*

Toward the problem of segmenting volume images of neuronal processes taken by an electron microscope (Knott *et al.*, 2008) we consider the instances of the multicut problem defined in (Andres *et al.*, 2012b). These instances are publicly available as part of the OpenGM benchmark (Kappes *et al.*, 2015a). Results are shown in Fig. 3.1 and Tab. 3.3 and Tab. 3.4.

These results are comparable to those for image segmentation (Sec. 3.4.1). Yet, differences between algorithms become more pronounced here, as the instances are larger. For instance, the time until convergence for GAEC divided by the time until convergence of B&C is of the orders 10^{-3} for seg-3d-300 and 10^{-4} for seg-3d-450. KLj converges faster than KL (Kappes *et al.*, 2015a), and to better feasible solutions.

Algorithm	T [s]	Value	VI	VI_p	VI_r	RI
zeros	0.000	0.00	4.67	0.00	4.67	0.09
ones	0.000	186787.03	7.94	7.87	0.07	0.91
Branch-and-Cut	4130.273	-78474.69	2.02	0.93	1.09	0.87
Lower Bound	10337.771	-78482.98	–	–	–	–
GAEC	0.340	-78281.49	2.11	0.94	1.17	0.85
GF	**0.207**	-78194.01	2.13	0.97	1.16	0.85
zeros + KL (Kappes *et al.*, 2015a)	16.007	-72984.82	4.16	1.25	2.91	0.67
zeros + CGC (Beier *et al.*, 2014)	64.850	-78252.24	2.32	0.96	1.36	0.83
zeros + KLj	12.687	-78390.69	2.09	0.95	1.14	0.86
ones + KL (Kappes *et al.*, 2015a)	628.895	-61755.28	2.95	2.13	0.82	0.89
ones + CGC (Beier *et al.*, 2014)	375.190	-78369.18	2.17	0.94	1.23	0.85
ones + KLj	12.636	-78388.89	2.08	0.94	1.14	0.86
GAEC + KL (Kappes *et al.*, 2015a)	122.214	-78358.89	2.11	0.95	1.17	0.85
GAEC + CGC (Beier *et al.*, 2014)	22.815	-78403.27	2.09	0.95	1.14	0.86
GAEC + KLj	6.565	**-78417.99**	2.04	0.94	1.10	0.86
GF + KL (Kappes *et al.*, 2015a)	127.386	-78345.25	2.12	0.96	1.16	0.85
GF + CGC (Beier *et al.*, 2014)	28.497	-78392.97	2.10	0.95	1.15	0.86
GF + KLj	7.109	-78412.11	2.08	0.95	1.13	0.86

Table 3.4: Empirical comparison of the algorithms on the problem of segmenting volume images (*seg-3d-450*).

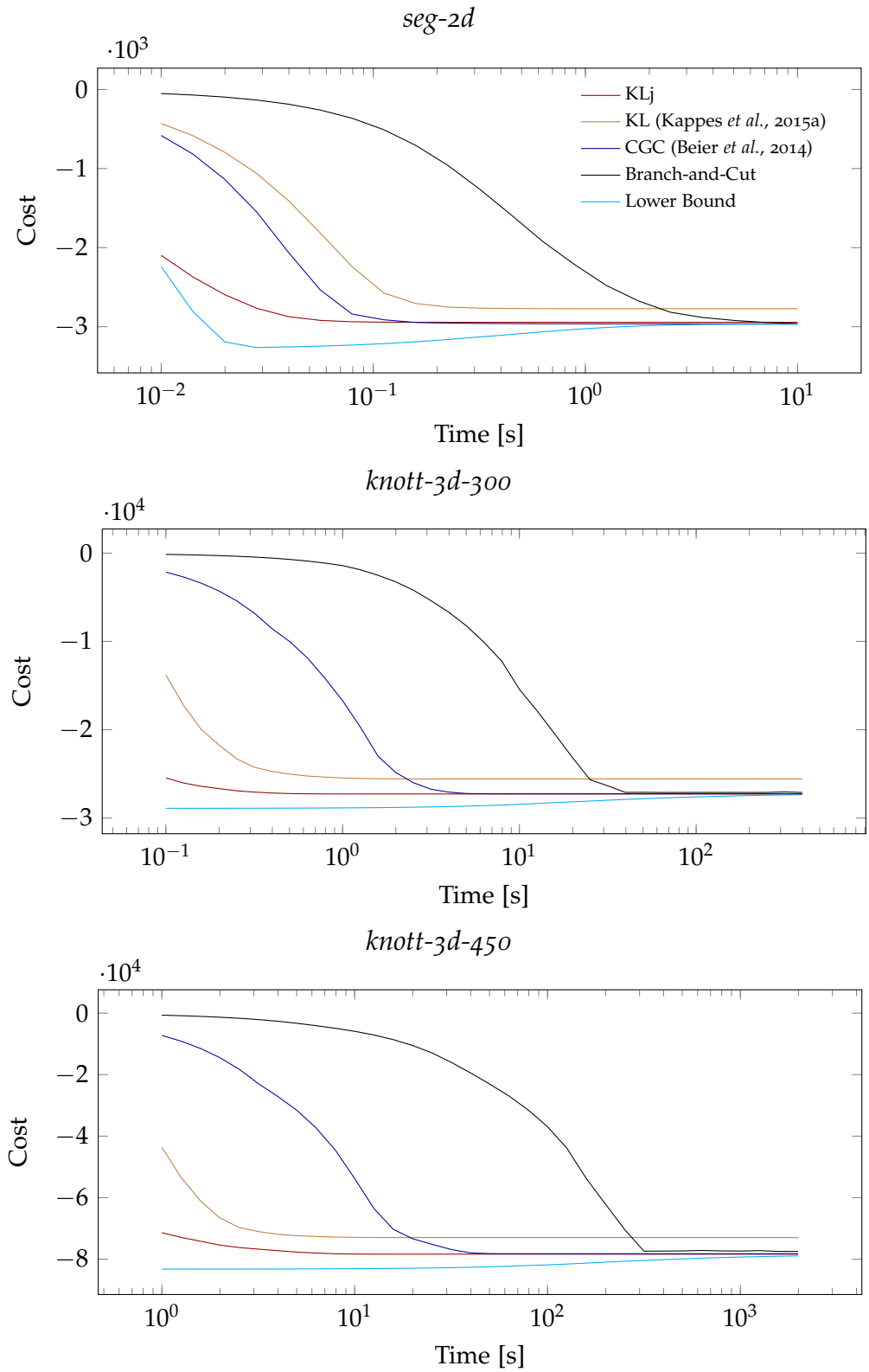


Figure 3.1: Convergence behaviour of the algorithms (mean objective vs. time), initialized with "zeros".

3.4.3 Image Collection Clustering: MNIST

Toward the problem of partitioning the MNIST test set of images of hand-written digits (LeCun *et al.*, 1998), we learn from the MNIST training set a Siamese CNN to predict, for any pair of images, whether they depict the same or distinct digits. The learning is described in detail below. The probabilities predicted by this Siamese CNN for the MNIST test set define an instance of the multicut problem for the complete graph whose nodes are the test images.

Results from applying the local search algorithms described in Section 3.3 to this instance are shown in Tab. 3.5 and 3.6. It can be seen from these results that local search algorithms are practical for this large instance. This is in contrast to the implementations of B&C and LB we have applied, which are impractical for this instance. Secondly, it can be seen from the VI and RI as well as from the confusion matrix in Tab. 3.6 that the clustering we obtain as a feasible solution is exceptionally accurate, defining 14 clusters, 4 of which contain only a single image. By relating the ten largest clusters to the ten digits optimally, this clustering can be associated with a misclassification rate of $M = 0.82\%$ (see Tab. 3.5) which is lower than the misclassification rate of LeNet trained for classification (1.1%). This result is surprising, given that we solve a clustering problem and do not prescribe the number of clusters. It encourages applications of multicut even in settings where the number of clusters *is* known.

Learning a Siamese CNN for relating images We train a Siamese CNN from scratch, with a cross entropy loss, built on the architecture of LeNet (LeCun *et al.*, 1998) which has a standard implementation in the Caffe framework (Jia *et al.*, 2014). We duplicate the convolution part of LeNet with shared weights to take two images. We feed outputs of these parts to the fully connected layer of the LeNet architecture. The CNN has two outputs, scores for two images depicting the same and distinct digits, respectively. We learn the parameters of this network from the MNIST training set, without any data augmentation. We stick to the solver parameters proposed in the Caffe example[1] in which LeNet is trained for a classification task. Specifically, we use stochastic gradient descent and an inverse decay learning policy with the base learning rate 0.01, $\gamma = 0.0001$, and power 0.75. The batch size is 64. We carry out 150000 iterations.

[1] http://caffe.berkeleyvision.org/gathered/examples/mnist.html

Algorithm	Time [s]	Value	VI	VI_p	VI_r	RI	M%
zeros	0.000	0.0	3.319	0.000	3.319	0.100	0
ones	0.000	-825928042.4	9.968	9.968	0.000	0.900	99.9
GAEC	369.77	-906355440.5	0.220	0.113	0.108	0.995	1.20
GF	148.33	-906355440.5	0.220	0.113	0.108	0.995	1.20
zeros + KL (Kappes *et al.*, 2015a)	1796.64	-906765155.8	0.162	0.083	0.079	0.997	0.82
zeros + CGC (Beier *et al.*, 2014)	–	–	–	–	–	–	–
zeros + KLj	**120.66**	-906765155.8	0.162	0.083	0.079	0.997	0.82
ones + KL (Kappes *et al.*, 2015a)	50197.41	-906765155.8	0.162	0.083	0.079	0.997	0.82
ones + CGC (Beier *et al.*, 2014)	120418.80	-906765155.8	0.162	0.083	0.079	0.997	0.82
ones + KLj	1403.22	-906765155.8	0.162	0.083	0.079	0.997	0.82
GAEC + KL (Kappes *et al.*, 2015a)	1197.80	-906765155.8	0.162	0.083	0.079	0.997	0.82
GAEC + CGC (Beier *et al.*, 2014)	1898.33	-906585148.4	0.174	0.089	0.085	0.997	0.91
GAEC + KLj	415.93	-906765155.8	0.162	0.083	0.079	0.997	0.82
GF + KL (Kappes *et al.*, 2015a)	896.51	-906765155.8	0.162	0.083	0.079	0.997	0.82
GF + CGC (Beier *et al.*, 2014)	1553.41	-906585148.4	0.174	0.089	0.085	0.997	0.91
GF + KLj	186.45	-906765155.8	0.162	0.083	0.079	0.997	0.82

Table 3.5: Empirical comparison of the algorithms described Section 3.3 on the problem of clustering the images of hand-written digits in the MNIST test set (LeCun *et al.*, 1998).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	Σ
0	976	0	0	0	0	0	1	1	1	0	0	0	1	0	980
1	0	1132	1	2	0	0	0	0	0	0	0	0	0	0	1135
2	1	0	1026	0	1	0	0	3	1	0	0	0	0	0	1032
3	0	0	1	1003	0	4	0	0	1	1	0	0	0	0	1010
4	0	0	0	0	976	0	1	0	0	5	0	0	0	0	982
5	2	0	0	8	0	880	1	0	0	1	0	0	0	0	892
6	4	2	0	0	1	1	946	0	2	0	0	1	0	1	958
7	0	2	5	1	0	0	0	1017	1	1	1	0	0	0	1028
8	2	0	2	1	0	1	0	1	965	2	0	0	0	0	974
9	1	0	0	1	2	4	0	4	0	997	0	0	0	0	1009
Σ	986	1136	1035	1016	980	890	949	1026	971	1007	1	1	1	1	

Table 3.6: Confusion matrix of the MNIST test images for the best obtained solution, with the outliers shown below.

	epinions		*slashdot*	
Algorithm	Time [s]	Value	Time [s]	Value
GAEC	4.54	-70205	2.93	-49071
GF	**2.03**	-70219	**1.33**	-49085
zeros + KLj	60219.8	-70702	35038.6	-49273
ones + KL	40368.4	-71170	25261.4	-49468
GAEC + KLj	6363.45	**-71288**	6259.56	**-49476**
GF + KLj	11031.16	-71242	5454.17	-49422

Table 3.7: Comparison of performance of different algorithms with different initializations on very large graphs of social networks.

3.4.4 Social Networks Analysis: *epinions* and *slashdot*

Toward the clustering of social networks, we consider the directed graphs called *epinions* and *slashdot* (Leskovec *et al.*, 2010). In these directed graphs $H = (V, A)$ with edge costs $c' : A \to \{-1, 1\}$, each node represents a person and every edge $(v, w) = a \in A$ encodes that a person v either trusts ($c'_a = 1$) or distrusts ($c'_a = -1$) a person w. We define a multicut problem w.r.t. these signed directed graphs by firstly removing self-loops from H and by then defining a cost matrix C w.r.t. the cost matrix C' as $C := \operatorname{sgn}(C' + C'^T)$.

Results for those local search algorithms described in Section 3.3 that are practical for these large instances of the multicut problem are shown in Tab. 3.7 and Fig. 3.2. To make use of the feasible solution with the lowest cost that we find, we report in Fig. 3.2 the cumulative distribution of cluster sizes, i.e. the number of clusters $N(x)$ of size less than or equal to x. For *epinions*, we observe one large cluster of 93984 people, 6704 clusters of a size between 2 to 1053, and 18842 totally disconnected individuals. This is consistent with the fraction of edges indicating trust, which is 82.9%. For *slashdot*, we observe one large cluster of 68993 people, 862 clusters of a size between 2 to 692, and 8246 totally disconnected individuals. This is consistent with the fraction of edges indicating trust, which is 76.1%.

3.5 CONCLUSION

We have applied four local search algorithms to a collection of instances of the multicut problem for the tasks of image segmentation, hand-written digit classification and social network analysis. A combination of greedy initialization (by additive edge contraction or fixation) and the Kernihan-Lin Algorithm with joins has proven practical for all instances. It has allowed us to cluster the MNIST test set of hand-written digits into 10 significant clusters with an associated misclassification rate of 0.82%. Moreover, it has allowed us to estimate the distribution of cluster sizes form social network graphs. The practicality of these local search algorithms encourages further applications of the multicut problem, especially in settings where

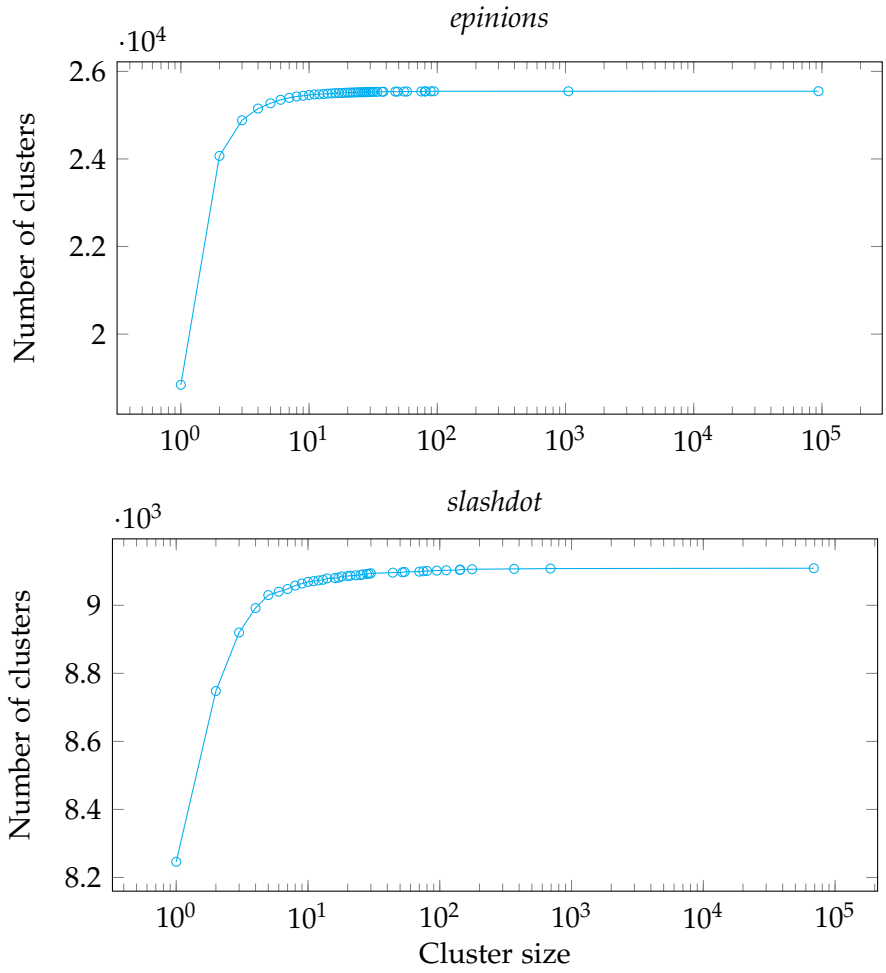


Figure 3.2: Cumulative distribution of cluster sizes in social networks, as defined by the best obtained solutions.

the number of clusters is not known a priori and needs to be estimated from data.

4 INTERACTIVE MULTICUT VIDEO SEGMENTATION

Contents

Video segmentation requires separating foreground from background, but the general problem extends to more complicated scene segmentations of different objects and their multiple parts. We develop a new approach to interactive multi-label video segmentation where many objects are segmented simultaneously with consistent spatio-temporal boundaries, based on intuitive multi-colored brush scribbles. From these scribbles, we derive constraints to define a multicut problem, that we can solve efficiently using the heuristics introduced in Chapter 2. These algorithms are fast enough to allow for a fully interactive video segmentation scenario. In addition, we developed a client-server web-based application, so that all the computations are carried out not on the user's machine, also allowing collaborative editing. As our solution generalizes typical binary segmentation tasks, while also improving efficiency in multi-label tasks, our work shows the promise of multicuts for interactive video segmentation.

4.1 INTRODUCTION

Video segmentation is a common task both in computer vision and computer graphics. In media production, it is key to composite or remove elements in a scene, or to apply effects to different objects independently. In computer vision, video segmentation is key to creating 'ground truth' sequences from which to train models for object classification. Professional visual effects (VFX) tools like SilhouetteFX, LLC. (2004–2018) and Mocha Imagineer Systems, Ltd. (2014) allow artists to segment a foreground from uncontrolled backgrounds; however, these often require hours of work for a few seconds of footage. Acceptable results are achievable in simpler tools, but these can still take time, e.g. Adobe After Effects Rotobrush (Bai *et al.*, 2009), or be too inflexible, e.g. Power Matte Digital Film Tools, LLC..

We aim to make a fast video segmentation tool which requires just a few user interactions, but which is still able to produce acceptable results for applications such as consumer VFX work or 'ground truth' labeling of video databases. One avenue for potential gains is to consider scenes where multiple objects or regions need to be segmented. Instead of many binary segmentations, it is possible to take an approach so that user interactions inform the segmentation of all parts simultaneously—producing a multicut.

To this end, we introduce an approach to quickly solve the video multicut problem with user constraints, and so create a fast interactive video segmentation system. We begin with a supervoxel over-segmentation of the video. This over-segmentation forms a non-planar graph in spacetime, which must be cut into meaningful segments. The user interactively scribbles onto the video with different colored brushes, one color per desired segment. From these scribbles, we resolve split and merge constraints to inform the solution to the multicut problem. If the constraints conflict with the underlying supervoxel graph, i.e., the over-segmentation is too coarse, then we automatically refine the graph to resolve the constraints. In principle, many different multicuts may exist which meet these constraints. To pick a good multicut, we assign costs to cutting graph edges by dynamically clustering video features based on user scribbles. With this approach, each added user stroke takes $\approx$1 second to resolve, and the solution automatically generalizes to segmenting any number of regions.

We perform a comparison of time and quality on 4 sequences against two expert users who took 10–12 hours with professional tools, and against novices who used RotoBrush. Our method was 50x faster than the experts, though with less output quality. Against Rotobrush, our tool was 3x faster with comparable quality. Based on these results, we believe our approach holds a promise as an interactive video segmentation tool for consumers, and as a quick tool to label 'ground truth' datasets for computer vision.

We state our contributions explicitly:

1. A formulation of the multicut problem that is appropriate for *interactive* video segmentation, with an efficient way to provide fast feedback to users.

(a) Supervoxels **(b)** User Interactions **(c)** Segmentation

Figure 4.1: Starting with a supervoxel segmentation of the input **(a)**, the user scribbles different regions to segment with multi-colored brushes **(b)**. These interactive constraints quickly cut the video into any number of segments, providing a fast and simple way to accomplish multi-part segmentation tasks **(c)**.

2. A system for interactive video segmentation which demonstrates the speed and quality achievable with a supervoxel-based multicut solution.

4.2 RELATED WORK

Interactive segmentation of videos into spatially and temporally connected components (Fig. 4.1) is a prerequisite for many video post-production applications, such as matting (Chuang *et al.*, 2002), color transfer (Bonneel *et al.*, 2013), or white balancing (Hsu *et al.*, 2008). It is most often performed as a binary segmentation task where a foreground object is extracted from the background (Wang *et al.*, 2005; Bai and Sapiro, 2007; Zhong *et al.*, 2012; Fan *et al.*, 2015; Lu *et al.*, 2016; Li *et al.*, 2016). Prominent examples include the consumer Rotobrush tool in Adobe After Effects, which is based on the work of (Bai *et al.*, 2009), and professional commercial tools such as Silhouette, Mocha, and Power Matte. These tools do not guaranteed to create consistent segment boundaries between different objects.

In the computer vision literature, consistent boundaries can be guaranteed by multi-label segmentations (Grundmann *et al.*, 2010; Ochs and Brox, 2011; Galasso *et al.*, 2013). However, existing work in this area has focused on fully automatic segmentation (Jain *et al.*, 2017). While automatic approaches will work some of the time, it is currently very difficult to reliably generate semantic-level segmentations. To complete the task of disambiguating regions and creating meaningful labels, a user is required, and so interactive segmentation tools are needed.

The closest approach to ours, that also uses user strokes for foreground vs. background segmentation, is Shankar Nagaraja *et al.* (2015); they pre-compute point trajectories that propagate user input in time. An easy and intuitive input from the user in the form of simple clicks is used in (Wang *et al.*, 2014a; Bearman *et al.*, 2016). A separate line of work (Khoreva *et al.*, 2017; Perazzi *et al.*, 2017; Jampani *et al.*, 2017; Caelles *et al.*, 2017) get as input the exact mask of an object in the first frame and then propagates it throughout the video. Cheng *et al.* (2017) jointly estimates optical flow and segmentation of one object in the video.

4.3 METHOD AND SYSTEM

A block diagram of our system is shown in Fig. 4.2. To begin, we preprocess the video to compute both supervoxels and an associated feature vector per supervoxel which describes its appearance. Then, interactively, the user employs a 'paint by numbers' approach with multiple colored brushes to scribble coarsely and sparsely over the video. These multiple labels are automatically resolved into a set of constraints which specify which supervoxels belong to the same segment and which belong to different segments. Should any constraints conflict with the underlying supervoxels, we automatically refine the supervoxel over-segmentation, e.g., to capture fine details.

Then, we solve the constrained multicut problem and find a good segmentation. To provide graph edge weights with which to score potential multicut solutions, we cluster all supervoxels by training on the provided user labels and appearance features. Once a multicut solution is optimized, the presented segmentation respects both the user constraints and the video appearance. Finally, after the user is satisfied, we matte the result.

To make this model applicable to video, there are two major challenges to overcome. The first challenge is making it fast while maintaining segmentation accuracy. Multicuts are typically computed on the graph of a supervoxel over-segmentation. Many supervoxels are usually used through a fine over-segmentation, but this implies a large graph which is slow to solve. In employing a very fast heuristic—KLj (Chapter 2)—we can start with many supervoxels (5,000) to increase output accuracy and still maintain interactivity with $\approx$1 second response.

The second challenge is implementing constraints to split different brush strokes. Whenever the user wants two pixels to belong to different segments, there must exist at least one segment boundary on *any* path between these two pixels. This is a large space, and many constraints must be added to the optimization problem to constrain the set of feasible segmentations. As this is expensive, we propose a solution to this problem in Sec. 4.3.2.

4.3.1 Multicuts of Supervoxel Graphs

We use the multicut framework to model the segmentation problem. Namely, we define a graph $G = (V, E)$ that connects all the supervoxels and aim at finding a 01-edge labeling y that gives an optimal decomposition. As not every possible 01-edge labeling defines a proper decomposition, we need to enforce it via the following set of constraints:

$$\forall C \in \text{cycles}(G) \forall e \in C \quad y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \tag{4.1}$$

User Interactions Users provides evidence about the correct segmentation via scribbling over each intended segment with a different colored brush. From these scribbles, we automatically extract a set of additional constraints:

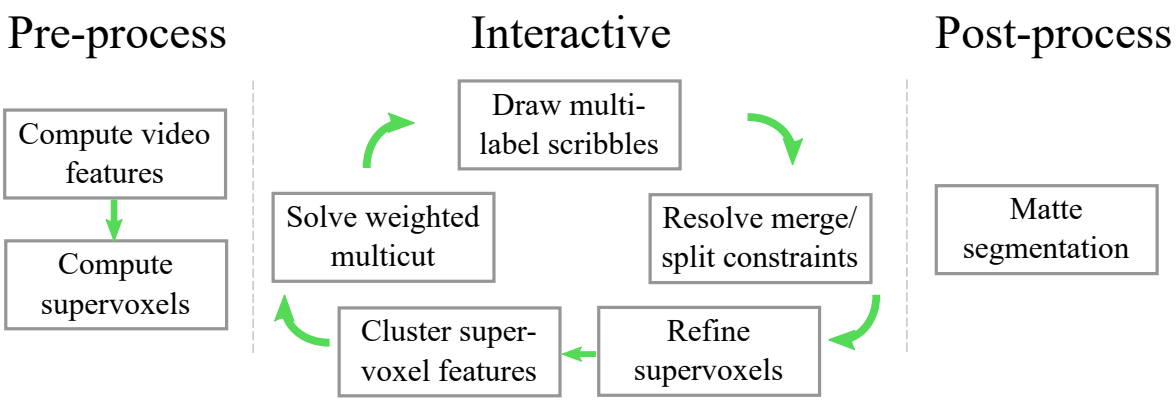


Figure 4.2: After an initial pre-process to compute superpixels and aggregated features, the user interactively refines the segmentation with multiple scribbles. Finally, the solution is matted.

- A set $E_0 \subseteq E$ of edges known to connect adjacent supervoxels of the same segment, i.e., supervoxels along a single brush stroke.
- A set $E_1 \subseteq E$ of edges known to connect adjacent supervoxels of different segments.
- A set $F_1 \subseteq (V \times V) \setminus E$ of pairs of non-adjacent supervoxels known to be in different segments, i.e., supervoxels along different colored brush strokes.

This evidence constrains the set of feasible segmentations.

We can write these constraints formally as linear inequalities:

$$\forall e \in E_0 \quad y_e = 0 \tag{4.2}$$
$$\forall e \in E_1 \quad y_e = 1 \tag{4.3}$$
$$\forall \{v, w\} \in F_1 \quad \forall \pi \in \text{path}_G(v, w) \quad 1 \leq \sum_{e \in \pi} y_e \tag{4.4}$$

where path (v, w) denotes the set of all paths in the graph (G) from node v to node w. The inequalities in (4.4) state that no path exists in the graph that connects supervoxels v and w. Constraining non-adjacent nodes to the same segment is possible (Nowozin and Lampert, 2010), but is a substantial complication that is beyond the scope of this work. This means that, in our solutions, disconnected but similarly-brush-colored segments are assigned different segment labels in the result. In practice, this is not a problem as we simply relabel the result segments based on their intersecting input brush stroke colors. Note that (4.4) is of the same form as (4.1), with $x_e = 1$, and this facilitates convenient user interactions which will be described in Section 4.3.5.

4.3.2 Problem

From the set of all possible segmentations which satisfy these constraints, we wish to favor segmentations in which similar appearing supervoxels belong to the same segments, and dissimilar supervoxels belong to different segments. To this end, we model the similarity between supervoxels by the edge features described below. Then, the segmentation problem is posed as a minimization over probabilities that adjacent supervoxels belong to different segments.

The probability p_e that an edge between adjacent supervoxels $e \in E$ belongs to a segment boundary with $x_e = 1$ can be modeled by any probabilistic model for classification. We convert the vector of probabilities $p \in [0,1]^{|E|}$ into a cost vector $c \in \mathbb{R}^{|E|}$ by $c_e = \log\left((1-p_e)/p_e\right)$. The problem of inferring the most probable segmentation x given the user-defined constraints (4.2)–(4.4) is an integer linear program (ILP):

$$\min_{y \in \{0,1\}^E} \quad \sum_{e \in E} c_e y_e \tag{4.5}$$

$$\text{subject to} \quad (4.1),\ (4.2)\text{–}(4.4) \tag{4.6}$$

This formulation is similar to the minimum cost multicut problem (Def. 2) with the exception of additional user-specific constraints (4.2)–(4.4).

We are aiming at creating a real-time system, therefore the speed of inference is very important for us. We want to use the efficient KLj heuristic, however, it cannot handle these additional constraints. We circumvent this problem by assigning very large positive or negative weights to edges which connect supervoxels that, through resolving constraints derived from the multi-colored brush strokes, were marked by a user as belonging to the same or different segments correspondingly. This restricts the search space of the heuristics to the solutions that respect the user input constraints, because the solutions that violate them will have a much higher objective values. One benefit of this approach is that if a user specifies contradicting constraints, we can still arrive at a solution that is feasible w.r.t. the multicut constraints (4.1), i.e., a valid segmentation. This makes our system more stable under arbitrary user interactions.

Edge Features and Classification The cost vector c in (4.5) describes the similarity between neighboring supervoxels. To describe this similarity, we compute the mean and variance of per pixel grayscale intensity, sum of the squared horizontal, vertical and temporal gradient magnitudes, Lab color, HSV color, and bi-directional optical flow (Liu, 2009). This results in a 24-d feature vector of means and variances per supervoxel.

These costs c depend on the probability that adjacent supervoxels belong to different segments. To predict these probabilities, we use the user labels to train a random forest classifier with 64 trees with features for every edge between adjacent supervoxels, where the element-wise distance between the feature vectors represents the overall distance. In principle, it is possible to pre-train an edge classifier from

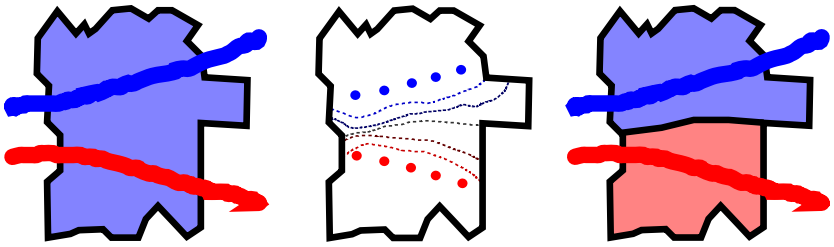


Figure 4.3: *Left:* A blue superpixel is in conflict with a new red scribble. *Middle:* To automatically resolve the conflict, we split the supervoxel in a 3D region-growing process (dotted lines converging on center line), where samples along the brush strokes act as seeds. *Right:* The supervoxel is split and the constraint resolved.

a database of videos. This could be useful for domain-specific videos, such as for sports videos. However, in general, retraining the classifier for every video is the most flexible approach to cope with unseen videos of widely-varying appearance.

4.3.3 Supervoxels

As we solve for a solution on a graph of supervoxels, we rely on these supervoxels for temporal consistency. Given that, an initial over-segmentation should not be too coarse so as to avoid missing important contours, and should not be too fine so as to remain computationally tractable. We favored fast feedback, and refine the supervoxel segmentation whenever user constraints deem it necessary. We compute SLIC supervoxels (Achanta *et al.*, 2012) which we re-label into connected components. Moreover, SLIC sometimes produces small cluttered regions. We remove regions smaller than 500 pixels per frame and relabel their pixels with that of their nearest neighbors. This provides a good trade-off between speed and quality among a set of tested alternatives (Sharon *et al.*, 2006; Felzenszwalb and Huttenlocher, 2004; Roerdink and Meijster, 2000).

Refining Supervoxels Our fast solver allows us to use many supervoxels, but this still may not be enough to capture fine details. This manifests as conflicting constraints in the system of equations, e.g., two different scribble colors in the same superpixel. We detect such conflicts and automatically refine the supervoxel segmentation to match the constraints (Fig. 4.3). Coordinates along the conflicting scribble lines are used as seeds to a 3D region growing algorithm, which uses the sum of squared gradient magnitudes in horizontal, vertical, and temporal directions as features. This refinement takes less than a second. This allows the user to quickly make coarse segmentations using the discovered supervoxel edges, but also to specify new segmentation edges in arbitrary locations by drawing conflicting brush strokes very close together.

	Ours Novice	RotoBrush Novice	Silhouette Expert 1	mocha Pro Expert 2
Video 1	11.29	27.61	405.75	174.25
Video 2	5.48	35.88	33.10	35.36
Video 3	8.99	15.31	103.50	147.34
Video 4	13.04	31.68	180.21	219.89
Mean	9.70	27.63	180.64	144.21

Table 4.1: Completion time in minutes. Participants completed the multi-label segmentation tasks faster with our tool than with Rotobrush (RB).

4.3.4 Matting

After segmentation, we compute a matting solution to refine our supervoxel edges and to cope with transparency. We use a spatio-temporal cross-bilateral filter (Tomasi and Manduchi, 1998) with fast bilateral grids (Chen *et al.*, 2007). This filter takes two video volumes as input: a binary mask, and per-pixel weights to modulate the Gaussian kernel used in filtering. We use a grayscale version of the video as our weights. Each label is filtered separately, then the resulting mattes are normalized so that they sum to 1. This solution can be parallelized across labels. Other state-of-the-art matting solutions would improve our results further (Bai *et al.*, 2011; Shahrian *et al.*, 2014).

4.3.5 Web Implementation and Collaborative Editing

Our system is built as a web application: All core algorithmic components are implemented in C++, then wrapped through Cython and served from Tornado Web Server. The client is written in JavaScript, with WebGL rendering. First, the video and supervoxel segmentation are sent from the server and cached on the client. During interaction, only the constraints are sent to the server, with the segmentation returned to the client. Updates to the supervoxels are sent to the client as incremental changes. This approach allows for new applications such as collaborative segmentation on difficult or long sequences, or for segmenting large video databases as the task can be crowdsourced and completed on light-weight terminals.

For interaction, the user selects a brush, scribbles, and the segmentation is returned. The user iterates until they are satisfied. There are two optional brushes: the first turns off retraining the classifier, for use when the clustering is well-defined and should not be swayed by newly-marked regions; the second is a pure non-constraint-forming scribble, to paint the segmentation labels of supervoxels directly. In practice, the novice user need not use these expert features.

4.4 EXPERIMENTS

For a 1280×720×150 video, on an Intel Xeon E5-2609 CPU, our system spends 12 min in unoptimized pre-processing: 10 min to compute edge features, then 2 min to compute supervoxels and aggregate features. During interaction, with 5k supervoxels and 25 user constraints, training and classifying takes 0.3 s, with the multicut problem solved in a further 0.25 s. After network transfer, the total response time to interaction is under one second. As an extreme case, with 25k supervoxels, training and classifying takes 1.4 s, with the multicut problem solved in 5 s.

To assess the accuracy and efficiency of our approach, we compare our tool with After Effects Rotobrush which is used in a repeated binary segmentation fashion by a novice user. For comparison, we also hired two professional VFX artists to segment our videos with professional tools of their choice (namely Silhouette and mocha Pro). In total, four videos were segmented with different levels of difficulty (Fig. 4.4). Completion times are shown in Table 4.1. We can see from the expert timings that high-quality multi-label segmentation is a laborious task. Ideally, one would re-use the shared edges between different segments. While this is possible in existing foreground/background segmentation tools, our approach solves edges jointly and guarantees consistent non-overlapping boundaries. Our novice user was able to perform multi-label segmentation tasks more quickly with our tool than with After Effects Rotobrush, and to a similar segmentation quality (Fig. 4.5). However, our automatic pre-process takes longer. While we only tested four videos, these results show some promise for the generality of the tool.

4.5 DISCUSSION

Pro-level tools can achieve high edge accuracy given enough time, and this is where our tool cannot compete yet. In principle, the system of arbitrary supervoxel refinement (Sec. 4.3.3) could be augmented to take input not from scribbles but from parametric curves, similar to Silhouette. This would provide the control necessary to describe arbitrary edges with more control, though it requires future work. That said, our current results are suitable for many consumer-level editing tasks, and the speed and multi-label nature of the tool makes it suitable for labeling large databases of videos to generate 'ground truth' for training computer vision models.

One limitation with re-learning a classifier at every new stroke is that the resulting segmentation can change in unexpected ways, e.g., an edge in the video which has similar content in appearance on either side may change segment as new strokes are added. This typically manifests itself in the very early stages of segmenting a video, when the classification boundaries have very few user labels to inform them. One solution is to simply educate the user to trust the process as it is a part of the learning system; another is to use a static pre-trained classifier, though this may be inflexible to new 'unseen' videos which do not match the training database.

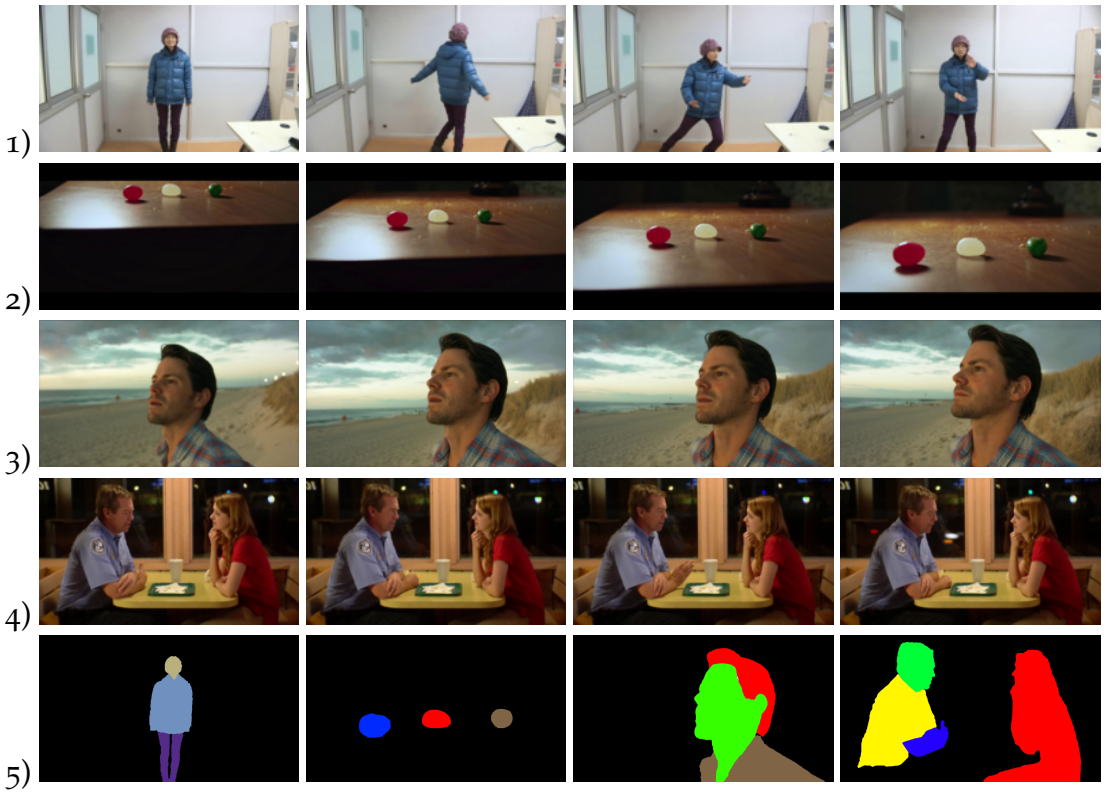


Figure 4.4: The four different videos in the comparison. *1)* 'Kung Fu', 960×540×270. *2)* 'The Secret Number', 1280×720×210. *3)* 'Sunset', 1280×960×60. *4)* 'Time Expired', 1280×960×150. *5)* The required segments for each sequence.

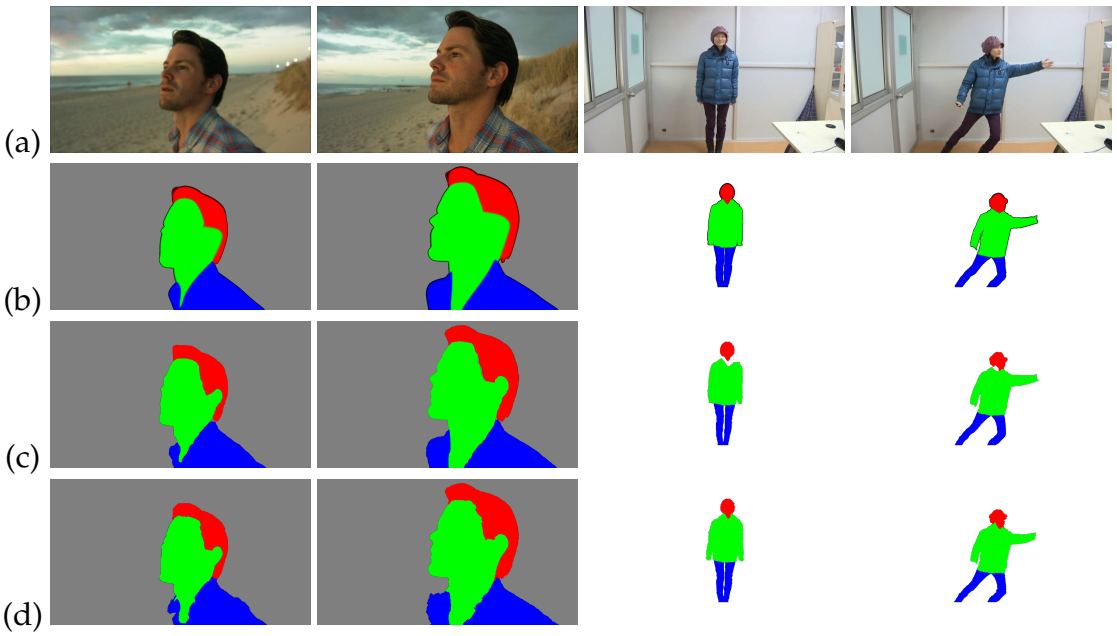


Figure 4.5: (a) 'Sunset' (video 3) and 'Kung Fu' (video 1). (b) Segmentation from professional VFX artist using Silhouette (SI). (c) Novice Rotobrush (RB) result. (d) Novice result with our tool.

4.6 CONCLUSION

We propose an interactive multicut video segmentation system. We solve two challenges with creating an interactive video multicut tool: making it fast through a state-of-the-art multicut solver, and integrating user constraints into the problem. Our system produces similar quality to current prosumer tools such as After Effects Rotobrush, while being faster to use. Overall, these results show the promise of interactive multicut video segmentation, easing the way for more advanced video editing and database labeling.

MORAL LINEAGE TRACING 5

Contents

Lineage tracing, the tracking of living cells as they move and divide, is a central problem in biological image analysis. Solutions, called lineage forests, are key to understanding how the structure of multicellular organisms emerges. We propose an integer linear program (ILP) whose feasible solutions define, for every image in a sequence, a decomposition into cells (segmentation) and, across images, a lineage forest of cells (tracing). In this ILP, path-cut inequalities enforce the morality of lineages, i.e., the constraint that cells do not merge. To find feasible solutions of this NP-hard problem, with certified bounds to the global optimum, we define efficient separation procedures and apply these as part of a branch-and-cut algorithm. To show the effectiveness of this approach, we analyze feasible solutions for real microscopy data in terms of bounds and run-time, and by their weighted edit distance to lineage forests traced by humans.

5.1 INTRODUCTION

Phenomenal progress in microscopy allows biologists to image large numbers of living cells as they move and divide (Keller *et al.*, 2010; Tomer *et al.*, 2012). Such observations are essential in developmental biology for studying embryogenesis and tissue formation (Keller *et al.*, 2008; Khairy and Keller, 2011; Megason and Fraser, 2007). Consequently, the tracing of cells and their lineages in sequences of images has become a central problem in biological image analysis (Amat and Keller, 2013; Amat *et al.*, 2014; Keller, 2013).

The lineage tracing problem consists of two sub-problems. The first sub-problem is to identify the cells in every individual image. The second sub-problem is to connect every cell identified in an image to the same cell and descendant cells identified in subsequent images. A joint solution of both sub-problems is a set of pairwise disjoint lineage trees (depicted in Fig. 5.1, in red and green) whose nodes are cells.

The first sub-problem is an image decomposition problem: If every pixel shows a part of a cell and no pixel shows a background, the objective is to decompose the pixel grid graph of the image into precisely one component per cell. If pixels potentially show background, the objective is to jointly select and decompose a subgraph of the pixel grid graph such that there is precisely one component for each cell and no component for the background.

The second sub-problem is a cell tracking problem: The objective is to connect every cell detected in one image to the same cell and descendant cells identified in subsequent images. A joint solution of both sub-problems is constrained by prior knowledge. In particular, every cell has at most one direct progenitor cell, *i.e.*, cells do not merge. Moreover, no cell splits into more than two cells at once. Yet, a cell can appear without a direct progenitor cell when entering the field of view, and a cell can disappear when dying or leaving the field of view. Finally, it can appear as if a cell was dividing into more than two cells at once if the temporal resolution is too low to separate consecutive divisions.

It is understood that errors in the image decomposition make it harder to reconstruct the true lineage forest. Attempts at reconstructing the lineage forest can help to avoid such errors. Thus, we state a joint optimization problem whose feasible solutions define, for every image, a decomposition (segmentation) into cells and, across images, a lineage forest. Unlike in prior work, we do not constrain the set of decompositions, except by contracting pixels to superpixels.

5.2 RELATED WORK

The image decomposition problem has been tackled by various abstractions in the form of optimization problems including the minimum cost spanning forest problem, *i.e.*, agglomerative clustering (Adams and Bischof, 1994), balanced cut problems, *i.e.*, spectral clustering (Arbeláez *et al.*, 2011; Shi and Malik, 2000), and the minimum

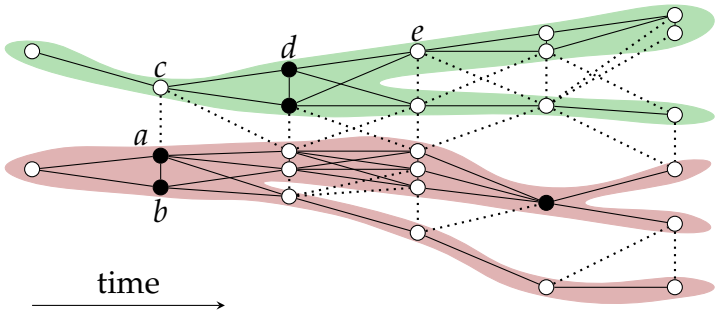


Figure 5.1: Given a sequence of images, taken at consecutive points in time, and given a decomposition of each image into *cell fragments* (depicted above as nodes), the objective of lineage tracing is to join fragments of the same cell within and across images, *e.g.* $\{a, b\}$ and $\{c, d\}$, and to join fragments of descendant cells across images, *e.g.* $\{d, e\}$. Joins (cuts) are depicted as solid (dotted) lines. Fragments of dividing cells are depicted as black nodes.

cost multicut problem, *i.e.*, correlation clustering (Kim *et al.*, 2014). We build on its formulation as a minimum cost multicut problem, an optimization problem studied in (Chopra and Rao, 1993; Deza and Laurent, 1997), which is NP-hard (Bansal *et al.*, 2004; Demaine *et al.*, 2006) and has been used for image segmentation in (Alush and Goldberger, 2012; Andres *et al.*, 2011, 2012b, 2013; Bagon and Galun, 2011; Beier *et al.*, 2015, 2014; Kappes *et al.*, 2011, 2016, 2015b; Kim *et al.*, 2014; Yarkony *et al.*, 2015, 2012).

The lineage forest reconstruction problem has been cast as an optimization problem in (Funke *et al.*, 2012; Jug *et al.*, 2014; Kausler *et al.*, 2012; Padfield *et al.*, 2011; Schiegg *et al.*, 2013; Türetken *et al.*, 2013a,b, 2012, 2011; Rempfler *et al.*, 2018). If cells neither die nor enter or leave the field of view and if one drops the constraint that cells split into at most two direct descendant cells, *e.g.* because temporal image resolution is too low to separate divisions, the problem can be formulated as a minimum cost k disjoint arborescence problem (Schrijver, 2003, Section 53.9), as shown in (Türetken *et al.*, 2012, 2011). Here, k is the number of cells visible in the first image. This problem can be solved in strongly polynomial time (Edmonds, 1975). With the additional constraint that cells split into at most two descendant cells, the problem becomes NP-hard, and so do generalizations (Funke *et al.*, 2012; Jug *et al.*, 2014; Kausler *et al.*, 2012; Padfield *et al.*, 2011; Schiegg *et al.*, 2013; Türetken *et al.*, 2013a,b) that model, *e.g.*, the (dis)appearance of cells.

One lineage tracing approach (Kausler *et al.*, 2012) copes with imperfect decompositions by over-segmenting individual images. This guarantees that every cell is represented by at least one component and that every component represents at most one cell. Advantageous there is the fact that the true lineage forest is represented by at least one set of disjoint arborescences. Disadvantageous is the loss of robustness: For the true decomposition, every component belongs to precisely one arborescence and thus, every error in the set of disjoint arborescences implies at least a second error. This renders solutions robust to perturbations of the objective function. For an

over-segmentation, this property is lost. Another disadvantage is the fact that the number of progenitor cells is not determined by the number of components of the first image. Over-estimates result in excessive arborescences that typically conflict with correct ones. Under-estimates result in a loss of lineage trees. As in (Kausler *et al.*, 2012), we consider an over-decomposition of each image into cell fragments. In contrast to (Kausler *et al.*, 2012) where each node of a lineage forest is a single representative fragment, each node in the lineage forests we consider is a clusters of fragments. This idea of clustering instead of selecting has been used in (Tang *et al.*, 2015) to track multiple people in a video sequence. The optimization problem defined in (Tang *et al.*, 2015) is a hybrid of a minimum cost multicut problem and a disjoint path problem. The optimization problem we propose here is a hybrid of a minimum cost multicut problem and a disjoint arborescence problem.

Two techniques have been proposed to deal with over and under-decomposition simultaneously: The first (Schiegg *et al.*, 2013) is to allow single image components to represent multiple cells and thus be part of multiple lineages. This relaxation of the disjointness constraint of the arborescence problem introduces additional feasible solutions that can represent the true lineage forest even in the presence of under-decomposition. The same idea is used in (Türetken *et al.*, 2013b) for the reconstruction of curvilinear structures and in (Wang *et al.*, 2014b) for the tracking of objects in containers. The second technique (Funke *et al.*, 2012; Jug *et al.*, 2014; Schiegg *et al.*, 2014) considers a hierarchy of alternative decompositions and casts lineage tracing as an optimization problem whose feasible solutions select and connect components from the hierarchy. Constraints guarantee that selected components are mutually consistent and consistent with a set of disjoint lineages. As in (Funke *et al.*, 2012; Jug *et al.*, 2014), the feasible solutions we propose define, for every image, a decomposition into cells and, across images, a lineage forest. In contrast to prior work, we do not constrain the set of decompositions, except by contracting pixels to superpixels. We compare our experimental results to a state-of-the-art software system for lineage tracing (Aigouy *et al.*, 2010).

Unlike in the work discussed above, feasible lineages and costs of feasible lineages can be defined recursively, as in particle filtering. *C.f.* (Chenouard *et al.*, 2014) for a recent comprehensive comparison and (Amat *et al.*, 2014) for a recent application to lineage tracing.

5.3 PROBLEM

In this section, we cast lineage tracing as an optimization problem that we call *moral lineage tracing*. In Section 5.3.1, we define the set of feasible solutions. In Section 5.3.2, we define the objective function and optimization problem.

5.3.1 Feasible Set

In order to encode a combinatorial number of feasible lineage forests, we define a *hypothesis graph*. In a hypothesis graph, every node corresponds to one superpixel of one image in a sequence and is referred to as a *cell fragment* (Def. 4). In order to encode a single feasible lineage forest, we define a *lineage graph* (Def. 5). A lineage graph is a subgraph of the hypothesis graph that defines, within each image, a clustering of cell fragments into cells and, across images, a lineage forest of cells (Lemma 1). In order to state in the form of an ILP an optimization problem whose feasible solutions are lineage graphs, we identify the characteristic functions of lineage graphs with 01-labelings of the edges of the hypothesis graph that satisfy a system of linear inequalities (Lemma 2).

Definition 4 A *hypothesis graph* is a node-labeled graph[1] $G = (V, E, \tau)$ in which every edge $\{v, w\} \in E$ holds $|\tau(v) - \tau(w)| \leq 1$. Every $v \in V$ is called a *(cell) fragment*, and $\tau(v)$ is called its *time index*.

The intuition is this: For any distinct fragments $v, w \in V$ with $\tau(v) = \tau(w)$, the presence of the edge $\{v, w\} \in E$ indicates the possibility that v and w are fragments of the same cell. For any fragments $v, w \in V$ with $\tau(w) = \tau(v) + 1$, the presence of the edge $\{v, w\} \in E$ indicates the possibility that v and w are fragments of the same cell, observed at successive points in time, as well as the possibility that v is a fragment of a progenitor cell of the cell of w.

Next, we characterize those subgraphs of a hypothesis graph that we consider as feasible solutions. For clarity, we propose some notation: For every $t \in \mathbb{N}$, let $V_t := \tau^{-1}(t)$ the set of all fragments having the time index t. Let $G_t = (V_t, E_t)$ be the subgraph of G induced by V_t. Let $E_{t,t+1} := \{\{v, w\} \in E | v \in V_t \wedge w \in V_{t+1}\}$ be the set of those edges of G that connect a fragment v having the time index t to a fragment w having the time index $t + 1$. Let $G_t^+ = (V_t^+, E_t^+)$ be the subgraph of G induced by $V_t^+ := V_t \cup V_{t+1}$.

Definition 5 For every hypothesis graph $G = (V, E, \tau)$, a set $C \subseteq E$ is called a *lineage cut* of G, and $(V, \bar{C})$ with $\bar{C} := E \setminus C$ is called a *lineage (sub)graph* of G, iff the following conditions hold:

1. For every $t \in \mathbb{N}$, the set $E_t \cap C$ is a multicut[2] of G_t

2. For every $t \in \mathbb{N}$ and every $\{v, w\} \in E_{t,t+1} \cap C$, v and w are not connected by any path in the graph $(V_t^+, E_t^+ \cap \bar{C})$

3. For every $t \in \mathbb{N}$, any $v_t, w_t \in V_t$ and any $v_{t+1}, w_{t+1} \in V_{t+1}$ such that $\{v_t, v_{t+1}\} \in E \cap \bar{C}$ and $\{w_t, w_{t+1}\} \in E \cap \bar{C}$, and for any path in $(V, E_{t+1} \cap \bar{C})$ from v_{t+1} to w_{t+1}, there exists a path in $(V, E_t \cap \bar{C})$ from v_t to w_t.

[1] All graphs are assumed to be finite, simple and undirected. A node labeling of a graph (V, E) is a map $\tau : V \to \mathbb{N}$.

[2] A multicut of $G_t = (V_t, E_t)$ is a subset $M \subseteq E_t$ of edges such that, for every cycle Y in G_t: $|M \cap Y| \neq 1$ (Chopra and Rao, 1993).

If these conditions are satisfied then, for every $t \in \mathbb{N}$ and every non-empty, maximal connected subgraph (V'_t, E'_t) of $(V_t, E_t \cap \bar{C})$, its node set V'_t is called a *cell* at time index t.

A lineage cut and lineage subgraph are called *binary* iff, in addition to Conditions 1–3, it holds:

4. For every $t \in \mathbb{N}$, every cell $V'_t \subseteq V_t$ is connected in the lineage subgraph to at most two distinct cells at $t + 1$.

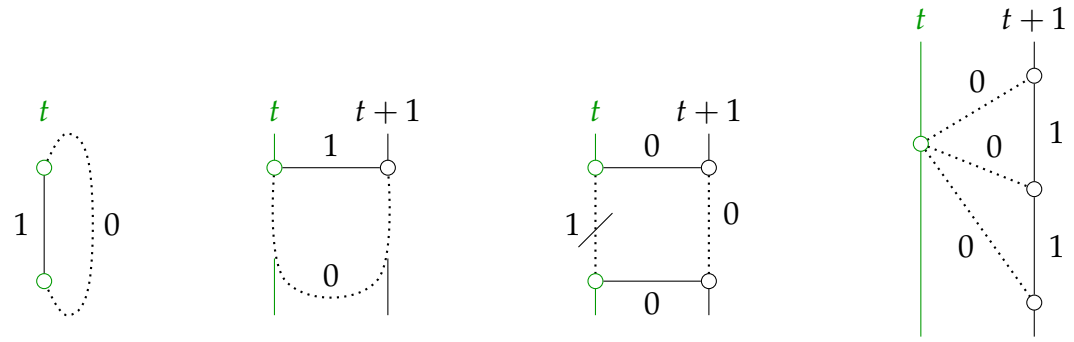
An intuition for Conditions 1–3 is offered by Lemma 1 and the proof.

Lemma 1 For every $t \in \mathbb{N}$, a lineage graph well-defines a decomposition of G_t whose components are the cells at time index t. Across time, a lineage graph well-defines a (lineage) forest of cells.

PROOF Condition 1 guarantees that every subgraph defining a cell is node-induced, i.e., for every $t \in \mathbb{N}$ and every $\{v, w\} \in E_t$: $\{v, w\} \in \bar{C}$ *iff* v and w are fragments of the same cell. Condition 2 guarantees, for every $t \in \mathbb{N}$, every cell V'_t at time index t, and every cell V'_{t+1} at time index $t + 1$ that either all edges of G between V'_t and V'_{t+1} are in $\bar{C}$, or none. Condition 3 guarantees, for every $t \in \mathbb{N}$ and every distinct cells V'_t, V''_t at time index t that these are not connected in $(V, E^+_t \cap \bar{C})$ to the same cell at time index $t + 1$. This guarentees, by induction, that V'_t, V''_t are not connected by any path in the graph $(V_{\geq t}, E_{\geq t} \cap \bar{C})$. This guarantees that distinct cells never merge. □

Lemma 2 For every hypothesis graph $G = (V, E, \tau)$ and every $x \in \{0, 1\}^E$, the set $x^{-1}(1)$ of edges labeled 1 is a lineage cut of G iff x satisfies the linear inequalities (5.1)–(5.3) stated below. It is sufficient in (5.1) to consider only chordless cycles. Moreover, the lineage cut is binary iff, in addition, x satisfies the linear inequality (5.4).

$$\forall t \in \mathbb{N}\ \forall Y \in \text{cycles}(G_t)\ \forall e \in Y :$$
$$x_e \leq \sum_{e' \in Y \setminus \{e\}} x_{e'} \tag{5.1}$$
$$\forall t \in \mathbb{N}\ \forall \{v, w\} \in E_{t,t+1}\ \forall P \in vw\text{-paths}(G^+_t) :$$
$$x_{vw} \leq \sum_{e \in P} x_e \tag{5.2}$$
$$\forall t \in \mathbb{N}\ \forall \{v_t, v_{t+1}\}, \{w_t, w_{t+1}\} \in E_{t,t+1}$$
$$\forall T \in v_t w_t\text{-cuts}(G_t)\ \forall P \in v_{t+1} w_{t+1}\text{-paths}(G_{t+1}) :$$
$$1 - \sum_{e \in T} (1 - x_e) \leq x_{v_t v_{t+1}} + x_{w_t w_{t+1}} + \sum_{e \in P} x_e \tag{5.3}$$
$$\forall t \in \mathbb{N}\ \forall v \in V_t\ \forall w_1, w_2, w_3 \in V_{t+1}$$
$$\forall P_1 \in vw_1\text{-paths}(G^+_t)\ \forall P_2 \in vw_2\text{-paths}(G^+_t)$$
$$\forall P_3 \in vw_3\text{-paths}(G^+_t)\ \forall C_{12} \in w_1 w_2\text{-cuts}(G_{t+1})$$
$$\forall C_{23} \in w_2 w_3\text{-cuts}(G_{t+1})\ \forall C_{13} \in w_1 w_3\text{-cuts}(G_{t+1}) :$$
$$1 - \sum_{e \in C_{12} \cup C_{23} \cup C_{13}} (1 - x_e) \leq \sum_{e \in P_1 \cup P_2 \cup P_3} x_e \tag{5.4}$$



(a) Space cycle **(b)** Spacetime cycle **(c)** Morality **(d)** Bifurcation

Figure 5.2: Depicted above are examples (graphs and 01-labelings of edges) in which inequalities (5.1)-(5.3) and, optionally, (5.4) are violated. **(a)** An inequality (5.1) is violated iff there exist $t \in \mathbb{N}$ and a cycle Y in G_t in which precisely one edge is labeled 1. **(b)** An inequality (5.2) is violated iff there exist $t \in \mathbb{N}$, an edge $\{v, w\} \in E_{t,t+1}$ labeled 1 and a path in G_t^+ connecting v to w in which all edges are labeled 0. **(c)** An inequality (5.3) is violated iff there exist $t \in T$ and nodes $v_t, w_t \in V_t$ and $v_{t+1}, w_{t+1} \in V_{t+1}$ connected by edges $\{v_t, v_{t+1}\}, \{w_t, w_{t+1}\} \in E_{t,t+1}$ labeled 0, such that v_t and w_t are separated by a cut in G_t with all edges labeled 1 and v_{t+1} and w_{t+1} are connected by a path in G_{t+1} with all edges labeled 0. **(d)** An inequality (5.4) is violated iff there exist $t \in T$ and a node $v \in V_t$ and nodes $w_1, w_2, w_3 \in V_{t+1}$ connected by edges $\{v, w_1\}, \{v, w_2\}, \{v, w_1\}, \{v, w_2\} \in E_{t,t+1}$ labeled 0, such that w_1, w_2, and w_3 are pairwise mutually separated by cuts in G_{t+1} with all edges labeled 1.

PROOF If Condition 1 in Def. 5 holds for a set $C \subseteq E$, then all inequalities (5.1) are satisfied by the $x \in \{0,1\}^E$ such that $x^{-1}(1) = C$. Otherwise, there would exist a $t \in \mathbb{N}$, a cycle Y of G_t and an $e \in Y$ such that $x_e = 1$ and $\forall e' \in Y \setminus \{e\}$: $x_{e'} = 0$. This implies $|Y \cap C| = 1$, in contradiction to the assumption that $C \cap E_t$ is a multicut of G_t. Conversely, if all inequalities (5.1) are satisfied by an $x \in \{0,1\}^E$, then $C := x^{-1}(1)$ satisfies Condition 1 in Def. 5. Otherwise, there would exist a $t \in \mathbb{N}$ for which $C \cap E_t$ is not a multicut of G_t. Thus, there would exist a cycle Y of G_t and an $e \in Y$ such that $Y \cap C = \{e\}$, by definition of a multicut. Hence, the inequality (5.1) for that cycle Y and that edge e of Y would be violated by x. The sufficiency of chordless cycles follows from (5.1) and is established, *e.g.*, in (Chopra and Rao, 1993).

If Condition 2 in Def. 5 holds for a set $C \subseteq E$, then all inequalities (5.2) are satisfied by the $x \in \{0,1\}^E$ such that $x^{-1}(1) = C$. Otherwise, there would exist $t \in \mathbb{N}$, $\{v, w\} \in E_{t,t+1}$ and a path $P \in vw\text{-paths}(G_t^+)$ such that $x_{vw} = 1$ and $x_P = 0$. From $x_{vw} = 1$ follows $\{v, w\} \in E_{t,t+1} \cap C$. From $x_P = 0$ follows that v and w are connected by P in $(V_t^+, E_t^+ \cap \bar{C})$. Both statements together contradict the assumption. Conversely, if all inequalities (5.2) are satisfied by an $x \in \{0,1\}^E$, then $C := x^{-1}(1)$ satisfies Condition 2 in Def. 5. Otherwise, there would exist $t \in \mathbb{N}$, $\{v, w\} \in E_{t,t+1} \cap C$ and a path $P \in vw\text{-paths}(V_t^+, E_t^+ \cap \bar{C})$. From this follows $x_{vw} = 1$ and $x_P = 0$, in contradiction to the assumption that (5.2) is satisfied.

If Condition 3 in Def. 5 holds for a set $C \subseteq E$, then all inequalities (5.3) are satisfied by the $x \in \{0,1\}^E$ such that $x^{-1}(1) = C$. Otherwise, there would exist $t \in \mathbb{N}$, $v_t, w_t \in V_t$, $v_{t+1}, w_{t+1} \in V_{t+1}$, a path $P \in v_{t+1}, w_{t+1}$-paths(G_{t+1}), and a cut $T \in v_t w_t$-cuts(G_t) such that $x_{v_t,v_{t+1}} = 0$ and $x_{w_t,w_{t+1}} = 0$ and $x_P = 0$ and $x_T = 1$. P witnesses the existence of a $v_{t+1}w_{t+1}$-path in $(V, E_{t+1} \cap \bar{C})$. The existence of T certifies the non-existence of a $v_t w_t$-path in $(V, E_t \cap \bar{C})$. Both statements together contradict the assumption. Conversely, if all inequalities (5.3) are satisfied by an $x \in \{0,1\}^E$, then $C := x^{-1}(1)$ satisfies Condition 3 in Def. 5. Otherwise, there would exist $t \in \mathbb{N}$, $v_t, w_t \in V_t$ and $v_{t+1}, w_{t+1} \in V_{t+1}$ such that $\{v, w\} \in E_{t,t+1} \cap \bar{C}$ and $\{v_{t+1}, w_{t+1}\} \in E_{t,t+1} \cap \bar{C}$ and such that there exist $P \in v_{t+1}w_{t+1}$-paths$(V_{t+1}, E_{t+1} \cap \bar{C})$ and $T \in v_t w_t$-cuts$(V_t, E_t \cap \bar{C})$. Hence, $x_{v_t,v_{t+1}} = 0$ and $x_{w_t,w_{t+1}} = 0$ and $x_P = 0$ and $x_T = 1$, in contradiction to the assumption that (5.3) is satisfied. □

Complementary to the proof, a discussion of (dis)connectedness w.r.t. a multicut can be found in (Horňáková *et al.*, 2017).

Here, the set of all $x \in \{0,1\}^E$ that satisfy (5.1)–(5.4) is denoted by X_G. Examples of violated inequalities are depicted in Fig. 5.2. Note that the path-cut inequalities (5.3) guarantee that any fragments of the same cell at time $t+1$ cannot be joined with fragments of distinct cells at time t, *i.e.*, morality is enforced.

5.3.2 Objective Function

Definition 6 A *priced hypothesis graph* is a tuple $(V, E, \tau, c, c^+, c^-)$ with (V, E, τ) a hypothesis graph, $c : E \to \mathbb{R}$ and $c^+, c^- : V \to \mathbb{R}_0^+$. For every $e \in E$, c_e is called the *cut cost* of e. For every $v \in V$, c_v^+ and c_v^- are called the *appearance* and *disappearance cost* of v, respectively.

The optimization problem we propose is defined below in the form of an ILP w.r.t. a priced hypothesis graph $G = (V, E, \tau, c, c^+, c^-)$. This ILP has the following properties: Every feasible solutions defines a lineage subgraph of G. For every $\{v, w\} = e \in E$, the objective function assigns the cost (or reward) c_e to all lineage graphs in which the cell fragments v and w belong to distinct cells. For every $t \in \mathbb{N}$ and every $v \in V_{t+1}$, the objective function assigns the (appearance) cost c_v^+ to all lineage graphs in which the fragment v is not joined with any fragment in V_t. For every $t \in \mathbb{N}$ and every $v \in V_t$, the objective function assigns the (disappearance) cost c_v^- to all lineage graphs in which the fragment v is not joined with any fragment in V_{t+1}.

Definition 7 For any priced hypothesis graph $G = (V, E, \tau, c, c^+, c^-)$, the instance of the *moral lineage tracing problem* w.r.t. G is the ILP in $x \in \{0,1\}^E$ and $x^+, x^- \in \{0,1\}^V$

written below.

$$\min_{x,x^+,x^-} \quad \sum_{e\in E} c_e x_e + \sum_{v\in V} c_v^+ x_v^+ + \sum_{v\in V} c_v^- x_v^- \tag{5.5}$$

$$\text{subject to} \quad x \in X_G \tag{5.6}$$

$$\forall t \in \mathbb{N}\ \forall v \in V_{t+1}\ \forall T \in V_t v\text{-cuts}(G_t^+) : \quad 1 - x_v^+ \leq \sum_{e\in T}(1 - x_e) \tag{5.7}$$

$$\forall t \in \mathbb{N}\ \forall v \in V_t\ \forall T \in vV_{t+1}\text{-cuts}(G_t^+) : \quad 1 - x_v^- \leq \sum_{e\in T}(1 - x_e) \tag{5.8}$$

If, in an inequality of (5.7), all edges in the cut T are labeled 1, then $x_v^+ = 1$. Otherwise, for every feasible solution x the same solution but with $x_v^+ := 0$ is not worse (as $0 \leq c_v^+$, by definition of c^+). Thus, a cost $c_v^+ \neq 0$ is payed iff fragment v appears at time $t+1$. The argument for (5.8) and disappearance is analogous.

5.4 ALGORITHMS

5.4.1 Efficient Separation Procedures

Below, we define, for each class of inequalities, (5.1)–(5.4), (5.7) and (5.8), an efficient separation procedure (Tab. 5.1) that takes any (x, x^+, x^-) as input. If any inequality is violated, it terminates and outputs at least one of these. If no inequality is violated, it terminates and outputs the empty set. We apply these procedures in a branch-and-cut algorithm described in the next section. We have also applied these procedures in the preparation of the experiments described in Section 5.5, to certify the well-definedness of lineages we traced manually.

To separate infeasible solutions by inequalities (5.1) for a given t, we label maximal subgraphs of G_t connected by edges labeled 0. Then, for every $\{v, w\} = e \in E_t$ with $x_e = 1$ and with v and w being in the same subgraph, we search for a shortest vw-path P in G_t such that $x_P = 0$, using breadth-first-search (BFS). If the path is chordless, we output the inequality defined by the cycle $P \cup \{e\}$ and e.

To separate infeasible solutions by inequalities (5.2) for a given t, we label maximal subgraphs of G_t^+ connected by edges labeled 0. Then, for every $\{v, w\} = e \in E_{t,t+1}$ with $x_e = 1$ and with v and w being in the same subgraph, we search for a shortest vw-path P in G_t^+ such that $x_P = 0$ using BFS. We output the inequality defined by the cycle $Y := P \cup \{e\}$ and $e \in Y$.

To separate infeasible solutions by inequalities (5.3) for a given t, we label maximal subgraphs of G_t connected by edges labeled 0. Then, for every pair $v, w \in V_t$ of nodes with different labels, we use BFS to search for (i) a shortest vw-path P in $(V_{t+1}, E_{t,t+1} \cup E_{t+1})$ such that $x_P = 0$, and (ii) a vw-cut T in G_t such that $x_T = 1$. We output the inequality defined by P and T.

Constraint	ILP (branch-and-cut)	LP (cutting-plane)
Space	$O(\|E_t\|^2)$	$O(\|E_t\|^2 \log \|V_t\|)$
Spacetime	$O(\|E_{t,t+1}\|\|E_t^+\|)$	$O(\|E_{t,t+1}\|\|E_t^+\| \log \|V_t^+\|)$
Morality	$O(\|V_t\|^2\|E_t^+\|)$	$O(\|V_t\|^2(m(\|V_t\|,\|E_t\|)$ $+\|E_{t+1}\| \log \|V_{t+1}\|))$
Termination	$O(\|V_t\| + \|E_t^+\|)$	$O(\|V_t\| m(\|V_t^+\|,\|E_t^+\|))$
Birth	$O(\|V_{t+1}\| + \|E_t^+\|)$	$O(\|V_{t+1}\| m(\|V_t^+\|,\|E_t^+\|))$
Bifurcation	$O(\|E\| \log \|V\| + \|V_t\|$ $+\|E_t^+\| \log \|V_t^+\|)$	$O(\|V_t\|\|V_{t+1}\|^3(m(\|V_{t+1}\|,\|E_{t+1}\|)$ $+\|E_t^+\| \log \|V_t^+\|))$

Table 5.1: Worst case time complexity of the separation procedures we implement for integral points (ILP) and fractional points (LP). Here, $m(|V|,|E|)$ denotes the worst case time complexity of a maximum st-flow algorithm for a graph (V, E).

To separate infeasible solutions by inequalities (5.4) for a given t, we label maximal subgraphs of G_t and G_{t+1}, resp., connected by edges labeled 0. From every $v \in V_t$, we start a BFS in the subgraph of G_t^+ whose edges are labeled 0. If nodes $w_1, w_2, w_3 \in V_{t+1}$ of three distinct components of G_{t+1} are reached, we output the inequality defined by the boundaries of these components and by paths from v to w_j.

To separate infeasible solutions by inequalities (5.7) for a given t, we start a BFS from every $v \in V_{t+1}$ in the subgraph of G_t^+ whose edges are labeled 0. We either find a vertex $w \in V_t$ (no violation) or a cut $T \in V_t v\text{-cuts}(G_t^+)$ which separates v from V_t. In the latter case, we output the inequality defined by the vertex v and the cut T. The separation of infeasible solutions by inequalities (5.8) is analogous, in the opposite order of time indices.

5.4.2 Branch-and-Cut Algorithm for the ILP

In order to find feasible solutions of the moral lineage tracing problem (Def. 7), with certified bounds, we implement the separation procedures defined in the previous section in C++ and call these form the branch-and-cut algorithm of the ILP solver Gurobi (Gurobi Optimization, 2015) whenever an *integer* feasible solution is found. In order to tighten intermediate LP relaxations, we resort to the cuts implemented in Gurobi.

In all experiments we conduct, less than 1% of the total run-time is spent on the separation of infeasible solutions by inequalities (5.1)–(5.4), (5.7) and (5.8) together. Objective values, bounds and numbers of added inequalities are shown w.r.t. run-time, for three instances of the problem, in Fig. 5.3.

Problem Instance	Variables	Fractional	Optimal
HeLa-small	1839	5	1834 (100%)
HeLa-test	41571	1180	40078 (99.2%)
Flywing	29063	1174	27740 (99.5%)

Table 5.2: Analysis of solutions of the canonical LP relaxation of the moral lineage tracing problem.

5.4.3 Cutting-Plane Algorithm for the LP Relaxation

In addition to the moral lineage tracing ILP and (integer) feasible solutions found by the branch-and-cut algorithm, we study the canonical LP relaxation and its (possibly fractional) solutions found by a cutting-plane algorithm. Results shown in Fig. 5.3 and Tab. 5.2 are discussed below:

It can be seen from Fig. 5.3 that the solution found by our cutting-plane algorithm for the LP (blue) converges slower than the lower bound found by our branch-and-cut algorithm for the ILP (black). This is simply because the separation of infeasible points by violated inequalities is more complex if the point can be fractional; see Tab. 5.1.

It can be seen from Fig. 5.3 and Tab. 5.2 that the LP relaxation is almost tight for the problem instance *HeLa-small* and less tight for the larger problem instances. This is expected, as *HeLa-small* is dominated by the disjoint arborescence sub-problem, which is in PTIME, while the larger problems are dominated by the minimum cost multicut sub-problem, which is NP-hard. The solution of the LP is not half-integral. Yet, it encourages future work on rounding procedures.

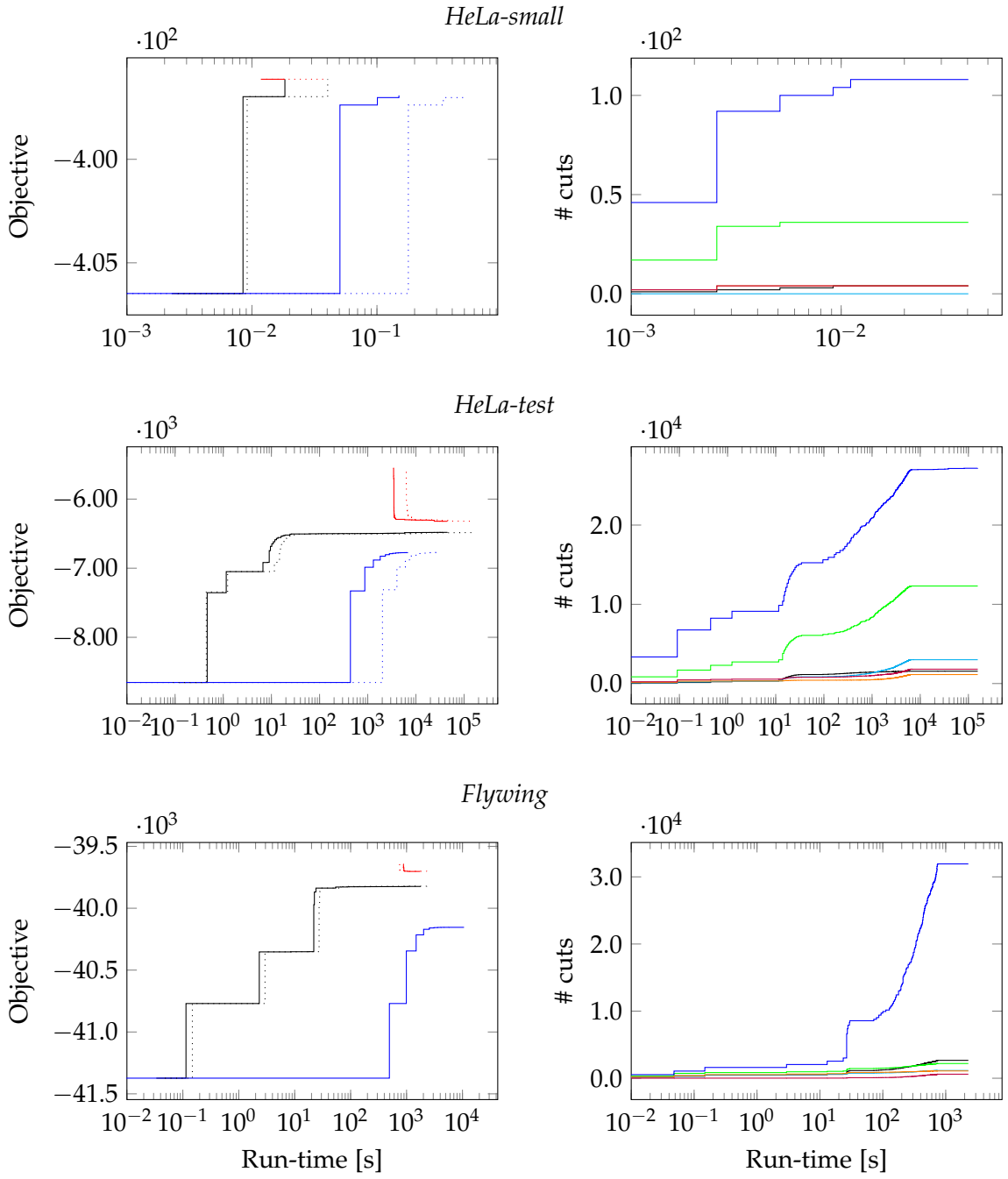


Figure 5.3: Depicted above is the convergence of the branch-and-cut algorithm that solves the ILP and of the cutting plane algorithm that solves the canonical LP relaxation. Graphs on the left show the objective values of intermediate integer feasible solutions (–) and lower bounds (–) found by the branch-and-cut algorithm, as well as the lower bound found by the cutting-plane algorithm (–). Dotted lines show the convergence with the bifurcation constraint (5.4). It can be seen that the LP relaxation is not tight for larger problems. Graphs on the right show numbers of cuts: morality (–), spacetime cycle (–), space cycle (–), appearance (–), disappearance (–), and bifurcation (–). It can be seen that violated morality constraints dominate.

5.5 EXPERIMENTS

In order to examine the effectiveness of moral lineage tracing (MLT) and the proposed branch-and-cut algorithm, we define three instances of the problem w.r.t. two biomedical data sets, *N2DL HeLa* and *Flywing Epithelium*.

5.5.1 N2DL HeLa Data

This microscopy data consists of three sequences of images which show HeLa cells that move and divide as bright objects in front of a dark background, *c.f.* Fig. 5.6(a). Two sequences are publicly available and one sequence is undisclosed for an annual competition (Maška *et al.*, 2014). Here, we use the two public sequences, one for learning a cost function, the other (*HeLa-test*) for experiments. To obtain, in addition, a shorter sequence of smaller images, we crop from *HeLa-test* a sub-problem (*HeLa-small*). For both sequences, we construct a priced hypothesis graph as shown in Fig. 5.4(a) and described below. The hypothesis graph for *HeLa-test* consists of 10882 nodes and 19807 edges. The hypothesis graph for *HeLa-small* consists of 512 nodes and 812 edges.

Optimization The convergence of the branch-and-cut algorithm for the instances *HeLa-small* and *HeLa-test* of the MLT problem is shown in the first two rows of Fig. 5.3. It can be seen that the small problem is solved to optimality, while the full problem is solved with an optimality gap. Most separating cuts are morality constraints.

Results A lineage forest for *HeLa-small* defined by the solution of the MLT problem is shown in Fig. 5.5. This lineage forest is in exact accordance with the ground truth. Corresponding decompositions of images are shown in Fig. 5.6. A lineage forest for *HeLa-test* defined by the feasible solution of the MLT problem is shown in Fig. 5.8. A comparison with ground truth provided in (Maška *et al.*, 2014) is shown in Tab. 5.3 in terms of metrics SEG and TRA as defined in (Maška *et al.*, 2014).

Technical details Before constructing a hypothesis graph we perform the following data pre-processing: We train a random forest to predict, for every pixel r, the probability p_r of this pixel being foreground (part of a cell). For every time index t, we consider the set S_t of pixels r at time t for which $p_r > 0.5$. A watershed search of the distance transform of S_t decomposes the subgraph of the pixel grid graph induced by S_t into cell fragments (superpixels) V_t. For every cell fragment $v \in V_t$, we compute its center of mass $r_v \in \mathbb{R}^2$ in the image plane. We train a second random forest to predict, for every pixel r, the probability p'_r of this pixel showing a cell boundary, *i.e.* the interface between a cell and the background or the interface between two cells that touch.

We then construct a hypothesis graph $G = (V, E, \tau)$ as follows: For every time index t and every pair of distinct cell fragments $v, w \in V_t$, we introduce the edge

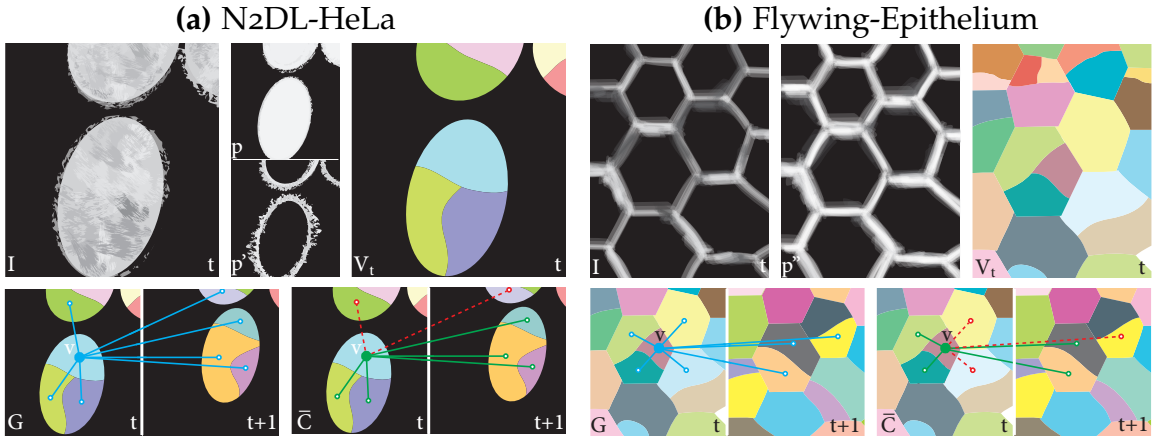


Figure 5.4: Sketched above is the construction of an instance of the MLT for **(a)** the N2DL-HeLa Data and **(b)** the Flywing-Epithelium Data. For every time index t and the respective image I in the sequence, foreground probabilities p of pixels showing part of a cell and probabilities p' and p'' of pixels showing object boundaries are estimated and used to decompose the image into cell fragments S_t. A hypothesis graph (shown for only one fragment each) connects nearby cell fragments within images and across successive images.

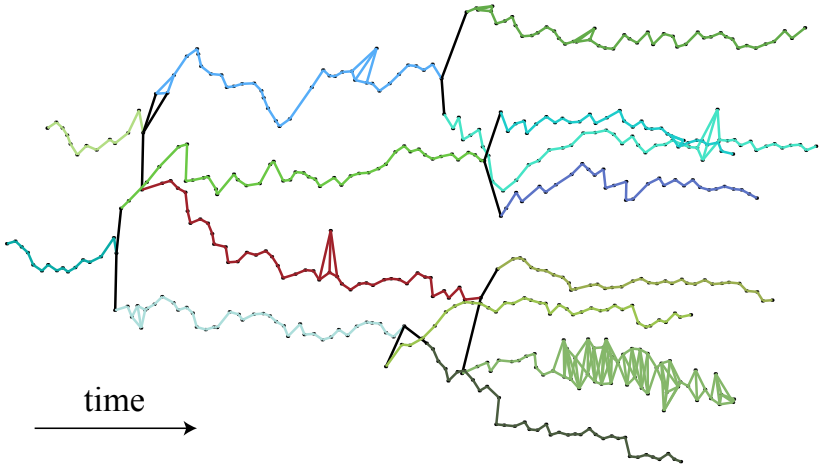


Figure 5.5: Depicted above is the lineage forest $(V, \bar{C})$ reconstructed by solving an instance of the moral lineage tracing problem (Def. 7) defined w.r.t. the image sequence HeLa-small. Edges connecting a fragment of one cell to a fragment of a descendant cell (depicted in black) indicate cell divisions. Edges connecting fragments of the same cell are depicted in a color representing that cell. Note the two progenitor cells in the first image, visible here on the l.h.s..

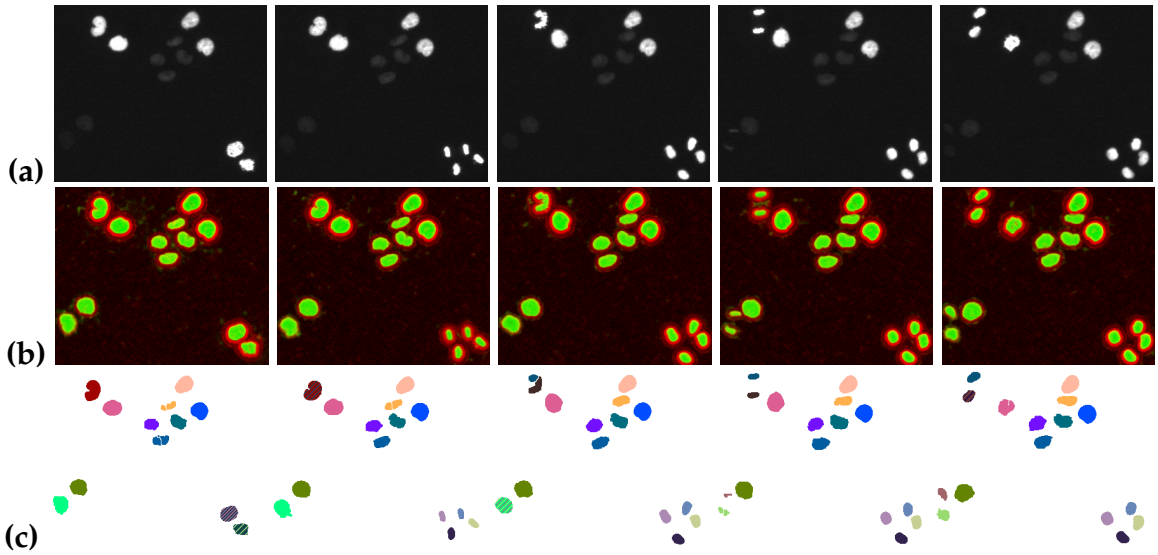


Figure 5.6: Depicted above are **(a)** image crops of the full HeLa-test data set, and **(c)** decompositions of the images defined by a feasible solution of the moral lineage tracing problem (Def. 7). Diagonally striped cells indicate cell division.

	Test Data	SEG	TRA
MLT (our)	public	0.7811	0.9747
KTH-SE (Magnusson *et al.*, 2015)	undisclosed	0.8932	0.9920
HEID-GE (Harder *et al.*, 2015)	undisclosed	0.8155	0.9871
LEID-NL (Dzyubachyk *et al.*, 2010)	undisclosed	0.8180	0.9558
HOUS-US (Merouane *et al.*, 2015)	undisclosed	0.7701	0.9865
NOTT-UK	undisclosed	0.5778	0.7811
IMCB-SG	undisclosed	0.3317	0.9327

Table 5.3: Quantified above is the distance from ground truth of decompositions (SEG) and lineage forests (TRA) obtained by MLT and contenders of the second ISBI Tracking Challenge (Maška *et al.*, 2014). Evaluation for (Maška *et al.*, 2014) is performed on undisclosed test data. We evaluate MLT on test data published on the challenge website.

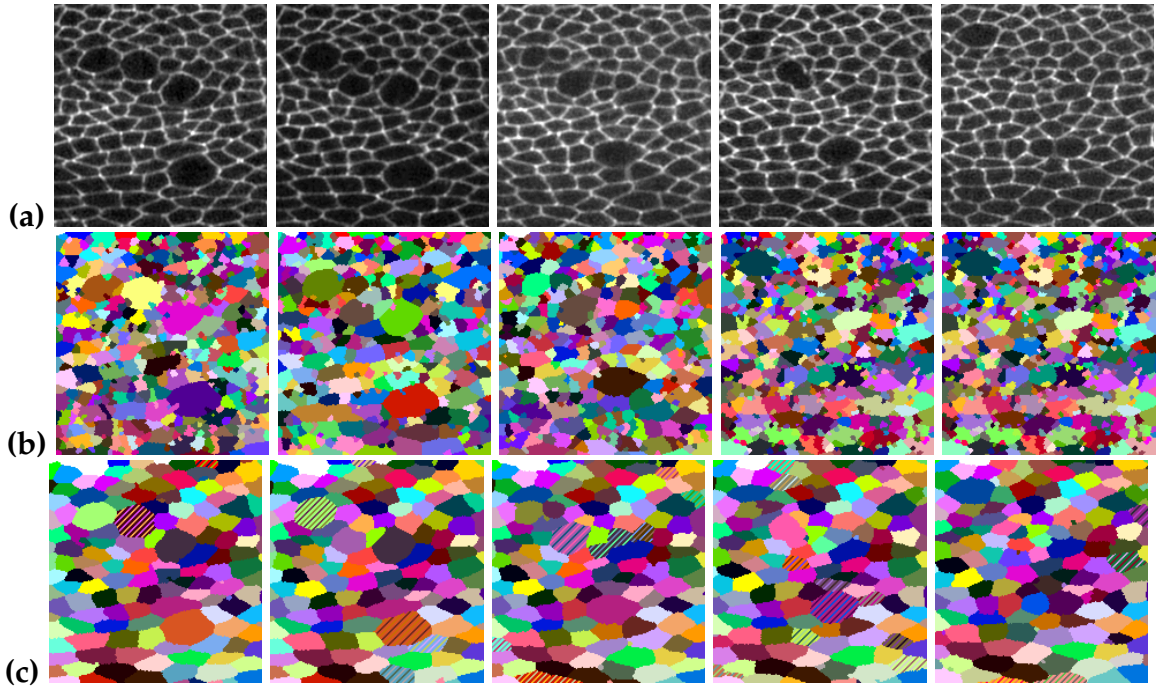


Figure 5.7: Depicted above are **(a)** images of the fly wing test set, **(b)** decompositions of these images into cell fragments, and **(c)** decompositions of the images defined by a feasible solution of the moral lineage tracing problem (Def. 7). Diagonally striped cells divide before the next time point (image).

$\{v, w\} \in E_t$ iff $\|r_v - r_w\| < d_1$, for a maximum distance $d_1 \in \mathbb{R}^+$. For every time index t and every pair $(v, w) \in V_t \times V_{t+1}$, we introduce the edge $\{v, w\} \in E_{t,t+1}$ iff $\|r_v - r_w\| < d_2$, for a maximum distance $d_2 \in \mathbb{R}^+$.

For every time index t and every edge $e = \{v, w\} \in E_t$, we define the cost

$$c_e = -\text{logit } \max\{||r_v - r_w||/d_1, p'(r_v, r_w)\} \ . \tag{5.9}$$

Here $p'(r_v, r_w)$ denotes the maximum of p' along the Bresenham line from r_v to r_w. For every time index t and every $e = \{v, w\} \in E_{t,t+1}$, we define the cost

$$c_e = -\text{logit } ||r_v - r_w||/d_2 \ . \tag{5.10}$$

All (dis)appearance costs are constant, $c^+ = c^- = c_0 \in \mathbb{R}^+$.

5.5.2 Flywing Epithelium Data

This dataset contains images that show a developing fly wing epithelium, *c.f.* Fig. 5.7(a). Every pixel is a part of a cell and no pixels show background. The data is divided into a training sequence and a test sequence. We have collected ground truth for both sequences by manually joining watershed superpixels. The construction of a priced hypothesis graph from the raw test sequence is sketched in Fig. 5.4(b) and described in more detail below. It consists of 5026 nodes and 19011 edges.

Method	SEG	TRA
MLT (our)	0.9722	0.9813
PA (on GT seg.)	0.9327	0.9898
PA (auto)	0.7980	0.9206

Table 5.4: Quantified above is the distance from ground truth of decompositions (SEG) and traced lineage forests (TRA) obtained by MLT and, alternatively, the *Packing Analyzer* (Aigouy *et al.*, 2010).

Optimization The convergence of the branch-and-cut algorithm is shown in the third row of Fig. 5.3. It be seen from this figure that the problem is solved with an optimality gap determined by the lower bound.

Results The lineage forest defined by the feasible solution of the problem is depicted in Fig. 5.8. Corresponding decompositions of images are depicted in Fig. 5.7(c). Decompositions and the lineage forests are compared in Tab. 5.4 to the ground truth in terms of the metrics SEG and TRA. It can be seen from this table that these results compare well to those found by the tracking system biologist use today (Aigouy *et al.*, 2010).

Technical details Data pre-processing consists of training a random forest classifier for detecting, for every pixel r, the probability p''_r of showing a cell membrane. For every time index t, we decompose the image taken at time t into cell fragments V_t by first applying a watershed transform on the raw image sequence and then progressively joining adjacent superpixels iff both the average image intensity and the average membrane probability along their shared boundary are below respective thresholds. We maximize these thresholds w.r.t. the training data subject to the constraint that no false joins occur at this stage. This leads to 3.09 ± 1.3 fragments per cell. Also as pre-processing, we estimate dense optical flow f for the image sequence and compute, for every cell fragment v, its center of mass $r_v \in \mathbb{R}^2$.

We then construct a hypothesis graph $G = (V, E, \tau)$ as follows: For every time index t and every pair of distinct cell fragments $v, w \in V_t$, we introduce the edge $\{v, w\} \in E_t$ iff v and w are adjacent components of the pixel grid graph of the image taken at time index t. For every time index t and every pair $(v, w) \in V_t \times V_{t+1}$, we introduce the edge $\{v, w\} \in E_{t,t+1}$ iff $||r_v + f(r_v) - r_w||_2 \leq d$, for a maximum distance $d \in \mathbb{R}^+$.

For every time index t and every edge $e = \{v, w\} \in E_t$, we define the cut-cost

$$c_e = -\text{logit} \sum_{r \in E(v,w)} p''(r) / |B(v, w)| \tag{5.11}$$

where $B(v, w)$ is the set of pixels in fragment v adjacent to fragment w and vice versa. For every time index t and every edge $e = \{v, w\} \in E_{t,t+1}$, we define the cut-cost $c_e = c_0 + c_1 m_e$ with m_e the maximum of p'' along a geodesic between pixels

r_v and r_w, and with $c_0, c_1 \in \mathbb{R}$ estimated from training data by logistic regression. All (dis)appearance costs are constant, $c^+ = c^- = c_0 \in \mathbb{R}^+$.

5.6 CONCLUSION

Building on recent work in image decomposition and multi-target tracking, we have proposed a rigorous mathematical abstraction of lineage tracing, a central problem in biological image analysis. The optimization problem we propose, a hybrid of the well-known minimum cost multicut problem and the minimum cost k disjoint arborescence problem, is a joint formulation of image decomposition and lineage forest reconstruction. Its feasible solutions define, for every image in a sequence of images, a decomposition into cells and, across images, a lineage forest of cells. Unlike previous formulations, it does not constrain the set of decompositions. We have studied three instances of this problem defined by two biologically relevant microscopy data sets. For all instances, we have obtained feasible solutions with certified optimality gap. One instance has been solved to global optimality, yielding a solution in exact accordance with decompositions and ground truth lineages. A follow-up work by Rempfler *et al.* (2017) considerably improved the runtime of the solver by obtaining tighter relaxations as well as implementing a Kernighan-Lin-type heuristic with optimal branching.

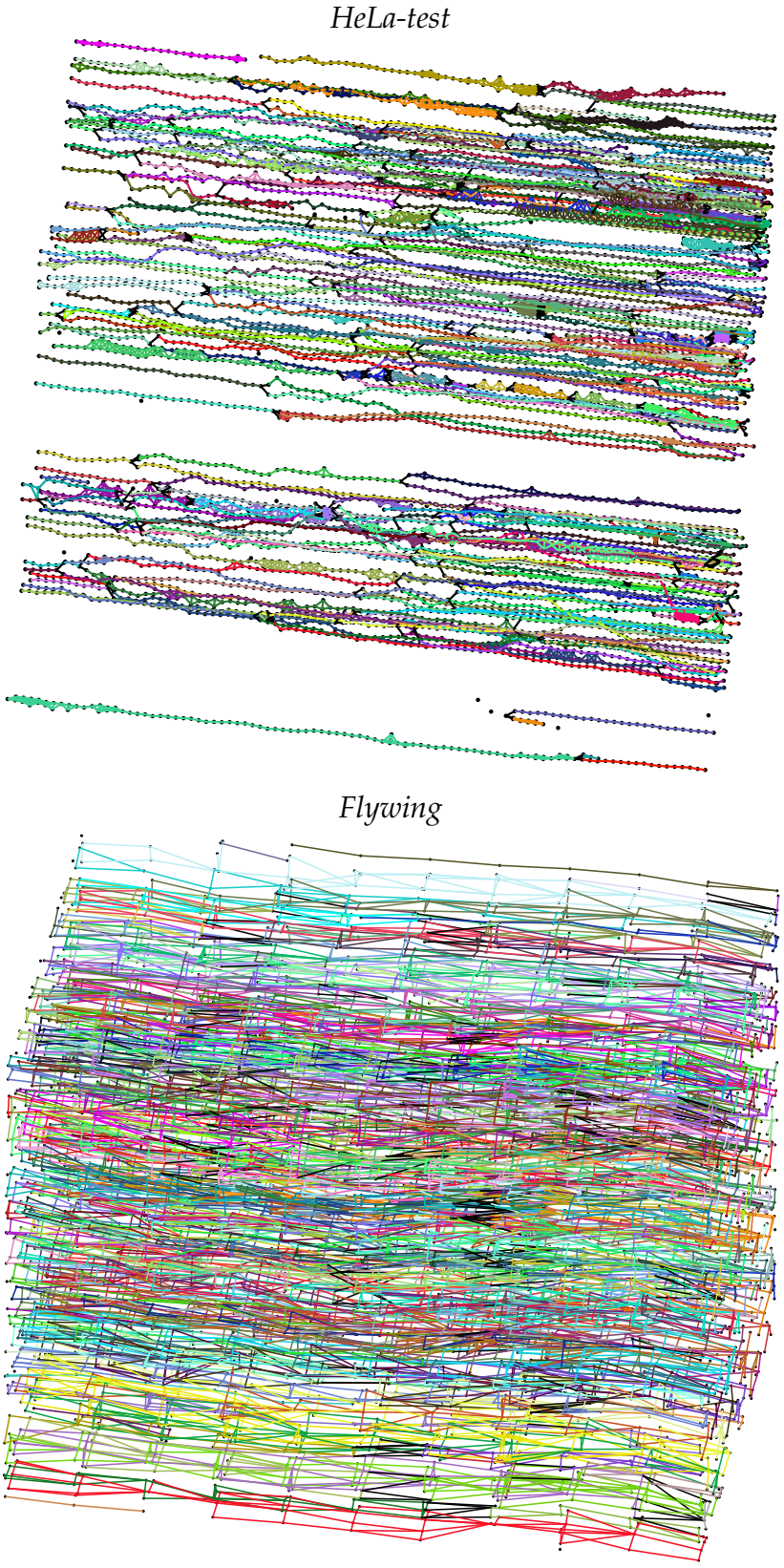


Figure 5.8: 3D rendered lineage forests as obtained by solving the moral lineage tracing problem. For better visibility, only the traced moral lineages are shown while all cut edges are not rendered. Time progresses from left to right.

Part II

Node Labeling Multicuts for Computer Vision

6 NODE LABELING LIFTED MULTICUT PROBLEM

Contents

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 problems that we construct from published data and improve over the corresponding baselines.

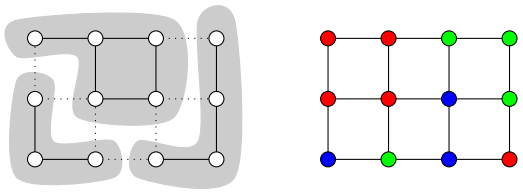
6.1 INTRODUCTION

Graphs are a ubiquitous and widely used structure in the field of computer science. They offer an abstract representation of different phenomena in life and algorithms to solve the arising problems. In computer vision continuous graph-based algorithms have been particularly successful over the last few decades. In this article, we propose a discrete optimization framework that simultaneously optimizes for a decomposition and a node labeling of a given graph (Fig. 6.1).

Our objective function is submodular and is therefore NP-hard to optimize exactly for general graphs. We extend a heuristic proposed in Chapter 2 in a non-trivial way and define and implement two local search algorithms, that allow us to perform efficient inference into real world problems. Note, that we do not perform any fine-tuning for a particular task, but rely solely on their generalization and efficiency. We apply our formulation to three distinct computer vision tasks: multiple object tracking, instance-separating semantic segmentation and articulated human body pose estimation. We report an improvement of application-specific accuracy for these three applications.

Articulated human body pose estimation can be seen as a task requiring two classes of decisions: For every putative detection of a part of the human body in an image, we need to decide whether it is a true or a false positive one. For every pair of detections that do depict body parts, we need to decide if they belong to the same body. Pishchulin *et al.* (2016) and Insafutdinov *et al.* (2016) 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 identify body part classes (head, wrist, ankle, etc.) and background. Due to high computational complexity of the problem they had to restrict themselves to a small number of detections per image. 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 detections obtained from their published data. This allows us to obtain more accurate pose estimates for the MPII Human Pose Dataset (Andriluka *et al.*, 2014) than the respective baseline. Also, our efficient inference technique allowed for a novel application of pose tracking (Insafutdinov *et al.*, 2017), that can further improve accuracy.

Multiple object tracking (Berclaz *et al.*, 2011; Brendel *et al.*, 2011; Choi, 2015; Fagot-Bouquet *et al.*, 2016; Kim *et al.*, 2015; Milan *et al.*, 2014; Pirsiavash *et al.*, 2011; Xiang *et al.*, 2015; Zamir *et al.*, 2012) can be seen as a task requiring two classes of decisions: For every given detection in an image, we need to decide whether it is a true or a false positive one. For every pair of detections that depict objects, we need to decide if the object is the same. Tang *et al.* (2015) 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. However, due to computational complexity of the problem they threshold detections on their scores and then perform clustering. We straightforwardly apply our algorithms to the data of (Tang *et al.*, 2016) and obtain more accurate tracks for the multiple object tracking



(a) Decomposition **(b)** Node Labeling

Figure 6.1: 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.

benchmark (Milan *et al.*, 2016) than the respective baseline.

Instance-separating semantic segmentation (Dai *et al.*, 2016; Hayder *et al.*, 2017; Arnab and Torr, 2017; He *et al.*, 2017; Liu *et al.*, 2018; Bai and Urtasun, 2017; Kendall *et al.*, 2018; Liu *et al.*, 2017) 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, etc.). For every pair of points of the same class, we need to decide if they belong ot the same object (instance of this class). Kroeger *et al.* (2014) 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 of the pixel grid graphs. While Kroeger *et al.* (2014) show only qualitative results, we apply our algorithms to instances of the problem from the Cityscapes (Cordts *et al.*, 2016) benchmarks.

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 (Kappes *et al.*, 2015a), and also generalizes the NP-hard minimum cost lifted multicut problem, LMP (Chapter 2). 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 (Chapter 2) 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.

6.2 PROBLEM

In this section, we define the minimum cost node labeling lifted multicut problem, NL-LMP. Sections 6.2.1–6.2.3 offer an intuition for its parameters, feasible solutions and cost function. Section 6.2.4 offers a concise and rigorous definition. Section 6.2.5 discusses special cases.

6.2.1 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 $v, w \in V$:

$$\{v, w\} \in E' \Leftrightarrow (v, w) \in A \vee (w, v) \in A \tag{6.1}$$
$$(v, w) \notin A \vee (w, v) \notin A \tag{6.2}$$

- 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.

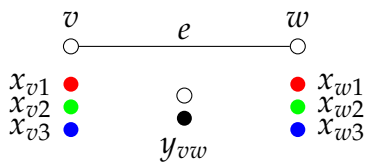


Figure 6.2: A solution to NL-LMP is a 01-labeling of edges (decomposition) and a 01-labeling of vertices' labels (node labeling).

6.2.2 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 $y \in \{0,1\}^{E'}$, *c.f.* Fig 6.2. 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

$$\forall v \in V : \quad \sum_{l \in L} x_{vl} = 1 \ . \tag{6.3}$$

  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 (Horňáková *et al.*, 2017). 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:

$$\forall C \in \text{cycles}(G)\, \forall e \in C : \ y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \tag{6.4}$$

$$\forall \{v, w\} \in E' \setminus E\, \forall P \in vw\text{-paths}(G) : \quad y_{\{v,w\}} \leq \sum_{e \in P} y_e \tag{6.5}$$

$$\forall \{v, w\} \in E' \setminus E\, \forall C \in vw\text{-cuts}(G) : \quad 1 - y_{\{v,w\}} \leq \sum_{e \in C} (1 - y_e) \ . \tag{6.6}$$

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

$$\begin{aligned}\varphi(x,y) = &\sum_{v \in V} \sum_{l \in L} c_{vl}\, x_{vl} \\ &+ \sum_{vw \in A} \sum_{ll' \in L^2} c^{\sim}_{vw,ll'}\, x_{vl}\, x_{wl'}\, (1 - y_{\{v,w\}}) \\ &+ \sum_{vw \in A} \sum_{ll' \in L^2} c^{\not\sim}_{vw,ll'}\, x_{vl}\, x_{wl'}\, y_{\{v,w\}} \quad . \end{aligned} \tag{6.7}$$

6.2.4 Definition

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

Definition 8 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

$$\min_{(x,y) \in X_{VL} \times Y_{GG'}} \varphi(x,y) \quad . \tag{6.8}$$

6.2.5 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.

6.2.5.1 *Unconstrained Integer Quadratic Program*

Definition 9 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

$$\min_{x \in X_{VL}} \sum_{v \in V} \sum_{l \in L} c_{vl}\, x_{vl} + \sum_{vw \in A} \sum_{ll' \in L^2} c'_{vw,ll'}\, x_{vl}\, x_{wl'} \quad . \tag{6.9}$$

Lemma 3 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 W.l.o.g. 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 (6.7) of the cost function of the NL-LMP specializes to the form (6.9) of the cost function of the UIQP.

6.2.5.2 *Minimum Cost Lifted Multicut Problem*

Definition 10 (Horňáková *et al.*, 2017) 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

$$\min_{y \in Y_{GG'}} \sum_{e \in E'} c'_e y_e \ . \tag{6.10}$$

Lemma 4 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

$$L = \{1\} \quad c = 0 \quad c^{\sim} = 0 \tag{6.11}$$

$$\forall (v, w) \in A : \quad c^{\not\sim}_{vw,11} = c'_{\{v,w\}} \ . \tag{6.12}$$

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 (6.7) of the NL-LMP assumes the special form below which is identical with the form in (6.10).

$$\varphi(x, y) \overset{(6.3),(6.11)}{=} \sum_{vw \in A} c^{\not\sim}_{vw,11} y_{\{v,w\}} \overset{(6.12)}{=} \sum_{e \in E'} c'_e y_e \ . \tag{6.13}$$

6.2.5.3 *Subgraph Selection*

Applications such as (Insafutdinov *et al.*, 2016; Pishchulin *et al.*, 2016; Tang *et al.*, 2015, 2016) 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 *suppressed*. Secondly, one chooses a large enough $c^* \in \mathbb{N}$, a $c^{\dagger} \in \mathbb{N}_0$ and $c^{\sim}, c^{\not\sim}$ such

that

$$\forall vw \in A \, \forall l \in L \setminus \{\epsilon\} : \quad c^{\sim}_{vw,l\epsilon} = c^{\sim}_{vw,\epsilon l} = c^* \tag{6.14}$$

$$c^{\not\sim}_{vw,l\epsilon} = c^{\not\sim}_{vw,\epsilon l} = 0 \tag{6.15}$$

$$\forall vw \in A : \quad c^{\sim}_{vw,\epsilon\epsilon} = c^{\dagger} \ . \tag{6.16}$$

By (6.14), suppressed nodes are not joined with the remaining nodes in the same component. By (6.15), cutting a suppressed node from the remaining nodes has zero cost. By (6.16), joining suppressed nodes has cost $c^{\dagger}$, possibly zero. Choosing $c^{\dagger}$ large enough implements an additional constraint proposed in (Tang *et al.*, 2015) that the suppressed nodes are necessarily isolated. It is by this constraint and by a two-elementary label set that (Tang *et al.*, 2015) is a specialization of the NL-LMP.

6.2.5.4 *(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 (Kroeger *et al.*, 2014). 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 Pishchulin *et al.* (2016), where every pair of not suppressed nodes necessarily belongs to the same human. Another example is connected foreground segmentation (Nowozin and Lampert, 2010; Rempfler *et al.*, 2016; Stühmer and Cremers, 2015; Vicente *et al.*, 2008) 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 and disconnectedness constraints by choosing 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 $c^{\sim}_{vw,ll'} = c^{\sim}_{vw,l'l} = c^*$.

6.3 LOCAL SEARCH 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 (1970) and generalized to the LMP in Chapter 2, 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 defining two local search algorithms for the NL-LMP.

6.3.1 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

$$\forall v \in V \, \forall l \in L : \quad x^\lambda_{vl} = 1 \Leftrightarrow \lambda(v) = l \ , \tag{6.17}$$

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

$$\forall \{v, w\} \in E' : \quad y^\mu_{\{v,w\}} = 0 \Leftrightarrow \mu(v) = \mu(w) \ . \tag{6.18}$$

6.3.2 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.

$$\forall w \in V : \quad \lambda'(w) := \begin{cases} l & \text{if } w = v \\ \lambda(w) & \text{otherwise} \end{cases} \ . \tag{6.19}$$

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.

$$\forall w \in V : \quad \mu'(w) := \begin{cases} m & \text{if } w = v \\ \mu(w) & \text{otherwise} \end{cases} \ . \tag{6.20}$$

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.

$$\forall w \in V : \quad \mu'(w) := \begin{cases} m' & \text{if } \mu(w) = m \\ \mu(w) & \text{otherwise} \end{cases} \ . \tag{6.21}$$

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 (6.20) whose output μ' violates this condition.

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.

6.3.3 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., (6.8) for the fixed labels $x \in X_{VL}$, by means of GAEC (Chapter 2).

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 we proposed in Chapter 2. 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 (6.19)–(6.21) 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 (1970), 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

Function 1: $(\Delta, \lambda') = \text{update-labeling}(\mu, \lambda)$

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

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.

6.3.4 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 (1970). 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

$$(\hat{v}, \hat{l}) \in \underset{(v,l) \in V \times L}{\operatorname{argmin}} \varphi(x^{T_{vl}(\lambda_t)}, y^{\mu_t}) - \varphi(x^{\lambda_t}, y^{\mu_t}) \ . \tag{6.22}$$

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,

Function 2: $(\Delta', \mu', \lambda') = \text{update-lifted-multicut}(\mu, \lambda)$

$\mu_0 := \mu \qquad t := 0$
$(\delta, \lambda_0) := \text{update-labeling}(\mu_0, \lambda)$
let $\alpha_0 : \mathbb{N} \to \{0,1\}$ such that $\alpha_0(\mathbb{N}) = 1$
repeat
  $\Delta := 0 \qquad \mu_{t+1} := \mu_t \qquad \lambda_{t+1} := \lambda_t$
  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
    $(\delta, \mu_{t+1}, \lambda_{t+1}) := \text{update-2-cut}(\mu_{t+1}, \lambda_{t+1}, m, m')$
    if $\delta < 0$
      $\alpha_{t+1}(m) := 1 \quad \alpha_{t+1}(m') := 1 \quad \Delta := \Delta + \delta$
  for each $m \in \mu(V)$
    if $\alpha_t(m) = 0$
      continue
    $m' := 1 + \max \mu(V)$ (new component)
    $(\delta, \mu_{t+1}, \lambda_{t+1}) := \text{update-2-cut}(\mu_{t+1}, \lambda_{t+1}, m, m')$
    if $\delta < 0$
      $\alpha_{t+1}(m) := 1 \quad \alpha_{t+1}(m') := 1 \quad \Delta := \Delta + \delta$
  $(\delta, \lambda_{t+1}) := \text{update-labeling}(\mu_{t+1}, \lambda_{t+1})$
  $\Delta := \Delta + \delta$
  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 $\Delta < 0$

considering only the neighbors w of v in G':

$$\begin{aligned}
&\varphi(x^{T_{vl}(\lambda_t)}, y^{\mu_t}) - \varphi(x^{\lambda_t}, y^{\mu_t}) \\
&= c_{vl} - c_{v\lambda_t(v)} \\
&\quad + \sum_{vw \in A} (1 - y_{\{v,w\}}) \left(c^{\sim}_{vw,l\lambda_t(w)} - c^{\sim}_{vw,\lambda_t(v)\lambda_t(w)} \right) \\
&\quad + \sum_{wv \in A} (1 - y_{\{v,w\}}) \left(c^{\sim}_{wv,\lambda_t(w)l} - c^{\sim}_{wv,\lambda_t(w)\lambda_t(v)} \right) \\
&\quad + \sum_{vw \in A} y_{\{v,w\}} \left(c^{\nsim}_{vw,l\lambda_t(w)} - c^{\nsim}_{vw,\lambda_t(v)\lambda_t(w)} \right) \\
&\quad + \sum_{wv \in A} y_{\{v,w\}} \left(c^{\nsim}_{wv,\lambda_t(w)l} - c^{\nsim}_{wv,\lambda_t(w)\lambda_t(v)} \right) \\
&=: \Delta_{t,vl} \ . \qquad (6.23)
\end{aligned}$$

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 (6.22) 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, (6.24) subtracts the costs due to $\hat{v}$ being labeled $\lambda_t(\hat{v})$ (which is possibly outdated), while (6.25) adds the costs due to $\hat{v}$ having obtained a new and possibly different label $\hat{l}$.

$$\begin{aligned}
\Delta_{t+1,wl} =& \Delta_{t,wl} \\
&- (1 - y_{\{w,\hat{v}\}}) \left(c^{\sim}_{w\hat{v},l\lambda_t(\hat{v})} - c^{\sim}_{w\hat{v},\lambda_t(w)\lambda_t(\hat{v})} \right) \\
&- y_{\{w,\hat{v}\}} \left(c^{\not\sim}_{w\hat{v},l\lambda_t(\hat{v})} - c^{\not\sim}_{w\hat{v},\lambda_t(w)\lambda_t(\hat{v})} \right) && (6.24) \\
&+ (1 - y_{\{w,\hat{v}\}}) \left(c^{\sim}_{w\hat{v},l\hat{l}} - c^{\sim}_{w\hat{v},\lambda_t(w)\hat{l}} \right) \\
&+ y_{\{w,\hat{v}\}} \left(c^{\not\sim}_{w\hat{v},l\hat{l}} - c^{\not\sim}_{w\hat{v},\lambda_t(w)\hat{l}} \right) \quad . && (6.25)
\end{aligned}$$

We solve (6.22) 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 (Chapter 2) for the minimum cost lifted multicut problem generalizes the Kernighan and Lin (1970) Algorithm 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

$$(\hat{v}, \hat{l}) \in \operatorname*{argmax}_{(v,l) \in V_t \times L} \varphi(x^{T_{vl}(\lambda_t)}, y^{T'_{vm'}(\mu_t)}) - \varphi(x^{\lambda}_t, y^{\mu_t}) \tag{6.26}$$

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 $\forall v \in A \cup B, \forall l \in L$:

$$\Delta_{vl} = c_{v\lambda_t(v)} - c_{vl} + \tag{6.27}$$

$$\begin{cases} \sum_{w \in A \setminus \{v\}} & c^{\sim}_{vw,\lambda_t(v)\lambda_t(w)} - c^{\not\sim}_{vw,l\lambda_t(w)} \\ \sum_{w \in B} & c^{\not\sim}_{vw,\lambda_t(v)\lambda_t(w)} - c^{\sim}_{vw,l\lambda_t(w)} \\ \sum_{w \notin A \cup B} & c^{\not\sim}_{vw,\lambda_t(v)\lambda_t(w)} - c^{\not\sim}_{vw,l\lambda_t(w)} \end{cases} , \tag{6.28}$$

where $w \in \mathcal{N}_{G'}(v)$. In eq. (6.28) the first two cases are exactly the same as given in (Kernighan and Lin, 1970) 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 $\forall w \in \mathcal{N}_{G'}(\hat{v}), \forall l \in L$:

$$\Delta_{wl} = \Delta_{wl} - \left(c^{\sim}_{\hat{v}w,\lambda_t(\hat{v})\lambda_t(w)} - c^{\not\sim}_{\hat{v}w,\lambda_t(\hat{v})l}\right) + c^{\not\sim}_{\hat{v}w,\hat{l}\lambda_t(w)} - c^{\sim}_{\hat{v}w,\hat{l}l} \ , \quad \text{if } w \in A \ , \tag{6.29}$$

$$\Delta_{wl} = \Delta_{wl} - \left(c^{\not\sim}_{\hat{v}w,\lambda_t(\hat{v})\lambda_t(w)} - c^{\sim}_{\hat{v}w,\lambda_t(\hat{v})l}\right) + c^{\sim}_{\hat{v}w,\hat{l}\lambda_t(w)} - c^{\not\sim}_{\hat{v}w,\hat{l}l} \ , \quad \text{if } w \in B \ . \tag{6.30}$$

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 (Kernighan and Lin, 1970), but multiplied by 2, because in our objective we have two terms that operate on the edges simultaneously.

As we generalize KLj (Chapter 2) by an additional loop over the set L of labels, the analysis of the time complexity carries over with an additional multiplicative factor $|L|$.

Function 3: $(\Delta', \mu', \lambda') = \text{update-2-cut}(\mu, \lambda, m, m')$

$\mu_0 := \mu \qquad \lambda_0 := \lambda \qquad t := 0$
if $\mu^{-1}(m') = \emptyset$
  $V_0 := \mu^{-1}(m)$
else
  $V_0 := \{v \in \mu^{-1}(m) \mid \exists w \in \mu^{-1}(m') : \{v, w\} \in E\}$
if $\mu^{-1}(m) = \emptyset$
  $W_0 := \mu^{-1}(m')$
else
  $W_0 := \{w \in \mu^{-1}(m') \mid \exists v \in \mu^{-1}(m) : \{v, w\} \in E\}$
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 $(\hat{v}, \hat{l}) \in \underset{(v,l) \in V_t \times L}{\operatorname{argmin}} \varphi(x^{T_{vl}(\lambda_t)}, y^{T'_{vm'}(\mu_t)}) - \varphi(x^{\lambda_t}, y^{\mu_t})$
    $\delta := \varphi(x^{T_{\hat{v}\hat{l}}(\lambda_t)}, y^{T'_{\hat{v}m'}(\mu_t)}) - \varphi(x^{\lambda_t}, y^{\mu_t})$
  if $W_t \neq \emptyset$
    choose $(\hat{w}, \hat{l}) \in \underset{(w,l) \in W_t \times L}{\operatorname{argmin}} \varphi(x^{T_{wl}(\lambda_t)}, y^{T'_{wm}(\mu_t)}) - \varphi(x^{\lambda_t}, y^{\mu_t})$
    $\delta' := \varphi(x^{T_{\hat{w}\hat{l}}(\lambda_t)}, y^{T'_{\hat{w}m}(\mu_t)}) - \varphi(x^{\lambda_t}, y^{\mu_t})$
  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 visited)
  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 visited)
  $V_{t+1} := \{v \in V \mid \mu_{t+1}(v) = m \wedge \alpha(v) = 1 \wedge \exists \{v, w\} \in E : \mu_{t+1}(w) = m'\}$
  $W_{t+1} := \{w \in V \mid \mu_{t+1}(w) = m' \wedge \alpha(w) = 1 \wedge \exists \{v, w\} \in E : \mu_{t+1}(v) = m\}$
  $t := t + 1$
$\hat{t} := \min \underset{t' \in \{0,\ldots,t\}}{\operatorname{argmin}} \varphi(x^{\lambda_{t'}}, y^{\mu_{t'}}) - \varphi(x^{\lambda_0}, y^{\mu_0})$
$\Delta_1 := \varphi(x^{\lambda_{\hat{t}}}, y^{\mu_{\hat{t}}}) - \varphi(x^{\lambda_0}, y^{\mu_0})$
$\Delta_2 := \varphi(x^{\lambda_0}, y^{T'_{mm'}(\mu)}) - \varphi(x^{\lambda_0}, y^{\mu_0})$ (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 $(\Delta_2, T_{mm'}(\mu), \lambda)$

6.4 EXPERIMENTS

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.

6.4.1 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.* (2016) and Insafutdinov *et al.* (2016) 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 6.2.5.3). In this case, nodes are putative detections of body parts and labels define body part classes (head, wrist, etc.).

Pishchulin *et al.* (2016) 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:

$$\forall d \in D \forall cc' \in \tbinom{C}{2} : \quad x_{dc} + x_{dc'} \leq 1 \tag{6.31}$$

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

$$y_{dd'} \leq \sum_{c \in C} x_{d'c} \tag{6.32}$$

$$\forall dd'd'' \in \tbinom{D}{3} : \quad y_{dd'} + y_{d'd''} - 1 \leq y_{dd''} \tag{6.33}$$

$x_{vl} = 1$ 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 the same human. The objective function has the form below with coefficients α and β.

$$\sum_{d \in D} \sum_{c \in C} \alpha_{dc}\, x_{dc} + \sum_{dd' \in \binom{D}{2}} \sum_{c,c' \in C} \beta_{dd'cc'} x_{dc} x_{d'c'} y_{dd'} \tag{6.34}$$

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

$$c_{vl} := \begin{cases} \alpha_{vl} & \text{if } l \in C \\ 0 & \text{if } l = \epsilon \end{cases} \tag{6.35}$$

$$c^{\sim}_{vw,ll'} := \begin{cases} \beta_{vwll'} & \text{if } l \in C \wedge l' \in C \\ 0 & \text{if } l = \epsilon \text{ xor } l' = \epsilon \\ \infty & \text{if } l = l' = \epsilon \end{cases} . \tag{6.36}$$

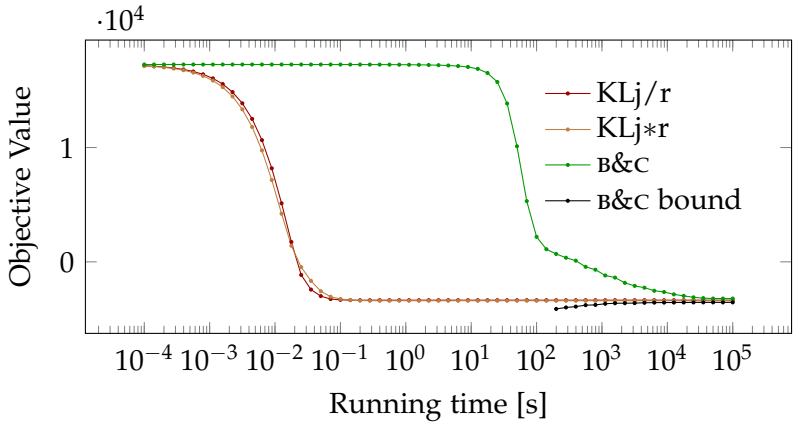


Figure 6.3: Convergence of B&C (Insafutdinov *et al.*, 2016), KLj/r, and KLj∗r on MPII Human Pose Dataset (Andriluka *et al.*, 2014). Plotted above is the mean objective value vs. the absolute runtime. It can be seen, that KLj/r and KLj∗r converge to solutions that are in between the bounds given by B&C, but obtained in five orders of magnitude less time.

Alg.	Mean cost	Mean time [s]	Median time [s]
B&C (Insafutdinov *et al.*, 2016)	-3013.30	9519.26	308.28
KLj/r	-3352.74	**0.033**	**0.031**
KLj∗r	**-3419.07**	0.119	0.100

Table 6.1: Mean cost and runtime averaged over 1758 instances of the MPII Human Pose Dataset (Andriluka *et al.*, 2014). We set a runtime limit of 3 hours per instance for B&C (Insafutdinov *et al.*, 2016). We used exactly the same data as in (Insafutdinov *et al.*, 2016).

The test set of MPII Human Pose Dataset (Andriluka *et al.*, 2014) consists of 1758 images. To tackle the instances of problems defined over each image, Insafutdinov *et al.* (2016) implement a branch-and-cut (B&C) algorithm in the integer linear programming software framework Gurobi. We refer to their published C++ implementation as B&C. We take unary c and pairwise $c^{\sim}$ costs exactly the same as in (Insafutdinov *et al.*, 2016).

Cost and time In Fig. 6.3, 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 KLj∗r arrive at near optimal feasible solutions after 10^{-1} seconds, five orders of magnitude faster than B&C. Exact numbers are presented in Tab. 6.1. 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.

$\|D\|$	Alg.	Head	Sho	Elb	Wri	Hip	Knee	Ank	AP
150	B&C (Insafutdinov *et al.*, 2016)	84.9	79.2	66.4	52.3	65.5	59.2	51.2	65.5
	KLj/r	**87.1**	80.0	**66.8**	**53.6**	66.1	60.0	51.8	66.5
	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**
	KLj∗r	89.8	**85.2**	**71.8**	**59.6**	71.1	63.0	**53.5**	**70.6**

Table 6.2: Our algorithms improve application-specific accuracy results over the baseline B&C (Insafutdinov *et al.*, 2016) on the MPII Human Pose Dataset (Andriluka *et al.*, 2014). By using all the available detections ($|D| = 420$) in the data, we are able to get a further 4% improvement.

	Head	Sho	Elb	Wri	Hip	Knee	Ank	AP
Insafutdinov *et al.* (2016)	89.4	84.5	70.4	59.3	68.9	62.7	54.6	70.0
Varadarajan *et al.* (2018)	92.1	85.9	72.9	61.6	72.0	64.6	56.6	72.2
Insafutdinov *et al.* (2017)	88.8	87.0	75.9	64.9	74.2	68.8	60.5	74.3
Cao *et al.* (2017)	91.2	87.6	77.7	66.8	75.4	68.9	61.7	75.6
Fang *et al.* (2017)	88.4	86.5	78.6	70.4	74.4	73.0	65.8	76.7
Newell *et al.* (2017)	92.1	89.3	78.9	69.8	76.2	71.6	64.7	77.5
Fieraru *et al.* (2018)	91.8	89.5	80.4	69.6	77.3	71.7	65.5	78.0
Our	89.8	85.2	71.8	59.6	71.1	63.0	53.5	70.6

Table 6.3: Application-specific accuracy results on the MPII Human Pose Dataset (Andriluka *et al.*, 2014).

Application-specific Accuracy and Results In Tab. 6.2, 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 better mean cost translates into higher average accuracy in the default setting.

Due to runtime issues, Insafutdinov *et al.* (2016) had to restrict themselves to at most 150 detections per image in order to compute reasonable results with B&C. The shorter absolute running time of KLj/r and KLj∗r allows us to use all the available detections in the data, up to 420 per image, *without additional learning*. It can be seen from the last two rows of Tab. 6.2 that this increases the accuracy by 4%. Our results are put into perspective relative to other methods in Tab. 6.3. Qualitative results are shown in Fig. 6.4. Also, this observation allowed for a novel application of pose tracking in videos (Insafutdinov *et al.*, 2017), that can further improve accuracy.

6.4.2 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.* (2015) approach this problem via a

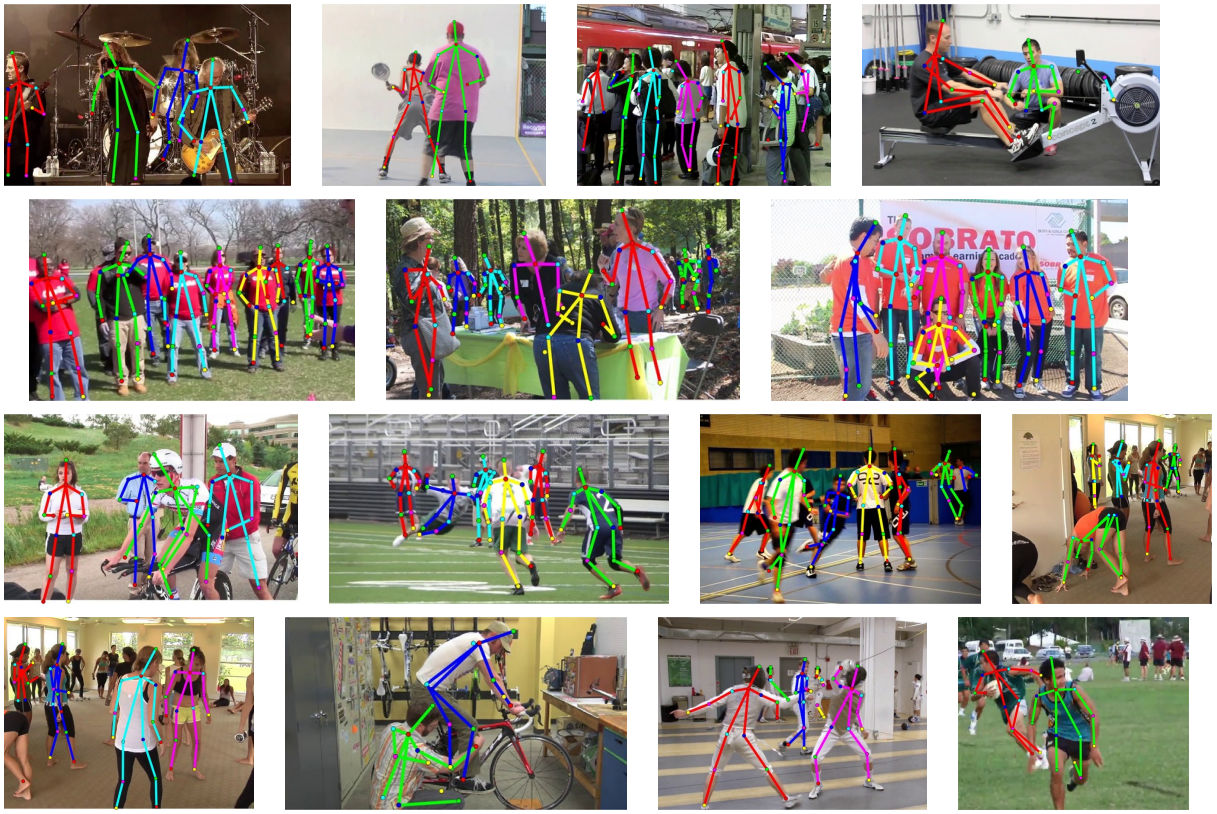


Figure 6.4: Pose estimation results on the MPII Human Pose Dataset (Andriluka *et al.*, 2014).

graph decomposition and node labeling problem that we identify as a special case of the NL-LMP with two labels and subgraph selection (Sec. 6.2.5.3). In this case, nodes are putative detections of persons.

Tang *et al.* (2015) 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

$$\forall \{v, w\} \in E: \quad y_{vw} \leq x_v \tag{6.37}$$

$$y_{vw} \leq x_w \tag{6.38}$$

$$\forall C \in \text{cycles}(G)\ \forall e \in C: \quad 1 - y_e \leq \sum_{f \in C \setminus \{e\}} (1 - y_f) \tag{6.39}$$

$x_{vl} = 1$ indicates that the putative detection v is not suppressed, and $y_{vw} = 1$ indicates that the putative detections v and w are of the same person. The objective function has the form below with coefficients α and β.

$$\sum_{v \in V} \alpha_v x_v + \sum_{e \in E} \beta_e y_e \tag{6.40}$$

We identify the solutions of this problem with the solutions of the NL-LMP

Method	MOTA ↑	FP ↓	FN ↓	Hz ↑
Tang *et al.* (2016)	46.3	6373	90914	0.8
Choi (2015)	46.4	9753	87565	2.6
Sadeghian *et al.* (2017)	47.2	2681	92856	1.0
Chen *et al.* (2018)	47.6	9253	85431	20.6
Henschel *et al.* (2018)	47.8	8886	85487	0.6
Ma *et al.* (2018)	48.2	5104	88586	2.8
Sheng *et al.* (2018)	48.7	6632	86504	4.8
Tang *et al.* (2017)	48.8	6654	86245	0.5
KLj/r	47.6	5844	89093	8.3
KLj∗r	47.6	5783	89160	0.7

Table 6.4: Application-specific accuracy on the MOT16 (Milan *et al.*, 2016) challenge (public detections). We used the same data as in (Tang *et al.*, 2016).

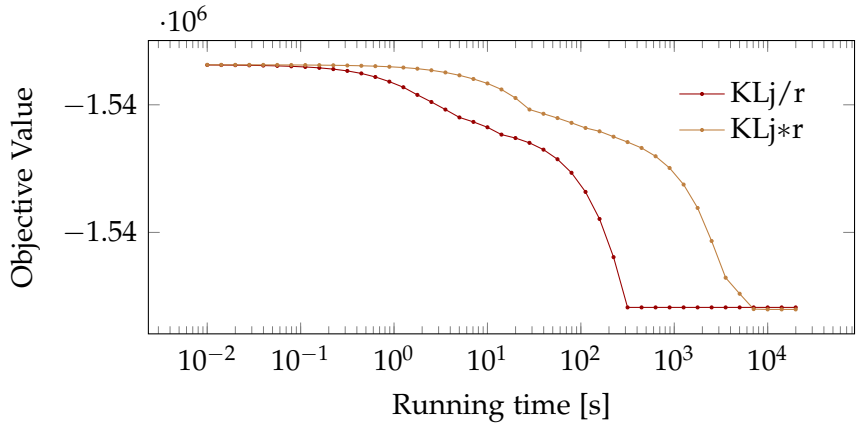


Figure 6.5: Convergence of the algorithms KLj/r and KLj∗r on the data from the MOT16 (Milan *et al.*, 2016) challenge.

w.r.t. the graphs $G' = G$, the label set $L = \{1, \epsilon\}$ and the costs $c^{\nsim} = 0$ and

$$c_{vl} := \begin{cases} \alpha_v & \text{if } l = 1 \\ 0 & \text{if } l = \epsilon \end{cases} \tag{6.41}$$

$$c^{\sim}_{vw,ll'} := \begin{cases} \beta_{vw} & \text{if } l = 1 \wedge l' = 1 \\ 0 & \text{if } l = 1 \text{ xor } l' = 1 \\ \infty & \text{if } l = l' = \epsilon \end{cases} . \tag{6.42}$$

The test set of the multiple object tracking benchmark (MOT16) (Milan *et al.*, 2016) consists of 7 video sequences and putative people detections are provided. To tackle the large instances defined for each sequence, Tang *et al.* (2016) 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 (Chapter 2), without re-iterating. Here, we solve the joint problem of detections suppression and graph decomposition by means of KLj/r and KLj∗r and compare their output to that of (Tang *et al.*, 2016) and of other top-performing algorithms (Choi, 2015; Fagot-Bouquet *et al.*, 2016; Kim *et al.*, 2015). We use the same data as in (Tang *et al.*, 2016), therefore the performance gain is due to our algorithms that solve the full problem.

Cost and time The convergence of the algorithms KLj/r and KLj∗r is shown in Fig. 6.5. It can be seen from this figure that KLj/r converges faster than KLj∗r.

Application-specific Accuracy and Results We compare the feasible solutions output by KLj/r and KLj∗r on the MOT16 benchmark (Milan *et al.*, 2016). To this end, we report in Tab. 6.4 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.

It is can be seen from Tab. 6.4 that the feasible solutions obtained by KLj/r and KLj∗r substantially improve over the baseline (Tang *et al.*, 2016). Compared to (Tang *et al.*, 2016), 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. Some visual results are presented in Fig. 6.6. A complete evaluation of our experimental results and complete video sequences with the tracking overlay can be found at `http://motchallenge.net/tracker/NLLMPa`. 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. We believe that the recently published method by Tang *et al.* (2017), that uses the lifted multicut framework for tracking, can also benefit from our new algorithms.

6.4.3 InstanceCut: Instance-Separating Semantic Segmentation

We turn toward applications of the NL-LMP and 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. No subgraph selection is performed in this application.

Our pipeline (shown in Fig. 6.7) consists of two independent branches: For unary costs $c^l_v, \forall v \in V, \forall l \in L$, we employ a publicly available Fully Convolutional Network Dilation10 (Yu and Koltun, 2016) for semantic labeling, pre-trained by the authors and publicly available. Note, that since this is an independent component it can easily be updated to the most recent FCN. Our contribution is a novel instance-aware edge detector network that produces probabilities $c^{\not\sim}_{vw}, \forall \{vw\} \in E$ for neighbouring

Figure 6.6: Some visual results on MOT16 (Milan *et al.*, 2016) challenge.

super-pixels to belong to the same instance. The two outputs are then combined in a joint NL-LMP (6.8) formulation.

6.4.3.1 *Instance-Aware Edge Detection*

Edge detection is a very well studied problem in computer vision. Some classical results were obtained already back in the 80's (Canny, 1986). More recent methods are based on spectral clustering (Shi and Malik, 2000; Arbeláez *et al.*, 2011, 2014; Isola *et al.*, 2014). These methods perform global inference on the whole image. An alternative approach suggests to treat the problem as a per-pixel classification task (Lim *et al.*, 2013; Dollár and Zitnick, 2015). Recent advances in deep learning have made this class of methods especially efficient, since they automatically obtain rich feature representation for classification (Ganin and Lempitsky, 2014; Kivinen *et al.*, 2014; Shen *et al.*, 2015; Bertasius *et al.*, 2015a,b; Xie and Tu, 2015; Bertasius *et al.*, 2016).

The recent per-pixel classification method (Bertasius *et al.*, 2016) constructs features, which are based on an FCN trained for semantic segmentation on Pascal VOC 2012 (Everingham *et al.*, 2010). The features for each pixel are designed as a concatenation of intermediate FCN features, corresponding to that particular pixel. The logistic regression trained on these features, followed by non-maximal sup-

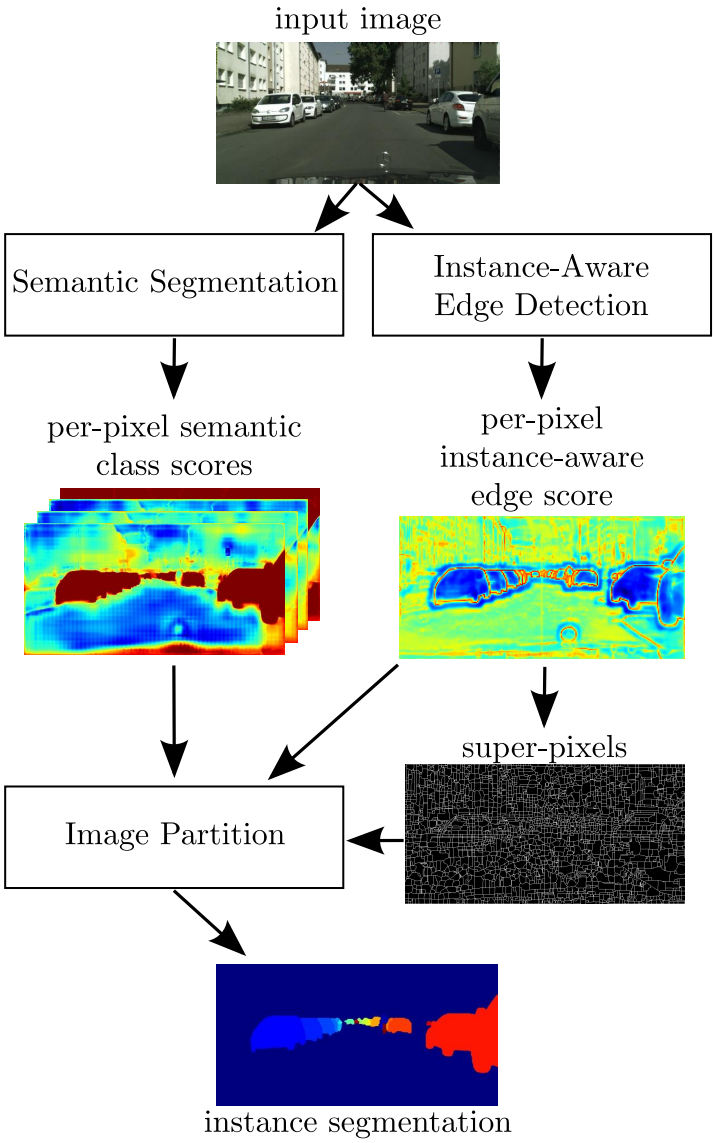


Figure 6.7: **Our InstanceCut pipeline - Overview.** Given an input image, two independent branches produce the per-pixel semantic class scores and per-pixel instance-aware edge scores. The edge scores are used to extract superpixels. The final image partitioning block merges the superpixels into connected components with a class label assigned to each component. The resulting components correspond to object instances and background.

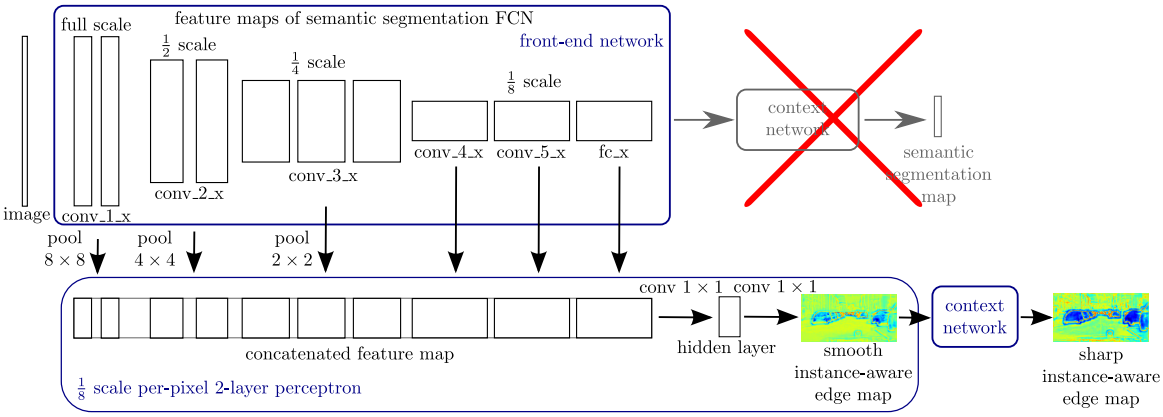


Figure 6.8: **Instance-aware edge detection block.** The semantic segmentation FCN is the front-end part of the network (Yu and Koltun, 2016) trained for semantic segmentation on the same dataset. Its intermediate feature maps are downsampled, according to the size of the smallest feature map, by a max-pooling operation with an appropriate stride. The concatenation of the downsampled maps is used as a feature representation for a per-pixel 2-layer perceptron. The output of the perceptron is refined by a context network of Dilation10 (Yu and Koltun, 2016) architecture.

pression, outputs a per-pixel edge probability map. The paper suggests that the intermediate features of an FCN trained for semantic segmentation form a strong signal for solving the edge detection problem. Similarly constructed features also have been used successfully for other dense labelling problems (Hariharan *et al.*, 2015).

For datasets like BSDS500 (Arbeláez *et al.*, 2011) most works consider general edge detection problem, where annotated edges are class- and instance-agnostic contours. In our work the instance-aware edge detection outputs a probability for each pixel, whether it touches a boundary. This problem is more challenging than canonical edge detection, since it requires to reason about contours and semantics jointly, distinguishing the true objects' boundaries and other not relevant edges, e.g., inside the object or in the background. Below (see Fig. 6.8), we describe a new network architecture for this task that utilizes the idea of the intermediate FCN features concatenation.

As a base for our network we use an FCN that is trained for semantic segmentation on the dataset that we want to use for object boundary prediction. In our experiments we use a pre-trained Dilation10 (Yu and Koltun, 2016) model, however, our approach is not limited to this architecture and can utilize any other FCN-like architectures. We form a per-pixel feature representation by concatenating the intermediate feature maps of the semantic segmentation network. This is based on the following intuition: during inference, the semantic segmentation network is able to identify positions of transitions between semantic classes in the image. Therefore, its intermediate features are likely to contain a signal that helps to find the borders between classes. We believe that the same features can be useful to determine

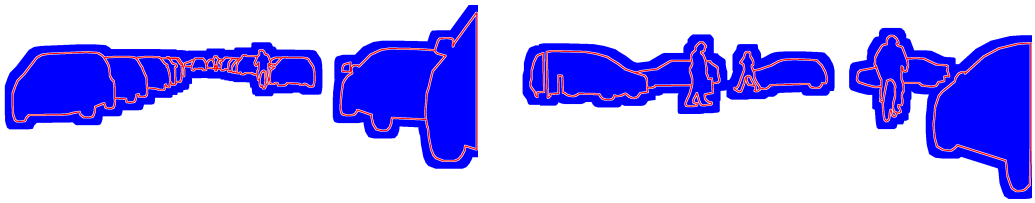


Figure 6.9: Ground truth examples for our instance-aware edge detector. Red indicates pixels that are labeled as edges, blue indicates background, i.e. no edge and white pixels are ignore.

boundaries between objects.

Commonly used approaches (Bertasius *et al.*, 2016; Hariharan *et al.*, 2015) suggest upscaling feature maps that have a size which is smaller than the original image to get per-pixel representation. However, in our experiments such an approach produces thick and over-smooth edge scores. This behavior can be explained by the fact that the most informative feature maps have an 8 times smaller scale than the original image. Hence, instead of upscaling, we downscale all feature maps to the size of the smallest map. Since the network was trained with rectified linear unit (ReLU) activations, the active neurons tend to output large values, therefore, we use max-pooling with a proper stride for the downscaling, see Fig. 6.8.

The procedure outputs the downscaled feature maps (of a *semantic segmentation FCN*, see Fig. 6.8) that are *concatenated* to get the downscaled per-pixel *feature map*. We utilize a *2-layer perceptron* that takes this feature map as input and outputs log-probabilities for edges (*smooth instance-aware edge map*, see Fig. 6.8). The perceptron method is the same for all spatial positions, therefore, it can be represented as two layers of 1×1 convolutions with the ReLU activation in between.

In our experiments we have noticed that the FCN gives smooth edge scores. Therefore, we apply *a context network* (Yu and Koltun, 2016) that refines the scores making them sharper. The new architecture is an FCN, i.e., it can be applied to images of arbitrary size, it is differentiable and has a single loss at the end. Hence, straightforward end-to-end training can be applied for the new architecture. We upscale the resulting output map to match an input image size.

Since the image partition framework, that comes next, operates on super-pixels, we need to transform the per-pixel edge scores to edge scores $c^{\angle}_{vw}$ for each pair $\{v, w\}$ of neighboring superpixels. We do this by averaging all scores of those pixels that touch the border between v and w.

In the following, we describe an efficient implementation of the 2-layer perceptron and also discuss our training data for the boundary detection problem.

Efficient implementation In our experiments, the input for the 2-layer perceptron contains about 13k features per pixel. Therefore, the first layer of the perceptron consumes a lot of memory. It is, however, possible to avoid this by using a more

efficient implementation. Indeed, the first layer of the perceptron is equivalent to the summation of outputs of multiple 1×1 convolutions, which are applied to each feature map independently. For example, `conv_1` is applied to the feature maps from the `conv_1_x` intermediate layer, `conv_2` is applied to the feature maps from `conv_2_x` and its output is summed up with the output of `conv_1`, etc. This approach allows reducing the memory consumption, since the convolutions can be applied during evaluation of the front-end network.

Training data Although it is common for ground truth data that object boundaries lie in-between pixels, we will use in the following the notion that a boundary lies on a pixel. Namely, we will assume that a pixel i is labeled as a boundary if there is a neighboring pixel j, which is assigned to a different object (or background). Given the size of modern images, this boundary extrapolation does not affect performance. As a ground truth for boundary detection we use the boundaries of object instances presented in CityScapes (Cordts *et al.*, 2016).

As mentioned in several previous works (Xie and Tu, 2015; Bertasius *et al.*, 2015b), highly unbalanced ground truth (GT) data heavily harms the learning progress. For example, in BSDS500 (Arbeláez *et al.*, 2011) less than 10% of pixels on average are labeled as edges. Our ground truth data is even more unbalanced: since we consider the object boundaries only, less than 1% of pixels are labeled as being an edge. We employ two techniques to overcome this problem of training with unbalanced data: *a balanced loss function* (Xie and Tu, 2015; Hwang and Liu, 2015) and *pruning of the ground truth data*.

The balanced loss function (Xie and Tu, 2015; Hwang and Liu, 2015) adds a coefficient to the standard log-likelihood loss that decreases the influence of errors with respect to classes that have a lot of training data. That is, for each pixel i the balanced loss is defined as

$$loss(p_{edge}, y^{GT}) = [\![y^{GT} = 1]\!] \log(p_{edge}) + \alpha [\![y^{GT} = 0]\!] \log(1 - p_{edge}),$$

where p_{edge} is the probability of the pixel i to be labeled as an edge, y^{GT} is the ground truth label for i (the label 1 corresponds to an edge), and $\alpha = N_1/N_0$ is the balancing coefficient. Here, N_1 and N_0 are numbers of pixels labeled, respectively, as 1 and 0 in the ground truth.

Another way to decrease the effect of unbalanced GT data is to subsample the GT pixels, see, e.g., (Bertasius *et al.*, 2016). Since we are interested in instance-aware edge detection and combine its output with our semantic segmentation framework, a wrong edge detection, which is far from the target objects (for example, in the sky) does not harm the overall performance of the InstanceCut framework. Hence, we consider a pixel to be labeled as background for the instance-aware edge detection if and only if it lies inside the target objects, or in an area close to it, see Fig. 6.9 for a few examples of the ground truth data for the CityScapes dataset (Cordts *et al.*, 2016). In our experiments, only 6.8% of the pixels are labeled as object boundaries in the pruned ground truth data.

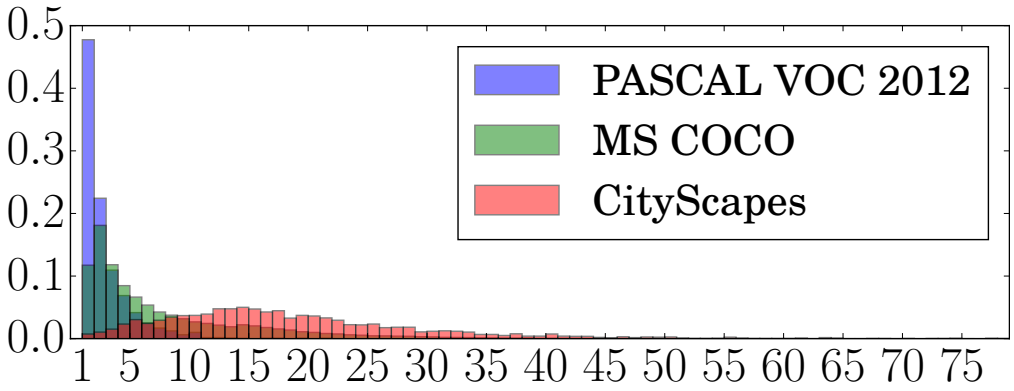


Figure 6.10: The histograms shows distribution of number of instances per image for different datasets. For illustrative reasons we cut long tails of CityScapes and MS COCO. We use CityScapes dataset since it contains significantly more instances per image.

6.4.3.2 *Experiments*

Dataset There are three main datasets with full annotation for the instance-aware semantic segmentation problem: PASCAL VOC2012 (Everingham *et al.*, 2010), MS COCO (Lin *et al.*, 2014) and CityScapes (Cordts *et al.*, 2016). We select the last one for our experimental evaluation for several reasons: (i) CityScapes has very fine annotation with precise boundaries for the annotated objects, whereas MS COCO has only coarse annotations, for some objects, that do not coincide with the true boundaries. Since our method uses an edge detector, it is important to to have precise object boundaries for training. (ii) The median number of instances per image in CityScapes is 16, whereas PASCAL VOC has 2 and MS COCO has 4. For this work a larger number is more interesting. The distribution of the number of instances per image for different datasets is shown in Fig. 6.10. (iii) Unlike other datasets, CityScapes annotation is dense, i.e., all foreground objects are labeled.

The CityScapes dataset has 5000 street-scene images recorded by car-mounted cameras: 2975 images for training, 500 for validation and 1525 for testing. There are 8 classes of objects that have an instance-level annotation in the dataset: person, rider, car, truck, bus, train, motorcycle, bicycle. All images have the size of 1024×2048 pixels.

Training details For the semantic segmentation block in our framework we test two different networks, which have publicly available trained models for CityScapes: Dilation10 (Yu and Koltun, 2016) and LRR-4x (Ghiasi and Fowlkes, 2016). The latter is trained using the additional coarsely annotated data, available in CityScapes. Importantly, CityScapes has 19 different semantic segmentation classes (and only 8 out of them are considered for instance segmentation) and both networks were trained to segment all these classes. We do not retrain the networks and directly use the log-probabilities for the 8 semantic classes, which we require. For the background

	AP	AP50%	AP100m	AP50m
PANet (Liu *et al.*, 2018)	36.4	63.1	49.2	51.8
Mask R-CNN (He *et al.*, 2017)	32.0	58.1	45.8	49.5
SGN (Liu *et al.*, 2017)	25.0	44.9	38.9	44.5
Arnab and Torr (2017)	23.4	45.2	36.8	40.9
Kendall *et al.* (2018)	21.6	39.0	35.0	37.0
DWT (Bai and Urtasun, 2017)	19.4	35.3	31.4	36.8
BAIS (Hayder *et al.*, 2017)	17.4	36.7	29.3	34.0
Pixel Encoding (Uhrig *et al.*, 2016)	8.9	21.1	15.3	16.7
MCG+R-CNN (Cordts *et al.*, 2016)	4.6	12.9	7.7	10.3
InstanceCut (our)	13.0	27.9	22.1	26.1

Table 6.5: CityScapes results for instance-aware semantic segmentation on the test set.

Metric	Mean	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle
AP	13.0	10.0	8.0	23.7	14.0	19.5	15.2	9.3	4.7
AP50%	27.9	28.0	26.8	44.8	22.2	30.4	30.1	25.1	15.7
AP100m	22.1	19.7	14.0	38.9	24.8	34.4	23.1	13.7	8.0
AP50m	26.1	20.1	14.6	42.5	32.3	44.7	31.7	14.3	8.2

Table 6.6: Detailed instance-aware semantic segmentation results on the test set of CityScapes, given for each semantic class.

label we take the maximum over the log-probabilities of the remaining semantic classes.

As an initial semantic segmentation network for the instance-aware edge detection block we use Dilation10 (Yu and Koltun, 2016) pre-trained on the CityScapes. We exactly follow the training procedure described in the original paper (Yu and Koltun, 2016). That is, we pre-train first the front-end module with the 2-layer perceptron on top. Then we pre-train the context module of the network separately and, finally, train the whole system end-to end. All the stages are trained with the same parameters as in (Yu and Koltun, 2016). In our experiments the 2-layer perceptron has 16 hidden neurons. On the validation set the trained detector achieves 97.2% AUC ROC.

Quantitative and qualitative results We evaluated our method using 4 metrics that are suggested by the CityScapes benchmark: AP, AP50%, AP100m and AP50m. We refer to the webpage of the benchmark for a detailed description.

The InstanceCut framework with Dilation10 (Yu and Koltun, 2016) as the semantic segmentation block gives AP $= 14.8$ and AP50% $= 30.7$ on the validation part of the

dataset. When we replace Dilation10 by LRR-4x (Ghiasi and Fowlkes, 2016) for this block the performance improves to AP = 15.8 and AP50% = 32.4, on the validation set.

Quantitative results for the test set are provided in Table 6.5. We compare our approach to all published methods that have results for this dataset. Class-level results are given in Tab. 6.6. Some qualitative results are shown in Fig. 6.11, failure cases are shown in Fig. 6.12.

6.5 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 competitive 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.

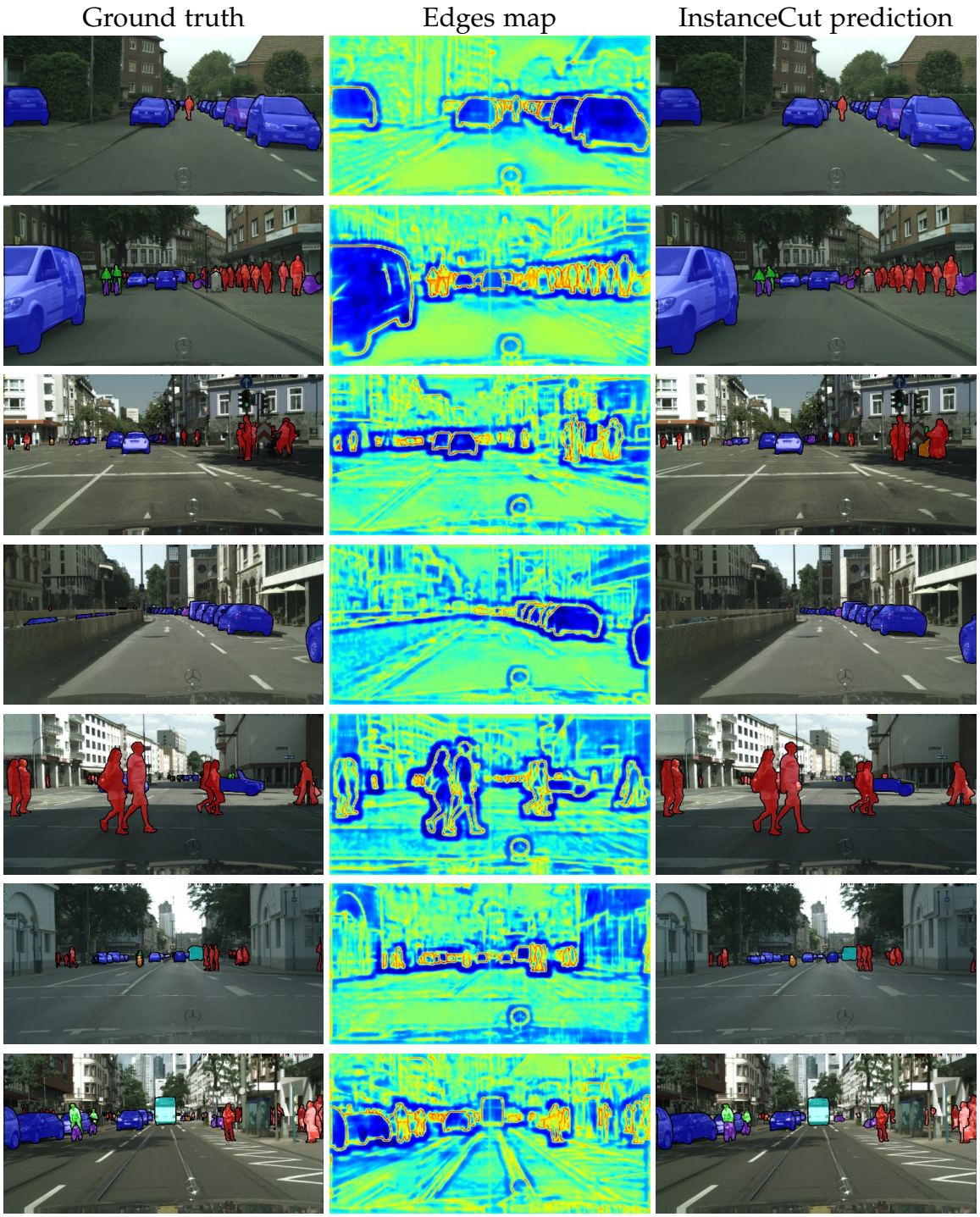


Figure 6.11: Curated difficult scene, where InstanceCut performs well. The left column contains input images with ground truth instances highlighted. The middle column depicts per-pixel instance-aware edge log-probabilities and the last column shows the results of our approach.

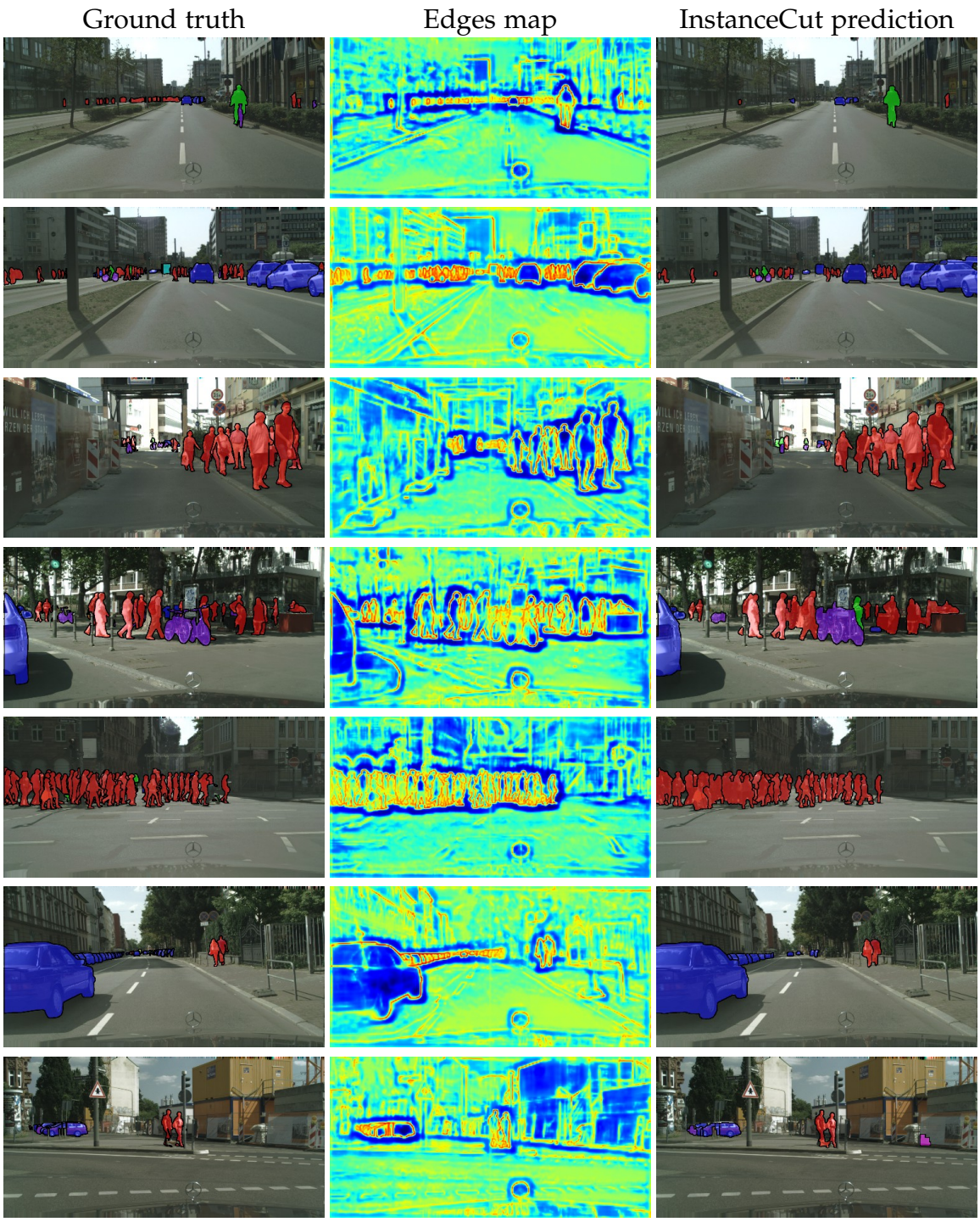


Figure 6.12: Failure cases. The left column contains input images with ground truth instances highlighted. The middle column depicts per-pixel instance-aware edge log-probabilities and the last column shows the results of our approach.

Part III

Higher Order Multicuts for Computer Vision

7 GEOMETRIC MULTIPLE MODEL FITTING WITH MODEL SELECTION

Contents

Multiple model fitting—the task of explaining noisy observations by models under certain assumptions—is a classical problem in pattern recognition. Its core challenges are two-fold: First, one needs to estimate the number of models along with the model parameters. Second, one usually has little prior knowledge on the noise level each individual model is observed with. While most previous approaches focus on either of the two challenges, we propose an pseudo-boolean formulation for the joint problem. Solutions to the proposed formulation fit multiple models to noisy data, while choosing, for each model, an appropriate noise level. Despite the problem being NP-hard, for the line fitting task, good solutions can be found in practice by a primal feasible heuristic we propose.

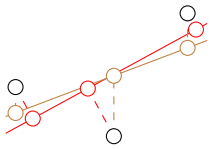
7.1 INTRODUCTION

We tackle the problem of joint model fitting and model selection, i.e., we attempt to i) fit an optimal number of models into noisy data and ii) simultaneously estimate the noise level of each model. Specifically, we attempt to recover the true set of models M, which are drawn from a pre-defined model class (for examples lines in the 2D plane). We assume that each model $m_i \in M$ is subject to an individual noise level σ_i.

The most traditional way to tackle the problem of estimating a model from noisy data is the random sampling consensus (RANSAC) approach (Fischler and Bolles, 1981), with the underlying idea of iteratively growing a model's inlier set by sampling points. It can be adapted to fitting a *pre-defined* number of multiple models for example by Zuliani *et al.* (2005). Preference analysis based methods such as (Magri and Fusiello, 2014, 2016; Tepper and Sapiro, 2017; Amayo *et al.*, 2018) sample a large but fixed set of model hypotheses and then optimize the inlier sets of these hypotheses based on preference matrices. Thus, they can recover any number of models from the initially sampled ones, but are restrained to this initial set.

In contrast, we follow the line of work of Tennakoon *et al.* (2016); Xiao *et al.* (2016); Purkait *et al.* (2017) and cast the multiple model fitting problem as a point grouping problem. Thus, our approach is based on local observations over which a global optimization is possible. Specifically, we consider residual errors r of k-tuples of points relative to the most plausible model given these points as local features.

We compute residual errors relative to a model fit using total least squares (TLS). Unlike least squares (brown), that minimize only axis-aligned residuals, total least squares (red) minimize the residuals as the distance between a point and its orthogonal projection onto the model.



Intuitively, these residuals are assumed to be informative about the points' likelihood of belonging together, i.e., belonging to the same model. Given this local evidence, we can phrase multiple model fitting as a graph decomposition problem, more specifically, as a *higher order minimum cost multicut problem* (Kim *et al.*, 2011; Kappes *et al.*, 2016). This formulation allows us to optimize over the number of models along with the model estimation itself. However, as any clustering-based technique, this formulation has to assume that certain hyperparameters of the models are fixed (e.g., width of a line). However, this is not realistic for real-world data, where each model might be captured with a unique noise level. One practically relevant example would be the fitting of lines on building facades, where the noise level of the data strongly depends on the objects' distances from the camera or 3D sensor. In such scenarios, traditional methods usually achieve best result when assuming the same average noise level for all models, which, however, might lead to under- or over-assignment of data points.

We solve this issue by a joint problem formulation. We generalize the higher order minimum cost multicut objective by introducing nodel labels (Chapter 6). Thus, we can formulate a pseudo-boolean program, which simultaneously groups points

and chooses one of the possible noise levels for each group. Because multicuts are NP-hard (Bansal *et al.*, 2004) and we expect to face large problem instances in practice, we focus on providing heuristic solutions. Therefore, we propose an extension of an efficient primal feasible heuristic (Keuper, 2017) to optimize our *higher order node labeling minimum cost multicut problem* instances. Our experiments on line fitting data show convincing results.

7.2 RELATED WORK

J-Linkage (Toldo and Fusiello, 2008) builds a preference matrix by assigning either 1 or 0 indicating if a point belongs to a hypothesis, determined by a threshold. Then they perform a greedy agglomerative clustering of points using the Jaccard distance, i.e. they group points with similar preference sets.To avoid hard 01-assignments, T-Linkage (Magri and Fusiello, 2014) relaxes the membership value and uses a different similarity metric (Tanimoto, 1957).

A separate line of research seeks to find low-rank representations of the preference matrix and thus discover the models. Robust Preference Analysis (RPA) (Magri and Fusiello, 2015) builds a symmetric kernel of pairwise similarities (measured with Tanimoto distance). Then it performs Robust PCA to obtain a rank-k representation and clusters points in this reduced space using Symmetric Non-negative Matrix Factorization. This approach requires the exact number of models to be provided and can be seen as a "robust spectral clustering".

Denitto *et al.* (2016) propose an approach to compute a sparse low-rank representation of the preference matrix using FABIA (Hochreiter *et al.*, 2010) and obtain *bi-clusters*. The latter being clustering both in row and column space of the matrix allows points to belong to multiple models.

Avoiding all intermediate operations, Non-negative Matrix Underapproximation (RS-NMU) (Tepper and Sapiro, 2017) directly finds a low-rank representation of the preference matrix and yields high-quality results. Notably, they make a further contribution to preference-based multiple model fitting by pointing out the importance of statistical observations. Their method gains robustness via a t-test that efficiently filters out statistically insignificant hypotheses. Whenever the application of such a test is possible in practice, it should be applied.

Magri and Fusiello (2016) cast multiple model fitting as a maximum set coverage problem, similar to sequential RANSAC. The key difference is that this approach finds a globally optimal coverage: Given an integer k, it selects exactly k subsets of hypotheses that cover the maximum number of points. As a by-product, this algorithm may leave certain (outlier) points uncovered.

Isack and Boykov Isack and Boykov (2012) also formulated a combinatorial optimization problem with a discrete label space of possible models parameters. They iteratively find inliers to the models by applying the efficient α-expansion algorithm Boykov *et al.* (2001) and then re-estimate model parameters and prune the redundant ones. In principle, it also possible to fit models with different noise

levels in their framework. Amayo et. al. Amayo *et al.* (2018) proposed an efficient primal-dual optimization method for a convex relaxation of the discrete energy of Isack and Boykov (2012). Recently, Barath and Matas Barath and Matas (2018) have introduced density modes into models' parameters space and showed that applying mean-shift Comaniciu and Meer (2002) can efficiently reduce the number of models.

Higher order clustering has previously been used in (Agarwal *et al.*, 2005; Jain and Madhav Govindu, 2013; Madhav Govindu, 2005; Tennakoon *et al.*, 2016; Xiao *et al.*, 2016; Purkait *et al.*, 2017; Wang *et al.*, 2018) for multiple model fitting. Higher order potentials are defined over k-tuples of points and a probability of the latter to belong together is computed based on the residuals to the sampled hypotheses. However, to perform clustering, these approaches transform higher order terms into pairwise potentials or use randomized techniques and apply spectral clustering. Thus, in principle, they also require the number of models to be given. In contrast, we work directly with the higher order potentials and perform correlation clustering so as to automatically recover the optimal number of clusters. However, the results presented in (Tennakoon *et al.*, 2016) and (Purkait *et al.*, 2017) can be seen as a motivation for the use of local evidence on larger than minimal sets, as employed in the proposed approach.

7.3 PRELIMINARIES

Recall, that a multicut of a graph $G = (V, E)$ is a subset of edges $MC \subseteq E$ that uniquely defines a decomposition of G. A characteristic function $y : E \mapsto \{0, 1\}$ assigns 1 to edges within the same component and 0 to edges that straddle distinct components. The multicut polytope Y_G can be fully described by linear, so called *cycle*, inequalities:

$$Y_G = \left\{ y \mid \forall C \in \text{cycles}(G), \forall e \in C : 1 - y_e \leq \sum_{f \in C \setminus \{e\}} (1 - y_f) \right\} \tag{7.1}$$

This definition is different from (1.2), because here 1 means join. Still, these inequalities ensure the consistency of a solution: If an edge along one path connecting two vertices is cut, there needs to exist a cut edge on all other paths connecting these two vertices. Chopra and Rao (1993) showed that it is sufficient to consider all chordless cycles, i.e., those correspond to the facets of Y_G.

Definition 11 For a graph $G = (V, E)$ and a cost function $c : E \mapsto \mathbb{R}$, that assigns a cost or reward for putting two vertices into the same component, the following integer linear program is called *minimum cost multicut problem* (Grötschel and Wakabayashi, 1989)

$$\arg\min_{y \in Y_G} \sum_{e \in E} c_e y_e. \tag{7.2}$$

It is important to note that at least some values of c *must be* negative to avoid a trivial solution. Yet, there are no assumptions on the number or (expected) size of components. Indeed, they are deduced from the solution. This makes multicuts useful, when the number of components needs to be estimated from data, which is the case in the multiple model fitting scenario.

Each cost c_e measures the agreement between corresponding vertices and has a probabilistic interpretation. Andres *et al.* (2011); Kappes *et al.* (2011) show that if $\mathbf{P}(y_e = 1)$ is a probability estimate of 2 vertices to be put in the same component, then solving the minimum cost multicut problem (7.2) with costs $c_e = \text{logit}(\mathbf{P}(y_e = 1)) = \log \frac{1-\mathbf{P}(y_e=1)}{\mathbf{P}(y_e=1)}$ is equivalent to finding a maximum likelihood decomposition. For probability $\mathbf{P}(y_e = 1) = 0.5$, c_e equals zero (random coin flip), while $c_e < 0$, when vertices are likely to belong together, and $c_e > 0$, when $\mathbf{P}(y_e = 1) < 0.5$. This allows to interpret solutions of (7.2) in terms of local probabilities.

Kim *et al.* (2011) proposed to extend the cost function c to include higher order terms. In this case, each hyperedge $e = \{v_0, \dots, v_k\} \subseteq V$ includes k vertices, and k is called the order of the multicut problem. Higher order edges can be represented as cliques of order k and (7.2) becomes a higher order problem by simply substituting $y_e = \prod_{\{u,v\} \in e} y_{uv}$, i.e., y_e is 1, iff all vertices of hyperedge e are put into the same component. In principle, the cost or reward c can include terms of different orders, but here we use a constant k. The probabilistic interpretation is straightforwardly extended by $\mathbf{P}(y_e = 1)$ denoting the probability of all vertices in e to belong to the same component.

In the following we describe how we compute costs c, introduce node labeling into higher order multicuts, and propose a primal feasible heuristic to infer into the emerging problem.

7.4 PROBLEM

We formulate the task of multiple model fitting as a higher order minimum cost multicut problem. Thus, we need to define an appropriate cost function taking into account the number of points needed to uniquely define a model as well as the expected noise level. If $k - 1$ points are required to define a model of the model class under consideration, the employed cost function needs to be at least of order k, such as to allow for the computation of model residuals. For example lines in the Euclidean space are uniquely defined by any two distinct points. Thus, three points allow for the estimation of a model and the computation of their residual model errors. In practice, since the space complexity of the higher order problem grows polynomially with the chosen problem order, we choose it to be exactly k for models defined with $k - 1$ points.

We assume, that data points belonging to the same model have been sampled from this model independently and with a unique noise distribution. A point's probability of belonging to a model is inversely proportional to its residual error w.r.t. to this model. Thus, we fit a model using TLS into each subset of vertices

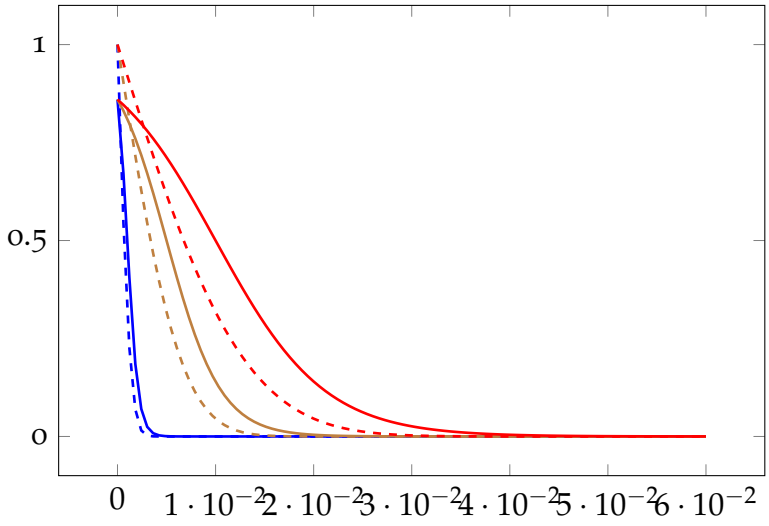


Figure 7.1: Complementary CDF of a Gaussian (dashed) and our approximation by logistic distribution with non-zero mean. For small values of σ it is quite accurate.

$U = \{u_i\}_{i=1}^k \in \binom{V}{k}$ and compute residuals $\{r_i\}_{i=1}^k$. Then the probability of vertices in U to belong to the same model is given as $\mathbf{P}(y_U = 1) = \phi(\sum_{i=1}^k r_i; \theta)$, where $\phi(\cdot;\theta) : \mathbb{R}^+ \mapsto [0,1]$ is some uncertainty model with hyperparameters θ.

Most commonly, one assumes the noise to be Gaussian normally distributed with standard deviation σ. For numerical reasons, we want to approximate this noise model with a logistic distribution with non-zero mean, i.e.,

$$f\left(\sum_{i=1}^k r_i; \mu, s\right) = \frac{e^{-\frac{\sum_{i=1}^k r_i - \mu}{s}}}{s \cdot \left(1 + e^{-\frac{\sum_{i=1}^k r_i - \mu}{s}}\right)^2}. \tag{7.3}$$

With mean $\mu = \sigma$ and scale $s := \frac{\sigma}{\theta_0}$ for a constant, positive θ_0, this approximation is reasonable for small σ, as can be seen in Fig. 7.1. This, accordingly, motivates to choose $\phi(\cdot;\theta)$ as a logistic complementary cumulative distribution function

$$\phi\left(\sum_{i=1}^k r_i; \theta\right) = \left(1 + e^{-\theta_0 + \theta_1 \sum_{i=1}^k r_i}\right)^{-1}, \tag{7.4}$$

with $\theta_1 = \frac{\theta_0}{\hat{\sigma}}$ for an appropriate estimation $\hat{\sigma}$ for the standard deviation σ. After the logit transformation, the cost function is linear: $c := \text{logit}(\phi(\sum_{i=1}^k r_i; \theta)) = -\theta_0 + \theta_1 \sum_{i=1}^k r_i$.

Parameters θ in (7.4) can be interpreted as follows: θ_0 is the maximum reward the optimization can get for grouping all vertices $\{u_i\}_{i=1}^k$ together (in case $\sum_{i=1}^k r_i = 0$). θ_1 defines the threshold when reward turns into penalty, i.e., when the cost function crosses the x-axis. It should be chosen inversely proportional to the standard deviation σ. Obviously, setting $\hat{\sigma}$ too large will result in grouping all points together, which

is probably not desired. Therefore $\hat{\sigma}$ needs to be tuned for optimal performance by grid search.

Interpreting these parameters in terms of mean μ and scale s, μ defines the boundary corresponding to probability of 0.5 and s defines the steepness of transition through this boundary. Indeed, in our experiments we noticed that setting $\hat{\sigma} \in [2\sigma, 3\sigma]$ gives the best results, assuming the model noise is zero-mean Gaussian distributed with standard deviation σ. This is not surprising, because 99.7% of points sampled from a Gaussian (inliers to a model) are expected to be found within 3σ range.

Having computed costs c_U for each subset U we can plug them directly into (7.2) and solve with the heuristic proposed by Keuper (2017). The solution will be a decomposition of points such that those points likely to belong together under uncertainty $\phi(\cdot;\theta)$ will be grouped and each such group corresponds to one model. We refer to this baseline as HO-MP further on.

7.4.1 Model Selection

Model fitting is a rather ill-posed problem and it requires certain assumptions on the models being fit. For example, most methods in the literature require certain hyperparameters (e.g., threshold value for a point to belong to a model) to be specified beforehand. In our case, one specifically needs to determine $\hat{\sigma}$ in eq. (7.4). Moreover, in real data, it is unlikely that all models have the same noise level. However, if one lets the optimization choose such thresholds (noise levels), it will always choose the largest possible value (indeed, infinity) as this will lead to the highest local probabilities and thus the lowest total objective value.

In our approach such a hyperparameter is $\hat{\sigma}$, which directly corresponds to the expected noise level and needs to be set correspondingly (θ_0 can be fixed). Instead of picking $\hat{\sigma}$ by hand, we propose to take a discrete finite set of possible hyperparameters $\{\hat{\sigma}^0, \hat{\sigma}^1, \ldots, \hat{\sigma}^{\ell-1}\}$ and would like the optimization to choose one for each model. In Chapter 6 we extended the multicut framework by including node labels which allows to specify different costs depending on the class labels of the nodes. The combination of this formulation with higher order minimum cost multicuts allows us to specify different costs c_U for each subset $U \subseteq V$ and each hyperparameter $\hat{\sigma}$. Thus, we propose the following

Definition 12 For any simple connected graph $G = (V, E)$ and a set of labels $\mathcal{L}$, unary costs $c : V \times \mathcal{L} \mapsto \mathbb{R}$, some integer $k \in \mathbb{N}, k > 1$, and costs $c : \binom{V}{k} \times \mathcal{L} \mapsto \mathbb{R}$ written below is the *minimum cost higher order node labeling multicut problem* (HO-

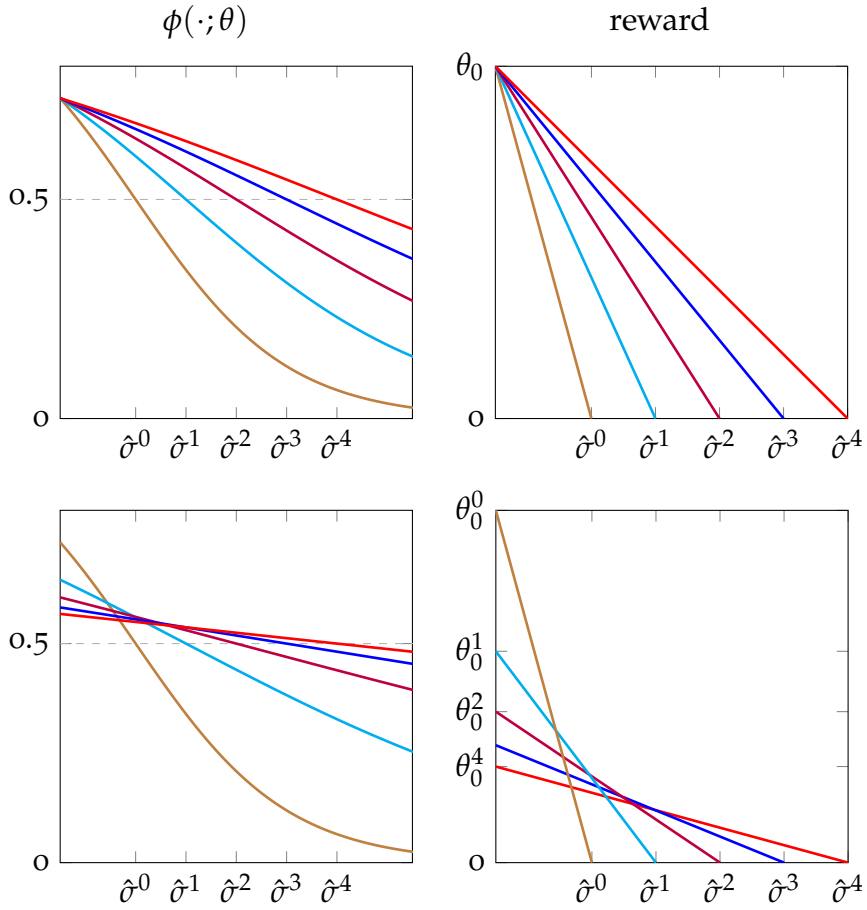


Figure 7.2: **(first row)** depicts probabilities $\phi(\cdot;\theta)$ and corresponding reward parts of cost functions c for different $\hat{\sigma}$, but the same θ_0. Functions corresponding to larger $\hat{\sigma}$ have more probability mass above 0.5 and, respectively, more reward. **(second row)** after normalization all functions have the same probability mass above 0.5 and the same amount of reward.

NLMP) of order k:

$$\begin{aligned} \underset{x\in\{0,1\}^{V\times\mathcal{L}},y\in Y_G}{\arg\min} & \sum_{v\in V}\sum_{l\in\mathcal{L}} x_{vl}c_{vl} \\ & + \sum_{U\in\binom{V}{k}}\sum_{l\in\mathcal{L}}\prod_{\{u,v\}\in\binom{U}{2}} y_{\{u,v\}}\prod_{v\in U} x_{v,l}\; c_{U,l} \qquad (7.5) \\ & \text{s. t.} \quad \forall v\in V: \quad \sum_{l\in\mathcal{L}} x_{vl} = 1 \qquad (7.6) \end{aligned}$$

We can associate $\mathcal{L} = \{\hat{\sigma}^0, \hat{\sigma}^1, \ldots, \hat{\sigma}^{\ell-1}\}$ and (7.6) ensures that each vertex can only belong to a model with a specific $\hat{\sigma}$. $\prod_{v\in U} x_{v,l}$ makes sure that vertices in U can be clustered together, iff they have the same label, i.e., the same $\hat{\sigma}$.

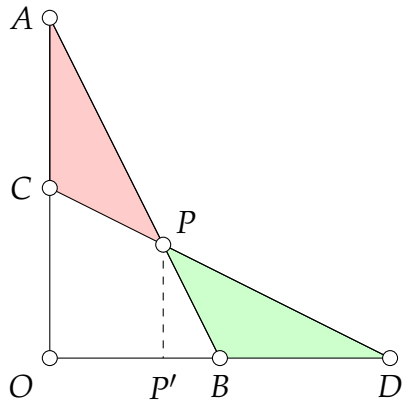


Figure 7.3: Depicted above is geometrical interpretation of our proposed normalization of cost functions OAB and OCD. If observations can be explained by a hyperparameter corresponding to OB, then choosing a larger one will result in loss of reward equal to the area of ACP. However, if there are enough data points with residuals in $[P', D]$, it will be compensated by the area of PBD. And because, by construction, areas ACP and PBD are equal, the total amount of reward remains constant.

7.4.2 Cost Functions Normalization

Solving (7.5) directly will lead to the above mentioned problem: the optimization will pick the largest $\hat{\sigma}$ possible. Recall, that θ_0 can be arbitrary and we fix it to 1 for all the values of $\theta_1 = \frac{\theta_0}{\hat{\sigma}}$. Fig. 7.2 depicts reward parts of several cost functions (multiplied by -1 for visualization purposes) corresponding to different $\hat{\sigma}$, which is a simple linear function $c_{U,l} = \text{logit}(\phi(\sum_{i=1}^{k} r_i; \theta^l)) = -\theta_0^l + \frac{\theta_0^l}{\hat{\sigma}^l} \sum_{i=1}^{k} r_i$. It is visible from the figure, that for the same residuals each larger $\hat{\sigma}$ will provide more reward and all the points will be assigned to the models with the largest $\hat{\sigma}$. To alleviate this problem we propose to normalize all the cost functions to have the same integral reward, i.e., area under the curve. We compute $A^0 = \frac{\theta_0^0 \hat{\sigma}^0}{2}$ and for all $l \geq 1$ we leave $\hat{\sigma}^l$ fixed and adjust $\theta_0^l = \frac{2A^0}{\hat{\sigma}^l}$. Thus, $\forall l \in \mathbb{N}, l \geq 1 : A^l = A^0$ and all cost functions have the same amount of reward. Fig. 7.3 shows the intuition: starting with the smallest $\hat{\sigma}$ the optimization may choose to switch to larger ones, only if such a move is supported by enough points with the corresponding residuals to compensate for the lost reward.

7.5 LOCAL SEARCH ALGORITHMS

Bansal *et al.* (2004) showed that the multicut problem is NP-hard; even for planar graphs (Voice *et al.*, 2012). This makes multicut-based frameworks correspondingly NP-hard as well. Recently, Fukunaga (2018) proposed an LP-based pivoting algorithm, that gives an $\mathcal{O}(k \log n)$ approximation for the higher-order multicut problem.

Algorithm 3: Kernighan-Lin Algorithm. The function update is given in Algorithm 4.

Data: weighted undirected higher-order graph $G = (V, E, c)$ of order k, node labels set $\mathcal{L}$, starting y^0 and x^0
Result: 01-edge labelings y and x

1 **while** $t < max\_iter$ **and** $(y^t \neq y^{t-1}$ **or** $x^t \neq x^{t-1})$ **do**
2 **foreach** $(a, b) \in adjacent\_components(y^{t-1})$ **do**
3 $(y^t, x^t) \leftarrow \text{update}(G, a, b)$
4 **foreach** $a \in components(y^t)$ **do**
5 $(y^t, x^t) \leftarrow \text{update}(G, a, \emptyset)$

However, it does not include node labeling and is still slow for large instances. A separate line of research focuses on local search algorithms, that do not provide any guarantees on runtime and quality of the solutions, but tend to perform well in practice; in Chapter 3 we provide a comparative study.

Kernighan and Lin (1970) proposed one such algorithm which is based on locally optimal moves of vertices between neighbouring components. In Chapter 2 we extended it to lifted multicuts, in Chapter 6 to node labeling multicuts, and Keuper (2017) to higher order multicuts. Here, we further extend it to higher order node labeling multicuts. The algorithm gets an initial decomposition and node labeling and consists of two parts. The first (Alg. 3) iteratively updates the boundary between pairs of neighbouring components (lines 2-3) or introduces new components (lines 4-5) into existing.

Alg. 4 receives graph G and two neighbouring components A and B from Alg. 3 (B can be an empty set). In each iteration (line 6), it finds a vertex whose move (with possible relabeling) to the other set will result in the largest decrease of objective (7.5). An efficient implementation is to pre-compute $\forall v \in \{A \cup B\}, \forall l \in \mathcal{L}$ these values (line 2) and then update them after every move. A special operation (line 3) computes the decrease of objective of joining two components A and B under the same label. This allows two smaller models to merge into a bigger one, if this is beneficial.

In the main loop it picks a vertex v^* to move and its new (possibly the same) label (line 7), records this move (line 8) and then updates pre-computed values of all other vertices that share an edge with v^*. There are different cases captured by lines 13-22; w.l.o.g. we assume that v^* is moved from A to B. A special condition in line 19 prevents moving of vertices if their label is different from the labels of vertices in the other component. We compute the cumulative total gain (line 23) and threshold on the maximum one (line 25). Note, that this way we allow sub-optimal (non-decreasing) individual moves in the hope to leave local minimum and arrive at a better one. We reverse the moves (possibly all), that did not lead to a decrease in objective in line 28.

Algorithm 4: Function update

Data: weighted undirected higher order graph $G = (V, E, c)$ of order k, a pair of partitions A and B
Result: 01-vectors y and x

$\Omega \leftarrow$ find_boundary_nodes(G, A, B)
$D^{\Omega \times \mathcal{L}} \leftarrow$ compute_differences(G, A, B)
$\Delta_{join} \leftarrow$ compute_gain_from_joining(G, A, B)
$S_0 = 0$ // cumulative gain
$M = []$ // record moves
for $i \leftarrow 1$ **to** $|\Omega|$ **do**
  $(v^*, l^*) \leftarrow \text{argmax}_{v,l}\ (D^{\Omega \times \mathcal{L}})$
  M.push_back(v^*, l^*)
  $l' \leftarrow$ old_node_label(x, v^*)
  foreach $e \in G.edges(v*)$ **do**
    $U \leftarrow$ G.nodes(e)
    foreach $l \in \mathcal{L}$ **do**
      if $|\{U \setminus v^*\} \cap A| = k - 1$ **then**
        foreach $w \in \{U \setminus v^*\} \cap A$ **do**
          $D_{w,l} \leftarrow D_{w,l} - c_{U,l'}$
      else if $|U \cap B| = k - 1$ **then**
        foreach $w \in U \cap B$ **do**
          $D_{w,l} \leftarrow D_{w,l} + c_{U,l^*}$
      if $|\{U \setminus v^*\} \cap A| = 1$ **and** $l = l^*$ **then**
        $D_{w,l} \leftarrow D_{w,l} - c_{U,l^*}$
      else if $|U \cap B| = 1$ **then**
        $D_{w,l} \leftarrow D_{w,l} + c_{U,l'}$
  $S_i \leftarrow S_{i-1} + D_{v^*,l^*}$
  $\Omega \leftarrow$ update_boundary(v^*, G, A, B)
$i^* \leftarrow \text{argmax}_i S_i$ // threshold max gain
if $\Delta_{join} > S_{i^*}$ **and** $\Delta_{join} > 0$ **then**
  $(y, x) \leftarrow$ join_components(A, B)
$(y, x) \leftarrow$ rollback_moves_after(M, i^*)

7.6 EXPERIMENTS

We evaluate our approach on line fitting data, which has always been a standard problem for multiple model fitting. We start with the data of Toldo and Fusiello (2008): it consists of 3 problem instances each containing a number of lines. Important feature of this data is that all lines in each instance have the same Gaussian distribution. To facilitate a thorough evaluation on more instances, we propose a novel line fitting dataset. It consists of 100 instances, where each line is perturbed by a unique and random noise level. In this scenario, our method, which is able to choose an individual noise level for each line, outperforms previous methods significantly.

7.6.1 Experimental setup

Here, we compare against J-Linkage (Toldo and Fusiello, 2008), T-Linkage (Magri and Fusiello, 2014), RansaCov (Magri and Fusiello, 2016), and RS-NMU (Tepper and Sapiro, 2017), whose code is publicly available. It has been shown in (Toldo and Fusiello, 2008) that J-Linkage outperforms multiRANSAC (Zuliani *et al.*, 2005), residual histogram analysis (Zhang and Kosecka, 2006), and mean-shift (Comaniciu and Meer, 2002).

As a performance measure, we use the widely accepted (Magri and Fusiello, 2014; Toldo and Fusiello, 2008; Magri and Fusiello, 2016; Tepper and Sapiro, 2017) misclassification error (ME), that is the ratio of the misclassified points over the total number of points. The classification is performed by matching the predicted and the ground truth clusters such that the number of misclassified points is minimized using the Hungarian algorithm (Kuhn, 1955) for minimum weight bipartite matching. Then the point is considered to be correctly classified if its cluster label corresponds to the matched ground truth.

Xiao *et al.* (2016) notice that it is impossible to estimate the number of models present in the data without introducing additional stronger assumptions. For example, if we assume that points lying on a line constitute a model, then many more lines can be discovered in the presence of heavy background noise. Knowing the ground truth number of models K, one can simply take the top-K largest clusters and assign the rest to noise as in (Toldo and Fusiello, 2008). Another approach would be to apply an efficient filtering procedure proposed by Tepper and Sapiro (2017) to keep only statistically significant models. In this paper we match the predicted models to the ground truth so as to minimize the misclassification error. Some algorithms, e.g., RansaCov, need the exact number of models *before* optimization.

7.6.2 Line fitting data from Toldo and Fusiello (2008)

This line fitting dataset consists of 3 instances—called *Stairs4*, *Star5*, and *Star11*—in which lines are arranged in corresponding shapes. Points sampled from lines are

	Misclassification error [%]								
	J-Linkage		T-Linkage		RansaCov		RS-NMU	HO-MP	HO-NLMP
	original	full	original	full	original	full			
Stairs4	11.9±2.3	10.6	31.1±3.2	8.6	4.0±1.5	4.0	3.0	6.2	5.2
Star5	13.2±3.1	5.4	14.8±1.9	7.4	3.3±0.8	2.0	1.0	2.4	3.0
Star11	21.8±2.6	12.0	22.4±1.5	9.55	3.3±0.6	2.73	1.91	2.82	3.64
	Runtime [s]								
Stairs4	12/4	58/30	16/43	58/120	11/8	58/42	58/36	9/5	88/115
Star5	12/4	54/15	15/38	54/54	11/9	54/15	54/11	9/6	88/153
Star11	22/15	294/112	22/188	294/1652	24/19	294/113	294/210	98/51	927/643

Table 7.1: Results on synthetic line fitting data from Toldo and Fusiello (2008). The total runtime is split into the sampling and solving time.

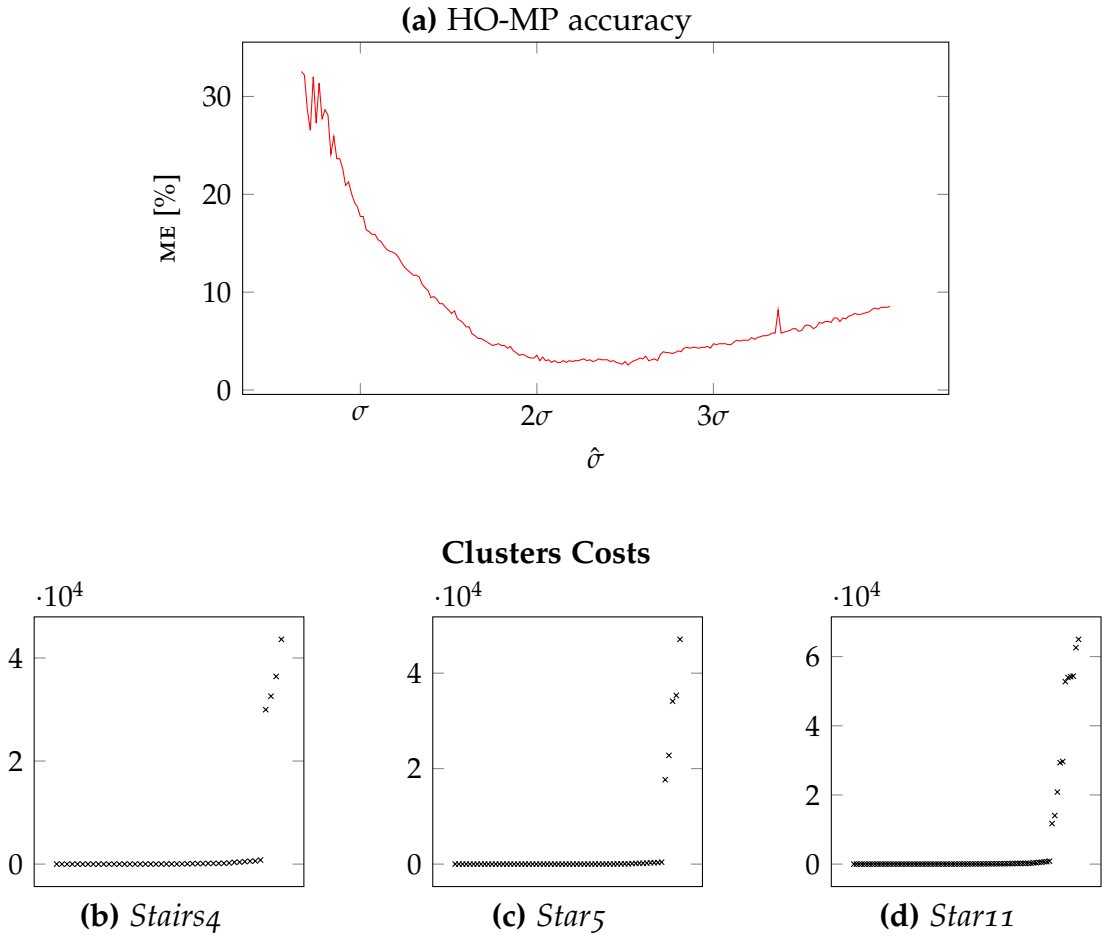
displaced by the same noise in each instance; here, 3σ corresponds to the actual width of a line. Half of the points are uniform background noise. The largest instance *Star11* has 1100 points in total. We use the 3rd order multicut problem and sample, correspondingly, all $\binom{|V|}{3}$ triplets of points and fit a line into each triplet using TLS. For the competing methods we used the best settings provided by the authors and chose their hyperparameters by grid search for optimal performance.

Results and discussion The results are compiled together in Tab. 7.1. In their default implementation J/T-Linkage and RansaCov perform a random sampling of a small number of hypotheses, which leads to a considerable variance ('original' in Tab 7.1). For our method we perform a complete sampling, thus, for fairness, competing methods should be evaluated using full sampling. During the experiments, we noticed that the statistical filtering of Tepper and Sapiro (2017) greatly reduces the number of irrelevant hypotheses. Therefore, we use full sampling with such filtering ('full' in Tab 7.1). This has a two-fold benefit: 1) the mean error is considerably reduced, 2) there is no more variance in the results.

Tab. 7.1 shows that, when the data has the same noise level, methods with one model naturally perform better. However, they still require hand-tuning or grid search for parameters. For example, Fig. 7.4 (a) shows how ME of our HO-MP approach changes depending on $\hat{\sigma}$. Although the shape is almost convex, it requires the user to try a number of different parameters (the same for all other approaches).

Having found the best parameter $\hat{\sigma}^*$ for our HO-MP approach for each instance, we construct instances of HO-NLMP with 9 different parameters $\hat{\sigma} = \hat{\sigma}^* \pm i\delta, \forall i \in \{0, \dots, 4\}$, i.e., the optimal one and other values to confuse the optimization. For *Stairs4* and *Star11* we took $\delta = 0.0025$ and for *Star5* we took $\delta = 0.005$, because its original noise level is twice higher. Our results in Tab. 7.1 show that even in this case the ME is rather close to the hand-tuned variant.

As other methods, without prior knowledge of the number of models, our



(b) *Stairs4* **(c)** *Star5* **(d)** *Star11*

Figure 7.4: **(a)** ME on *Star11* as a function of the parameter $\hat{\sigma}$; the latter is shown in terms of the original noise level of the data σ. **(b)–(d)** sorted absolute costs per cluster in the optimal solutions. For each problem the corresponding 4, 5, and 11 most expensive clusters are clearly separated form the lower cost clusters, the latter being lines fit into background noise.

approach produces an over-segmentation of the data, i.e., it finds small models in the background noise. However, if we plot the costs of clusters in our solution, for example, Fig. 7.4 (b)–(d), it is easy to spot the good clusters, that have noticeably larger costs.

7.6.3 Our novel line fitting dataset

To facilitate a clean evaluation of our and competing methods without possible overfitting to few problem instances and specific noise levels, we propose here a novel line fitting dataset. It provides: 1) 100 instances, which allows a more statistically significant evaluation, 2) from 3 up to 10 lines in each instance with a unique, random noise level from the range $[0.005, 0.025]$. Some examples are depicted in Fig. 7.5.

	Misclassification error [%]					
	J-Linkage	T-Linkage	RansaCov	RS-NMU	HO-MP	HO-NLMP
Mean	14.44	15.45	6.51	6.65	6.82	4.65
Median	13.96	16.01	6.21	5.47	6.83	4.11

Table 7.2: Results on our novel line fitting dataset.

We use exactly the same setting for our and competing approaches as in the previous section. We construct instances of HO-NLMP with 5 parameters $\hat{\sigma} = 0.01 + 0.01i, \forall i \in \{0, \ldots, 4\}$. The results are compiled together in Tab. 7.2. The absolute mean ME are significantly higher than on the data from Toldo and Fusiello (2008), but the relative rankings of algorithms remain.

Note that we did not tune parameters of all methods for each problem instance, but rather for the entire dataset. As a result, for all the methods a mean parameter showed the best performance. Our HO-NLMP approach is able to choose a suitable noise level for each line individually and therefore significantly improves accuracy. This shows that, in case each model has a unique nose level, it is beneficial to let the optimization choose an individual parameter for each model.

7.7 CONCLUSIONS

We presented an approach to model fitting using higher order minimum cost multicuts. It groups points that are likely to belong to the same model based on local feature estimates. We also presented a novel problem setting, when the optimization algorithm may choose an optimal parameter for each model. This closer corresponds to real world data, where different models may have different noise levels. We introduced higher order node labeling multicut problem to do so. As it is NP-hard, we proposed an efficient primal feasible heuristic. We showed that our approach outperforms previous methods in cases when models have different noise levels. The main disadvantage of our method is its computational and memory complexity.

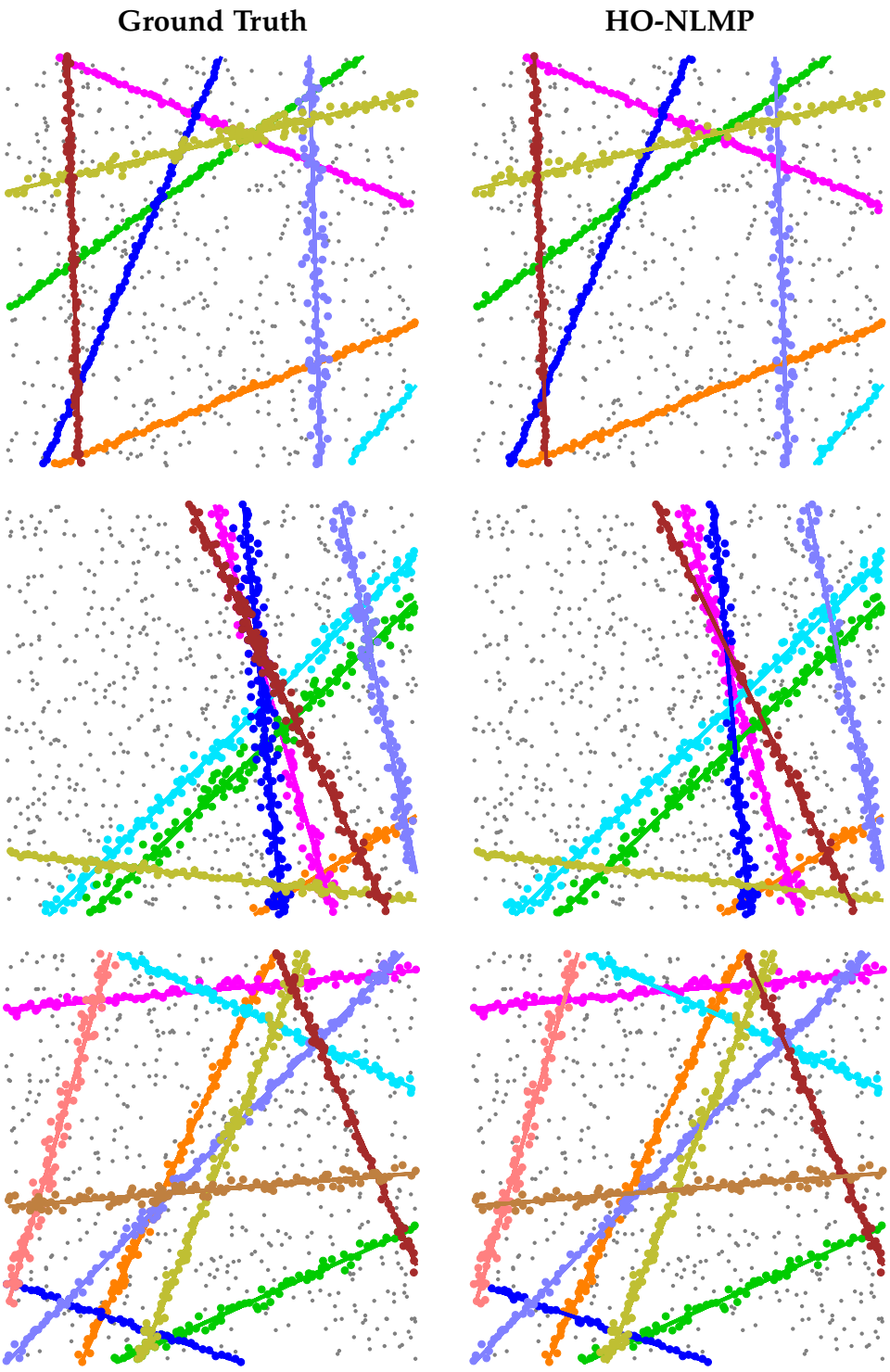


Figure 7.5: Depicted are our results for our novel line fitting data set where each line has a unique noise level.

8 CONCLUSIONS

In this thesis we studied multicuts that require only local probability estimates, but can find a globally consistent decomposition. The main advantage of multicuts is that the number of components is not required to be known in advance, but can rather be deduced from the solution. However, this comes at the cost of the minimum cost multicut problem being NP-hard to solve exactly (Bansal *et al.*, 2004). Also, the multicuts problem allow to specify a cost or a reward only for direct neighbours in the graph and the objective does not include a unary term. In this thesis, we proposed extensions and solutions that tackle exactly these problems.

We introduced new cost edges that allow to specify for non-neighbouring nodes a cost or a reward for being put into distinct components. The crucial property is that these additional edges do not change connectivity on the graph, thus preserving the feasible set of solutions. We term this new formulation the minimum cost lifted multicut problem. Although, in principle, the costs for the new edges can be computed in any probabilistically consistent way, we proposed an automatic construction called probabilistic geodesic lifting. For each pair of non-neighbouring nodes it finds the most probable path connecting them and the total probability is computed as a product of probabilities on the path. We showed that the new formulation significantly improve the results in image and mesh segmentation.

Because multcuts are NP-hard to solve, they have mostly been applied to superpixel graphs. To this end we defined and implemented two local search algorithms, that do not provide any guarantees on the runtime or the quality of the solutions. We performed an intensive analysis of them and other solvers for the multicut problem in terms of runtime and metric/objective value. We applied it to graphs of different size (from hundreds to more than 10^5 nodes) and different properties (e.g., complete and sparse, planar and non-planar graphs with unitary and general costs). We showed that our algorithms strike a favourable spot between runtime and solution quality. The improvement in runtime allowed us to apply multicuts to big social network graphs. This also enabled us to create a tool for online video segmentation: This is a client-server application, where after a pre-processing of a video into supervoxels a user draws scribbles on individual frames and a time-coherent segmentation is computed on-the-fly on the server. This might enable some novel methods of working, for example, collaborative video editing.

We introduced node labels into the multicut framework, that enabled us to optimize jointly for a node labeling (e.g., node classification) and a graph decomposition. We extended our local search algorithms, so that the inference time into emerging models remained feasible. This novel formulation allowed us to solve three distinct computer vision tasks using the same algorithms. In multiple person pose estimation

node labels correspond to possible type of body joints and the task is to fit a skeleton into an unknown number of persons present in an image. Due to improved runtime we were able to achieve better results, than the prior work (Insafutdinov *et al.*, 2016), using exactly the same input data. Success in application to images allowed us to develop a method for pose tracking in videos (Insafutdinov *et al.*, 2017), where estimated poses help tracking of people and tracking in turn helps to locate joints more accurately. In multiple object tracking nodes correspond to putative detections connected by edges inside and across frames and the task is to decompose the graph so as to recover tracks corresponding to objects, while possibly getting rid of false detections. We showed improvement over the prior work (Tang *et al.*, 2016) using exactly the same data.

Another important application, that has recently gained considerable attention, is instance-aware semantic segmentation. Here the task is to not only assign a semantic label to each pixel of an image (this task is almost solved by modern FC-CNNs), but to group pixels corresponding to each instance of each class. We formulate this as node labeling multicut problem and apply our local search algorithms for inference. We trained a fully convolutional neural network to predict instance boundaries that are combined in our multicut together with the unaries obtained from a well-performing semantic labeling CNN and got competitive results on CityScapes (Cordts *et al.*, 2016).

We also tackled the problem of lineage tracing, i.e., the task of tracking cells in microscopy data together with building their parental hierarchy. The problem is complicated by the fact that we do not rely on any detections of cells, but rather perform segmentation ourselves. We formulated it as a multicut-based ILP, that can be seen as a more constrained lifted multicut problem. Unfortunately, due to new types of constraints we could not apply our efficient heuristics and had to resolve to a branch-and-cut approach. We proposed efficient separating procedures, as well as a canonical LP relaxation to produce lower bounds. The obtained solutions compared favourably to competing methods, however, runtime was an issue. A follow-up work by Rempfler *et al.* (2017) proposed a tighter description of the polytope and a variant of KLj with optimal branching, that greatly reduced the runtime.

Multicuts are not limited to only pairwise terms. Indeed, one can define a higher-order multicut problem of any order (Kappes *et al.*, 2016), or a problem that even combines several different orders. To this end, we applied it to the task of multiple model fitting, where one needs to explain observations by a set of models. We constructed higher order problems by computing the probability of k-points to belong together. For this, we fit a model into k data points using TLS and use their residuals w.r.t. this model as local features. However, this approach (same as most other methods) assumes, for each model, hyperparameters fixed, e.g., width of a line, which is unlikely to be true in real data. To this end, we propose a higher order node labeling multicut problem that is able to simultaneously fit models and choose a proper parameter for each model. The latter greatly improves results on our novel line fitting data set, where each line has a unique noise level.

Future perspectives One of the most important issues of multicut-based framework is that they are NP-hard to solve, that limits their possible application. Therefore, the most important direction of future research, as we think, is about improving algorithms. We see several possible ways to go:

1) Better understanding of the polytopes of emerging problems is utterly important to create efficient solvers. For example, Horňáková *et al.* (2017) propose tighter inequalities for lifted multicuts, that dramatically decrease runtime of branch-and-bound algorithms.

2) We believe, that one of the most promising directions for future research is about persistency and re-parametrization of edge values as done by Alush and Goldberger (2012, 2015); Swoboda and Andres (2017); Lange *et al.* (2018). Persistency allows to specify some edge values and guarantee that they will retain their values also in the optimal solution. This decrease the search space of the solvers and improve the runtime. Lange *et al.* (2018) show, that by computing a re-parametrization of the multicut problem one can obtain both better solutions (with the heuristics proposed in this thesis) and shorter runtime. In principle, this might allow to come up with new local search algorithms that might make use of such information.

3) While improvements of branch-and-cut solvers (Horňáková *et al.*, 2017) make it feasible to apply multicut-based approaches to problems of increasingly larger size, most real-world problems are still too big. Thus, we think, that local search algorithms will play an important role in various applications. KLj heuristic, proposed in Chapter 2, performs an exhaustive search of vertices to move. It might be possible to improve the runtime by considering only a subset of such possibilities using, for example, persistency information. Also, KLj processes a pair of partitions at a time, so parallelization might be beneficial. However, to ensure correctness of results and to prevent conflicts between multiple threads a 4-coloring of the graph is necessary, which is NP-hard for general graphs.

4) A separate line of research considers meta-heuristics, i.e., general algorithms that have some multicut solver as their core component. Beier *et al.* (2015, 2016) proposed to contract graph, solve multicut problem, and expand the solution back to the original graph. Re-parametrization of edge weights and faster lower bounds (Lange *et al.*, 2018) might help here. Pape *et al.* (2017) proposed to solve multicuts on intersecting blocks of the graph and then merge individual solutions together; persistency information may provide better global solutions in this case.

5) Certainly, an interesting research question is whether it is possible to introduce multicuts into a deep learning framework, at least for certain graph type, e.g., planar graphs. The main obstacle here is that the number of components is not known a priori and the multicut constraints make the whole framework non-differentiable. The former problem can be addressed in specific applications by first counting the number of components in the graph as is done, for example, by Newell *et al.* (2017). There they use some strong features, for example, heads of people, to count them. But this approach fails in scenarios, when such an estimate is difficult or impossible. It is possible to introduce multicut constraints into the loss function by assigning high penalty for infeasible 01-edge labelings. However, in this case the loss might

become exponentially large for general graphs due to exponentially many cycles in such graphs. But even for complete graphs it will contain $\binom{V}{3}$ additional terms, that makes evaluation of cost function at each forward pass rather expensive. Another question is how to mine positive and negative examples for learning.

On the application side we see a possibility to use multicuts for non-maximum suppression, which is a crucial step in a variety of applications, most importantly in object detection. It the latter one gets as output from a neural network a score for each pixel to be a center for a bounding box of an object. Normally, way more bounding boxes are predicted, than there are objects in an image. The usual way to filter out irrelevant predictions is to keep only one maximum score in a certain small window. However, in case of highly overlapping objects it is impossible to distinguish cases when two local maxima correspond to two different objects or one of them is a false positive. By estimating a pairwise probability for each pair of predicted bounding boxes to depict the same object, we may resolve such difficult cases by means of multicuts.

LIST OF FIGURES

LIST OF TABLES

BIBLIOGRAPHY

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk (2012). SLIC Superpixels Compared to State-of-the-Art Superpixel Methods, *IEEE Trans. Patt. Anal. and Machine Intell.*, vol. 34(11). 49

R. Adams and L. Bischof (1994). Seeded Region Growing, *TPAMI*, vol. 16(6), pp. 641–647. 56

S. Agarwal, J. Lim, L. Zelnik-manor, P. Perona, D. Kriegman, and S. Belongie (2005). Beyond pairwise clustering, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2005*. 114

B. Aigouy, R. Farhadifar, D. B. Staple, A. Sagner, J.-C. Röper, F. Jülicher, and S. Eaton (2010). Cell flow reorients the axis of planar polarity in the wing epithelium of Drosophila., *Cell*, vol. 142(5), pp. 773–786. 58, 71

A. Alush and J. Goldberger (2012). Ensemble Segmentation Using Efficient Integer Linear Programming., *TPAMI*, vol. 34(10), pp. 1966–1977. 5, 14, 31, 57, 129

A. Alush and J. Goldberger (2015). Hierarchical Image Segmentation Using Correlation Clustering, *Neural Networks and Learning Systems, IEEE Transactions on*, vol. PP(99), pp. 1–10. 5, 31, 129

F. Amat and P. J. Keller (2013). Towards comprehensive cell lineage reconstructions in complex organisms using light-sheet microscopy., *Dev. Growth Differ.*, vol. 55(4), pp. 563–578. 56

F. Amat, W. Lemon, D. P. Mossing, K. McDole, Y. Wan, K. Branson, E. W. Myers, and P. J. Keller (2014). Fast, accurate reconstruction of cell lineages from large-scale fluorescence microscopy data., *Nature Methods*, vol. 11(9), pp. 951–958. 56, 58

P. Amayo, P. Piniés, L. M. Paz, and P. Newman (2018). Geometric Multi-Model Fitting with a Convex Relaxation Algorithm, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018*. 112, 114

B. Andres (). *Graph Processing Library*. 20, 21

B. Andres, T. Beier, and J. H. Kappes (2012a). OpenGM: A C++ Library for Discrete Graphical Models, *CoRR*, vol. abs/1206.0111. 15

B. Andres, J. H. Kappes, T. Beier, U. Köthe, and F. A. Hamprecht (2011). Probabilistic Image Segmentation with Closedness Constraints, in *ICCV 2011*. 2, 3, 5, 14, 18, 33, 34, 57, 115

B. Andres, T. Kröger, K. L. Briggman, W. Denk, N. Korogod, G. Knott, U. Köthe, and F. A. Hamprecht (2012b). Globally Optimal Closed-surface Segmentation for Connectomics, in *ECCV 2012*. 5, 14, 31, 32, 33, 35, 57

B. Andres, J. Yarkony, B. S. Manjunath, S. Kirchhoff, E. Turetken, C. Fowlkes, and H. Pfister (2013). Segmenting Planar Superpixel Adjacency Graphs w.r.t. Non-planar Superpixel Affinity Graphs, in *EMMCVPR 2013*. 14, 17, 57

M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele (2014). 2D Human Pose Estimation: New Benchmark and State of the Art Analysis, in *CVPR 2014*. 78, 93, 94, 95

P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik (2011). Contour Detection and Hierarchical Image Segmentation, *TPAMI*, vol. 33(5), pp. 898–916. 13, 14, 15, 19, 21, 22, 24, 28, 56, 98, 100, 102

P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik (2014). Multiscale Combinatorial Grouping, in *CVPR 2014*. 15, 22, 23, 24, 98

A. Arnab and P. H. Torr (2017). Pixelwise instance segmentation with a dynamically instantiated network, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017*. 79, 104

Y. Bachrach, P. Kohli, V. Kolmogorov, and M. Zadimoghaddam (2013). Optimal Coalition Structure Generation in Cooperative Graph Games, in *AAAI 2013*. 5, 31

S. Bagon and M. Galun (2011). Large Scale Correlation Clustering Optimization, *CoRR*, vol. abs/1112.2903. 5, 14, 15, 32, 57

M. Bai and R. Urtasun (2017). Deep watershed transform for instance segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017*. 79, 104

X. Bai and G. Sapiro (2007). A Geodesic Framework for Fast Interactive Image and Video Segmentation and Matting, in *IEEE Int. Conf. Comp. Vision (ICCV) 2007*. 45

X. Bai, J. Wang, and D. Simons (2011). Towards Temporally-Coherent Video Matting, in *Computer Vision/Computer Graphics Collaboration Techniques 2011*, vol. 6930. 50

X. Bai, J. Wang, D. Simons, and G. Sapiro (2009). Video SnapCut: Robust Video Object Cutout Using Localized Classifiers, in *ACM Trans. Graph. (SIGGRAPH) 2009*. 44, 45

N. Bansal, A. Blum, and S. Chawla (2004). Correlation Clustering, *Machine Learning*, vol. 56(1–3), pp. 89–113. 3, 4, 14, 15, 30, 57, 113, 119, 127

D. Barath and J. Matas (2018). Multi-class model fitting by energy minimization and mode-seeking, in *ECCV 2018*. 114

A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei (2016). What's the point: Semantic segmentation with point supervision, in *European Conference on Computer Vision 2016*. 45

T. Beier, B. Andres, U. Köthe, and F. A. Hamprecht (2016). An efficient fusion move algorithm for the minimum cost lifted multicut problem, in *European Conference on Computer Vision 2016*. 129

T. Beier, F. A. Hamprecht, and J. H. Kappes (2015). Fusion Moves for Correlation Clustering, in *CVPR 2015*. 5, 14, 15, 33, 57, 129

T. Beier, T. Kröger, J. H. Kappes, U. Köthe, and F. A. Hamprecht (2014). Cut, Glue & Cut: A Fast, Approximate Solver for Multicut Partitioning, in *CVPR 2014*. 5, 14, 15, 23, 31, 32, 33, 34, 35, 36, 37, 39, 57

H. Benhabiles, G. Lavoué, J.-P. Vandeborre, and M. Daoudi (2011). Learning Boundary Edges for 3D-Mesh Segmentation, *Computer Graphics Forum*, vol. 30(8), pp. 2170–2182. 16, 23

J. Berclaz, F. Fleuret, E. Türetken, and P. Fua (2011). Multiple Object Tracking using K-Shortest Paths Optimization, *TPAMI*, vol. 33(9), pp. 1806–1819. 78

G. Bertasius, J. Shi, and L. Torresani (2015a). DeepEdge: A Multi-Scale Bifurcated Deep Network for Top-Down Contour Detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2015*. 15, 98

G. Bertasius, J. Shi, and L. Torresani (2015b). High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision, in *Proceedings of the IEEE International Conference on Computer Vision 2015*. 98, 102

G. Bertasius, J. Shi, and L. Torresani (2016). Semantic segmentation with boundary neural fields, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016*. 98, 101, 102

N. Bonneel, K. Sunkavalli, S. Paris, and H. Pfister (2013). Example-Based Video Color Grading, *ACM Trans. Graph. (SIGGRAPH)*, vol. 32(4). 45

D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein (2016). Learning shape correspondence with anisotropic convolutional neural networks, in *Advances in Neural Information Processing Systems 2016*. 15

Y. Boykov, O. Veksler, and R. Zabih (2001). Fast approximate energy minimization via graph cuts, *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23(11), pp. 1222–1239. 113

W. Brendel, M. Amer, and S. Todorovic (2011). Multiobject Tracking as Maximum Weight Independent Set, in *CVPR 2011*. 78

S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool (2017). One-shot video object segmentation, in *CVPR 2017*. 45

J. Canny (1986). A computational approach to edge detection, *IEEE Transactions on pattern analysis and machine intelligence*. 98

Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh (2017). Realtime multi-person 2d pose estimation using part affinity fields, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017*. 94

M. Charikar, V. Guruswami, and A. Wirth (2005). Clustering with Qualitative Information, *J. Comput. Syst. Sci.*, vol. 71(3), pp. 360–383. 4, 31

J. Chen, S. Paris, and F. Durand (2007). Real-time Edge-aware Image Processing with the Bilateral Grid, *ACM Trans. Graph. (SIGGRAPH)*, vol. 26(3). 50

L. Chen, H. Ai, Z. Zhuang, and C. Shang (2018). Real-Time Multiple People Tracking with Deeply Learned Candidate Selection and Person Re-Identification, in *IEEE International Conference on Multimedia and Expo (ICME) 2018*. 96

X. Chen, A. Golovinskiy, and T. Funkhouser (2009). A benchmark for 3D mesh segmentation, *TOG*, vol. 28(3), p. 73. 13, 15, 23, 28

J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang (2017). Segflow: Joint learning for video object segmentation and optical flow, in *Computer Vision (ICCV), 2017 IEEE International Conference on 2017*. 45

N. Chenouard, I. Smal, F. de Chaumont, M. Maška, I. F. Sbalzarini, Y. Gong, J. Cardinale, C. Carthel, S. Coraluppi, M. Winter, A. R. Cohen, W. J. Godinez, K. Rohr, Y. Kalaidzidis, L. Liang, J. Duncan, H. Shen, Y. Xu, K. E. G. Magnusson, J. Jaldén, H. M. Blau, P. Paul-Gilloteaux, P. Roudot, C. Kervrann, F. Waharte, J.-Y. Tinevez, S. L. Shorte, J. Willemse, K. Celler, G. P. van Wezel, H.-W. Dan, Y.-S. Tsai, C. Ortiz-de Solorzano, J.-C. Olivo-Marin, and E. Meijering (2014). Objective comparison of particle tracking methods., *Nature Methods*, vol. 11(3), pp. 281–289. 58

W. Choi (2015). Near-Online Multi-target Tracking with Aggregated Local Flow Descriptor, in *ICCV 2015*. 78, 96, 97

S. Chopra and M. Rao (1993). The partition problem, *Mathematical Programming*, vol. 59(1–3), pp. 87–115. 2, 4, 14, 16, 17, 30, 31, 57, 59, 61, 114

Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski (2002). Video Matting of Complex Scenes, *ACM Trans. Graph. (SIGGRAPH)*, vol. 21(3). 45

D. Comaniciu and P. Meer (2002). Mean shift: a robust approach toward feature space analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 114, 122

M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding, in *CVPR 2016*. 79, 102, 103, 104, 128

E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis (1994). The Complexity of Multiterminal Cuts, *SIAM Journal on Computing*, vol. 23, pp. 864–894. 30

J. Dai, K. He, and J. Sun (2016). Instance-aware semantic segmentation via multi-task network cascades, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016*. 79

E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica (2006). Correlation clustering in general weighted graphs, *Theoretical Computer Science*, vol. 361(2–3), pp. 172–187. 4, 14, 15, 30, 31, 57

M. Denitto, L. Magri, A. Farinelli, A. Fusiello, and M. Bicego (2016). Multiple Structure Recovery via Probabilistic Biclustering, in *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition (S+SSPR) 2016*. 113

M. Deza, M. Grötschel, and M. Laurent (1991). Complete descriptions of small multicut polytopes, *Applied Geometry and Discrete Mathematics*, vol. 4. 4

M. Deza, M. Grötschel, and M. Laurent (1992). Clique-web facets for multicut polytopes, *Mathematics of Operations Research*, vol. 17(4), pp. 981–1000. 4

M. M. Deza and M. Laurent (1997). *Geometry of Cuts and Metrics*, Springer. 14, 17, 57

P. Dollár and C. L. Zitnick (2015). Fast edge detection using structured forests, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37(8), pp. 1558–1570. 15, 19, 22, 24, 98

O. Dzyubachyk, W. A. van Cappellen, J. Essers, W. J. Niessen, and E. Meijering (2010). Advanced Level-Set-Based Cell Tracking in Time-Lapse Fluorescence Microscopy, *IEEE Transactions on Medical Imaging*, vol. 29(3), pp. 852–867. 69

J. Edmonds (1975). Some Well-Solved Problems in Combinatorial Optimization, in B. Roy (ed.), *Combinatorial Programming: Methods and Applications 1975*, vol. 19 of *NATO Advanced Study Institutes Series*, pp. 285–301, Springer. 57

M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman (2010). The pascal visual object classes (voc) challenge, *International journal of computer vision*, vol. 88(2), pp. 303–338. 98, 103

L. Fagot-Bouquet, R. Audigier, Y. Dhome, and F. Lerasle (2016). Improving Multi-frame Data Association with Sparse Representations for Robust Near-online Multi-object Tracking, in *ECCV 2016*. 78, 97

Q. Fan, F. Zhong, D. Lischinski, D. Cohen-Or, and B. Chen (2015). JumpCut: Non-successive Mask Transfer and Interpolation for Video Cutout, *ACM Trans. Graph.*, vol. 34(6), pp. 195:1–195:10. 45

H. Fang, S. Xie, Y.-W. Tai, and C. Lu (2017). Rmpe: Regional multi-person pose estimation, in *The IEEE International Conference on Computer Vision (ICCV) 2017*. 94

P. F. Felzenszwalb and D. P. Huttenlocher (2004). Efficient Graph-Based Image Segmentation, *Int. J. Comput. Vision*, vol. 59(2). 49

M. Fieraru, A. Khoreva, L. Pishchulin, and B. Schiele (2018). Learning to Refine Human Pose Estimation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops 2018*. 94

M. A. Fischler and R. C. Bolles (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, *Communications of the ACM*, vol. 24(6), pp. 381–395. 112

T. Fukunaga (2018). LP-Based Pivoting Algorithm for Higher-Order Correlation Clustering, in *Computing and Combinatorics 2018*. 4, 119

J. Funke, B. Anders, F. A. Hamprecht, A. Cardona, and M. Cook (2012). Efficient automatic 3D-reconstruction of branching neurons from EM data, in *CVPR '12: Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2012*. 57, 58

F. Galasso, N. S. Nagaraja, T. J. Cardenas, T. Brox, and B. Schiele (2013). A Unified Video Segmentation Benchmark: Annotation, Metrics and Analysis, in *IEEE Int. Conf. Comp. Vision (ICCV) 2013*. 45

Y. Ganin and V. Lempitsky (2014). N^4-Fields: Neural Network Nearest Neighbor Fields for Image Transforms, in *Asian Conference on Computer Vision 2014*. 98

G. Ghiasi and C. C. Fowlkes (2016). Laplacian pyramid reconstruction and refinement for semantic segmentation, in *European Conference on Computer Vision 2016*. 103, 105

O. Goldschmidt and D. S. Hochbaum (1994). A Polynomial Algorithm for the K-cut Problem for Fixed K, *Math. Oper. Res.*, vol. 19(1), pp. 24–37. 30

M. Grötschel and Y. Wakabayashi (1989). A cutting plane algorithm for a clustering problem, *Mathematical Programming*, vol. 45(1), pp. 59–96. 4, 30, 31, 114

M. Grötschel and Y. Wakabayashi (1990). Facets of the clique partitioning polytope, *Mathematical Programming*, vol. 47(1), pp. 367–387. 4

M. Grundmann, V. Kwatra, M. Han, and I. Essa (2010). Efficient Hierarchical Graph Based Video Segmentation, *IEEE CVPR*. 45

I. Gurobi Optimization (2015). *Gurobi Optimizer Reference Manual*. 64

S. Hallman and C. Fowlkes (2015). Oriented Edge Forests for Boundary Detection, in *CVPR 2015*. 15

N. Harder, R. Batra, N. Diessl, S. Gogolin, R. Eils, F. Westermann, R. König, and K. Rohr (2015). Large-scale tracking and classification for automatic analysis of cell migration and proliferation, and experimental optimization of high-throughput screens of neuroblastoma cells, *Cytometry Part A*, vol. 87(6), pp. 524–540. 69

B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik (2015). Hypercolumns for object segmentation and fine-grained localization, in *Proceedings of the IEEE conference on computer vision and pattern recognition 2015*. 100, 101

Z. Hayder, X. He, and M. Salzmann (2017). Boundary-aware instance segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017*. 79, 104

K. He, G. Gkioxari, P. Dollár, and R. Girshick (2017). Mask r-cnn, in *Proceedings of the IEEE International Conference on Computer Vision 2017*. 79, 104

R. Henschel, L. Leal-Taix'e, D. Cremers, and B. Rosenhahn (2018). Fusion of Head and Full-Body Detectors for Multi-Object Tracking, in *CVPR Workshops 2018*. 96

S. Hochreiter, U. Bodenhofer, M. Heusel, A. Mayr, A. Mitterecker, A. Kasim, T. Khamiakova, S. Van Sanden, D. Lin, W. Talloen, L. Bijnens, H. W. H. Ghlmann, Z. Shkedy, and D.-A. Clevert (2010). FABIA: factor analysis for bicluster acquisition, *Bioinformatics*. 113

J. Hopcroft and R. Tarjan (1973). Algorithm 447: Efficient Algorithms for Graph Manipulation, *Commun. ACM*, vol. 16(6), pp. 372–378. 21

A. Horňáková, J.-H. Lange, and B. Andres (2017). Analysis and Optimization of Graph Decompositions by Lifted Multicuts, in *ICML 2017*. 4, 17, 30, 62, 81, 83, 129

E. Hsu, T. Mertens, S. Paris, S. Avidan, and F. Durand (2008). Light mixture estimation for spatially varying white balance, *ACM Trans. Graph. (SIGGRAPH)*, vol. 27(3). 45

J.-J. Hwang and T.-L. Liu (2015). Pixel-wise Deep Learning for Contour Detection, *CoRR*, vol. abs/1504.01989. 102

E. Insafutdinov, M. Andriluka, L. Pishchulin, S. Tang, E. Levinkov, B. Andres, and B. Schiele (2017). ArtTrack: Articulated Multi-person Tracking in the Wild, in *CVPR 2017*. 78, 94, 128

E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele (2016). DeeperCut: A Deeper, Stronger, and Faster Multi-Person Pose Estimation Model, in *ECCV 2016*. 5, 78, 83, 92, 93, 94, 128

H. Isack and Y. Boykov (2012). Energy-based geometric multi-model fitting, *International journal of computer vision*, vol. 97(2), pp. 123–147. 113, 114

P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson (2014). Crisp Boundary Detection Using Pointwise Mutual Information, in *ECCV 2014*. 15, 98

S. Jain and V. Madhav Govindu (2013). Efficient Higher-Order Clustering on the Grassmann Manifold, in *Proceedings of the IEEE International Conference on Computer Vision 2013*. 114

S. D. Jain, B. Xiong, and K. Grauman (2017). Fusionseg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos, in *Proc. CVPR 2017*. 45

V. Jampani, R. Gadde, and P. V. Gehler (2017). Video propagation networks, in *Proc. CVPR 2017*. 45

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell (2014). Caffe: Convolutional Architecture for Fast Feature Embedding, *arXiv preprint arXiv:1408.5093*. 38

F. Jug, T. Pietzsch, D. Kainmüller, J. Funke, M. Kaiser, E. van Nimwegen, C. Rother, and G. Myers (2014). Optimal Joint Segmentation and Tracking of Escherichia Coli in the Mother Machine, in *Bayesian and grAphical Models for Biomedical Imaging 2014*, pp. 25–36, Springer International Publishing, Cham. 57, 58

E. Kalogerakis and A. Hertzmann (2010). Learning 3D mesh segmentation and labeling, *TOG*, vol. 29(4), p. 102. 15, 16, 19, 23, 26, 28

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 (2015a). A Comparative Study of Modern Inference Techniques for Structured Discrete Energy Minimization Problems, *International Journal of Computer Vision*, vol. 115(2), pp. 155–184. 5, 15, 31, 32, 33, 34, 35, 36, 37, 39, 79

J. H. Kappes, M. Speth, B. Andres, G. Reinelt, and C. Schnörr (2011). Globally Optimal Image Partitioning by Multicuts, in *EMMCVPR 2011*. 3, 4, 14, 30, 32, 57, 115

J. H. Kappes, M. Speth, G. Reinelt, and C. Schnörr (2016). Higher-order Segmentation via Multicuts, *Computer Vision and Image Understanding*. 4, 14, 15, 57, 112, 128

J. H. Kappes, P. Swoboda, B. Savchynskyy, T. Hazan, and C. Schnörr (2015b). Probabilistic Correlation Clustering and Image Partitioning Using Perturbed Multicuts, in *SSVM 2015*. 14, 33, 57

B. X. Kausler, M. Schiegg, B. Andres, M. Lindner, U. Koethe, H. Leitte, J. Wittbrodt, L. Hufnagel, and F. A. Hamprecht (2012). A discrete chain graph model for 3d+t

cell tracking with high misdetection robustness, in *ECCV'12: Proceedings of the 12th European conference on Computer Vision 2012*. 57, 58

P. J. Keller (2013). Imaging morphogenesis: technological advances and biological insights., *Science (New York, N.Y.)*, vol. 340(6137), p. 1234168. 56

P. J. Keller, A. D. Schmidt, A. Santella, K. Khairy, Z. Bao, J. Wittbrodt, and E. H. K. Stelzer (2010). Fast, high-contrast imaging of animal development with scanned light sheet-based structured-illumination microscopy., *Nature Methods*, vol. 7(8), pp. 637–642. 56

P. J. Keller, A. D. Schmidt, J. Wittbrodt, and E. H. K. Stelzer (2008). Reconstruction of zebrafish early embryonic development by scanned light sheet microscopy., *Science*, vol. 322(5904), pp. 1065–1069. 56

A. Kendall, Y. Gal, and R. Cipolla (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018*. 79, 104

B. W. Kernighan and S. Lin (1970). An efficient heuristic procedure for partitioning graphs, *Bell Systems Technical Journal*, vol. 49, pp. 291–307. 5, 14, 15, 19, 31, 85, 86, 87, 89, 90, 120

M. Keuper (2017). Higher-Order Minimum Cost Lifted Multicuts for Motion Segmentation, in *Proceedings of the IEEE International Conference on Computer Vision 2017*. 4, 5, 113, 117, 120

M. Keuper, B. Andres, and T. Brox (2015). Motion Trajectory Segmentation via Minimum Cost Multicuts, in *ICCV 2015*. 5, 22, 33

K. Khairy and P. J. Keller (2011). Reconstructing embryonic development., *Genesis*, vol. 49(7), pp. 488–513. 56

A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele (2017). Lucid Data Dreaming for Multiple Object Tracking, *arXiv preprint arXiv:1703.09554*. 45

C. Kim, F. Li, A. Ciptadi, and J. M. Rehg (2015). Multiple Hypothesis Tracking Revisited, in *ICCV 2015*. 78, 97

S. Kim, S. Nowozin, P. Kohli, and C. Yoo (2011). Higher-Order Correlation Clustering for Image Segmentation, in *NIPS 2011*. 4, 14, 15, 30, 112, 115

S. Kim, C. Yoo, S. Nowozin, and P. Kohli (2014). Image Segmentation Using Higher-Order Correlation Clustering, *TPAMI*, vol. 36, pp. 1761–1774. 4, 14, 15, 30, 57

O. Kin-Chung Au, Y. Zheng, M. Chen, P. Xu, and C.-L. Tai (2011). Mesh Segmentation with Concavity-Aware Fields., *IEEE Transactions on Visualization and Computer Graphics*, vol. 18(7), pp. 1125 – 1134. 15, 19, 26, 28

J. Kivinen, C. Williams, and N. Heess (2014). Visual boundary prediction: A deep neural prediction network and quality dissection, in *Artificial Intelligence and Statistics 2014*. 98

P. N. Klein, C. Mathieu, and H. Zhou (2015). Correlation Clustering and Two-edge-connected Augmentation for Planar Graphs, in *32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015) 2015*. 5, 31

G. Knott, H. Marchman, D. Wall, and B. Lich (2008). Serial Section Scanning Electron Microscopy of Adult Brain Tissue Using Focused Ion Beam Milling, *Journal of Neuroscience*, vol. 28(12), pp. 2959–2964. 35

I. Kokkinos (2016). Pushing the boundaries of boundary detection using deep learning, in *Proceedings of the International Conference on Learning Representations 2016*. 15

I. Kokkinos (2017). Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017*. 15

T. Kroeger, J. H. Kappes, T. Beier, U. Koethe, and F. A. Hamprecht (2014). Asymmetric Cuts: Joint Image Labeling and Partitioning, in *GCPR 2014*. 79, 84

H. W. Kuhn (1955). The Hungarian Method for the assignment problem, *Naval Research Logistics Quarterly*, pp. 83–97. 122

J.-H. Lange and B. Andres (2017). Decomposition of Trees and Paths via Correlation, *arXiv preprint arXiv:1706.06822*. 4

J.-H. Lange, A. Karrenbauer, and B. Andres (2018). Partial Optimality and Fast Lower Bounds for Weighted Correlation Clustering, in *Proceedings of the International Conference on Machine Learning 2018*. 5, 31, 129

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner (1998). Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, vol. 86(11), pp. 2278–2324. 30, 33, 38, 39

J. Leskovec, D. Huttenlocher, and J. Kleinberg (2010). Signed Networks in Social Media, in *CHI 2010*. 30, 33, 40

W. Li, F. Viola, J. Starck, G. J. Brostow, and N. D. F. Campbell (2016). Roto++: Accelerating Professional Rotoscoping Using Shape Manifolds, *ACM Trans. Graph.*, vol. 35(4), pp. 62:1–62:15. 45

J. J. Lim, C. L. Zitnick, and P. Dollár (2013). Sketch tokens: A learned mid-level representation for contour and object detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2013*. 98

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick (2014). Microsoft coco: Common objects in context, in *European conference on computer vision 2014*. 103

C. Liu (2009). *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*, Ph.D. thesis, Massachusetts Institute of Technology. 48

S. Liu, J. Jia, S. Fidler, and R. Urtasun (2017). Sgn: Sequential grouping networks for instance segmentation, in *Proceedings of the IEEE International Conference on Computer Vision 2017*. 79, 104

S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia (2018). Path aggregation network for instance segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018*. 79, 104

Y. Liu and M. S. Lew (2016). Learning relaxed deep supervision for better edge detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016*. 15

Y. Lu, X. Bai, L. Shapiro, and J. Wang (2016). Coherent Parametric Contours for Interactive Video Object Segmentation, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016*. 45

C. Ma, C. Yang, F. Yang, Y. Zhuang, Z. Zhang, H. Jia, and X. Xie (2018). Trajectory Factory: Tracklet Cleaving and Re-Connection by Deep Siamese Bi-GRU for Multiple Object Tracking, in *IEEE International Conference on Multimedia and Expo (ICME) 2018*. 96

V. Madhav Govindu (2005). A Tensor Decomposition for Geometric Grouping and Segmentation., in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2005*. 114

K. E. G. Magnusson, J. Jaldén, P. M. Gilbert, and H. M. Blau (2015). Global linking of cell tracks using the Viterbi algorithm, *IEEE Trans. Med. Imag.*, vol. 34(4), pp. 911–929. 69

L. Magri and A. Fusiello (2014). T-Linkage: A Continuous Relaxation of J-Linkage for Multi-Model Fitting, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2014*. 112, 113, 122

L. Magri and A. Fusiello (2015). Robust Multiple Model Fitting with Preference Analysis and Low-rank Approximation, in *Proceedings of the British Machine Vision Conference 2015*. 113

L. Magri and A. Fusiello (2016). Multiple Models Fitting as a Set Coverage Problem, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016*. 112, 113, 122

D. Martin, C. Fowlkes, D. Tal, and J. Malik (2001). A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics, in *Proc. 8th Int'l Conf. Computer Vision 2001*. 34

M. Maška, V. Ulman, D. Svoboda, P. Matula, P. Matula, C. Ederra, A. Urbiola, T. España, S. Venkatesan, D. M. W. Balak, P. Karas, T. Bolcková, M. Štreitová, C. Carthel, S. Coraluppi, N. Harder, K. Rohr, K. E. G. Magnusson, J. Jaldén, H. M. Blau, O. Dzyubachyk, P. Křížek, G. M. Hagen, D. Pastor-Escuredo, D. Jimenez-Carretero, M. J. Ledesma-Carbayo, A. Muñoz-Barrutia, E. Meijering, M. Kozubek, and C. Ortiz-de Solorzano (2014). A benchmark for comparison of cell tracking algorithms, *Bioinformatics*, p. btu080. 67, 69

S. G. Megason and S. E. Fraser (2007). Imaging in systems biology., *Cell*, vol. 130(5), pp. 784–795. 56

M. Meilă (2007). Comparing Clusterings—an Information Based Distance, *J. Multivar. Anal.*, vol. 98(5), pp. 873–895. 22, 33

A. Merouane, N. Rey-Villamizar, Y. Lu, I. Liadi, G. Romain, J. Lu, H. Singh, L. J. N. Cooper, N. Varadarajan, and B. Roysam (2015). Automated profiling of individual cell–cell interactions from high-throughput time-lapse imaging microscopy in nanowell grids (TIMING), *Bioinformatics*, p. btv355. 69

A. Milan, L. Leal-Taixé, I. D. Reid, S. Roth, and K. Schindler (2016). MOT16: A Benchmark for Multi-Object Tracking, *arXiv:1603.00831*. 79, 96, 97, 98

A. Milan, S. Roth, and K. Schindler (2014). Continuous Energy Minimization for Multitarget Tracking, *TPAMI*, vol. 36(1), pp. 58–72. 78

Mocha Imagineer Systems, Ltd. (2014). *mocha Pro v3.1 software*, . 44

A. Newell, Z. Huang, and J. Deng (2017). Associative embedding: End-to-end learning for joint detection and grouping, in *Advances in Neural Information Processing Systems 2017*. 94, 129

S. Nowozin and S. Jegelka (2009). Solution Stability in Linear Programming Relaxations: Graph Partitioning and Unsupervised Learning, in *ICML 2009*. 14, 34

S. Nowozin and C. H. Lampert (2010). Global Interactions in Random Field Models: A Potential Function Ensuring Connectedness, *SIAM J. Img. Sci.*, vol. 3(4). 47, 84

P. Ochs and T. Brox (2011). Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions, in *IEEE Int. Conf. Comp. Vision (ICCV) 2011*. 45

D. Padfield, J. Rittscher, and B. Roysam (2011). Coupled minimum-cost flow cell tracking for high-throughput quantitative analysis., *Med Image Anal*, vol. 15(4), pp. 650–668. 57

X. Pan, D. S. Papailiopoulos, S. Oymak, B. Recht, K. Ramchandran, and M. I. Jordan (2015). Parallel Correlation Clustering on Big Graphs, in *NIPS 2015*. 31

C. Pape, T. Beier, P. Li, V. Jain, D. D. Bock, and A. Kreshuk (2017). Solving Large Multicut Problems for Connectomics via Domain Decomposition., in *ICCV Workshops 2017*. 129

F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A. Sorkine-Hornung (2017). Learning video object segmentation from static images, in *Computer Vision and Pattern Recognition 2017*. 45

H. Pirsiavash, D. Ramanan, and C. C. Fowlkes (2011). Globally-Optimal Greedy Algorithms for Tracking a Variable Number of Objects, in *CVPR 2011*. 78

L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. Gehler, and B. Schiele (2016). DeepCut: Joint Subset Partition and Labeling for Multi Person Pose Estimation, in *CVPR 2016*. 78, 83, 84, 92

Power Matte Digital Film Tools, LLC. (). *Power Matte v2.0.1.3 software*, . 44

P. Purkait, T. J. Chin, A. Sadri, and D. Suter (2017). Clustering with Hypergraphs: The Case for Large Hyperedges, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39(9), pp. 1697–1711. 112, 114

M. Rempfler, B. Andres, and B. Menze (2016). The Minimum Cost Connected Subgraph Problem in Medical Image Analysis, in *MICCAI 2016*. 84

M. Rempfler, J.-H. Lange, F. Jug, C. Blasse, E. W. Myers, B. H. Menze, and B. Andres (2017). Efficient Algorithms for Moral Lineage Tracing, in *ICCV 2017*. 8, 72, 128

M. Rempfler, V. Stierle, K. Ditzel, S. Kumar, P. Paulitschke, B. Andres, and B. H. Menze (2018). Tracing Cell Lineages in Videos of Lens-free Microscopy, *Medical Image Analysis*. 57

J. B. T. M. Roerdink and A. Meijster (2000). The watershed transform: definitions, algorithms and parallelization strategies, *Fundam. Inf.*, vol. 41(1-2). 49

C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer (2007). Optimizing binary MRFs via extended roof duality, in *CVPR 2007*. 32

A. Sadeghian, A. Alahi, and S. Savarese (2017). Tracking the Untrackable: Learning to Track Multiple Cues with Long-Term Dependencies, in *IEEE International Conference on Computer Vision 2017*. 96

M. Schiegg, P. Hanslovsky, C. Haubold, U. Koethe, L. Hufnagel, and F. A. Hamprecht (2014). Graphical Model for Joint Segmentation and Tracking of Multiple Dividing Cells., *Bioinformatics*, p. btu764. 58

M. Schiegg, P. Hanslovsky, B. X. Kausler, and L. Hufnagel (2013). Conservation Tracking, *ICCV 2013*. 57, 58

N. N. Schraudolph and D. Kamenetsky (2009). Efficient Exact Inference in Planar Ising Models, in *NIPS 2009*. 32

A. Schrijver (2003). *Combinatorial optimization. Polyhedra and efficiency*, Springer. 57

E. Shahrian, B. Price, S. Cohen, and D. Rajan (2014). Temporally Coherent and Spatially Accurate Video Matting, *Computer Graphics Forum*, vol. 33. 50

N. Shankar Nagaraja, F. R. Schmidt, and T. Brox (2015). Video segmentation with just a few strokes, in *Proceedings of the IEEE International Conference on Computer Vision 2015*. 45

L. Shapira, A. Shamir, and D. Cohen-Or (2008). Consistent mesh partitioning and skeletonisation using the shape diameter function, *The Visual Computer*, vol. 24(4), pp. 249–259. 23

E. Sharon, M. Galun, D. Sharon, R. Basri, and A. Brandt (2006). Hierarchy and adaptivity in segmenting visual scenes., *Nature*, vol. 442(7104). 49

W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang (2015). Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2015*. 98

H. Sheng, J. Chen, Y. Zhang, W. Ke, Z. Xiong, and J. Yu (2018). Iterative Multiple Hypothesis Tracking with Tracklet-level Association, *IEEE Transactions on Circuits and Systems for Video Technology*. 96

J. Shi and J. Malik (2000). Normalized Cuts and Image Segmentation, *TPAMI*, vol. 22(8), pp. 888–905. 14, 56, 98

SilhouetteFX, LLC. (2004–2018). *Silhouette v5 software*, . 44

J. Stühmer and D. Cremers (2015). A Fast Projection Method for Connectivity Constraints in Image Segmentation, in *EMMCVPR 2015*. 84

P. Swoboda and B. Andres (2017). A Message Passing Algorithm for the Minimum Cost Multicut Problem, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017*. 5, 31, 129

S. Tang, B. Andres, M. Andriluka, and B. Schiele (2015). Subgraph Decomposition for Multi-Target Tracking, in *CVPR 2015*. 6, 58, 78, 83, 84, 94, 95

S. Tang, B. Andres, M. Andriluka, and B. Schiele (2016). Multi-Person Tracking by Multicut and Deep Matching, in *ECCV Workshops 2016*. 78, 83, 96, 97, 128

S. Tang, M. Andriluka, B. Andres, and B. Schiele (2017). Multiple people tracking by lifted multicut and person re-identification, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017*. 96, 97

T. Tanimoto (1957). Technical report, *IBM Internal Report*. 113

R. B. Tennakoon, A. Bab-Hadiashar, Z. Cao, R. Hoseinnezhad, and D. Suter (2016). Robust Model Fitting Using Higher Than Minimal Subset Sampling, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38(2), pp. 350–362. 112, 114

M. Tepper and G. Sapiro (2017). Nonnegative Matrix Underapproximation for Robust Multiple Model Fitting, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017*. 112, 113, 122, 123

P. Theologou, I. Pratikakis, and T. Theoharis (2015). A comprehensive overview of methodologies and performance evaluation frameworks in 3D mesh segmentation, *CVIU*, vol. 135, pp. 49–82. 15

R. Toldo and A. Fusiello (2008). Robust Multiple Structures Estimation with J-Linkage, in *Proceedings of the European Conference on Computer Vision 2008*. ix, 111, 113, 122, 123, 125, 133

C. Tomasi and R. Manduchi (1998). Bilateral Filtering for Gray and Color Images, in *IEEE Int. Conf. Comp. Vision (ICCV) 1998*. 50

R. Tomer, K. Khairy, F. Amat, and P. J. Keller (2012). Quantitative high-speed imaging of entire developing embryos with simultaneous multiview light-sheet microscopy, *Nature Methods*, vol. 9(7), pp. 755–763. 56

E. Türetken, C. Becker, P. Glowacki, F. Benmansour, and P. Fua (2013a). Detecting Irregular Curvilinear Structures in Gray Scale and Color Imagery using Multi-Directional Oriented Flux, in *ICCV 2013*. 57

E. Türetken, F. Benmansour, B. Andres, H. Pfister, and P. Fua (2013b). Reconstructing Loopy Curvilinear Structures Using Integer Programming, in *CVPR 2013*. 57, 58

E. Türetken, F. Benmansour, and P. Fua (2012). Automated Reconstruction of Tree Structures using Path Classifiers and Mixed Integer Programming, in *CVPR 2012*. 57

E. Türetken, G. Gonzalez, C. Blum, and P. Fua (2011). Automated Reconstruction of Dendritic and Axonal Trees by Global Optimization with Geometric Priors, *Neuroinformatics*, vol. 9, pp. 279–302. 57

J. Uhrig, M. Cordts, U. Franke, and T. Brox (2016). Pixel-level Encoding and Depth Layering for Instance-level Semantic Labeling, in *GCPR 2016*. 104

S. Varadarajan, P. Datta, and O. Tickoo (2018). A greedy part assignment algorithm for real-time multi-person 2d pose estimation, in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV) 2018*. 94

S. Vicente, V. Kolmogorov, and C. Rother (2008). Graph cut based image segmentation with connectivity priors, in *CVPR 2008*. 84

T. Voice, M. Polukarov, and N. R. Jennings (2012). Coalition Structure Generation over Graphs, *Journal of Artificial Intelligence Research*, vol. 45, pp. 165–196. 5, 31, 119

H. Wang, X. Guobao, Y. Yan, and D. Suter (2018). Searching for Representative Modes on Hypergraphs for Robust Geometric Model Fitting, *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 114

J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen (2005). Interactive Video Cutout, in *ACM Trans. Graph. (SIGGRAPH) 2005*. 45

T. Wang, B. Han, and J. Collomosse (2014a). Touchcut: Fast image and video segmentation using single-touch interaction, *Computer Vision and Image Understanding*, vol. 120, pp. 14–30. 45

X. Wang, E. Turetken, F. Fleuret, and P. Fua (2014b). Tracking Interacting Objects Optimally Using Integer Programming, in *ECCV 2014*. 58

Y. Wang, X. Zhao, and K. Huang (2017). Deep Crisp Boundaries, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017*. 15

Y. Xiang, A. Alahi, and S. Savarese (2015). Learning to Track: Online Multi-Object Tracking by Decision Making, in *ICCV 2015*. 78

G. Xiao, H. Wang, T. Lai, and D. Suter (2016). Hypergraph modelling for geometric model fitting, *Pattern Recognition*, vol. 60, pp. 748 – 760. 112, 114, 122

S. Xie and Z. Tu (2015). Holistically-nested edge detection, in *Proceedings of the IEEE international conference on computer vision 2015*. 15, 98, 102

Z. Xie, K. Xu, L. Liu, and Y. Xiong (2014). 3D Shape Segmentation and Labeling via Extreme Learning Machine, *Computer Graphics Forum*, vol. 33(5). 16, 23

J. Yarkony (2014). Analyzing PlanarCC: Demonstrating the Equivalence of PlanarCC and The Multi-Cut LP Relaxation, in *NIPS Workshop on Discrete Optimization 2014*. 4, 30

J. Yarkony, A. Ihler, and C. Fowlkes (2012). Fast Planar Correlation Clustering for Image Segmentation, in *ECCV 2012*. 4, 5, 14, 30, 31, 32, 57

J. Yarkony, C. Zhang, and C. Fowlkes (2015). Hierarchical Planar Correlation Clustering for Cell Segmentation, in *EMMCVPR 2015*. 14, 57

L. Yi, V. G. Kim, D. Ceylan, I. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, L. Guibas, *et al.* (2016). A scalable active framework for region annotation in 3d shape collections, *ACM Transactions on Graphics (TOG)*, vol. 35(6), p. 210. 15

L. Yi, H. Su, X. Guo, and L. Guibas (2017). SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017*. 15

F. Yu and V. Koltun (2016). Multi-scale context aggregation by dilated convolutions, in *ICLR 2016*. 97, 100, 101, 103, 104

A. R. Zamir, A. Dehghan, and M. Shah (2012). GMCP-Tracker: Global Multi-object Tracking Using Generalized Minimum Clique Graphs, in *ECCV 2012*. 78

J. Zhang, J. Zheng, C. Wu, and Jianfei Cai (2012). Variational Mesh Decomposition, *TOG*, vol. 31(3). 15, 19, 26, 28

W. Zhang and J. Kosecka (2006). Nonparametric estimation of multiple structures with outliers, in *Workshop on Dynamic Vision within European Conference on Computer Vision 2006*. 122

F. Zhong, X. Qin, Q. Peng, and X. Meng (2012). Discontinuity-Aware Video Object Cutout, *ACM Trans. Graph. (SIGGRAPH Asia)*, vol. 31(6). 45

M. Zuliani, C. S. Kenney, and B. S. Manjunath (2005). The multiransac algorithm and its application to detect planar homographies, in *Proceedings of the IEEE International Conference on Image Processing 2005*. 112, 122