DISSERTATION
submitted
to the
Combined Faculty for the Natural Sciences and Mathematics
of
Heidelberg University, Germany
for the degree of
Doctor of Natural Sciences

Put forward by
MSc Thorsten Beier
Born in Heidelberg, Germany
Oral examination:

Multicut Algorithms for Neurite Segmentation

Thorsten Beier

Advisor: Prof. Dr. Fred A. Hamprecht

Abstract

Correlation clustering, or multicut partitioning is widely used for image segmentation and graph partitioning. Given an undirected edge weighted graph with positive and negative weights, correlation clustering partitions the graph such that the sum of cut edge weights is minimized. Since the optimal number of clusters is automatically chosen, multicut partitioning is well suited for clustering neural structures in EM connectomics datasets where the optimal number of clusters is unknown a-priori. Due to the NP-hardness of optimizing the multicut objective, exact solvers do not scale and approximative solvers often give unsatisfactory results.
In chapter 2 we investigate scalable methods for correlation clustering. To this end we define *fusion moves* for the multicut objective function which iteratively fuses the current and a proposed partitioning and monotonously improves the partitioning. Fusion moves scale to larger datasets, give near optimal solutions and at the same time show state of the art anytime performance.
In chapter 3 we generalize the *fusion moves* frameworks for the lifted multicut objective, a generalization of the multicut objective which can penalize or reward all decompositions of a graph for which any given pair of nodes are in distinct components. The proposed framework scales well to large datasets and has a cutting edge anytime performance.
In chapter 4 we propose a framework for automatic segmentation of neural structures in 3D EM connectomics data where a membrane probability is predicted for each pixel with a neural network and superpixels are computed based on this probability map. Finally the superpixels are merged to neurites using the techniques described in chapter 3. The proposed pipeline is validated with an extensive set of experiments and a detailed lesion study. This work substantially narrows the accuracy gap between humans and computers for neurite segmentation.
In chapter 5 we summarize the software written for this thesis. The provided implementations for algorithms and techniques described in chapters 2 to 4 and many other algorithms resulted in a software library for graph partitioning, image segmentation and discrete optimization.

Zusammenfassung

Correlation Clustering oder Multicut Partitionierung ist eine weit verbreitete Technik zur Bildsegmentierung oder Graphpartitionierung. Correlation Clustering partitioniert einen kantengewichteten Graph mit positiv und negativ gewichteten Kanten sodass die Summe der Kantengewichte der geschnittenen Kanten minimiert wird. Da die optimale Anzahl der Kluster automatisch ausgewählt wird, ist die Multicut Paritionierung gut geeignet um neuronale Strukturen in sogenannten EM-Konnektom Datensätzen zu segmentieren, da dort die optimale Anzahl von Klustern nicht a-priori bekannt ist. Da es NP-hart ist die Multicut Zielfunktion zu optimieren skalieren exakte Algorithmen nicht und approximative Verfahren geben schlechte Resultate.
In Kapitel 2 untersuchen wir skalierende Methoden für Correlation Clustering. Wir definieren *Fusion Moves* für die Multicut Zielfunktion. Fusion Moves ist ein iteratives Verfahren das die momentane Partitionierung mit einer Kandidatenpartitionierung fusioniert und so monoton die Partitonierung verbessert. Fusion Moves skaliern zu großen Datensätzen, geben nahezu optimale Lösungen und haben eine gute Performance selbst wenn sie vor der Terminierung unterbrochen werden.
In Kapitel 3 generalisieren wir *Fusion Moves* für die Lifted Multicut Zielfunktion, eine Generalisierung der Multicut Zielfunktion welche alle Partitionierungen eines Graphes belohnen oder bestrafen kann in der ein beliebiges paar von Knoten in verschiedenen Klustern ist. Die vorgeschlagenen Methoden skalieren gut und haben ein guten Performance selbst wenn sie vor der Terminierung unterbrochen werden.
In Kapitel 4 wird ein Framework zur automatischen Segmentierung von neuronalen Strukturen in 3D EM Daten vorgestellt. Startend von einer mit einem neuronalen Netz gelernten pixelweisen Membranwahrscheinlichkeit wird eine Superpixel Übersegmentierung erzeugt. Die Superpixel werden mit den in Kapiteln 2 und 3 vorgeschlagenen Methoden zu Neuronen zusammengefügt. Das vorgeschlagene Framework wird durch umfangreiche Experimente und eine detailreiche Läsion Studien validiert. Der Qualitätsunterschied zwischen menschlich erzeugten Segmentierungen und automatisch erzeugten Segmentierungen wurde durch das vorgeschlagene Framework deutlich verringert.
In Kapitel 5 wird die für diese Thesis geschriebene Software zusammengefasst. Die bereitgestellten Implementierungen für die Algorithmen aus Kapitel 2-4 und viele andere Algorithmen resultierten in einer Software Bibliothek zur Graph Partitionierung und Bildsegmentierung.

Acknowledgments

First, I like to thank Professor Fred Hamprecht for being the Supervisor for this thesis. I have been working in his research group *Image Analysis and Learning* for almost a decade. Since I joined the group as bachelor student, it has always been a pleasure to collaborate with so many friendly, intelligent and helpful people. I like to thank Bjoern Andres and Jörg Kappes for introducing me to the field of discrete optimization and many hours of fruitful discussions about algorithms and software. I like to thank Ullrich Köthe for many fruitful discussion about image segmentation, algorithms and software in general. I like to thank Thorben Kroeger, Niko Krasowki, Anna Kreshuk and Constantin Pape for our strong collaboration and joint work on automatic segmentation for 3D-EM Data. In particular I like to thank Constantin Pape for the many hours of work he invested in joint research papers and software projects. I had many enjoyable moments with Christoph Straehle and Luca Fiasch and Sven Wanner outside of the building. I enjoyed the special humor of Philipp Hanslovsky and Robert Walecki very much. I enjoyed many cups of coffees with Steffen Wolf, Lorenzo Cerrone, Constantin Pape, Phillip Schmidt, Peter Neigel, Christoph Straehle, Nasim Rahaman and Anna Kreshuk where we hand many enjoyable discussions. A special thanks to Barbara Werner for all her incredible work.

Contents

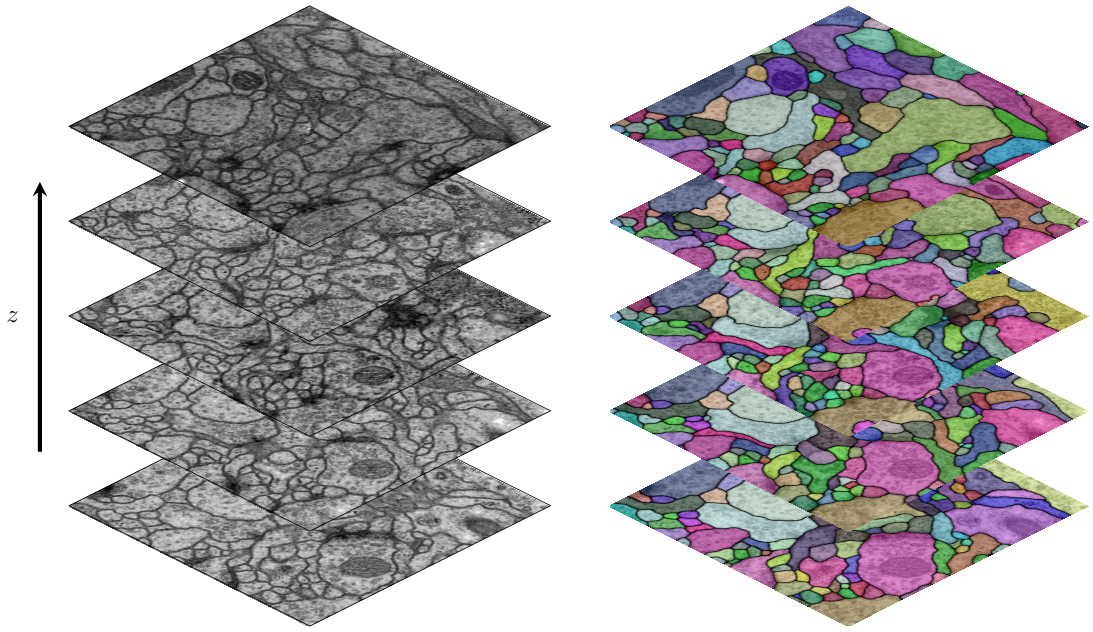
1 Introduction

1.1 Image Segmentation

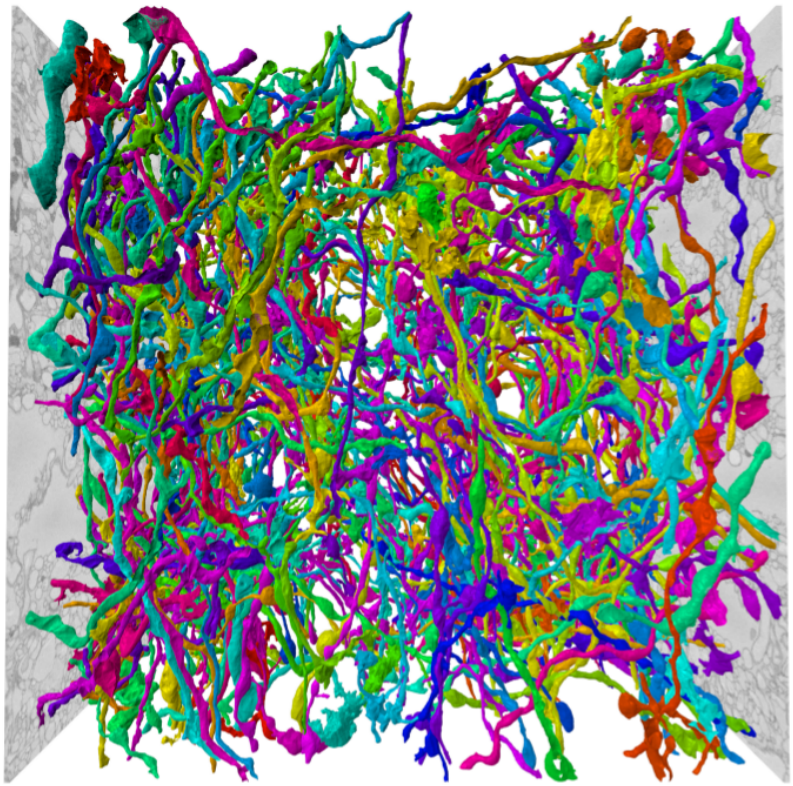
Image segmentation is arguably the most important task in computer vision. It is the fundamental building block for many application and therefore fast and accurate segmentations algorithms are needed. Many flavors of segmentation exists: i) *Class-level segmentation* where each pixel is assigned to a single class from a discrete set as $\{\textbf{sky}, \textbf{car}, \textbf{road}, \textbf{person}\}$. ii) *Instance-level segmentation* where each pixel is not only assigned to a single class from a discrete set as $\{\textbf{sky}, \textbf{car}, \textbf{road}, \textbf{person}\}$ but also a unique instance id, e.g. *person-1, person-2, car-1.* The number of instances per class is not known beforehand in this setting and iii) *One-Class-Instance-Level* as a special case of *Instance-level segmentation* with only a single class. Again, the number of instances is not known in advance. An example for this kind of segmentation is given in fig. 1.1 where each pixel is assigned to the id of the corresponding neural process, e.g. *neuron-1, neuron-2, neuron-3.*
In this thesis we will focus on the latter one, *One-Class-Instance-Level* segmentation for connectomics as described in the following section.

1.2 Segmentation for Connectomics

To understand how the brain is working neuroscientists are acquiring large volumes of electron microscopy (EM) images of the brain of animals with the aim of analyzing the neural circuit connectivity of the brain. This circuit formed by neurons which are connected via synapses is the so called *connectome*. *Connectomics* is the field of science acquiring and studying *connectomes*. The connectome can be acquired by sequencing techniques [149] or by segmentation based approaches. Here, we only discuss the latter one. Given a segmentation of the neurons in EM volumes, synapses and their synaptic partners need to be detected [88, 90, 128] to form the graph known as connectome. In this thesis we will focus on automated segmentation of neurons. Detection of synapses and their synaptic partners is beyond the scope of this thesis.



(a) Stack of raw data and corresponding segmentation



(b) 3D visualization of result from a multicut approach [9]

Figure 1.1: Figure 1.1a: A stack of 2×2 microns slices of from a transmission Electron Microscopy (ssTEM) data set of the Drosophila first instar larva ventral nerve cord (VNC) with a resolution of $4 \times 4 \times 50$nm/pixel [19] and a manual created segmentation where each individual neural process is assigned a random color. Figure 1.1b: Andres et al. [9] used a muticut approach to automatically segment the neuron in a volumetric 3D EM dataset. The elongated structures are the individual neurons. Each instances of a neuron is depicted in a random color. Image source: With permission from [94].

Despite impressive progress in collaborative annotation [79], the sheer size of these volumes make manual analysis infeasible. To handle large whole-brain datasets automated segmentation is needed. In chapters 4 and 5 we present algorithms and a software package to automatically segment such data sets with low error rates. The algorithms are based based on the multicut [8, 68] and lifted multicut [7, 78] formulation which we will briefly described in the following sections.

1.2.1 Multicut

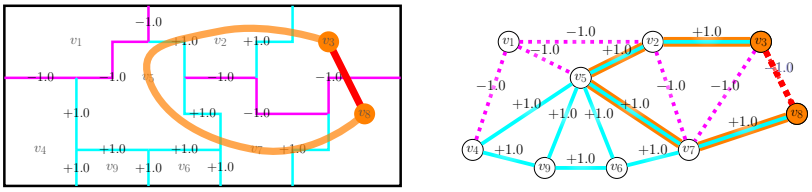


Figure 1.2: The multicut formulation guarantees a valid segmentation by disallowing violated cycle constraints (eq. (3.2)): The edge between node $v3$ and $v8$ is cut. But there is a path of uncut edge connecting $v3$ and $v8$ depicted in orange. The multicut objective can be optimized by adding violated constraints to an ILP in a cutting plane fashion [8, 68].

The multicut [25] and lifted multicut problem [7, 78] have become increasingly popular in the recent years [8, 23, 26, 27, 29, 68, 81, 83, 92, 99, 112, 113, 135, 146, 147].
Given a graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$ the minimum multicut minimizes the sum of weights between clusters. Formally the minimum multicut is defined as:

$$y^* = \underset{y \in \{0,1\}^E}{\arg\min} \quad \sum_{e \in E} w_e y_e \tag{1.1}$$

$$\text{subject to} \quad \forall C \in \text{cycles}(G)\ \forall e \in C : \ y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \tag{1.2}$$

Ensures valid segmentations without dangling edges as illustrated in fig. 1.2

Multicuts have several advantages compared to traditional algorithms operating on a weighted graph:
i) Graph-cuts [85] and normalized cuts [123] can only model positive weight (attraction) and ii) suffer from a shrinking bias [45, 141] while the multicut formulation allows

for positive (attraction) and negative (repulsion) edge weights and does not suffer from a shrinking bias. iii) While QPBO [118] and multi-label variants [84] can handle positive and negative weights, the maximum number of classes/ clusters needs to be fixed beforehand. The multicut approach does not need a specified number clusters, the optimal number of classes is implicitly choose by the optimal solution.
On the down-side, solving the multicut problem is in general NP-hard. Since the set of constraints in eq. (1.2) is of exponential size any exact solver will have scalability issues.
A detailed review of the multicut objective and optimizer is given in sections 2.1.2 and 3.2.1.

1.2.2 Lifted Multicut

The minimum lifted multicut problem [7, 78] is an optimization problem whose feasible solutions are decompositions of a graph. The objective function can penalize or reward all decompositions for which any given pair of nodes are in distinct components. Given a graph $G = (V, E)$ and a larger graph $G' = (V, E')$ with $E \subseteq E'$ and edge weights $w : E' \to \mathbb{R}$, where the weights penalize or reward precisely the decompositions of G for which the nodes v and w are in distinct components. The lifted multicut problem is defined as:

$$y^* = \underset{y \in \{0,1\}^{E'}}{\arg\min} \quad \sum_{e \in E'} c_e y_e \tag{1.3}$$

$$\text{subject to} \quad \forall Y \in \text{cycles}(G)\ \forall e \in Y :\ y_e \leq \sum_{e' \in Y \setminus \{e\}} y_{e'} \tag{1.4}$$

Ensures valid segmentations without dangling edges as depicted in fig. 1.2

$$\forall vw \in E' \setminus E\ \forall P \in vw\text{-paths}(G) :\ y_{vw} \leq \sum_{e \in P} y_e \tag{1.5}$$

If additional edges wv **is cut**, ensure that no path of uncut edges between u and v in G exists

$$\forall vw \in E' \setminus E\ \forall C \in vw\text{-cuts}(G) :\ 1 - x_{vw} \leq \sum_{e \in C} (1 - y_e) \tag{1.6}$$

If additional edges uv **is not cut**, ensure that no cut in G exists which separates uv

Like the multicut, solving the lifted multicut problem is in general NP-hard. A detailed review of the lifted multicut objective and optimizer is given in sections 3.1 and 3.2.2. In chapters 2 and 3 we will propose a *fusion moves* based algorithm for optimizing the multicut and lifted multicut algorithm respectively. The concept of *fusion moves* is described in the following section.

1.3 Fusion Moves

For energy minimization problems *fusion moves* have become increasingly popular [66, 96, 102, 148]. The fusion move is an algorithm to combine pairs of suboptimal solutions using graph-cut [85] or QPBO [118]. For many large scale computer vision applications fusion moves yield good approximations with state of the art anytime performance [66]. Fusion moves can be described as a class of constrained search algorithms. They consist of two procedures. The first procedure is proposal generation that computes a feasible solution in a possible randomized randomized fashion. Second is *fusion* where the proposal is combined with the current best solution. This can be formalized in the following way: Given pairwise MRFs / CRFs in the form of

$$y^* = \arg\min_{y \in Y} \sum_{u \in V} U_u(y_u) + \sum_{uv \in E} V_{uv}(y_u, y_v) \tag{1.7}$$

and pair of labels $y^a, y^b \in Y$ (also called proposals), the fusion move is defined as:

$$\begin{aligned} y^* = \arg\min_{y \in Y} & \sum_{u \in V} U_u(y_u) + \sum_{uv \in E} V_{uv}(y_u, y_v) \\ s.t. \quad & x_i \in \{y_i^a, y_i^b\} \quad \forall y_i \in Y \end{aligned} \tag{1.8}$$

Equation (1.8) can be optimized with graph-cut [85] or QPBO [118]. We will use FM_{MRF} to refer to eq. (1.8). The proposals themselves can be computed by a domain specific method most suitable for the given task.

1.4 Contribution and Overview of this Thesis

The chapters are structured in the following way: In chapter 2 we generalize fusion moves [96] for the minimum multicut objective: Instead of directly optimizing the multicut objective, we iteratively *fuse* a current best solution with candidate solutions

such that the best energy is improved. The *fusion* procedure in itself is again a minimum multicut optimization problem with additional must-link constraints. We show how to formulate this as an *unconstrained*[1] minimum multicut problem with a smaller number of variables and constraints. We investigate how to generate high quality candidate solutions in an efficient manner. To this end we define two candidate solution generators based on the watershed transform and agglomerative clustering in conjunction with perturbed edge weights. We use these generators in an iterative manner and fuse the generated solutions with the current best solution. Based on this we derive a set of scalable algorithms with state-of-the art anytime performance yielding solutions close to global optimality.

In chapter 3 we generalize the *fusion moves* framework [96] and the algorithm presented in chapter 2 to the minimum lifted multicut problem [7, 78]: Again, we iteratively *fuse* a current best solution with candidate solutions to minimize the minimum lifted multicut objective function. The *fusion* procedure in itself is a minimum lifted multicut problem with additional must-link constraints. We show how to reformulate this as an unconstrained minimum lifted multicut problem[2]. We propose efficient candidate solution generators to quickly generate diverse high quality solutions which are *fused* with the current best solution. Based on this, we derived a set of scalable algorithms with state-of-the art anytime performance.

In chapter 4, we apply the algorithms proposed in chapters 2 and 3 to the problem of segmentation of neural structures in EM data. We propose a state-of-the art pipeline and validate every step in the pipeline with extensive experiments. We predict the membrane probability for each pixel using a convolutional neural network. We use the watershed transform on a distance transform height map based on the membrane probabilities to generate superpixels for each 2D slice of the 3D stack. Finally, we use a Random Forest to learn and predict which pairs of superpixels should be merged and jointly optimize this with the multicut and lifted multicut algorithms proposed in chapter 2 and chapter 3 respectively.

In chapter 5 we discuss the software implemented to conduct the experiments throughout this thesis. We provide implementations for algorithms and techniques described in chapters 2 to 4 resulting in a C++ software framework for graph partitioning and image segmentation. Not only do we provide fast and readable modern C++ code, but also a fully functional Python API.

In chapter 6 is a enumeration of all peer reviewed publication where I was author or co-author. Chapters 2 to 4 are based on publications [26, 27, 30] in top ranked venues.

[1] The initial multicut constraints are still part of the model, but no additional must-link constrains

[2] See footnote 1

2 Fusion Moves for Multicut Partitioning

Correlation clustering, or multicut partitioning, is widely used in image segmentation for partitioning an undirected graph or image with positive and negative edge weights such that the sum of cut edge weights is minimized. Due to its NP-hardness, exact solvers do not scale and approximative solvers often give unsatisfactory results.
We investigate scalable methods for correlation clustering. To this end we define fusion moves for the correlation clustering problem. Our algorithm iteratively fuses the current and a proposed partitioning which monotonously improves the partitioning and maintains a valid partitioning at all times. Furthermore, it scales to larger datasets, gives near optimal solutions, and at the same time shows a good anytime performance.

2.1 Introduction

Correlation clustering [24], also known as the multicut problem [41] is a basic primitive in computer vision [5, 8, 9, 146] and data mining [16, 38, 40, 120]. See Sec. 2.2 for its formal definition of clustering the nodes of a graph.
Its merit is, firstly, that it accommodates both positive (attractive) *and* negative (repulsive) edge weights. This allows doing justice to evidence in the data that two nodes or pixels do not wish or do wish to end up in the same cluster or segment, respectively. Secondly, it does not require a specification of the number of clusters beforehand.
In signed social networks, where positive and negative edges encode friend and foe relationships, respectively, correlation clustering is a natural way to detect communities [38, 40]. Correlation clustering can also be used to cluster query refinements in web search [120]. Because social and web-related networks are often huge, heuristic methods, *e.g.* the PIVOT-algorithm [3], are popular [40].
In computer vision applications, unsupervised image segmentation algorithms often start with an over-segmentation into superpixels (superregions), which are then clustered into "perceptually meaningful" regions by correlation clustering. Such an approach has been shown to yield state-of-the-art results on the Berkeley Segmentation Database [5, 8, 80, 146].
While it has a clear mathematical formulation and nice properties, correlation clustering suffers from NP-hardness. Consequently, partition problems on large scale data,

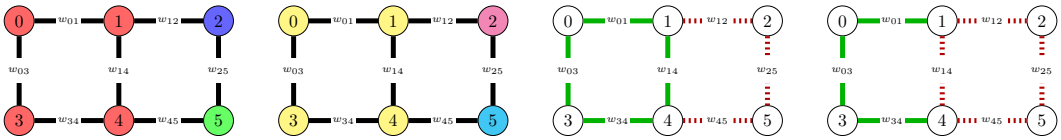
e.g. huge volume images in computational neuroscience [9] or social networks [97], are not tractable because reasonable solutions cannot be computed in acceptable time.

2.1.1 Contribution

In this chapter we present novel approaches that are designed for large scale correlation clustering problems. First, we define a novel energy based agglomerative clustering algorithm that monotonically increases the energy. With this at hand we show how to improve the anytime performance of Cut, Clue & Cut [29]. Second, we improve the anytime performance of polyhedral multicut methods [71] by more efficient separation procedures. Third, we introduce cluster-fusion moves, which extend the original fusion moves [96] used in supervised segmentation to the unsupervised case and give a polyhedral interpretation of this algorithm. Finally, we propose two versatile proposal generators, and evaluate the proposed methods on existing and new benchmark problems. Experiments show that we can improve the computation time by one to two magnitudes without worsening the segmentation quality significantly.

2.1.2 Related Work

A natural approach is to solve the integer linear program (ILP) directly 2.2. To this end, efficient separation procedures have been found [68, 71] that allow to iteratively augment the set of constraints until a valid partitioning is found. Alternatively, it is possible to relax the integrality constraints of the ILP formulation [71]. Such an outer relaxation can be iteratively tightened. However, intermediate solutions are fractional and therefore rounding is required to obtain a valid partitioning. For the latter approach column generating methods exist, which work best on planar graphs [146].
Another line of work uses move making algorithms to optimize correlation clustering [23, 29, 76]. Starting with an initial segmentation, auxiliary max-cut problems are (approximately) solved, such that the segmentation is strictly improved. As shown in [29] only Cut, Glue & Cut (CGC) can deal with large scale problems, but can also suffer from very large auxiliary problems.
Recently, a promising dual decomposition algorithm has been proposed [131] which relies on fast primal heuristics as proposed here for so called *rounding*.
Outside computer vision, greedy methods [3, 48, 55, 110, 126] have been suggested for correlation clustering problems, see [49] for an overview. The PIVOT Algorithm [3] iterates over all nodes in random order. If the node is not assigned it constructs a cluster containing the node and all its unassigned positively linked neighbors. A widely used post-processing method is Best One Element Move (BOEM) [55], which iteratively



(a) node labels (b) node labels (c) edge labels (d) dangling edge

Figure 2.1: Representing a clustering by node labels is ambiguous. 2.1a and 2.1b encode the same partition. Edge labels as in 2.1c do not suffer from such ambiguities, but can have *dangling edges* as in 2.1d. Node 1 and 4 are in the same connected component, even tough e_{14} is cut. This is not a valid partition, and must be ruled out by constraints.

reassigns nodes to clusters.
For energy minimization problems fusion moves have become increasingly popular [66, 96]. For many large scale computer vision applications fusion moves lead to good approximations with state of the art anytime performance [66] Due to the ambiguity of a node-labeling, classical fusion moves [96] cannot be applied directly for correlation clustering (see fig. 2.3). We will show how to overcome this problem in Sec. 2.4.

2.1.3 Outline

In Sec. 2.2 we give a detailed problem definition and introduce the correlation clustering objective. Next we give a description of energy based hierarchical clustering in Sec. 2.3 and our proposed correlation clustering fusion moves in Sec. 2.4. We evaluate the proposed methods in Sec. 2.5 and conclude in Sec. 2.6.

2.2 Notation and Problem Formulation

Let $G = (V, E, w)$ be a weighted graph of nodes V and edges E. The function $w : E \to \mathbb{R}$ assigns a weight to each edge. We will use w_e as a shorthand for $w(e)$. A positive weight expresses the desire that two adjacent nodes should be merged, whereas a negative weight indicates that these nodes should be separated into two distinct regions. A segmentation of the graph G can be either given by a node labeling $l \in \mathbb{N}^{|V|}$ or an edge labeling $y \in \{0, 1\}^{|E|}$, *cf*. Fig. 2.1. An edge labeling is only consistent if it does not violate any cycle constraint [41]. We denote the set of all consistent edge labelings by $P(G) \subset \{0, 1\}^{|E|}$. The convex hull of this set is known as

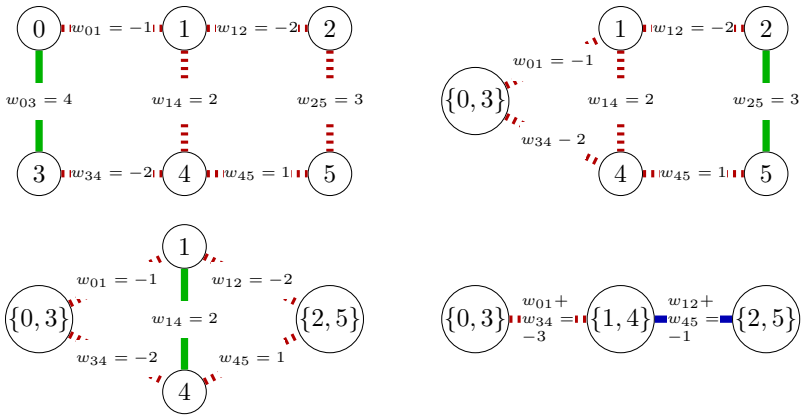


Figure 2.2: Energy-based Hierarchical clustering can be used to greedily optimize Eq. 2.2. In each step, the two nodes connected via the edge with the highest weight are merged by contracting this edge. (edge to be contracted is shown in green). Due to edge contraction parallel edges can occur, which are merged into single ones, and their weights are summed up. The algorithm terminates when the highest edge weight is smaller or equal to zero (edge shown in blue).

the *multicut polytope* $MC(G) = \text{conv}(P(G))$. By $l(y)$ we denote some node labeling for a segmentation given by y.
Given a weighted graph $G = (V, E, w)$ we consider the problem of segmenting G such that the costs of the edges between distinct segments is minimized. This can be formulated in the node domain by assigning each node i a label $l_i \in \mathbb{N}$

$$l^* = \underset{l \in \mathbb{N}^{|V|}}{\arg\min} \sum_{(i,j) \in E} w_{ij} \cdot [l_i \neq l_j], \tag{2.1}$$

or in the edge domain, by labeling each edge e as cut $y_e = 1$ or uncut $y_e = 0$

$$y^* = \underset{y \in P(G)}{\arg\min} \sum_{(i,j) \in E} w_{ij} \cdot y_{ij}. \tag{2.2}$$

As shown in [71] both problems are equivalent, but formulation 2.1 suffers from ambiguities in the representation, *cf*. Fig. 2.1.

2.3 Energy Based Hierarchical Clustering

Agglomerative hierarchical clustering (HC) is widely used in graph / image segmentation [18]. In each step, the edge with the highest weight w is contracted (green edges

in Fig. 2.2). Doing so, parallel edges can occur. In agglomerative clustering, weights of parallel edges are merged into single edges. For image segmentation, the length weighted mean is used to do this update [18].
Because we, contrary to [18], directly work on energies, we use energy based agglomeration with the following update rule: Whenever there are multiple edges between a pair of nodes, these edges are merged into a single edge and the weights are summed up, since we minimize the sum of the cut edges. We call HC with this update method Energy based Hierarchical Clustering (EHC).
We stop EHC if the highest edge weight is smaller or equal to zero (blue edge in Fig. 2.2). Any further edge contraction does not improve the energies.
Given the intrinsic greediness of hierarchical clustering, we cannot expect EHC to yield optimal solutions in general.
However, EHC is very fast and can be used to initialize CGC [29]. Excessive time in CGC is spent in the *cut phase* to solve the first two coloring on the complete graph. As shown in Sec. 2.5, allowing CGC to start from the EHC solution instead can improve performance drastically.

2.4 Correlation Clustering Fusion Moves

Fusion moves as defined in [96] work in the node domain and do not work properly for objective functions as Eq. 2.1 since the node coloring is ambiguous and has no semantic meaning, *cf*. Fig. 2.3. In the following, we propose a more suitable fusion move for correlation clustering which works on the edge domain. Given two proposal solutions y' and y'', $\mathcal{E}_0^{\breve{y}}$ is the set of edges which are uncut in y' and y''.

$$\breve{y}_{ij} = \max\{y'_{ij}, y''_{ij}\} \qquad \forall ij \in E \tag{2.3}$$

$$\mathcal{E}_0^{\breve{y}} = \{ij \in E \mid \breve{y}_{ij} = 0\} \tag{2.4}$$

The fusion move for correlation clustering is solving Eq. 2.2 with additional *must-link constraints* for all edges in $\mathcal{E}_0^{\breve{y}}$.

$$y^* = \arg\min_{y \in P(G)} \sum_{(i,j)\in E} w_{ij} \cdot y_{ij}. \tag{2.5}$$

$$\text{s.t.} \quad y_{ij} = 0 \qquad \forall (i,j) \in \mathcal{E}_0^{\breve{y}}$$

By construction, solving Eq. 2.5 cannot increase the energy w.r.t. the proposals y' and y'', because y' and y'' are feasible solutions for problem 2.5.

As Lempitsky *et al.* [96], we iteratively improve the best solution by fusing it with proposal solutions. The inherent difference is how we define the fusion.
As classical fusion, CC-Fusion does not provide a lower bound on the objective and has no sound stopping condition. For the latter we use a maximal number of iterations and maximal number of iterations without improvement.
A further difference is how we efficiently calculate the correlation clustering fusion move and how we generate proposals. Both will be discussed next. The overall framework is sketched in Fig. 2.5.

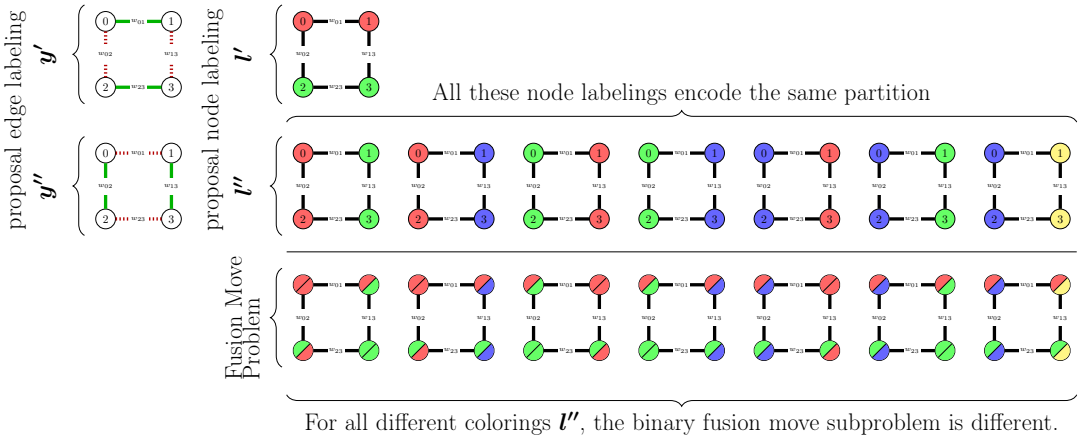


Figure 2.3: To fuse two edge labelings y' and y'' with fusion moves as defined by Lempitsky *et al.* [96] y' and y'' need to be transferred to the node domain. The mapping from edge labels to node labels is ambiguous and even for this small graph there are seven node labels which result in different binary fusion move problems. Enumerating all labelings for a graph of non trival size becomes intractable.

2.4.1 Fast Optimization of CC-Fusion Moves

In general the auxiliary fusion problem 2.5 is, as for classical fusion [96], NP-hard. However, many variables have been fixed to be zero and we can reformulate 2.5 into a correlation clustering problem on a coarsened graph, where all nodes which are connected via must-link constraints are merged into single nodes. We call this graph a *contracted graph*.

Definition 1. *(Contracted Graph)* Given a weighted graph $G = (V, E, w)$ and a segmentation of G given by $y \in P(G)$, we define the contraction of graph $G_y = (V_y, E_y, \bar{w})$ by $V_y = \{l_i(y) | i \in V\}$, $E_y = \{l_i(y) l_j(y) | ij \in E\}$, and $\forall \bar{u}\bar{v} \in E_y : \bar{w}_{\bar{u}\bar{v}} = \sum_{ij \in E, l_i(y)=\bar{u}, l_j(y)=\bar{v}} w_{ij}$

Any clustering $\bar{y}$ of the contracted graph $G_y = (V_y, E_y)$ can be *back projected* to a clustering $\tilde{y}$ of the original graph $G = (V, E)$ by

$$\tilde{y}_{ij} = \begin{cases} \bar{y}_{l_i(y) l_j(y)} & \text{if } l_i(y) \neq l_j(y) \\ 0 & \text{else} \end{cases} \qquad \forall uv \in E \tag{2.6}$$

Theorem 1 (Equivalence)**.** *The back projection of the optimal segmentation $\bar{y}'$ of the contracted graph $G_y = (V_y, E_y, \bar{w})$ is an optimal solution of problem 2.5.*

Proof. Let y' be the back propagation of $\bar{y}'$, which is by definition feasible for 2.5. If y' would not be an optimal solution, there must be a y'' with $\sum_{e \in E} w_e y'_e > \sum_{e \in E} w_e y''_e$. Since y'_e and y''_e are 0 for all $e \in \mathcal{E}_0^{\check{y}}$ we would have

$$\begin{aligned} \sum_{\bar{e} \in E_y} \bar{w}_{\bar{e}} \bar{y}'_{\bar{e}} &= \sum_{e \in E \setminus \mathcal{E}_0^{\check{y}}} w_e y'_e = \sum_{e \in E} w_e y'_e \\ &> \sum_{e \in E} w_e y''_e = \sum_{e \in E \setminus \mathcal{E}_0^{\check{y}}} w_e y''_e = \sum_{\bar{e} \in E_y} \bar{w}_{\bar{e}} \bar{y}''_{\bar{e}} \end{aligned}$$

where $\bar{y}''$ is the projection from y'' on G_y. This contradicts that y' is a optimal segmentation of G_y. □

Instead of problem 2.5 we can now solve problem 2.2 on the contracted graph $G_{\check{y}}$. This is, depending on the intersection of the current and proposed solution, magnitudes smaller than G. The correlation clustering problem on $G_{\check{y}}$ can be solved by any correlation clustering solver. Since $G_{\check{y}}$ is smaller, exact methods or good approximative methods like multicuts [65] or CGC [29] are very fast.

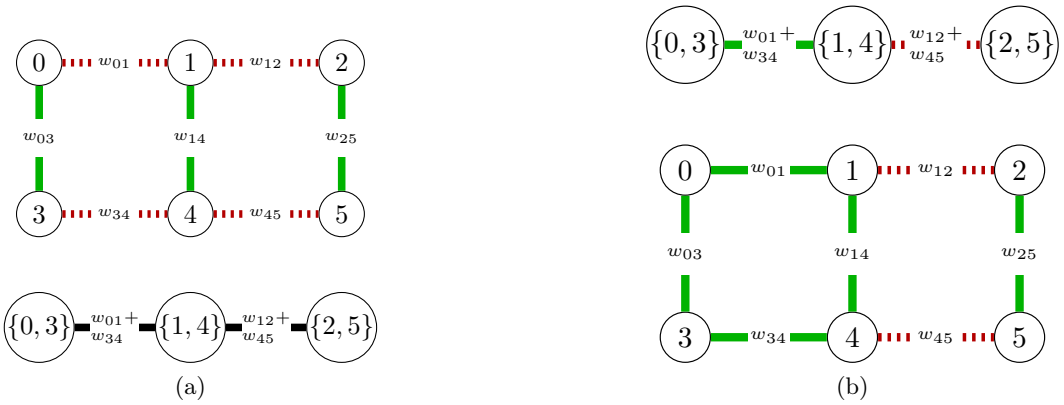


Figure 2.4: **(a)** Given a graph $G = (V, E, w)$ and a consistent edge labeling $y \in P(G)$, shown by solid and dotted lines, the contraction graph $G_y = (V_y, E_y, w_y)$ is constructed by contracting uncut edges in G w.r.t. y. **(b)** Given an edge labeling $\bar{y}$ of a contracted graph $G_y = (V_y, E_y, w_y)$, we can back project the edge labeling to the original graph.

2.4.2 Polyhedral Interpretation

A polyhedral interpretation of fusion moves is shown in Fig. 2.6. In each iteration the current and proposed segmentation define an inner polyhedral approximation of the original polytope. This interpretation holds for original fusion moves [96] as well as for the proposed CC-Fusion.

In our case, optimizing over the inner polytope is the same kind of problem as the original multicut polytope, but much smaller. Furthermore, the cost do not change and an improvement in the smaller polytope will be the same in the original graph, as shown in Theorem 1.

The choice of the proposal defines the shape of the inner polytope. In the given toy example, the first (red) polytope gives a huge improvement, the second proposal defines the blue polytope which does not lead to an improvement. The third proposal generates the green polytope that includes the globally optimal solution.

This procedure is fundamentally different from common polyhedral multicut methods [65, 67], which tighten an outer relaxation of the multicut polytope and contrary to our method do not operate in the feasible domain.

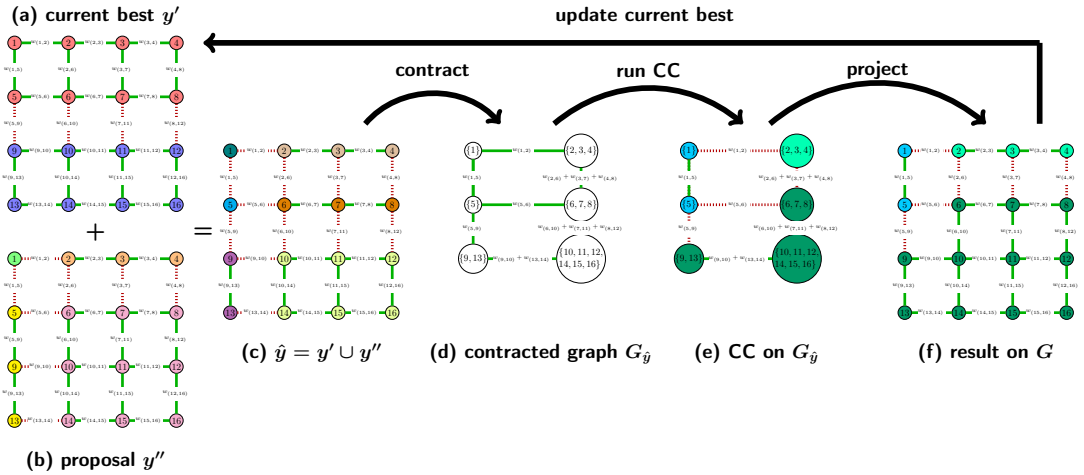


Figure 2.5: To fuse a current best segmentation y' (2.5a) with a proposal segmentation y'' (2.5b) we propose the following algorithm: $\hat{y}$ is defined as $y' + y''$ as in (2.5c). The contraction graph (2.5d) $G_{\hat{y}}$ is constructed by contracting all uncut edges in $\hat{y}$. The actual fusion move is solving eq. (2.2) for $G_{\hat{y}}$ as in (2.5e) and projecting the result back to G as in (2.5f). The result of the fusion move is guaranteed to be no worse than y' or y''. Therefore the current best solution can be updated from the result of the fusion move. In summary, the correlation clustering fusion move algorithm iteratively fuses the current best solution with different proposals.

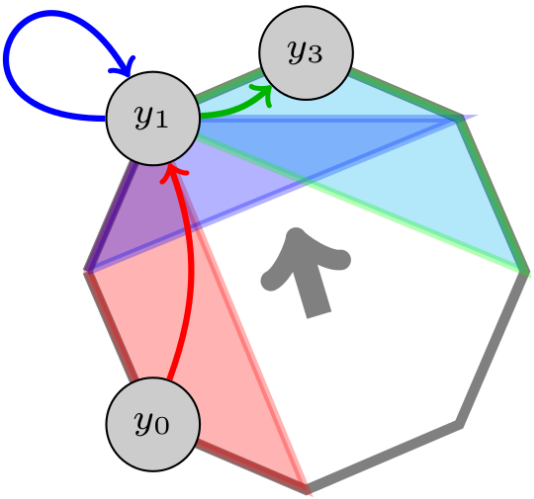


Figure 2.6: Each fusion move can be interpreted as an optimization of an inner polytope. Each inner polytope includes the current vertex. Starting with y_0 we optimize over the red polytope and find y_1 as optimum. Finally, when optimizing over the blue polytope we stay in y_1 as optimum, when optimizing over the green polytope we find y_2 which we will never leave again.

2.4.3 Proposal Generators

As discussed in [96], proposals should have two properties: *high quality* and *large diversity*.
A proposal has a high quality if it has a low energy at least in some regions. For high quality proposals the chance that the inner polytope includes a better solution (vertex) is larger than for those with low quality.
Diversity between the individual proposals increases the chances to span diverse internal polytopes, cover with the intersection of inner polytopes a large part of the original polytope and find more likely the globally optimal solution or escape from local minima.
For correlation clustering fusion we add a third property: *size*. The size of the contracted graph directly depends on the number of connected components of the intersection of the proposal solution and the current best solution. In one extreme case, where each node is in a separate connnected component, the fusion move is equivalent to solving the original problem. In the other extreme, where the proposal has a single connected component, the current best solution will not change. Therefore the size of the proposals should be small enough, such that solving eq. 2.5 can be done fast enough, but on the other side large enough to define a large internal polytope and therefore a powerful move. To this end we suggest two proposal generators.
Randomized Hierarchical Clustering (RHC): To generate fast energy aware proposals we can use energy based hierarchical clustering (EHC) as defined in Sec. 2.3. EHC follows the energy function, therefore the *quality* of the proposals is high. To get *diversity* among the different proposal, we add normally distributed noise $\mathcal{N}(0, \sigma_{ehc})$ to each edge weight. To get proposals of the desired *size*, we use a different stop condition for EHC, and stop only if a certain number of connected components is reached.
Randomized Watersheds (RWS): Watersheds have become quite popular for graph segmentation and have a strong connection to energy minimization [45]. The edge weighted watershed algorithm [109] with random seeds can be used to find cheap proposals. To improve *quality* we do not use n seeds distributed uniformly over all nodes but use the following. We draw $n/2$ negative edges, and assign different seeds to the endpoints of each edge. Doing so, a random subset of negative edges is forced to be cut within each proposal. For additional *diversity*, noise $\mathcal{N}(0, \sigma_{ws})$ is added to the edge weighs [130].

2.5 Experiments

In our experiments we compare to the following methods with publicly available implementation. For **CGC** [29] we used a branch of OpenGM [1] and for **KL** [76] the implementation in OpenGM[2]. For integer multicuts (**MC-I**) and relaxed multicuts (**MC-R**) [65] we modified OpenGM[2], as described in Sec. 2.5.2.
From the field of data-mining we compare to the PIVOT-algorithm [3] followed by a round of BOEM [55] denoted by **PIVOT-BOEM**[3]. This implementation uses full adjacency matrices it does not scale and cannot be applied to all datasets. We also run classical fusion moves [96] (**Fusion**) and select distinct labels for the two candidate segmentations. According to [29], CGC is faster and gives better energies than PlanarCC [146] and Expand & Explore [23]. Therefore we exclude those in our experiments.
We compare all of the above to the following methods suggested in the present paper: Energy Based Hierarchical Clustering (**EHC**), as described in Sec. 2.3. CGC warm started with the solution from EHC (**EHC-CGC**). The proposed correlation cluster fusion algorithm with EHC-based and watershed-based proposals and MC-I and CGC as subproblem solvers (**CC-Fusion-HC-MC**, **-HC-CGC**, **-WS-MC**, and **-WS-CGC**) respectively. We set the number of connected components in the proposals to 10% of the number of nodes of and use random edge noise with $\sigma = 1.5$. As stopping condition we choose 10^4 iterations and 100 iterations with no improvement. All experiments were run on Intel Core i5-4570 CPUs with 3.20 GHz, equipped with 32 GB of RAM. In our evaluation we make no use of multiple threads. The methods were stopped once they exceed 30 minutes at the next possible interrupt point.

2.5.1 Datasets

Social Networks. One important application for large scale correlation clustering are social networks. We consider two of those networks from the Stanford Large Network Dataset Collection[4]. Both networks are given by weighted directed graphs with edge weights -1 and $+1$. The first network is called *Epinions*. This is a who-trust-whom online social network of a general consumer review site. Each directed edge $a \to b$ indicates that user a trusts or does not trust user b by a positive or negative edge-weight, respectively. The network contains 131828 nodes and 841372 edges from which

[1] github.com/opengm/opengm/tree/cgc-cvpr2014
[2] github.com/opengm/opengm
[3] http://www.ling.ohio-state.edu/~melsner/resources/correlation-readme.html
[4] http://snap.stanford.edu/data/index.html

85.3% are positively weighted. The second network is called *Slashdot*. Slashdot is a technology-related news website known for its specific user community. In 2002 Slashdot introduced the Slashdot Zoo feature which allows users to tag each other as friend or foe. The network was obtained in November 2008 and contains 77350 nodes and 516575 edges of which 76.73% are positively weighted.
We consider the problem to cluster these graphs such that positively weighted edges ($E^+_{\rightarrow}$) link inside and negatively weighted edges ($E^-_{\rightarrow}$) between clusters. In other words friends and people who trust each other should be in the same segment and foes and non-trusting people in different clusters. To compensate the large impact of nodes with high degree we can normalize the edge weights such that each person has the same impact on the overall network, by enforcing.

$$\sum_{i \to j \in E_{\rightarrow}} |w_{i \to j}| = 1 \qquad \forall i \in V, deg^{\text{out}}(i) \geq 1 \tag{2.7}$$

We define the following energy function

$$\begin{aligned} J(y) &= \sum_{i \to j \in E^+_{\rightarrow}} y_{ij} \cdot w_{i \to j} + \sum_{i \to j \in E^-_{\rightarrow}} (y_{ij} - 1) \cdot w_{i \to j} \\ &= \sum_{ij \in E} y_{ij} \cdot \underbrace{(w_{i \to j} + w_{j \to i})}_{w_{ij}} + \text{const} \end{aligned} \tag{2.8}$$

which is zero if the given partitioning does not violate any relation and larger otherwise. We name these two datasets *social nets* and *normalized social nets*.
Network Modularity Clustering. As another example for network clustering we use the *modularity-clustering* models from [70] which are small but fully connected.
2D and 3D Image Segmentation To segment images or volumes into a previously unknown number of clusters, correlation clustering has been used [8, 9].
Starting from a super-pixel/-voxel segmentation, correlation clustering finds the clustering with the lowest energy. The energy is based on a likelihood of merging adjacent super-voxels. Each edge has a probability to keep adjacent segments separate ($p(y_{ij} = 1)$) or to merge them ($p(y_{ij} = 0)$). The energy function is

$$J(y) = \sum_{ij \in E} y_{ij} \cdot \underbrace{log\left(\frac{p(y_{ij} = 0)}{p(y_{ij} = 1)}\right) + log\frac{1 - \beta}{\beta}}_{w_{ij}} \tag{2.9}$$

where β is used as a prior [8].

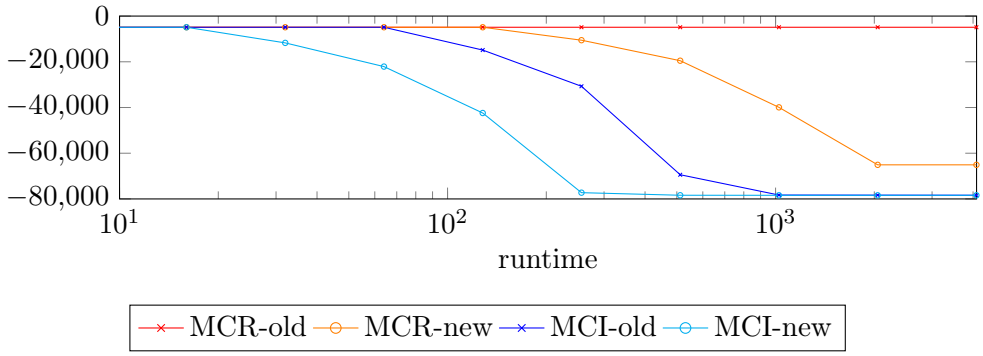


Figure 2.7: Comparison of the Multicut implementation of OpenGM and our modified implementation, which improves the runtime. However, for large scale problems it still does not scale.

We use the publicly available benchmark instances from [69, 70]. For 2D images from the Berkeley Segmentation Database [107] we took the segmentation problems called *image-seg* [8, 69]. For 3D volume segmentation we use the models *knott-3d-150*, *-300* and *-450* from [9, 70] as well as the large instance from the *3d-seg* model [8, 69]. These instances have underlying cube sizes of 150^3, 300^3, 450^3, and 900^3, respectively. We also requested larger instances from the authors of [9] who kindly provided us the dataset *knott-3d-550* with cube size 550^3.

2.5.2 Improvements for the Multicut Algorithm

When using the publicly available implementation in OpenGM[2], we have noticed that their implementation has some limitations on large problems. This results in a very slow separation and we make the following modifications. Firstly, we used *index-min-heap* [122] within the shortest path search by the Dykstra algorithm, which speeds up MC-R. Secondly, we follow [9] and search for shortest paths and add those only if they are non-chordal,instead of searching for the shortest non-chordal path during the separation procedure. In [9] this was used for MC-I only. For MC-R this search procedure is not sufficient and needs to be followed by a search for shortest non-chordal paths. Fig. 2.7 shows the improvements with our modifications compared to the implementation in OpenGM for the knott-3d-450 dataset. This procedure is one magnitude faster, but might cause a few extra outer iterations.

2.5.3 Parameter Choice for CC-Fusion

Beside the choice of the proposal generator and fusion method, CC-Fusion has some more parameters, which need to be set. The most crucial one is the number of segments in the proposal. For HC we also have to set the noise which is used to generate diversity. Fig. 2.8 shows an evaluation of the impact of this parameters for a single instance of knott-3d-450 averaged over several random seeds. The runtime depends on the number of clusters in the proposal (Fig. 2.8 left), which controls the size of the auxiliary move problems. The level of noise has no major impact on the runtime. The energy of the final solution improves with finer proposals since this increases the search space of the moves. The level of noise has to be large enough to generate diverse proposals, but not too large as this would lead to proposals with low quality. As shown in Fig. 2.8 right the useful parameter set is quite large. This allows us to use the same parameters for *all* experiments. However we would like to note that in practice we can improve the performance by adjusting these parameters for the specific problem setting.

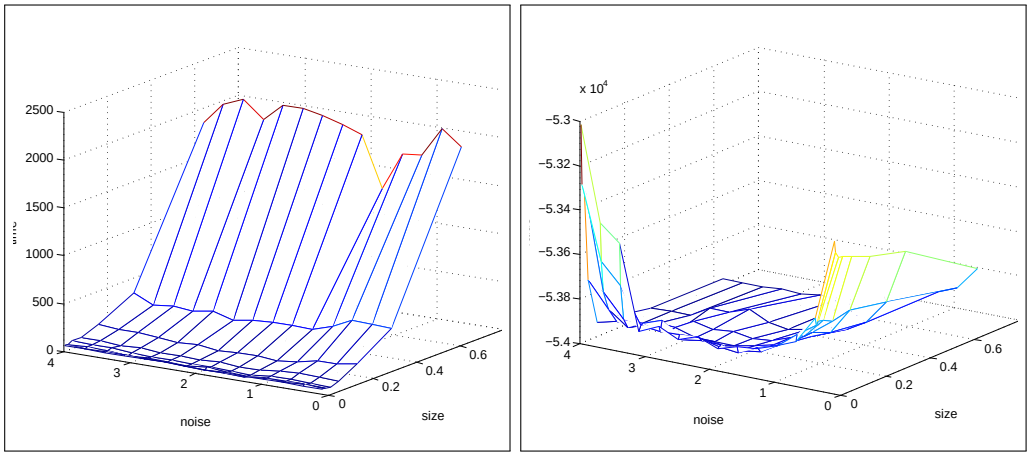


Figure 2.8: Empirical evaluation of the impact of noise used for proposal generation and the size of the proposals. Proposals with many segments cause longer runtime. Noise seemed not to be a critical parameter but should be selected large enough.

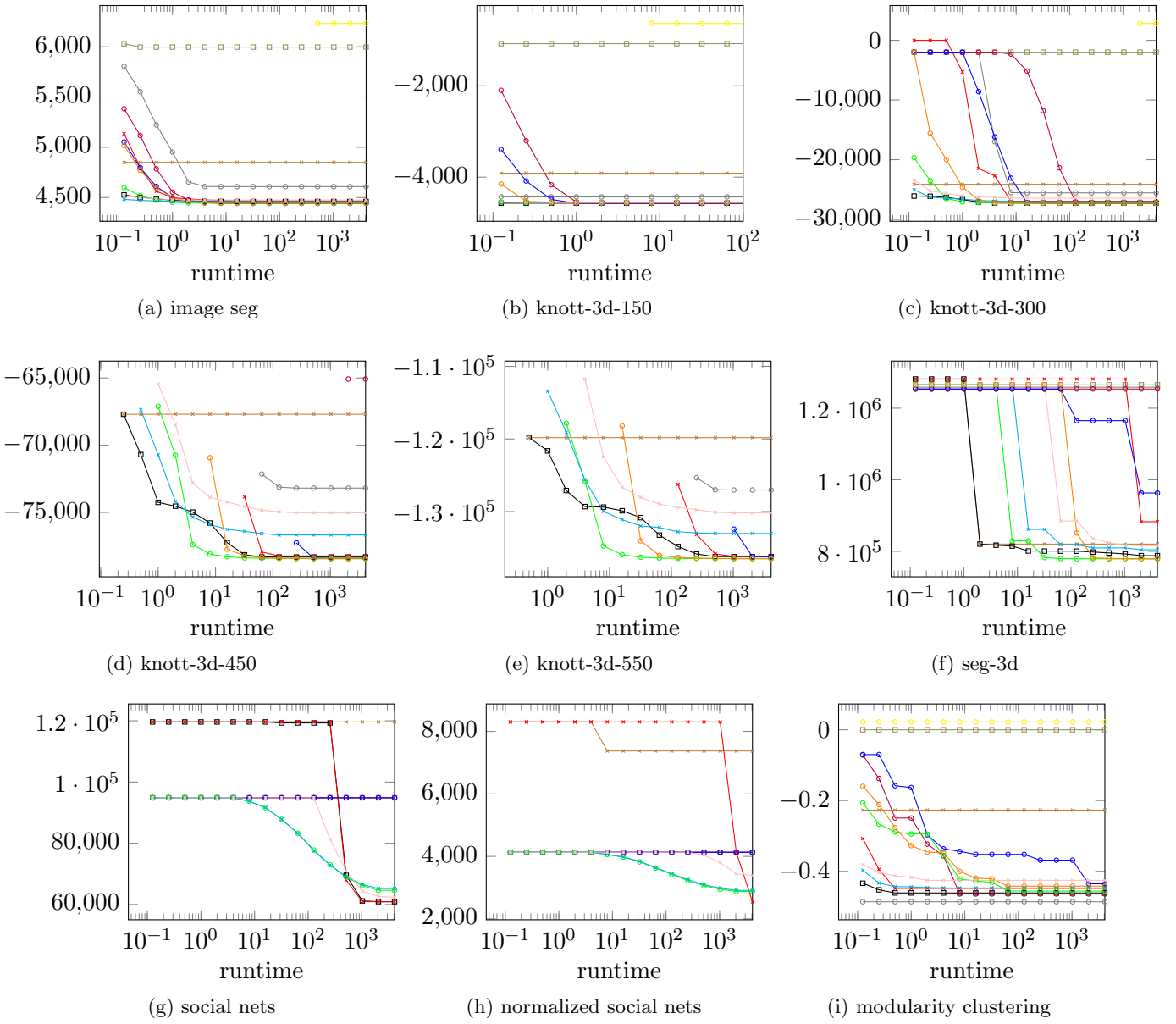
2.5.4 Evaluation

For the evaluation of the different methods on the datasets introduced in Sec. 2.5.1, we show zoomed anytime plots in Fig. 2.9 and variation of information (VOI) [108] and rand index (RI) [116] of the final solutions in Tab. 2.1. Anytime plots with no

zooming and a detailed evaluation are given in the supplementary material. For *social-nets* CC-Fusion methods provide the best results for the first minutes . Only CGC and HC-CGC are able to find better solutions after more than 1000 seconds. MC-I and MC-R cannot be applied to such large problems. We believe that with better proposal generators, which are more suited for such network problems, we can improve CCFusion. One candidate for such a generator would be a scalable implementation of the PIVOT algorithm. For *modularity-clustering* CC-Fusion performs on par with competitive methods, even though CC-Fusion and the used parameters have not been designed and chosen for this type of problem. In particular, it does a better job than MC-I. For *image-seg* CC-Fusion is faster than other methods and competitive in terms of energy, VOI, and RI. Because the models are designed to have a high boundary recall (oversegmentation), classical fusion, which returns undersegmentations, has best VOI but worse RI and energy. Proposals generated by EHC are a bit better than WS-based ones. For the *knott-datasets* CC-Fusion-HC-MC and CC-Fusion-WS-MC have a better performance with increasing problem size compared to competitive methods, *cf*. Fig. 2.9(b-e). Also in terms of VOI and RI they are only slightly worse than the globally optimal solution found by MC-I. The initialization of CGC by HC, denoted by HC-CGC, also improves the performance compared to native CGC. For the largest 3D volume seg-3d, HC-CGC gives the first useful solution, *cf*. Fig. 2.9(f). However, after a few minutes CC-Fusion-HC-MC and CC-Fusion-WS-MC give much better results and are also overall best in terms of energy, VOI, and RI. Pure EHC, Fusion and PIVOT-BOEM do not give useful results on any dataset.

2.6 Conclusion

We have presented a fast and scalable approximate solver for correlation clustering, named Correlation Clustering Fusion (CC-Fusion). It is orthogonal to previous research, *i.e.* it can be combined with any correlation clustering solver. The best solution is iteratively improved by a fusion with proposal solutions. The fusion move itself is formulated as correlation clustering on a smaller graph with fewer edges and nodes and can therefore be solved much faster than the original problem. Our evaluation shows that several CC-Fusion algorithms outperform many existing solvers w.r.t. anytime performance with increasing problem size.



(a) image seg (b) knott-3d-150 (c) knott-3d-300

(d) knott-3d-450 (e) knott-3d-550 (f) seg-3d

(g) social nets (h) normalized social nets (i) modularity clustering

Figure 2.9: Among all proposed solvers, Fusion-HC-MC has the best overall anytime performance. With increasing problem size (2.9b-2.9e and 2.9f) the runtimes of MC-I, MC-R and CGC increase drastically, while the proposed solvers still scale well. For these instances, the EHC started version of CGC outperforms GCG in terms of runtime and energy.

Overall, energy hierarchical clustering based proposals work better than watershed based proposals. They converge to similar energies but the clustering based approach is faster on all tested instances. On all instances except for modularity clustering, it is better to solve the fusion move to optimality (Fusion-HC-MC) than using approximations (FUSION-HC-GCG). The warm EHC started version of GCG (EHC-CGC) performs better than GCG itself, but both are outperformed by the proposed algorithms w.r.t. anytime performance.

For the modularity clustering instances in fig. 2.9i we see an interesting behavior. On these complete graphs, Kernighan Lin (KL) has the best performance. The proposed methods perform reasonably, but KL is faster and leads to better energies.

Table 2.1: Evaluation by Variation of Information (VOI) and Rand Index (RI) for datasets with available ground truth.

VOI	image-seg	knott-3d-150	knott-3d-300	knott-3d-450	3d-seg
PIVOT-BOEM	4.9633	2.9936	4.4986	–	–
HC	2.5967	1.5477	2.3513	2.9155	2.8395
HC-CGC	2.5164	0.9052	1.7636	2.2256	1.7603
CGC	2.5247	0.9267	1.8822	2.3104	6.8908
KL	2.6432	2.0648	4.1318	4.9270	7.1057
FUSION	**2**.1406	2.8787	4.0744	4.6616	6.5366
MC-R	2.5471	0.9178	1.6369	2.8710	6.5058
MC-I	2.5367	**0**.9063	**1**.6352	**2**.0037	4.3319
CC-Fusion-HC-MC	2.5319	0.9629	1.6516	2.0801	**1**.3347
CC-Fusion-HC-CGC	**2**.4961	0.9679	1.7673	2.3809	2.1347
CC-Fusion-WS-MC	2.5340	0.9629	1.6742	2.0739	**1**.3334
CC-Fusion-WS-CGC	2.5192	1.0585	2.1344	2.7487	3.3514

RI	image-seg	knott-3d-150	knott-3d-300	knott-3d-450	3d-seg
PIVOT-BOEM	0.7438	0.7851	0.8792	–	–
HC	0.7560	0.8139	0.8084	0.7610	0.9651
HC-CGC	0.7724	0.9226	0.8713	0.8433	0.9861
CGC	0.7590	0.9206	0.8666	0.8341	0.6024
KL	0.6400	0.8085	0.6858	0.6409	0.5849
FUSION	0.5480	0.2849	0.1420	0.0998	0.0345
MC-R	0.7822	0.9232	**0**.8849	0.6713	0.0432
MC-I	0.7821	**0**.9236	**0**.8849	**0**.8670	0.5461
CC-Fusion-HC-MC	0.7801	0.9042	0.8824	0.8573	**0**.9906
CC-Fusion-HC-CGC	0.7780	0.9031	0.8763	0.8470	0.9775
CC-Fusion-WS-MC	**0**.7825	0.9042	0.8802	0.8582	0.8895
CC-Fusion-WS-CGC	0.7750	0.8951	0.8596	0.8394	**0**.9906

3 Fusion Moves for Lifted Multicut Partitioning

Many computer vision problems can be cast as an optimization problem whose feasible solutions are decompositions of a graph. The minimum cost lifted multicut problem is such an optimization problem. Its objective function can penalize or reward all decompositions for which any given pair of nodes are in distinct components. While this property has many potential applications, such applications are hampered by the fact that the problem is NP-hard. We propose a fusion move algorithm for computing feasible solutions, better and more efficiently than existing algorithms. We demonstrate this and applications to image segmentation, obtaining a new state of the art for a problem in biological image analysis.

3.1 Introduction and Related Work

In 2011, Andres et al. [8], Bagon and Galun [23], Kim et al. [80, 81] and Yarkony et al. [146] independently proposed formulating the image segmentation problem [17] as a minimum cost multicut problem [25, 46] on a suitable graph. Given, for every pair of neighboring nodes, a cost or reward (negative cost) to be paid if these nodes are assigned to distinct components, the minimum cost multicut problem consists in finding a decomposition of the graph with minimal sum of costs. In 2015, Keuper et al. [78], using a construction from [7], proposed the minimum cost *lifted* multicut problem, a generalization with an identical feasible set whose objective function can assign a cost or reward to *every* pair of nodes, not just neighboring ones. These non-local interactions are represented in the graph by "lifted" edges which are subjected to slightly different constraints than the regular edges. The introduction of lifted edges is appealing for image segmentation, because non-local interactions can now be added without losing two key advantages of the multicut: (i) Every feasible solution of the optimization problem corresponds to a decomposition of the graph, i.e. to a consistent segmentation. (ii) No assumptions on the number or size of segments are made, making the method applicable in the typical and important scenario where such prior knowledge is not available. Since standard and lifted multicut are both NP-hard in-

teger linear programming problems [25, 46] – even for planar graphs [22, 142] – this paper proposes a new family of efficient heuristics inspired by [44, 96] and on the basis of *fusion moves* [66, 96] .
So far, the computer vision community has studied three classes of algorithms addressing optimization problems of this type: (i) branch-and-cut algorithms [8, 9, 72] that converge to an optimal integer solution but do not admit polynomial time complexity bounds and are too slow for lifted multicuts; (ii) linear programming relaxations with subsequent rounding to an integer solution [68, 72, 146] which can yield a log-factor approximation [46] in polynomial time; (iii) constrained search algorithms [12, 29, 78] that find approximate integer solutions directly in polynomial time. Although no theoretical guarantees are known for the latter approximations, they tend to be better than relaxation followed by rounding.
Constrained search algorithms for the lifted multicut problem were introduced in [78]. They generalize multicut algorithms of the Kernighan/Lin [76] type from [12] and greedy additive edge contraction from [29]. We show in this chapter that fusion move algorithms for the multicut as proposed in [27] can be generalized as well and actually perform better in terms of approximation quality and speed.

3.1.1 Contribution

With this chapter, we make the following contributions:

1. We generalize the fusion move algorithm [27] into a new constrained search algorithm for the minimum cost lifted multicut problem defined in [78].

2. We show that our algorithm outperforms the constrained search algorithms of [78] on the same problem instances in approximation quality and speed.

3. We introduce novel non-local potentials for the segmentation problem and incorporate them into a lifted multicut formulation of the objective.

4. We apply the proposed algorithm to the biological image segmentation benchmark [21, 37], achieving the highest accuracy known at the time of writing.

3.2 Optimization Problem

3.2.1 Minimum Cost Multicut Problem

The minimum cost multicut problem is an optimization problem whose feasible solutions can be identified with the decompositions of a graph. Below, we recall only the

necessary basic definitions and otherwise refer to [41, 56] for details.
A *decomposition* of a graph is a partition of the node set into connected subsets. More rigorously, a decomposition of a graph $G = (V, E)$ is a partition Π of the node set V such that, for every $U \in \Pi$, the subgraph of G induced by U is connected. Every decomposition of a graph can be identified with the set of edges that straddle distinct components. Such subsets of edges are called the multicuts of the graph.
A subset $M \subseteq E$ of edges is a *multicut* of G iff there exists a decomposition Π of G such that M is the set of edges straddling distinct components. Moreover, M is a multicut of G iff no cycle in the graph intersects with M precisely once. Rigorously, for every cycle $Y \subseteq E$ of G: $|M \cap Y| \neq 1$. This characterization is intuitive: If one transitions from one component to another along the cycle, one needs to transition back before returning to the node from which one has started. It is used to state the minimum cost multicut problem:
For every graph $G = (V, E)$ and every $c : E \to \mathbb{R}$, the instance of the *minimum cost multicut problem* w.r.t. G and c is the optimization problem

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

$$\text{subject to} \quad \forall Y \in \text{cycles}(G)\ \forall e \in Y : \ x_e \leq \sum_{e' \in Y \setminus \{e\}} x_{e'} \ . \tag{3.2}$$

3.2.2 Minimum Cost Lifted Multicut Problem

The minimum cost multicut problem has a limitation: A multicut makes explicit only for *neighboring* nodes whether these nodes are in distinct components of the decomposition induced by the multicut. It does not make this explicit for *non-neighboring* nodes. Thus, the cost function can introduce only for pairs of neighboring nodes a cost or reward to be paid by feasibles solutions that assign these nodes to distinct components. It cannot introduce such a cost for pairs of non-neighboring nodes. As illustrated in Fig. 3.1, simply considering a graph with more edges does not overcome this limitation in general.
This limitation led Andres [7] to define the minimum cost lifted multicut problem w.r.t. one graph $G = (V, E)$ whose decompositions are identified with feasible solutions, and a possibly larger graph $G' = (V, E')$ with $E \subseteq E'$ for whose every edge $vw \in E'$ it is made explicit whether the nodes v and w are in distinct components. By assigning a cost $c_{vw} \in \mathbb{R}$ to this edge, one can penalize or reward precisely those decompositions of G (!) for which the nodes v and w are in distinct components. This property is used

for image segmentation in [78]. We recall the minimum cost lifted multicut problem from [7, Def. 10].
For any graphs $G = (V, E)$ and $G' = (V, E')$ with $E \subseteq E'$ and every $c : E' \to \mathbb{R}$, the instance of the *minimum cost lifted multicut problem* w.r.t. G, G' and c is the optimization problem

$$\min_{x \in \{0,1\}^{E'}} \quad \sum_{e \in E'} c_e x_e \tag{3.3}$$

$$\text{subject to} \quad \forall Y \in \text{cycles}(G)\ \forall e \in Y :\ x_e \leq \sum_{e' \in Y \setminus \{e\}} x_{e'} \tag{3.4}$$

$$\forall vw \in E' \setminus E\ \forall P \in vw\text{-paths}(G) :\ x_{vw} \leq \sum_{e \in P} x_e \tag{3.5}$$

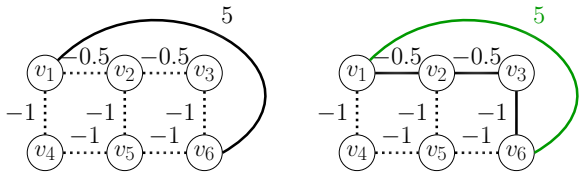
$$\forall vw \in E' \setminus E\ \forall C \in vw\text{-cuts}(G) :\ 1 - x_{vw} \leq \sum_{e \in C} (1 - x_e)\ . \tag{3.6}$$

The cycle constraints (3.4) are identical to those in (3.2). Additional constraints (3.5) and (3.6) ensure, for every edge $vw \in E' \setminus E$ that $x_{vw} = 0$ if (3.5) and only if (3.6) v and w are connected in G by a path of edges labeled 0, i.e., iff v and w are in the same component of G defined by the multicut $M := \{e \in E | x_e = 1\}$ of G. Or in other words, iff a lifted edge ($vw \in E' \setminus E$) is not cut, there must be a path of non-cut edges in the original graph connecting v and w.

3.3 Optimization Algorithm

3.4 Constrained Search Algorithms

Constrained search is a class of heuristic optimization algorithms. In the computer vision community, they are also commonly referred to as move making algorithms. Examples are α-expansion [86] $\alpha\beta$-swap [86], lazy flipping [10] and fusion [96].
Given a map $f : X \to \mathbb{R}$ and the optimization problem $\min \{f(x) \,|\, x \in X\}$, the idea of constraint search is this: Instead of optimizing f over the entire feasible set X, which might be hard, start from an initial feasible solution $x_0 \in X$, optimize f over a neighborhood $N(x_0) \subseteq X$ to obtain a new feasible solution x_1. Iff $f(x_1) < f(x_0)$, re-iterate, starting from x_1. Note that this algorithm does not require that x_1 be optimal.
Typically, the neighborhood function $N : X \to 2^X$ is chosen such that, for every $x \in X$, we have $x \in N(x)$. If N is chosen such that, for every $x \in X$, the problem



(a) Multicut (b) Lifted Multicut

Figure 3.1: Depicted above in (a) is an instance of the minimum cost multicut problem (3.1)–(3.2). The solution is the multicut consisting of those edges that are depicted as dotted lines. I.e. all edges except v_1v_6 are cut. Depicted above in (b) is an instance of the minimum cost lifted multicut problem (3.3)–(3.6) with one edge in $E' \setminus E$ depicted in green. Here as well, the solution is the lifted multicut consisting of those edges depicted as dotted lines. Note that, unlike in (a), the lifted edge with cost 5 causes the nodes v_1 and v_6 to be connected in G by a path of edges labeled 0. Thus, positive costs assigned to lifted edges are called an *attraction*.

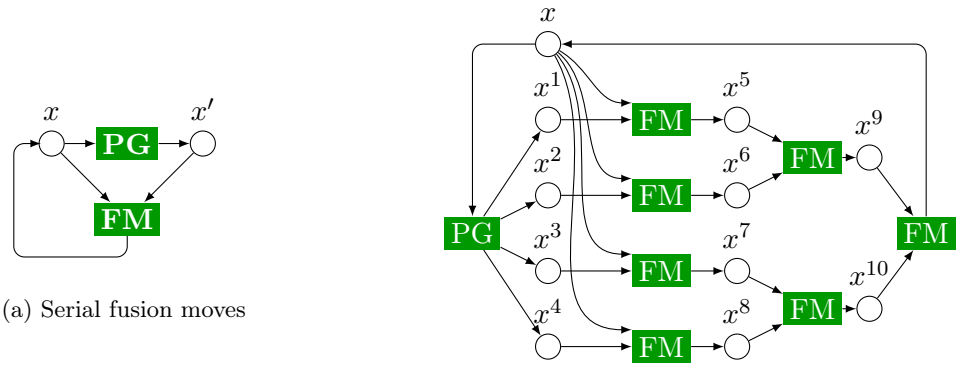
$\min \{f(x') \,|\, x' \in N(x)\}$ is of polynomial time complexity, then every iteration of the algorithm is efficient. If the optimization over the neighborhood is not known to be of polynomial complexity, it can still be less complex or smaller than the original problem and can thus be tractable in practice.

3.4.1 Fusion Move Algorithms

Fusion move algorithms [96] are a class of constrained search algorithms. They consist of two procedures. First is *proposal generation* that computes, for every feasible solution $x \in X$ given as input, another feasible solution $\text{PG}(x) \in X$ as output, possibly in a randomized fashion. Second is *fusion*, an optimization algorithm that computes a feasible solution of an optimization problem $\min \{f(x) \,|\, x \in N(x)\}$ for a neighborhood $N(x)$ defined w.r.t. x and $\text{PG}(x)$ such that $x \in N(x)$ and $\text{PG}(x) \in N(x)$, to obtain a feasible solution x' with $f(x') \leq f(x)$ and $f(x') \leq f(\text{PG}(x))$. In a fusion move algorithm, proposal generation and fusion can be combined in different ways, as depicted in Fig. 3.2.

3.4.2 Fusion Moves for the Lifted Multicut Problem

Lempitsky introduced fusion moves for unconstrained quadratic programming in [96]. In chapter 2 we define a fusion move algorithm for the minimum cost multicut problem. Here, we generalize the idea from chapter 2 to the minimum cost lifted multicut



(a) Serial fusion moves

(b) Parallel fusion moves

Figure 3.2: In a fusion move algorithm, proposal generation (PG) and fusion moves (FM) can be combined in different ways. We implement and study serial fusion moves (a) and parallel fusion moves (b).

problem. The fusion moves are defined in this section. Proposal generators are defined in the next section.

Given any feasible solutions x^1 and x^2 of the minimum cost lifted multicut problem (3.3)–(3.6), a constrained minimum cost lifted multicut problem in the variables $x \in \{0,1\}^{E'}$ is defined by (3.3)–(3.6) and the additional constraints

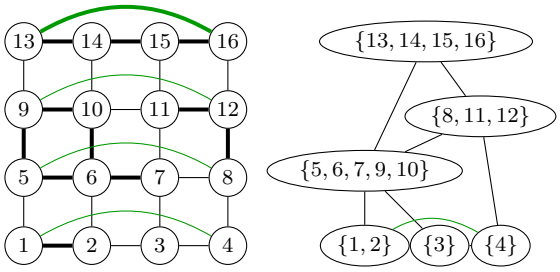
$$\forall e \in E : \quad x_e \leq x^1_e + x^2_e \ . \tag{3.7}$$

That is, all edges which are labeled 0 (join) in the feasible solution x^1 *and* the feasible solution x^2 are constrained to be labeled 0 in the problem (3.3)–(3.7). By construction, x^1 and x^2 are feasible solutions of the constrained problem (3.3)–(3.7).

Next, we reduce the *constrained* minimum cost lifted multicut problem (3.3)–(3.7) to an *unconstrained* minimum cost lifted multicut problem w.r.t. a smaller graph (Lemma 1). The latter problem can be solved by existing algorithms. In practice, we solve it approximatively by means of the Kernighan-Lin-type algorithm published by Keuper et al. [78]. The construction of the smaller graph is depicted in Fig. 3.3 and is described below.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph obtained from the graph G by contracting the edges $\{e \in E \,|\, x^1_e = 0 \wedge x^2_e = 0\}$[1]. Moreover, let $\mathcal{E}' \subseteq \binom{\mathcal{V}}{2}$ such that $V'W' \in \mathcal{E}'$ iff there exist $v \in V'$ and $w \in W'$ such that $vw \in E'$. Finally, let $C : \mathcal{E}' \to \mathbb{R}$ such that, for every

[1] I.e., $\mathcal{V}$ is a decomposition of G with every $V' \in \mathcal{V}$ a maximal subset $V' \subseteq V$ of nodes of G connected



(a) Graph G and E' (b) Graph $\mathcal{G}$ and $\mathcal{E}'$

Figure 3.3: To perform a fusion move, we solve a minimum cost lifted multicut problem with some edge labels fixed to 0 (join). In (a) such edges are depicted by bold lines. To solve this constrained problem, we reduce it to an unconstrained minimum cost lifted problem w.r.t. a contracted graph, depicted for this example in (b).

$V'W' \in \mathcal{E}'$:

$$C_{V'W'} = \sum_{\{vw \in E' \,|\, v \in V' \wedge w \in W'\}} c_{vw} \tag{3.8}$$

Lemma 1. *For every feasible solution $X : \mathcal{E}' \to \{0,1\}$ of the instance of the minimum cost lifted multicut problem w.r.t. $\mathcal{G}$, $\mathcal{G}' := (\mathcal{V}, \mathcal{E}')$ and C, the $x : E' \to \{0,1\}$ such that*

$$\forall vw \in E' : \quad x_e = \begin{cases} X_{V'W'} & \text{if } \exists V'W' \in \mathcal{E}' : v \in V' \wedge w \in W' \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

is well-defined and a feasible solution of the constrained minimum cost lifted multicut problem (3.3)–(3.7). Moreover,

$$\sum_{vw \in E'} c_{vw} x_{vw} = \sum_{V'W' \in \mathcal{E}'} C_{V'W'} X_{V'W'} \ . \tag{3.10}$$

Proof. If there exist $V'W' \in \mathcal{E}'$ such that $v \in V'$ and $w \in W'$, then V' and W' are unique (because $\mathcal{V}$ is a partition of V). Thus, x is well-defined.
The feasible solution X defines a decomposition of $\mathcal{G}$ (because $\mathcal{M} := \{V'W' \in \mathcal{E} \,|\, X_{V'W'} = 1\}$ is a multicut of $\mathcal{G}$). Every decomposition of $\mathcal{G}$ induces a decomposition of G (as the node set $\mathcal{V}$ of $\mathcal{G}$ is itself a decomposition of G). The multicut

by edges $e \in E$ for which $x^1_e = 0$ and $x^2_e = 0$. In addition, for every $V'W' \in \binom{\mathcal{V}}{2}$, we have $V'W' \in \mathcal{E}$ iff there exist $v \in V'$ and $w \in W'$ such that $vw \in E$.

$M := \{vw \in E \,|\, x_{vw} = 1\}$ of this decomposition of G is defined by the multicut $\mathcal{M}$ of $\mathcal{G}$ by (3.9) (by definition of $\mathcal{G}$). Thus, x satisfies (3.4).
Moreover, for every $vw \in E' \setminus E$, we have $x_{vw} = 0$ iff v is connected to w by a path P in G with $x_P = 0$ (by (3.9) and definition of $\mathcal{G}$ and $\mathcal{E}'$). Thus, x satisfies (3.5) and (3.6). Finally, (3.10) holds by (3.8) and (3.9). □ □

3.4.3 Proposal Generation for the Lifted Multicut Problem

As pointed out in [96], a proposal generator is designed with four objectives in mind. Firstly, proposed feasible solutions should be *diverse*. Otherwise, the fusion move algorithms can get trapped in local minima. Secondly, some proposed feasible solutions should be *good*. Otherwise, the fusion move algorithms cannot get close to the optimum. In the context of the minimum cost lifted multicut problem, a feasible solution is good if the recall of edges that are cut in an optimal solution is close to 1. Thirdly, the proposed feasible solutions should be *sparse*. In the context of the minimum cost lifted multicut problem, a feasible solutions is sparse if the precision of edges that are cut in an optimal solution is close to 1. Fourthly, the proposed feasible solutions should be *cheap*, i.e., proposals should be computable efficiently and in parallel. We study three proposal generators that emphasize different design objectives.
Randomly Perturbed Proposals. In order to obtain a proposal of high quality efficiently, we apply greedy additive edge contraction (GAEC) [78]. The key idea of this algorithm is to greedily contract edges with maximum cost until this maximum cost is equal to or smaller than zero. In order to get diverse solutions, we follow [27] and add normally distributed noise of zero mean to edge costs. In order to control the sparsity of the proposal, we replace the stopping criterion of GAEC and continue until a maximum allowed number of components is reached.
Subgraph Proposals. In order to obtain an objective-aware proposal for a large problem instance, we solve the minimum cost lifted multicut problem for a small subgraph. Technically, the procedure works as follows: We choose a center node $v \in V$ and the subgraph induced by the set U of all nodes within a fixed path-length distance from v. For $E_0 := \{vw \in E \,|\, v \notin U \wedge w \notin U\}$ and $E_1 := \{vw \in E \,|\, v \in U \wedge w \notin U\}$, we solve the instance of the minimum cost lifted multicut problem w.r.t. the graph G and the cost function c, with the additional constraints

$$\forall e \in E_0 : \quad x_e = 0 \tag{3.11}$$
$$\forall e \in E_1 : \quad x_e = 1 \ . \tag{3.12}$$

Watershed Proposals. In order to obtain diverse proposals cheaply, we follow [27] in using the weighted watershed algorithm [109] with random seeds. From the set

$\{vw \in E' \setminus E \,|\, c_{vw} < 0\}$ of lifted edges with negative cost, we draw a fixed number without replacement and assign different seeds to v and w. Thus, a random subset of lifted edges with negative cost is cut.

3.5 Experiments

We now describe experiments in which we compare the fusion move algorithm for the minimum cost lifted multicut problem with the Kernighan/Lin-type algorithm (KLj) and Greedy Additive Edge Contraction (GAEC) of [78] for the same problem.
In the tables below, FM-R, FM-SG and FM-WS stand for the fusion move algorithm with the randomized, subgraph and watershed proposal generators, respectively. Individual fusion problems, i.e., those problems denoted by boxes labeled "FM" in Fig. 3.2, are solved by KLj initialized with the output of GAEC.
In each experiment, the outer loop of fusion is terminated when no improvement is achieved for 5 consecutive iterations. Each experiment is conducted with 1, 2, 4 and 8 threads, respectively, to examine concurrency. All experiments are conducted on an Intel Core i7-4700MQ CPU operating at 2.40GHz $\times$ 8, and equipped with 32 GB of RAM.

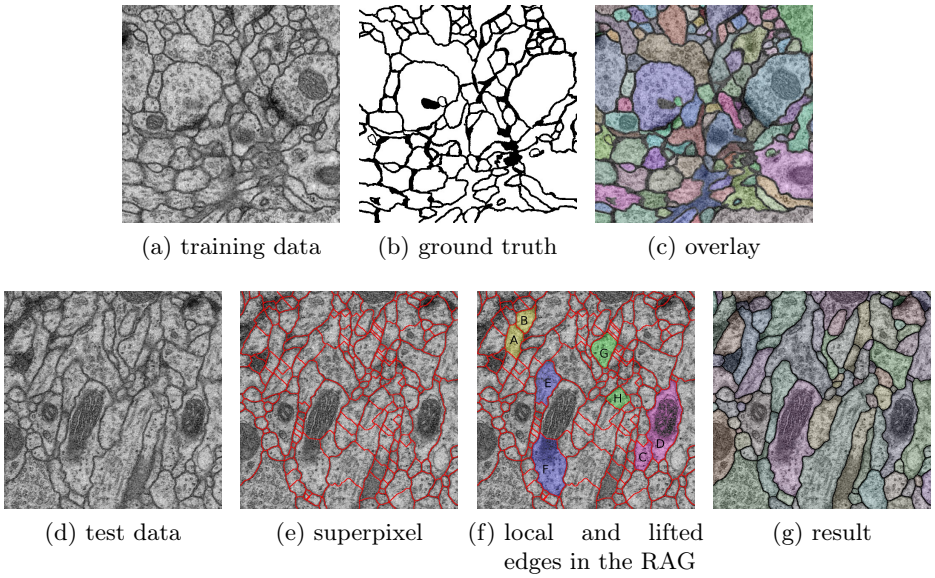
3.5.1 ISBI 2012 Challenge

The ISBI 2012 Challenge [21, 37] offers a set of segmentation tasks where images of the Drosophila larva ventral nerve cord acquired by a serial section transmission electron microscope are to be decomposed into distinct neurons, as depicted in Fig. 3.4c. The data set contains of 30 training images and 30 test images. Human annotations (Fig. 3.4b) are provided for each training image.
We propose a processing pipeline. Describing this pipeline in every technical detail is beyond the scope of this work. For the sake of reproducibility, the source code is available [2] . Overall, the pipeline consists of the following steps:

1. Start from the region adjacency graph (RAG) of an over-segmentation generated by seeded region growing [33], as shown in Fig. 3.4e.

2. Add lifted edges F for all pairs of superpixels within a path-length distance of $r_{nl} = 4$. The difference between lifted and non-lifted edges can be seen in Fig. 3.4f.

[2] https://github.com/DerThorsten/lifted_fusion_moves_eccv_2016



(a) training data (b) ground truth (c) overlay

(d) test data (e) superpixel (f) local and lifted edges in the RAG (g) result

Figure 3.4: The ISBI 2012 Challenge [21, 37] offers a set of segmentation tasks where neurons are to be delineated correctly in two-dimensional electron microscopy images, cf. (a)–(c). We start from the region adjacency graph of a superpixel segmentation (e) and train two classifiers to estimate the probability of adjacent and, respectively, non-adjacent superpixel pairs to belong to the same neuron. I.e., for edges like A-B and C-D in (f) or lifted edges E-F and G-H in (f). Solving, by fusion moves, a minimum cost multicut problem with costs defined in (3.13), our results on independent test images (with undisclosed ground truth) achieve the highest accuracy known at the time of writing. See (g) and Tab. 3.1.

3. Train two random forest classifiers: A first classifier RF_l learns to predict if a pair of adjacent superpixels should be in the same neuron or not. A second classifier RF_{nl} predicts the same for non-adjacent pairs of superpixels.

4. Solve an instance of the minimum cost lifted multicut problem (3.3)–(3.6) with superpixels as nodes, non-lifted and lifted edges and costs defined w.r.t. the probabilities estimated by RF_l and RF_{nl} as

$$c_{vw} := \log \frac{p(x_{vw} = 0)}{p(x_{vw} = 1)} \quad . \tag{3.13}$$

To train RF_l we use features on local image statistics as described in [9, 14]. To train RF_{nl}, we compute the following features for of lifted edges:

1. Features based on hierarchical clustering inspired by [18, 145]: We apply UCM to generate the complete dendrogram and use the thus defined ultrametric distance between pairs of nodes (height in the dendrogram at the moment when the nodes are merged) as a feature for the corresponding lifted edge, if it exists.

2. Features inspired by maximum intervening contours [51, 98, 106]: We compute simple statistics of local image features (e.g. average gradient) along multiple straight lines between two superpixels.

3. Shortest path based features: Using various local features (raw intensities, gradients etc.), we compute multiple shortest paths between non-adjacent superpixels and measure statistics along these paths.

4. Candidate segmentation features: We compute multiple candidate segmentations using the minimum multicut objective (with varying parameter and without lifted edges), and each edge is assigned the proportion of the segmentation where it got cut.

For all features above we use the raw data itself as input, but also a pixel wise probability map learned with a CNN [101].
A quantitative evaluation is shown in Tab. 3.1. It can be seen from this table that segmentations of the images defined by feasibles solutions of the minimum cost lifted multicut problem define a new state of the art on this highly competitive segmentation challenge. FM-R, FM-SG and KLj yield the same objective. Even with only a single thread, FM-R and FM-SG are slightly faster than KLj. With 8 threads, the proposed methods outperform KLj by a factor of 4.

3.5.2 Image Decomposition

Keuper et al. [78] pose the image decomposition problem [17] as a minimum cost lifted multicut problem. Instances of this problem are defined w.r.t. pixel grid graphs and lifted edges connecting each pixel to the (about 300) pixels within a path-length distance of 10. Costs of non-lifted edges are derived from structured edge detection according to [47]. Costs of lifted edges are defined by probabilistic geodesic lifting [78]. These large instances of the minimum cost multicut problem pose a challenge to optimization algorithms and are thus suitable for benchmarking. Here, we compare the

Table 3.1: At the time of publishing [30] feasible solutions of the minimum cost lifted multicut problem define the state of the art on the ISBI 2012 Challenge [21, 37]. The performance measures VRand and VInfo are defined in [21]. A value of 1 indicates a perfect segmentation; values close to zero indicate poor segmentations. Using 8 threads, the proposed methods (FM-R, FM-SG) outperform KLj by a factor of 4. Leader board: http://brainiac2.mit.edu/isbi_challenge/leaders-board-new

Algorithm	Objective	Time to convergence [s] (1/2/4/8 threads)	VRand (higher is better)	VInfo (higher is better)
FM-SG	-13560.18	0.62 / 0.37 / 0.28 / 0.21	0.9804	0.9884
FM-R	-13560.18	0.77 / 0.42 / 0.32 / 0.28	0.9804	0.9884
KLj	-13560.18	0.89	0.9803	0.9884
Leader Board 2	-	-	0.9796	0.9870
Leader Board 3	-	-	0.9768	0.9886
Humans	-	-	0.9978	0.9990

fusion move algorithm with watershed proposal generator (FM-WS) with GAEC and KLj initialized with the output of GAEC.
Results are shown in Tab. 3.2. It can be seen from these results that FM-WS outperforms the current state of the art (KLj) in terms of runtime and objective value. Moreover, FM-WS is about twice as fast with one thread and about six times as fast with 8 threads. The gap between FM-WS and KLj is comparatively larger than that between of KLj and GAEC. Therefore, we consider FM-WS a significant improvement over the state of the art.

3.5.3 Averaging Multiple Segmentations

Fusing multiple segmentations into a single one is not only important as an image analysis sub-task, but can also be used to combine multiple *manually* derived ground truth solutions into a "master" ground truth image. Multiple user-provided solutions are, for example, available for the BSDS-500 data set [17].
Recently, [4] proposed to solve this problem with an EM-algorithm based on the multicut objective. Their algorithm is defined on a complete graph derived from the region adjacency graph of an initial superpixel segmentation. In contrast to our approach, they use the plain multicut objective where all edges of the complete graph are considered local, and there are no lifted edges. Before constructiong the complete graph, every proposed segmentation x^l from the given set L is projected on the superpixel RAG, and all edges which are not cut in any proposal are contracted, resulting in a dramatic reduction of the graph's size. The edge costs of the remaining edges measure how often this edge is cut in L. Furthermore, a weight p_l measuring the estimated reliability of each segmentation relative to the others is assigned to each member of L. The multicut objective is then optimized with p_l kept fixed, and the p_l are updated according to the proportion of edges in x^l that agree with the current master segmentation. This is repeated in an EM manner until convergence.
We modify this aproach as follows: We optimize directly on the *pixel-level*, i.e. on a 4-connected grid graph instead of a superpixel RAG, to eleminate superpixel computation as an additional source of error. Moreover, we replace the multicut objective with a *lifted multicut* objective containing only a sparse set of lifted edges up to a graph distance of 5. We do not contract any edges in pre-processing. Edge costs are defined as in [4] by

$$c_{vw} := \log \sum_{l \in |L|} (1 - x^l_{vw}) p_l - \log \sum_{l \in |L|} x^l_{vw} p_l \tag{3.14}$$

As in [4], we use an EM-type algorithm to update p_l according to the number of edges in x^l that agree with the current master segmentation $\hat{x}$:

$$p_l = \frac{1}{|EF|} \sum_{x_{vw} \in EV} 1 - |x^l_{vw} - \hat{x}_{vw}| \tag{3.15}$$

In every iteration of EM, we solve an instance of the minimum cost lifted multicut problem using FM-SG and, for comparison, KLj. Both are initialized with the output of GAEC. We only use FM-SG results to update the p_l since they were always better than the KL results. In addition to the proposals generated by the subgraph method, all x^l are included into the proposal set, leading to a significant speed-up.
Results are shown in Tab. 3.3 and Fig. 3.5. It can be seen from Tab. 3.3 that FM-SG outperforms KLj in terms of objective value and run-time. Even with a single thread, FM-SG is twice as fast as KL. Using 8 threads, the FM-SG is six times as fast.

3.6 Conclusion

We have defined a fast, scalable and easy to implement fusion move algorithm for the minimum cost lifted multicut problem. Experiments with diverse instances of the problem have shown that this algorithm typically outperforms existing methods in terms of objective value and run-time. We conjecture that efficient algorithms such as the one proposed in this chapter facilitate a variety of applications of the minimum cost lifted multicut problem in computer vision of which the averaging of multiple segmentations is just one example.

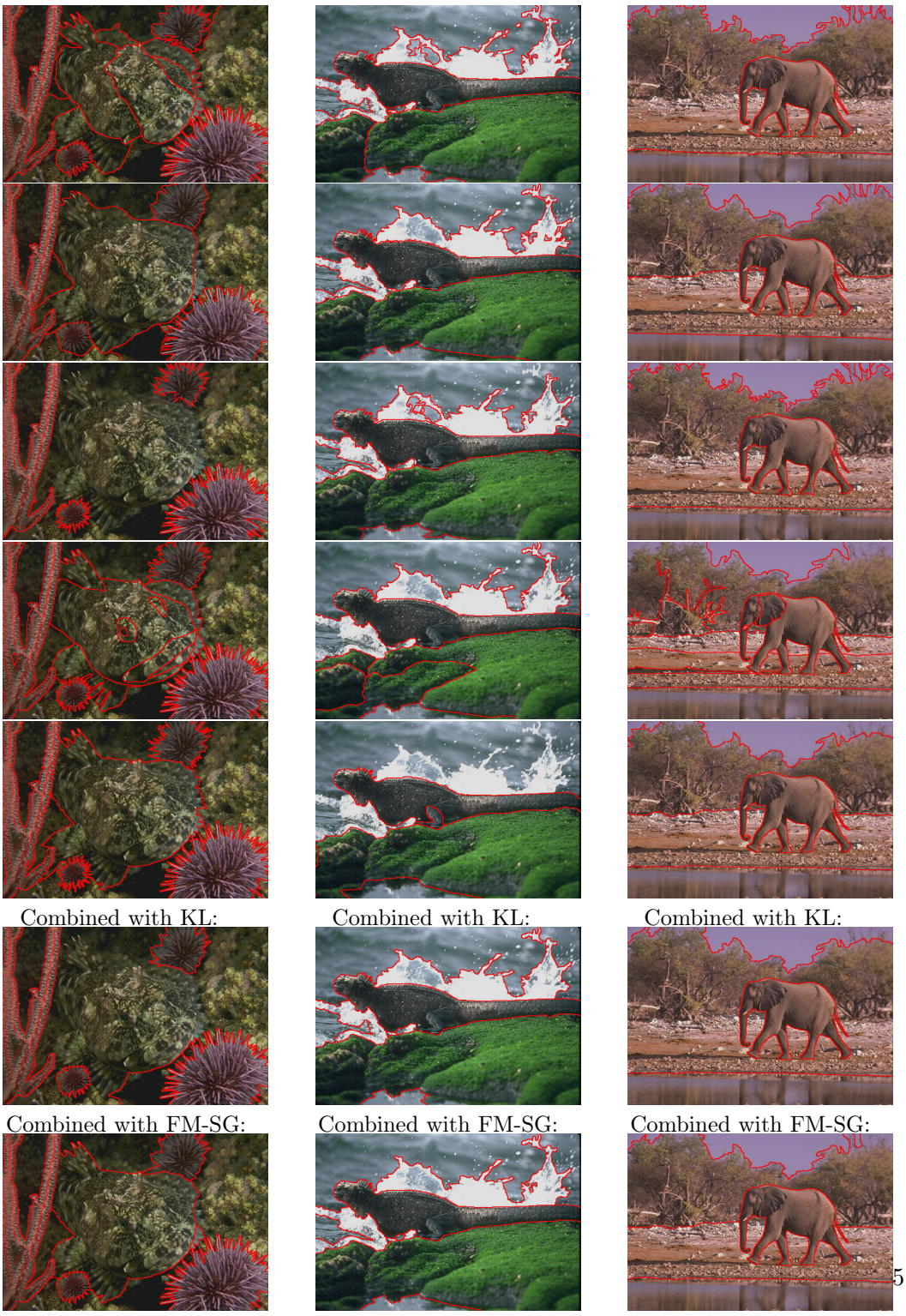


Figure 3.5: To average multiple segmentations, we solve instances of a minimum cost lifted multicut problem as in (3.14)–(3.15). Above, Rows 1–5 show different man-made segmentations of images from the BSDS-500 benchmark [17]. Row 6 shows the combination of these segmentations by the solution using KL, row 7 shows the result with the proposed algorithm (FM-SG).

Table 3.2: The proposed algorithm FM-WS outperforms KLj and GAEC on the large and hard instances of the minimum cost lifted multicut problem of [78].

Algorithm	Objective	Time to convergence [s] (1,2,4,8 threads)
FM-WS	−62748200	61 / 32 / 25 / 22
GAEC	−62744700	10 / n.a.
KLj	−62745500	121 / n.a.

Table 3.3: To average multiple segmentations, we solve instances of a minimum cost lifted multicut problem as part of the EM algorithm proposed in [4]. FM-SG is an efficient algorithm to solve these instances.

Algorithm	Objective	Time to convergence [s] (1/2/4/8 Threads)
FM-SG	**-2.29e+07**	14.8 / 8.83/ 6.33/ 5.21
GAEC	-1.53e+07	13.8
GAEC + KLj	-2.27e+07	29.3

4 Multicut brings automated neurite segmentation closer to human performance

This chapter is based on [26] where many authors have contributed to. In particular Nasim Rahaman contributed the neural network in section 4.3, Timo Prange contributed the superpixels in section 4.4. Constantin Pape conducted the experiments in section 4.5 and section 4.7 and contributed to the software library (section 5.2).

4.1 Introduction

The connectomics community is acquiring volumetric electron microscopy (EM) images of the brain at an unprecedented rate with the aim of mapping out and understanding in detail the physical correlates of information processing in animals. Reliable automatic segmentation is urgently needed for upcoming whole-brain data sets (>100 terabytes (TB) per volume). Manual analysis, despite impressive progress in collaborative annotation [79], will not scale to this massive task. We present an algorithm and software package to segment such data sets with low error rates. The software is made available open source at online repositories, and we also provide precompiled binaries (see section 5.2).

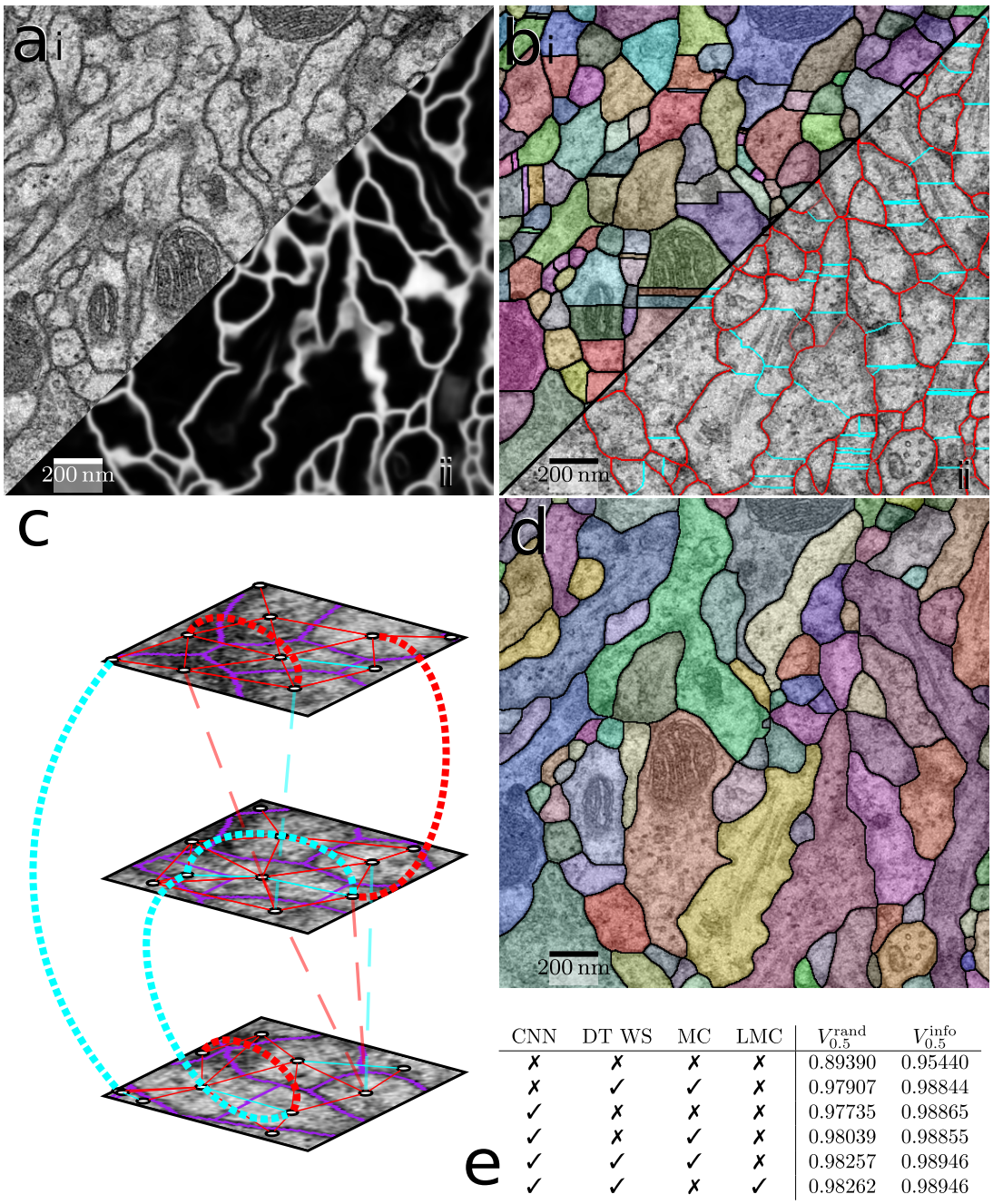
At the ISBI 2012 conference, a challenge for segmenting anisotropic 3D EM images was launched [19]. In this "blind" challenge, which remains open to new submissions, participants can submit tentative segmentations of the test data set. The organizers then measure the accuracy of the submitted segmentation in terms of Rand error. The latter is a statistic summarizing—for each and every pair of points—how often these points are correctly assigned to the same segment, or to different segments, as dictated by ground truth. The organizers publish the Rand error of a submission without giving away the ground truth segmentation itself, thus ensuring fair comparison and minimal bias [21].

At the time of writing, our algorithm produces the best known result on the ISBI 2012 blind challenge, halving the error of the 2012 winner. Our pipeline comprises three

major steps (see section 4.3). First, we apply a cascaded random forest (which needs less training data) or a convolutional neural network (which gives even better accuracy) to predict membrane probabilities. In the neural network, we found skip layers, elastic data augmentation during both training and prediction, and inception-like modules to be critical for performance (Section 4.3).
Second, we aggregate pixels into *superpixels* to coarse grain the problem and to extract higher order region information in a data- dependent fashion. Superpixels should be few (and thus large) to reduce the problem size for the final processing stage; but superpixel boundaries must also form a strict superset of true neurite boundaries. Distance transform watershed superpixels (see section 4.4) offered the best trade-off in our experiments, yielding large superpixels that are robust against minor gaps in the boundary probability maps. Finally, we merge superpixels to tentative neurites while respecting consistency constraints across distances that are larger than a neural network's field of view. Specifically, we solve the Lifted [78] Multicut [9] problem, which introduces attractive or repulsive potentials between (nonadjacent) superpixels, and we find the graph partitioning that optimally balances these cues. We always reason in 3D, even for
anisotropic data (Figure 4.1 and sections 4.5 and 4.6). This NP-hard partitioning problem is solved approximately using the the methods from chapters 2 and 3. Each of these choices is the result of extensive experimentation, and the lesion study summarized in Figure 1 and in section 4.7. Section 4.7.2 shows how performance degrades when deviating from these choices. The same pipeline works well on the anisotropic murine neocortex "SNEMI3D" data and the isotropic Drosophila medulla "Neuroproof" data (section 4.7).
This work substantially narrows the accuracy gap between humans and computers for neurite segmentation (section 4.7. Section 4.7.2). We expect this gap to close within the decade, at least for high-quality data, allowing neuroscientists to make the most of the impressive data sets that are currently being acquired.

4.2 Related Work

Electron microscopy is producing images at a rate that surpasses the human capacity of neurite tracing [6, 32, 34, 36, 73, 95, 134], even when allowing for massively parallel annotation [35, 37, 58, 75, 79, 100, 119, 143]. In response, the computer vision community is developing segmentation algorithms that decrease the manual proof-reading effort, with the ultimate aim of generating accurate segmentations fully automatically. Open segmentation challenges [19, 20] have been instrumental in this improvement.



CNN	DT WS	MC	LMC	$V_{0.5}^{\text{rand}}$	$V_{0.5}^{\text{info}}$
✗	✗	✗	✗	0.89390	0.95440
✗	✓	✓	✗	0.97907	0.98844
✓	✗	✗	✗	0.97735	0.98865
✓	✗	✓	✗	0.98039	0.98855
✓	✓	✓	✗	0.98257	0.98946
✓	✓	✗	✓	0.98262	0.98946

Figure 4.1: The automated neurite segmentation pipeline and the influence of its components on performance. **(a, i)** Example of ISBI 2012 data. Membrane probability estimates**(a, ii)** are used to find superpixels **(b,i)**. Pairs of these regions are associated with attractive (cyan) or repulsive (red) potentials that are informed by local appearance **(b, ii)**. A region adjacency graph is constructed in 3D, even for strongly anisotropic data **(c)**. We consider next neighbor interactions (straight lines and straight dashed lines) and longer range interactions (curved lines) for the Lifted Multicut. Solving the (Lifted) Multicut graph partitioning problem yields tentative neurites **(d)**. The table **(e)** shows the performance reached on the ISBI 2012 challenge using; from top to bottom: a cascaded Random Forest; the same with distance transform watershed superpixels (DT WS) and multicut (MC); our neural network; the same with standard watershed superpixels and MC; the same with DT WS and MC; and finally the proposed pipeline. Accuracies are measured by scores derived from the Rand index (RI) and the variation of information (VI), and higher scores are better.

Since many neurites are locally similar, most segmentation procedures rely on boundary information (alone). Such cell membrane probabilities can be estimated using neural networks (e.g. [42, 61, 137]) or other classifiers such as Random Forest [13], possibly augmented with a conditional random field [74].
Ideally, the connected components of thresholded boundary predictions would already correspond to neurites. Given that all boundary predictions available to date are imperfect, it is useful to first conservatively group pixels into larger clusters, so-called superpixels, that afford the extraction of more expressive features. These superpixels can then be grouped into tentative neurites in a second step, using a variety of techniques ([9, 13, 52, 63, 103, 111, 139, 140] and others).

4.3 Neurite Boundary Probability Prediction

Multiple neural network architectures have been used to predict pixel-wise membrane probability or pixel affinity given stacks of raw EM data (e.g. [42, 50, 61, 117, 129, 138]). Recent networks are both wider and deeper than their predecessors, and often combine high-resolution "geometric" with low-resolution but deep "semantic" pathways [104, 117].

4.3.1 Architecture of our Network

Figure 4.2 illustrates the network architecture, which we call ICv1. The network builds on important previous work, notably Inception Modules [132], uses strided convolutions, max-pool operations and deconvolutions in a fully convolutional setting [104], and combines high- and low-resolution pathways [117] to aggregate geometric and semantic features.
The key idea is to have the network, through a succession of stages, trade spatial resolution for a rich semantic representation; and to then restore the former, in a further succession of layers, to obtain a high-resolution but single channel prediction: for all pixels in the image, the probability of it showing a neurite boundary. A single skip connection helps conserve high-resolution information and propagate gradients back to the early layers.
Throughout the network, inception-like structures feed the output from the previous layer into independent pathways or "towers" whose outputs are concatenated downstream. Compared to a standard architecture with the same number of layers and neurons, this structure induces topological sparsity in the network architecture, thereby (a) reducing computational cost and (b) clustering features in layer activations.

Deviating from recent trends [124], we use relatively large convolution kernels (up to 9×9 pixels) to enhance the field of view without introducing significant memory overhead. For instance, a 9×9 field of view could also be obtained by four subsequent 3×3 convolutions that, assuming no intermediate dimensionality reduction, would consume more memory to store layer activations for the backward pass. To mitigate the internal covariate shift [60] to some extent, we use Exponential Linear Units [43] instead of Rectified Linear Units [91]. An ensemble of three ICv1's together with test time data augmentation (see below) is currently the best performing neural model on the ISBI 2012 dataset.

4.3.2 Data Augmentation

Our network makes extensive use of train and test time data augmentation. Test time data augmentation with linear transformations has been proposed by [50] and [117] as a computationally cheap way to build an ensemble at test time. In addition to random linear transformations, we also include nonlinear transformations in our ensemble to further boost the results. In this process, the input image is elastically transformed (by displacing all pixels according to the realization of a smooth bivariate 2D Gaussian random field) and fed into an ensemble. Then, its outputs are transformed back with the inverse elastic transformation. The process is repeated using different random transformations, and the results are averaged over. We profit from the fact that the resulting synthetic images are clearly distinct from the originals, but in appearance are still similar to real tissue images.
One single network has approximately 35 million parameters, which can be learned from the limited training data only thanks to extensive regularization and data augmentation.

4.3.3 Experimental Setup

The network was trained as a regression model with pixel-wise binary cross-entropy loss. Binary boundary / no-boundary target labels were smoothed [133] by computing the pixelwise negative exponential of the Euclidean distance transform, while the raw image batches were normalized to zero mean and unit variance. The training dataset (for the ISBI 2012 challenge: 28 out of the 30 slices in the training volume) was iterated over for 1000 epochs, and the set of parameters yielding the smallest pixel-wise mean squared error (for ISBI 2012: on the remaining 2 out of the 30 slices) was chosen for inference. The networks were optimized with ADAM [82] with the following optimization hyperparameters: step size $\alpha = 0.0002$, exponential decay rates

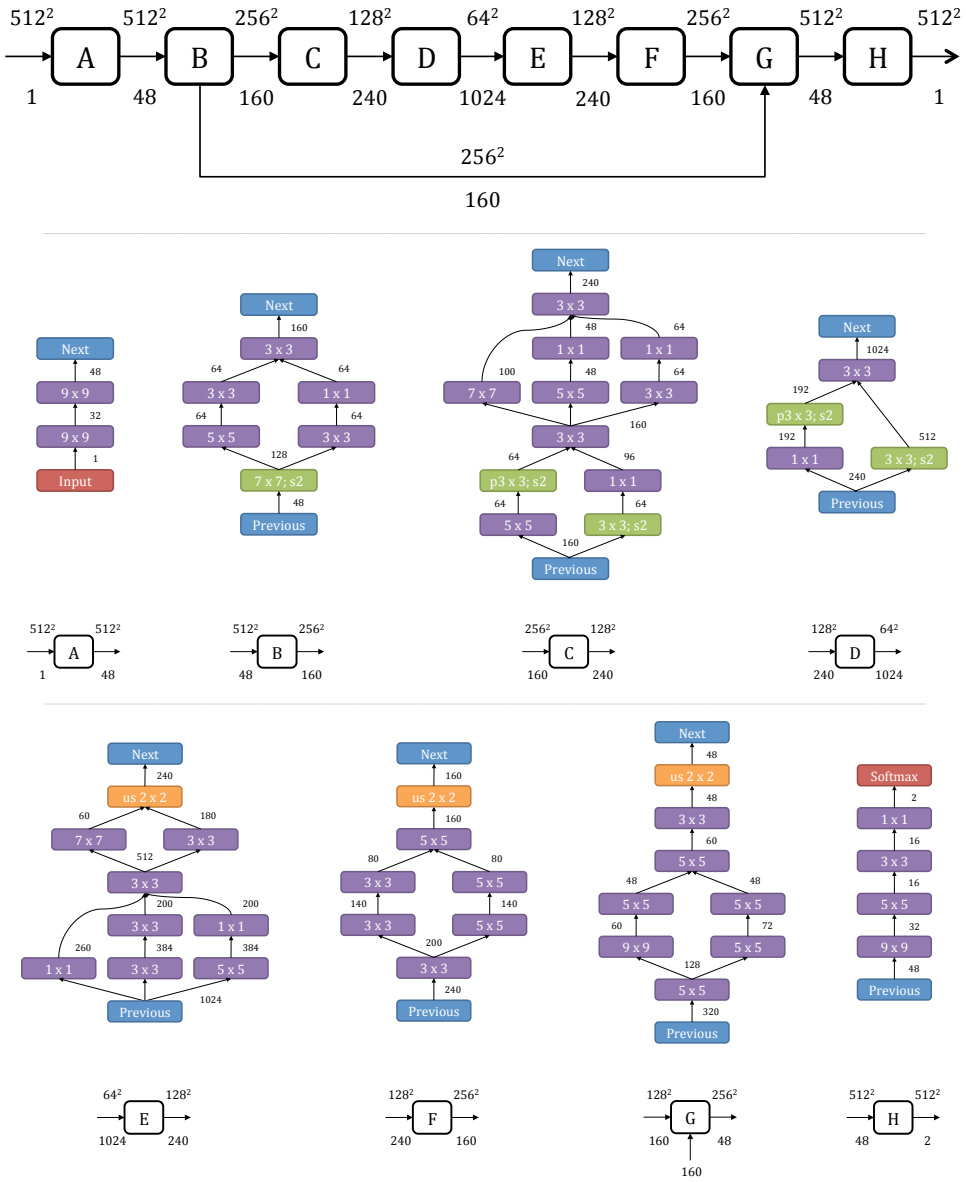


Figure 4.2: ICv1 neural network architecture. Purple boxes represent convolutional layers with kernel size $M \times N$, ELU non-linearity, and "same" border handling (i.e. the input and output images have identical shapes). Green boxes annotated with $pM \times N$; sS denote max-pooling layers over $M \times N$ patches with stride S; while those annotated with $M \times N$; sS represent convolutional layers with stride S. Both layers reduce the image width and height by a factor S. Orange boxes annotated us $M \times N$ show upscaling layers, where pixels in the input feature maps are repeated M times vertically and N times horizontally. The numbers next to arrows specify the number of feature maps flowing from one layer to the next. Boxes with M outgoing arrows imply that the output of the layer is replicated M times, while boxes with N incoming arrows imply that the layer receives N inputs, which are concatenated depth-wise (i.e. along the feature axis) before being processed.

for moment estimates $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a fuzz factor $\epsilon = 10^{-8}$ for numerical stability. A L2 weight decay term with coefficient $\lambda = 0.0005$ was added to the objective and dropout [127] ($p = 0.5$) was used. For test-time data augmentation, 20 elastic transformations (including also random rotations and flips) were used. The network was implemented using Theano [31] and CuDNNv4 [39]. Training a network for 1000 epochs with a batch size of 4 required 20 hours on a GeForce Titan X GPU. We built an ensemble of three networks by training in parallel and independently on three GPUs.

4.3.4 Baseline: Boundary Prediction with a Cascaded Random Forest

At the time of writing, the above neural network architecture achieves higher accuracy than all previous work on the ISBI 2012 challenge (except for the full pipeline introduced here). However, the many free parameters in neural networks generally call for copious amounts of labeled training data. Seeking to alleviate this requirement, we also performed membrane probability prediction by a cascaded Random Forest trained interactively with sparse labels.
Following ideas from Autocontext [64, 136], we perform two rounds of prediction using Random Forest. More specifically, the first Random Forest is trained interactively with ilastik [125], using multiple semantic classes: membrane, cytoplasm, mitochondria, mitochondrial membrane, intracellular structure, synaptic sites and "everything else". The predictions for all classes are concatenated to the raw data as new channels. In a second round, the linear and nonlinear features from the ilastik filter bank are computed both on the original raw data and the predictions from the first round. At this stage, labels are again provided interactively in ilastik, this time annotating only "boundary" and "background" classes. For the ISBI challenge dataset, such interactive labeling took about 15 hours in total for both stages.

4.4 Superpixel Generation

Even present-day deep neural networks have difficulty in modeling very long-range interactions. For instance, if the receptive field of a neural network is 512^2 pixels, it cannot reconcile cues that may be a thousand or more pixels apart. Given that neurites are tube- or tree-like structures that extend across thousands of pixels at standard electron microscopy resolutions, this is a practically relevant limitation.
One way forward is to first aggregate those pixels that, with a very high confidence, can be assumed to belong to the same neurite into a "superpixel". A second stage in the pipeline can then represent the original image, and reason in terms of, these

superpixels. Besides spanning larger distances and reducing the number of objects involved in the second-stage reasoning, superpixels have the advantage that they allow the extraction of a rich set of features that are not defined at the pixel level. For example, at the level of superpixels it is possible to answer questions like "how many mitochondria are in this superpixel" or "what is the diameter of this section of neurite" that would be ill-posed at the level of individual pixels.
In the downstream processing of superpixels, it is typically easy to merge sets of superpixels, but difficult to split these. As a consequence, a good superpixel should

- encompass as many pixels as possible, but
- not extend beyond a single neurite.

4.4.1 Standard superpixels

In standard computer vision tasks, algorithms such as SLIC [2] that group pixels of similar appearance are popular. Such algorithms are not useful for the present task, where neurite diameters have a large range and boundaries are the strongest cues that delineate a neurite from its neighbor. In the present setting, the watershed (e.g. [33]) works well when applied to a smoothed pixelwise boundary probability map [13].

4.4.2 Distance transform watershed superpixels

Standard electron microscopy sample preparation and staining protocols often lead to seeming small perforations of neurite boundaries, or even to thin slashes in such boundaries. These artifacts are a nuisance for the standard watershed, as it may result in undesirable superpixels that span two adjacent neurites.
As a consequence, we here propose to first threshold a pixelwise boundary probability map as obtained from a neural network or cascaded Random Forest; to then filter out tiny connected boundary components; and to finally compute the smoothed signed Euclidean distance transform on the remaining boundaries. See Fig. 4.3 for an illustration.
This is related to prior work on the stochastic watershed [15] and has the effect of closing slashes that are narrower than the diameter of either adjacent neurite in the proximity of the slash.
Finally, following [89] we use ilastik [125] pixel classification to train a myelin detector. Large myelin connected components are then turned into additional superpixels, see Fig. 4.4.

Figure 4.3: Generating distance transform watershed superpixels. From left to right: raw data, boundary probability map, thresholding thereof, distance transform on the latter, and watershed superpixels seeded by distance maxima. Note how the resulting superpixels are confined to single neurites, in spite of ambiguous boundary evidence (arrow).

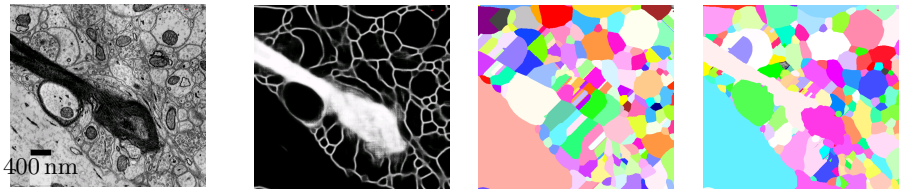


Figure 4.4: Generating superpixels in the presence of heavy myelination. From left to right: Part of SNEMI3D test data, boundary probability map courtesy of [42], distance transform watershed before and after myelin correction.

With suitable choice of parameters for both the standard and distance transform watershed, the former generates around $12k$ and the latter less than half that number of superpixels. This both reduces the number of objects to be reasoned with downstream, and allows aggregating features over larger areas.

4.5 Multicut Segmentation

The electron microsocopic images can now be represented at a higher level of abstraction, in terms of a region adjacency graph. Each node in this graph corresponds to one superpixel, and every edge represents the boundary between two adjacent superpixels (we will use the terms "edge" and "boundary" interchangeably). The segmentation problem can then be expressed as a graph partitioning problem. Here, *cutting* an edge means *preserving* the boundary between two incident superpixels. Vice versa, preserving an edge means that the two respective superpixels are merged into a single component. Note that only select edge labelings amount to a valid partitioning. For example, consider a triangular graph with nodes A, B, C. The labeling

$\{(AB) = 0, (BC) = 0, (CA) = 1\}$ is not a consistent partitioning: nodes A and C are assigned to the same cluster via their mutual connection to B; and at the same time, they should be in distinct clusters due to the cut edge between them: a contradiction. In keeping with literature, we have used the arbitrary convention that a cut edge has value 1 and a preserved edge has value 0.
Each edge is associated with a positive (attractive) or negative (repulsive) score. By solving the multicut or correlation clustering problem as described in chapter 2, we identify amongst all consistent edge labelings the one that best obeys these attractive and repulsive potentials.

4.5.1 Multicut for Anisotropic Data

The attractive or repulsive edge scores can be estimated based on the appearance of the boundary itself, and from the appearance of the incident superpixels. For isotropic data, we first generate 3D superpixels and then train a Random Forest to predict, based on these features, for each edge whether its score should be positive / attractive or negative / repulsive. For isotropic data, the fundamental procedure is described in [9].
For anisotropic data, some adjustments are important. To the best of our knowledge, this is the first time that the multicut is adapted to anisotropic connectomic data. For large degrees of anisotropy, no good 3D superpixels can be generated using the kind of procedure described in the previous section. As a consequence, we find 2D superpixels and connect these in a 3D region adjacency graph, that is, edges are also introduced between superpixels in adjacent slices.

4.5.2 Edge Features

Edge Features Based on the Appearance of the Corresponding Boundary These features are computed in three steps: First, filter bank outputs with Gaussian smoothing, Hessian of Gaussian eigenvalues, Laplacian of Gaussian, each at scales (using the vigra [87] convention) $\sigma = \{1.6, 4.2, 8.3\}$ are calculated for one or more inputs. These inputs are the raw data plus possibly probability maps for certain semantic classes in the data, such as specific organelles. In our experiments, we use the raw data and the very membrane probability maps that were used to find superpixels. For anisotropic data, these filter banks are calculated either strictly in 2D (for high anisotropy) or in 3D but with reduced filter size in the anisotropic direction. Next, the filter bank outputs are accumulated over the boundary of interest and summarized in terms of the following aggregate statistics: mean, sum, minimum, maximum, variance, skewness,

kurtosis and the $\{0.1, 0.25, 0.5, 0.75, 0.9\}$ quantiles.

Edge Features Based on the Appearance of Incident Superpixels For each superpixel, we compute: its size, the eigenvalues of its inertial tensor, as well as the histogram (64 bins), kurtosis, maximum, minimum, $\{0.1, 0.25, 0.5, 0.75, 0.9\}$ quantiles, skewness, sum and variance of the raw image. These numbers are then mapped to the edges by taking the min, max, sum and absolute difference of the incident superpixels' values. In addition, we compute the squared distance between the incident supervoxels' centers of mass. The latter are found both with uniform pixel weights, and with pixel weights given by the raw intensity.

Inter-slice Features for Anisotropic Data When constructing 2D superpixels, the inter- and intra-slice edges are different in nature: intra-slice edges correspond to 1D curves in image space, whereas inter-slice edges correspond to 2D surfaces in image space.
For such inter-slice edges we compute all of the above features, plus the following:

- size of the union (when projecting orthogonally onto the same slice) of the two superpixels of interest
- size of their intersection
- ratio of intersection and union

We also compute the ratio of the area of a 2D superpixel to its circumference, and map this information to an edge by taking the min, max and absolute difference between the values obtained for its incident superpixels.
In total, all of the above result in 625 features per superpixel edge. Empirically, we find that it is best to train a single Random Forest to predict scores for all types of edges. For intra-slice boundaries, the last set of features is set to zero. Finally, we normalize the predicted score for each edge by its length (for 1D edges) or area (for 2D edges).

4.6 Lifted Multicut for Anisotropic Data

The lifted multicut as described in chapter 3 is a generalization of the multicut objective . Its objective function can accommodate a cost or reward for *every* pair of nodes, not just neighboring ones; and its constraints make sure that all nodes ending up in the same cluster are also connected in the spatial domain, not merely by the non-local "lifted" edges. Overall, the lifted multicut is empirically computationally harder; but

it also represents a more expressive model (the standard multicut model is a strict subset of the lifted multicut model).
To obtain an attractive / repulsive score for each edge connecting two adjacent superpixels, we follow the procedure outlined in section 4.5. For the lifted edges, we use additional features as outlined next.

4.6.1 Lifted Edge Features

To predict if two non-neighboring nodes should be in the same segment we use the following features:

Multicut Connectedness The multicut procedure from section 4.5 is executed five times, where a bias is added to / subtracted from all edge scores, making them more attractive / repulsive. For each lifted edge and for each of the five biases, we record in a binary variable whether or not the two incident regions ended up in the same connected component.

Ultrametric distance We apply the ultrametric contour map transform [18] to generate a complete dendrogram. The ultrametric distance (the height in the dendrogram at which the two regions merge) is used as an additional feature for each lifted edge.

Region features Equivalent to those described in "Edge Features Based on the Appearance of Incident Superpixels" in section 4.5.2.

To account for the fact that we have a considerably larger number of lifted edges than local edges, we weight the scores for both by their number. For anisotropic data, inter-slice lifted edges that are d slices apart are further weighted by $1/(1+d)$. Finally, the lifted multicut objective is optimized approximately using using the fusion move algorithms descried in chapter 3.

4.7 Benchmark Experiments

We have evaluated the Multicut and Lifted Multicut on the blind ISBI 2012 challenge, as well as on the publicly available SNEMI3D and Neuroproof datasets, each containing separate training and test volumes of EM images. For evaluation, we report the measures $V_{0.5}^{\text{rand}}$ and $V_{0.5}^{\text{info}}$ as defined in [19]. These are the F1-scores derived from the structured segmentation accuracy measures "Rand Index" and "Variation of

Information" [19]. Both measures go beyond aggregating single pixel errors, and instead summarize statistics of point pairs, verifying if they are or are not in the same segment, as prescribed by ground truth. Note that the border thinning procedure from [19] is not needed for our (Lifted) Multicut segmentations because the resulting region labelings are already dense.

4.7.1 ISBI 2012

The ISBI 2012 challenge [19] is the most popular and competitive connectomics challenge to date. Here, we used the anisotropic pipeline to obtain the accuracies summarized in Fig. 1g in the main text.
The training data consists of images which have neurite boundaries and extracellular space marked. We used the neurite connected components as seeds for a watershed on the probability map to obtain ground truth with thin boundaries. These were then projected to the superpixel boundaries, according to the overlap of a superpixel with ground truth segments. Ground truth boundaries with ambiguous assignment to our superpixels are ignored at the training stage. An example for the resulting segmentation and labels can be found in Figure 4.5.

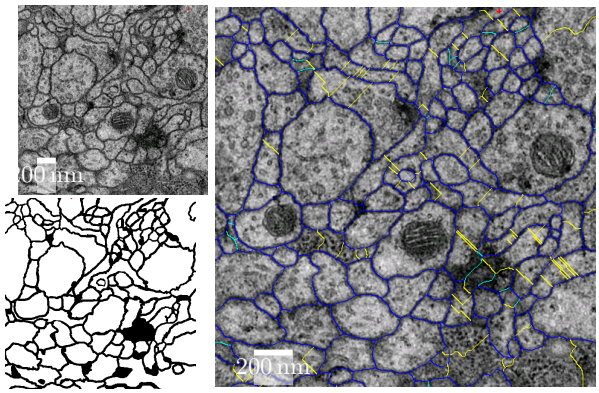


Figure 4.5: ISBI 2012 training data generation. Left: Raw data and ground truth. Right: Inferred superpixel boundary labels for training (blue = repulsive, yellow = attractive, cyan = unlabeled).

Given that ground truth is available only for intra-slice boundaries, we could not explicitly train the classifier for inter-slice edges. Even so, these were still used in the 3D Multicut. The Lifted Multicut results reported for this dataset were produced using all the features described in 4.6, and lifted edges were created between all superpixels

within a distance of four in the region adjacency graph.

4.7.2 SNEMI3D

The SNEMI3D dataset (http://brainiac2.mit.edu/SNEMI3D/) has been a blind challenge from 2013 to 2015, when a labeled superset of training and test block was released with [73].
There are a number of important differences in the nature of raw data and annotations with respect to ISBI 2012. First, the training ground truth is given in terms of a 3D segmentation (as opposed to a stack of 2D segmentations, as in ISBI 2012). Second, the data is better resolved along the z-axis. Third, segment boundaries in the ground truth are not always properly aligned with the actual membranes in the raw data. Fourth, individual neurites are separated by a "negative" class that covers membranes, but also intercellular space, myelin sheaths and a few erroneously omitted thin processes.
To exploit the higher z-resolution, we supply each slice along with its two adjacent slices as input to the network. To mitigate the mentioned inaccuracies in the training data, we weigh the positive pixels (intracellular space) uniformly; but downweight the negative pixels (covering everything else plus a few omitted processes) with increasing distance from the positive regions, reaching zero weight beyond a distance of seven pixels. Seeds for the distance transform watershed are found in 3D, and superpixels are grown in 2D on smoothed probability maps.
Training labels for intra- and inter-slice superpixel pairs are obtained with the mapping procedure described in Section 4.7.1, and all edge weights are normalized by edge size. Lifted edges were introduced between all pairs of superpixels with a maximum distance of three in the region adjacency graph.
The SNEMI3D website provides boundary probability maps graciously shared by the authors of [42], which we also feed into our postprocessing. Both the Ciresan and ICv1 networks give similar peformance on SNEMI3D when followed up by our superpixels and (lifted) multicut, see Table 4.1.

SNEMI3D accuracy ($V_{0.5}^{\text{rand}}$)	ICv1	Ciresan
Multicut	0.92698	0.92568
Lifted Multicut	0.93122	0.92892

Table 4.1: Results of Lifted and standard Multicut on SNEMI3D for probability maps generated with the proposed ICv1 architecure and the Ciresan network [42]. The evaluation is done completely in 3D, and higher numbers are better.

4.7.3 Neuroproof

To study performance on fully isotropic data, we have turned to the example datasets provided with the NeuroProof software[1] [134]), which includes focused ion beam scanning electron microscopic (FIBSEM) volumes along with pixelwise membrane probabilities and 3D superpixels. All of these were used as input for the final stage of our pipeline, producing the results reported in Table 4.2. For the Lifted Multicut, only the ultrametric distance and region features were used, with lifted edges between superpixels within a distance of two in the region adjacency graph. The resulting segmentation accuracies are higher than the published state of the art. The numbers given are relative to an updated ground truth where we have detected and fixed a single obvious segmentation error.

Neuroproof accuracy	$V_{0.5}^{\text{rand}}$	$V_{0.5}^{\text{info}}$
Multicut	0.93646	0.96173
Lifted Multicut	0.94047	0.96400

Table 4.2: Results for the isotropic neuroproof dataset. The evaluation is done completely in 3D.

Computational footprint All experiments were run on either a commodity notebook or a workstation with 20-core intel Xeon processor and 256 GB of RAM. Predicting boundary probability maps for the ISBI 2012 test set with the neural network took less than an hour on a single GPU. Running the remainder of the pipeline took less than an hour for the smallest problem (Multicut on Watershed Distance Transform Superpixels for the ISBI 2012 data) and less than a day for the largest problem (Lifted Multicut on SNEMI3D data).

4.8 Lesion Study

To study the effectiveness of different alternatives in our pipeline, we have conducted a lesion study with the data from the ISBI2012 challenge, the best-studied of the three data sets. Note that all measures reported for this dataset are calculated in 2D for each slice individually and then averaged.

[1] https://github.com/janelia-flyem/neuroproof_examples

ISBI 2012 accuracy	$V_{0.5}^{\text{rand}}$	$V_{0.5}^{\text{info}}$
Cascaded RF	0.89390	0.95440
ICv1	0.97735	0.98865

Table 4.3: Results of different probability maps on the ISBI test block.

ISBI 2012 accuracy	standard watershed		DT watershed	
	$V_{0.5}^{\text{rand}}$	$V_{0.5}^{\text{info}}$	$V_{0.5}^{\text{rand}}$	$V_{0.5}^{\text{info}}$
Cascaded RF & MC	0.96229	0.98436	0.97907	0.98844
Cascaded RF & LMC	0.97040	0.98375	0.97852	0.98798
ICv1 & MC	0.98039	0.98855	0.98257	0.98946
ICv1 & LMC	0.97510	0.98757	0.98262	0.98946

Table 4.4: Performance of different boundary probability estimators, and different superpixel generators combined with standard and Lifted Multicut.

First, we have investigated the use of different classifiers for predicting the membrane probability. Table 4.3 shows the scores for the probability maps without any additional processing by a (Lifted) Multicut. Specifically, the segmentation accuracy of a probability map alone is evaluated by thresholding at a particular probability value and using the resulting connected components as segmentation. In this case, the border thinning described in [19] is relevant. Our best neural network, ICv1, performs substantially better than the Cascaded Random Forest.
Table 4.4 shows that the (Lifted) Multicut greatly improves the segmentation accuracy beyond what even the best boundary probability estimators can deliver. Apparently, the (Lifted) Multicut can compensate for many errors made in the estimation of the probability map, because this additional processing substantially reduces the performance difference between neural network and Cascaded Random Forest.
Furthermore, we compare the superpixel generation using standard and distance trans-

ISBI 2012 accuracy	$V_{0.5}^{\text{rand}}$	$V_{0.5}^{\text{info}}$
2D	0.97775	0.98886
3D	0.98257	0.98946

Table 4.5: 2D vs. 3D Multicut on the ISBI test set.

form watersheds: according to Table 4.4 again, the latter clearly yields better results in all combinations. The results shown above were obtained from a 3D Multicut. This is better than solving the Multicut problems separately for each slice, as shown in Table 4.5.

5 Software

5.1 Nifty

We implemented a C++ software library with a fully functional Python API for graph based image segmentation and discrete optimization for image segmentation named nifty [28] in approximate 45000 line of code (see section 5.1 for details).
Nifty was developed with the following design aspects in mind:

- C++ is used as core implementation language: Following [1, 12, 87, 115] the main implementation language is C++. Support for other programing languages as Python is achieved by using wrapper code as pybind11 [62].
- Modern C++ 14 is utilized to write maintainable, elegant and readable code: We use C++ 14 since i) all major compiler support C++ 14[1], ii) new features as the `auto` keyword and range based for loops (`for(auto & var : vector){}`) have increased the readability of C++ 11/14 code drastically compared to C++ 98. iii) New features as smart pointer decrease the risk of bugs and memory leaks and iv) to be compatible with modern C++ libraries as xtensor [115] and pybind11 [62]. Code sample 1 clearly illustrates the benefits and importance of proper C++ 14 usage.
- A fully functional Python API: Not only is Python well known for its rapid prototyping capabilities, Python is the currently leading language for machine learning[2]. To serve the Python community we implemented a fully functional Python API using pybind11 [62] to generate wrapper code.

Not only do we provide reference implementations for the algorithms proposed in chapter 2 and chapter 3 but also a broad set of other optimizers for the multicut and lifted multicut objective. Furthermore we provided a flexible agglomerative clustering framework and implemented many flavors of agglomerative clustering within it. In the

[1] http://en.cppreference.com/w/cpp/compiler_support
[2] https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Language_Is_Best_For_Machine_Learning_And_Data_Science?lang=en

following we will give a brief overview of the aforementioned features implemented within *nifty*.

Nifty

```
for(auto u : g.nodes()){
    for(auto adj : g.adjacency(u)){
        auto edge = adj.edge();
        auto v = adj.node();
    }
}
```

Andres Graph

```
for (auto u = 0; u < g.numberOfNodes(); ++u){
    for (auto it = g.adjacenciesFromVertexBegin(u); it !=
 ↪   g.adjacenciesFromVertexEnd(u); ++it){
        auto edge it.edge();
        auto v = it.vertex();
    }
}
```

Lemon

```
for (lemon::ListGraph::NodeIt u(g); u!=lemon::INVALID; ++u){
    for (lemon::ListGraph::OutArcIt edge(g, u); edge!=lemon::INVALID; ++edge){
        auto v = edge.target();
    }
}
```

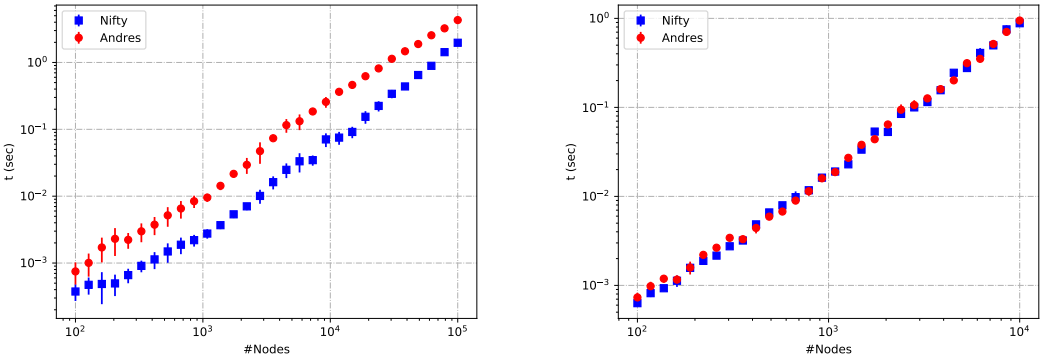
Code Sample 1: C++ code to Iteration over all nodes and their respective neighbors nodes for a graph `g` using different graph frameworks. With the *nifty* framework we can fully utilize modern C++ 11/14 features as the `auto` keyword and range based for loops. Therefore, the code using the *nifty* framework is shorter and arguably better than the competitors in terms of understandability.

5.1.1 Multicut

Not only is *nifty* currently the largest collection of open source multicut solvers available, but also *nifty* provides arguably the most convenient interface to these solvers as illustrated in code sample 2. Advanced examples can be found in the online example-gallery[3].
Below is a list of all the solvers which are currently available within *nifty*.

- Energy Based Hierarchical Clustering / Greedy Additive [27]: Optimizes the multicut via agglomerative clustering. The algorithm is very fast but suffers from

[3] http://derthorsten.github.io/nifty/docs/python/html/auto_examples/index.html#gallery-of-multicut-related-examples



(a) Greedy-Additive (b) Kernighan-Lin

Figure 5.1: We compared the runtime of different multicut algorithms implemented within *nifty* and the *Andres graph library* [11]. We use a synthetic graph with a sparse randomized connectivity structure and random edge weights as input. On the left we used *Greedy-Additive* and on the right *Kernighan-Lin*. Note that x and y axis are in log-scale in both plots. *Greedy-Additive* implemented within *nifty* runs about twice as fast while *Kernighan-Lin* is approximately equally fast in both frameworks.

Language	files	comment	code
C++	312	4373	48714
Python	68	1221	7085
CMake	45	253	1232
others	8	203	822
SUM:	433	6050	45329

Table 5.1: Code Metrics of the *nifty* library [28].

the intrinsic greediness. See section 2.3 for more details. The implementation within *nifty* runs twice as fast as the same algorithm implemented within the *Andres graph library* [11] (see fig. 5.1a).

- *Kernighan-Lin* [76]: Local move making heuristic which has been adapted to solve the multicut objective. The *Kernighan-Lin* algorithm strongly benefits from a good starting point and the solutions form *Greedy Additive* can serve as such.

- *Cut&Glue&Cut* [29]: A move making algorithm which iteratively refines the cut between two adjacent connected components. We implemented this solver such that any other solver implemented within *nifty* can be used to optimize the internal two-coloring problem.

- *Decomposing Solver* [5]: As proven by [5] one can decompose the multicut objective into individual subproblems and solve these independently if and if only there are only negative weighted edges between the individual subproblems. We implemented this solver such that any other solver implemented in *nifty* can be used to optimize the subproblems. Since the subproblems can be solved independently we allow for parallel optimization. The overhead to construct the subproblems is usually negligible.

- *Multicut ILP* [9, 68]: A global optimal cutting plane integer linear programming (ILP) solver. We support i) *Cplex* [59] ii) *Gurobi* [57] and iii) *GLPK* [54] as ILP back-end. For any real world problem we strongly support to use either *Cplex* or *Gurobi*. *GLPK* does not scale well for ILPs and is supported for educational purpose since it is free and open source.

- *Message Passing Multicut* [131]: A dual decomposition based convergent message

passing solver. We implemented a wrapper around [105], such that any of solver implemented within *nifty* can be injected to compute the primal solution.

- *Fusion-Moves* [27]: Fusion Moves for Multicut Partitioning as described in chapter 2. Any solver within *nifty* can be used to optimize the internal subproblem. The proposal generator are implemented in an easy extendable way such that problem specific proposal generator can easily be integrated.

- *Chained-Solvers*: This allows to chain multiple solvers which can be warm started into a single meta-solver. This new meta-solver can be used to optimize the internal subproblem in multiple algorithms. We found that it is very useful to chain *Greedy-Additive* with *Kernighan-Lin*.

5.1.2 Lifted Multicut

We provide the largest collection of open source solvers for the lifted multicut objective, but also we make lifted multicut solvers as accessible as possible via a convenient interface as illustrated in code sample 3. The implementations within *nifty* run 1.5-500 times faster as the the same algorithms implemented within the *Andres graph library*[11] (see fig. 5.2a).

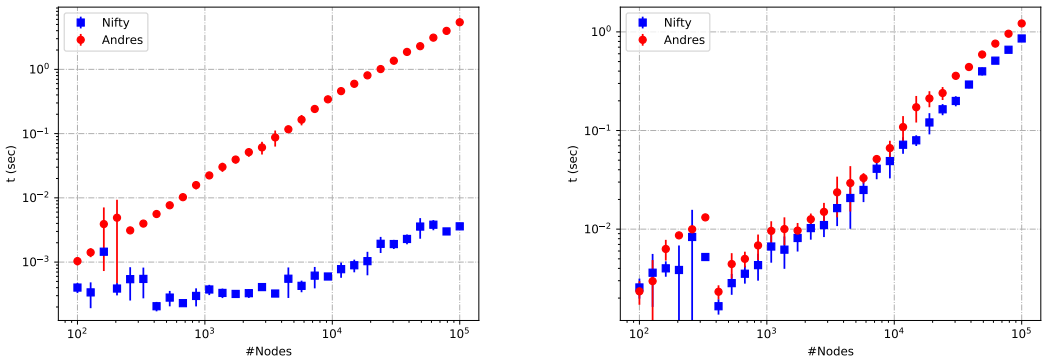
- Greedy Additive [78]: Optimizes the lifted multicut via agglomerative clustering. Plain agglomerative clustering is extended, such that lifted non-graph edges are only considered for merges once they are merged with local graph edges. The algorithm is very fast but but suffers from the intrinsic greediness. See section 2.3 for more details.

- *Kernighan-Lin* [76, 78]: Local move making heuristic which has been adapted to solve the lifted multicut objective. The *Kernighan-Lin* algorithm strongly benefits from a good starting point and the solutions form *Greedy Additive* can serve as such.

- *Decomposing Solver* [5]: As proven by [5] one can decompose the multicut objective into individual subproblems and solve these independently if and if only there are only negative weighted edges between the individual subproblems. We generalized this solver to also work for lifted multicut objectives. We implemented this solver such that any other solver implemented in *nifty* can be used to optimize the subproblems. Since the subproblems can be solved independently we allow for parallel optimization. The overhead to construct the subproblems is usually negligible.

```
import nifty,numpy

# create a graph with random edges and weights between -0.5 and 0.5
g = nifty.graph.randomGraph(numberOfNodes=20, numberOfEdges=30)
w = numpy.random.rand(g.numberOfEdges)  - 0.5

# the multicut objective function
Obj = g.MulticutObjective
objective = Obj(g, w)

# create different solvers factories
factories = {
    'greedy' : Obj.greedyAdditiveFactory(),
    'kl' : Obj.kernighanLinFactory(),
    # simple fusion move sover
    'fm' : Obj.ccFusionMoveBasedFactory(),
    # advanced fusion move solver
    'fm-ws' : Obj.ccFusionMoveBasedFactory(
        proposalGenerator=Obj.watershedCcProposals(),
        # fusion move settings
        fusionMove=Obj.fusionMoveSettings(
            # chain 2 solvers and use chained solver
            # to optimize the fusion move problem
            Obj.chainedSolversFactory([Obj.greedyAdditiveFactory(),
                                       Obj.kernighanLinFactory()])
        )
    )
}
# commercial solvers
if nifty.Configuration.WITH_CPLEX:
    factories['ilp_cplex'] =  Obj.multicutIlpFactory(ilpSolver='cplex')
if nifty.Configuration.WITH_GUROBI:
    factories['ilp_gurobi'] =  Obj.multicutIlpFactory(ilpSolver='gurobi')

# optimize the objective with different solvers
for solver_name in factories.keys():
    solver = factories[solver_name].create(objective)
    argmin = solver.optimize(objective.verboseVisitor())
```

Code Sample 2: This code sample illustrates the usage of different solver to optimize the multicut objective with the *nifty* library. For the sake of simplicity a randomized graph is used. While most solvers do not need any parameters, fusion move solvers can be parameterized in a very sophisticated way. For `'fm-ws'` we explicitly specify the proposal generator and the solver for the fusion move. We use a *chained solver* to combine *greedy additive* and *kernighanLin* into a single solver and use this meta-solver to optimize the fusion move subproblems.



(a) Greedy-Additive (b) Kernighan-Lin

Figure 5.2: We compared the runtime of different lifted multicut algorithms implemented within *nifty* and the *Andres graph library* [11]. We use a synthetic graph with a sparse randomized connectivity structure and random edge weights as input. On the left we used *Greedy-Additive* and on the right *Kernighan-Lin*. Note that x and y axis are in log-scale in both plots. It is clearly visible that the *Greedy-Additive* algorithm implemented within *nifty* scales much better. The *Kernighan-Lin* implementation within *nifty* is about 1.5 faster.

- *Lifted multicut ILP* [7]: A global optimal cutting plane integer linear programming (ILP) solver. We support i) *Cplex* [59] ii) *Gurobi* [57] and iii) *GLPK* [54] as ILP back-end. For any real world problem we strongly support to use either *Cplex* or *Gurobi*. *GLPK* does not scale well for ILPs and is supported for educational purpose since it is free and open source. Due to the NP-hardness and highly combinatorial nature of the problem, ILP solvers are only applicable for problems where not many violated constraints are expected.

- *Message Passing Lifted Multicut* [105, 131]: A dual decomposition based convergent message passing solver. We implemented a wrapper around [105], such that any of solver implemented within *nifty* can be injected to compute the primal solution.

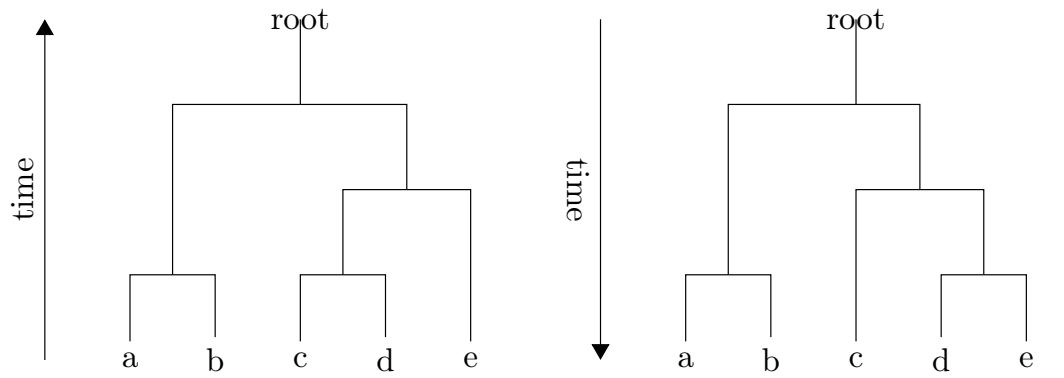
- *Fusion-Moves* [27]: Fusion Moves for lifted multicut partitioning as described in chapter 3. Any solver within *nifty* can be used to optimize the internal fusion move subproblem. The proposal generator are implemented in an easy extendable way such that problem specific proposal generator can easily be integrated.

- *Chained-Solvers*: This allows to chain multiple solvers which can be warm started into a single meta-solver. As for the multicut, we found that it is very useful to chain *Greedy-Additive* with *Kernighan-Lin*.

```
import nifty
import numpy as np
import nifty.graph.opt.lifted_multicut as lmc

# graph with 5 nodes and 6 edges:
#             0 - 2
#             |   | \
#             1 - 3 - 4
g = nifty.graph.undirectedGraph(numberOfNodes=5)
g.insertEdges([[0,1],[0,2],[1,3],[2,3], [2,4], [3,4]])

# setup the lifted multicut objective:
objective = lmc.liftedMulticutObjective(g)
Objective = objective.__class__

# add costs for some pair of nodes u,v
# u,v does not need to be an edge in g
uv = [[0,1], [2,3], [0,4], [2,3]]
weights = [0.2, 0.1, -0.9, -0.3]
objective.setCosts(uv, weights)

# solver factory (here we use an ilp solver)
factory = Objective.liftedMulticutIlpFactory()

# create solver from factory
solver = factory.create(objective)

# optimize
result = solver.optimize()
print(result)
```

Code Sample 3: Lifted multicut example: The code sample above illustrates how to construct a lifted multicut objective for a simple graph. Here, we optimize the objective with a cutting plane ILP solver.

5.1.3 Agglomerative Clustering



(a) Agglomerative Clustering: Clusters are merged with increasing time

(b) Top-Down: Clusters are divided with increasing time

Figure 5.3: Agglomerative clustering: Clusters are merged with increasing time. Top-Down clustering: Clusters are divided with increasing time.

The main idea behind agglomerative clustering is very simple: Initially, all observations start in a single cluster. Next, adjacent clusters which have highest similarities will be merged iteratively as illustrated in fig. 5.3a. Due to this merging, similarities change and need to be updated or recomputed. As described in section 3.4.3 we use agglomerative clustering to optimize eq. (3.1) and eq. (3.3). To this end we implemented an easy to use and flexible agglomerative clustering framework (see code sample 4). Different flavors of agglomerative clustering are implemented by providing so called *cluster-policies* as illustrated in code sample 5.
Within *nifty* we implemented the following clustering algorithms:

- Ultrametric Contour Maps UCM [18]: UCMs allow to encode a whole dendrogramm of agglomerative clustering in a single scalar per edge merely by remembering the priority/similarity at the time where the endpoints of the edge merge. Iff the similarities are monotonously decreasing, thresholding the UCM yields a closed contour segmentation for any possible threshold. We implemented this technique in a generic fashion such that the UCM can be computed for any *cluster policy* implemented within *nifty*.

- Median weighted clustering [53]: Funke et al. [53] successfully applied agglomerative clustering where edge weights are updated according to their median to

```python
import nifty
import nifty.graph.agglo as agglo
import numpy as np

# toy graph with 5 nodes and 6 edges:
g = nifty.graph.undirectedGraph(numberOfNodes=5)
g.insertEdges([[0,1],[0,2],[1,3],[2,3], [2,4], [3,4]])

# cluster-policy / the rules how to cluster
clusterPolicy = agglo.edgeWeightedClusterPolicy(graph=g,
    edgeIndicators=np.random.rand(g.numberOfEdges),
    edgeSizes=np.ones(g.numberOfEdges),
    nodeSizes=np.ones(g.numberOfNodes),
    numberOfNodesStop=2)

# - construct cluster algorithm, run clustering and get resutls
agglomerativeClustering = agglo.agglomerativeClustering(clusterPolicy)
agglomerativeClustering.run()
result = agglomerativeClustering.result()

print(result)
```

Code Sample 4: Agglomerative clustering example: The code sample above illustrates how segment a graph using agglomerative clustering. We use random edge weights for illustrative purpose only.

segment a region adjacency graph for the CREMI-challenge[4]. We implemented this *cluster-policies* using a fast histogram based implementation.

- Energy Based Hierarchical Clustering / Greedy Additive Clustering [27, 78]: We implemented algorithms for the multicut and lifted multicut objective based on agglomerative clustering where the edge weights are update according to their sum to approximate the multicut objective and lifted multicut objective respectively (see section 2.3 for details). This implementation has led to solvers outperforming other frameworks by factor of 2 in the case of the multicut objective and by a factor 500 in the case of the lifted multicut objective (see figs. 5.1a and 5.2a).

[4] https://cremi.org/

5.2 Multicut-Pipeline

In chapter 4 we proposed a framework for automated segmentation of neural structures in 3D EM data. To ensure maximum reproducible and accessibility the algorithms and software proposed in chapter 4 have been made available open source at online repositories, and we also provide precompiled binaries.
For software source code, binaries, sample data, and neural network parameters, please visit the following link: http://files.ilastik.org/multicut/. The source code is also available on github: http://github.com/ilastik/nature_methods_multicut_pipeline.
Our pipeline uses the vigra image processing library [87] for image manipulation, the OpenGM library [12, 27] and nifty [28] for solving discrete optimization problems and the scikit-learn library [114] for the Random Forest. The pipeline itself is written in Python and the complete source code has been made available. Besides the source code, we provide pre-compiled binary files for recent releases of Linux, Windows and OSX. Given the current limitations in GPU standardization, the neural network code is provided open source with this publication, but is not precompiled into the executables. The user of the binaries needs to provide probability maps by compiling the neural network by them self, or by training a classifier as made available by open source software including ilastik (http://ilastik.org), the ImageJ trainable Weka segmentation (http://imagej.net/Trainable_Weka_Segmentation) or some other tool.
The proposed pipeline has been well received by the community and is used by different labs around the world. At the time of writing, half of the top 10 entries in the ISBI challenge use the provided source-code in conjunction with their own probability maps.

```
class ClusterPolicyAPI{
    // which edge to contract next
    std::pair<uint64_t, double> edgeToContractNext() const;

    // stopping-criterion / is the clustering done
    bool isDone() const;

    // indicate that contraction of edge is started
    void contractEdge(const uint64_t edgeToContract);

    // merge endpoints of contracted edge into single node
    void mergeNodes(const uint64_t aliveNode, const uint64_t deadNode);

    // merge (possible) parallel edges into a single edge
    void mergeEdges(const uint64_t aliveEdge, const uint64_t deadEdge);

    // indicate that contraction of edge is started
    void contractEdgeDone(const uint64_t edgeToContract);
};
```

Code Sample 5: We implemented a flexible agglomerative clustering framework within *nifty*. To implement a new clustering algorithm only a so called *cluster-policy* needs to be implemented. The *cluster-policy* informs the clustering algorithm which edge to contract next via `edgeToContractNext()`. This will update the internal *edge-contraction-graph* and trigger four callbacks: i) `contractEdge(const uint64_t edgeToContract)` indicates that the edge contraction procedure has been started, ii) `mergeNodes(const uint64_t aliveNode, const uint64_t deadNode)` indicates how the endpoints are merged into a single note and iii) `mergeEdges(const uint64_t aliveEdge, const uint64_t deadEdge)` is called whenever an edge contraction yields parallel edges. iv) `contractEdgeDone(const uint64_t edgeToContract)` is called when the edge contraction is done. To terminate the clustering process `isDone()` has to return `true` .

6 Conclusion

In chapters 2 and 3 we have presented a fast and scalable approximate solver for the minimum multicut objective and minimum lifted multicut objective respectively. The presented algorithms are orthogonal to previous research, i.e. they can be combined with any other multicut solver as [131]. Problem specific knowledge can easy be incorporated by providing a problem specific proposal generator and blockwise solvers as [93, 113] are easy to formulate as fusion move when proposals are generated in a blockwise fashion.
The proposed fusion move framework can be generalized to higher order multicut [65] and higher order lifted multicut [77] problems, however this is part of future work. We conjecture that efficient algorithms such as the one proposed in this thesis facilitate a variety of applications of the minimum cost multicut and minimum cost lifted multicut problem in computer vision of which image segmentation and the averaging of multiple segmentations are just two examples.
In chapter 4 we proposed and validated a pipeline for automatic segmentation of neural structured in 3D EM connectomics data. Any CNN can be incorporated in the proposed framework and therefore the pipeline can easily be adapted to new state-of-the-art networks. While a CNN is used all predictions on a pixel level, we rely on a set of predefined features and a shallow classifier to predict if a pair of superpixels should be in the same connected component or not.
Future work should focus on replacing the non learned components and predefined features with their learned equivalent. A learned variant of the watersheds transform [144] could be used to replace the distance transform watersheds in section 4.4 and furthermore it might be possible to use Evolutionary Strategies [121] to end-to-end learn the complete segmentation pipeline including the non differentiable NP-hard multicut and lifted multicut optimization.

Publications

List of Peer Reviewed Publications

I presented [5] as an oral at CVPR and [6] as an oral at ECCV and contributed to the following peer reviewed publications:

[1] Bjoern Andres, Jörg H. Kappes, Thorsten **Beier**, Ullrich Köthe, and Fred A. Hamprecht. "Probabilistic Image Segmentation with Closedness Constraints." In: *ICCV*. IEEE. 2011, pp. 2611–2618.

[2] Bjoern Andres, Jörg H Kappes, Thorsten **Beier**, Ullrich Köthe, and Fred A Hamprecht. "The lazy flipper: Efficient depth-limited exhaustive search in discrete graphical models." In: *ECCV*. Springer, 2012, pp. 154–166. DOI: 10.1007/978-3-642-33786-4_12.

[3] T **Beier** et al. "Multicut brings automated neurite segmentation closer to human performance." In: *Nature Methods* 14 (2017), pp. 101–102. DOI: 10.1038/nmeth.4151. URL: http://rdcu.be/oVDQ.

[4] Thorsten **Beier**, Fred Hamprecht, and Joerg Kappes. "Fusion Moves for Correlation Clustering." In: *CVPR. Proceedings*. 1. 2015, pp. 3507–3516.

[5] Thorsten **Beier**, Thorben Kroeger, Jörg Hendrik Kappes, Ullrich Koethe, and Fred Hamprecht. "Cut, Glue & Cut: A Fast, Approximate Solver for Multicut Partitioning." In: *IEEE Conference on Computer Vision and Pattern Recognition 2014*. 2014.

[6] Thorsten **Beier**, Björn Andres, Ullrich Köthe, and Fred A. Hamprecht. "An Efficient Fusion Move Algorithm for the Minimum Cost Lifted Multicut Problem." In: *ECCV (2)*. Vol. 9906. Lecture Notes in Computer Science. Springer, 2016, pp. 715–730.

[7] Jörg H. Kappes, Thorsten **Beier**, and Christoph Schnörr. "MAP-Inference on Large Scale Higher-Order Discrete Graphical Models by Fusion Moves." In: *Computer Vision - ECCV 2014 Workshops - Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part II*. 2014. DOI: 10.1007/978-3-319-16181-5_37.

[8] N Krasowki, T **Beier**, G. Knott, U Köthe, F. A. Hamprecht, and A. Kreshuk. "Neuron Segmentation with High-Level Biological Priors." In: *IEEE Transactions on Medical Imaging* (2017). DOI: 10.1109/TMI.2017.2712360.

[9] Niko Krasowski, Thorsten **Beier**, G. W. Knott, Ullrich Köthe, Fred A. Hamprecht, and Anna Kreshuk. "Improving 3D EM Data Segmentation by Joint Optimization over Boundary Evidence and Biological Priors." In: *12th IEEE International Symposium on Biomedical Imaging, ISBI 2015, Brooklyn, NY, USA, April 16-19, 2015*. 1. 2015, pp. 536–539. DOI: 10.1109/ISBI.2015.7163929.

[10] Thorben Kroeger, Jörg Hendrik Kappes, Thorsten **Beier**, Ulrich Koethe, and Fred A. Hamprecht. "Asymmetric Cuts: Joint Image Labeling and Partitioning." In: *36th German Conference on Pattern Recognition*. 2014.

[11] C Pape, T **Beier**, P Li, V Jain, D. D. Brock, and A. Kreshuk. "Solving Large Multicut Problems for Connectomics via Domain Decomposition." In: *Bioimage Computing Workshop. ICCV, in press* (2017).

[12] Julian Yarkony, Thorsten **Beier**, Pierre Baldi, and Fred A. Hamprecht. "Parallel Multicut Segmentation via Dual Decomposition." In: *New Frontiers in Mining Complex Patterns - Third International Workshop, NFMCP 2014, Held in Conjunction with ECML-PKDD 2014, Nancy, France, September 19, 2014, Revised Selected Papers*. 2014, pp. 56–68. DOI: 10.1007/978-3-319-17876-9_4. URL: http://dx.doi.org/10.1007/978-3-319-17876-9_4.

Bibliography

[1] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. "SLIC superpixels compared to state-of-the-art superpixel methods." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 34.11 (2012), pp. 2274–2282.

[3] Nir Ailon, Moses Charikar, and Alantha Newman. "Aggregating Inconsistent Information: Ranking and Clustering." In: *J. ACM* 55.5 (Nov. 2008), 23:1–23:27. ISSN: 0004-5411. DOI: 10.1145/1411509.1411513. URL: http://doi.acm.org/10.1145/1411509.1411513.

[4] Amir Alush and Jacob Goldberger. "Ensemble segmentation using efficient integer linear programming." In: *Pattern Analysis and Machine Intelligence* 34.10 (2012), pp. 1966–1977.

[5] Amir Alush and Jacob Goldberger. "Break and Conquer: Efficient Correlation Clustering for Image Segmentation." In: *2nd International Workshop on Similarity-Based Pattern Analysis and Recognition*. 2013.

[6] James R. Anderson et al. "Exploring the retinal connectome." In: *Mol. Vis.* 17 (2011), pp. 355–379. ISSN: 1090-0535. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3036568/.

[7] Bjoern Andres. "Lifting of Multicuts." In: *CoRR* abs/1503.03791 (2015). http://arxiv.org/abs/1503.03791.

[8] Bjoern Andres, Jörg H. Kappes, Thorsten Beier, Ullrich Köthe, and Fred A. Hamprecht. "Probabilistic Image Segmentation with Closedness Constraints." In: *ICCV*. IEEE. 2011, pp. 2611–2618.

[9] Bjoern Andres, Thorben Kröger, Kevin L. Briggman, Winfried Denk, Natalya Korogod, Graham Knott, Ullrich Köthe, and Fred A. Hamprecht. "Globally Optimal Closed-surface Segmentation for Connectomics." In: *ECCV*. Springer, 2012, pp. 778–791.

[10] Bjoern Andres, Jörg H Kappes, Thorsten Beier, Ullrich Köthe, and Fred A Hamprecht. "The lazy flipper: Efficient depth-limited exhaustive search in discrete graphical models." In: *ECCV*. Springer, 2012, pp. 154–166. DOI: 10.1007/978-3-642-33786-4_12.

[11] Bjoern Andres, Duligur Ibeling, Giannis Kalofolias, Margret Keuper, Evgeny Levinkov, Mark Matten, and Markus Rempfler. *Andres Graph. Graphs and Graph Algorithms in C++*. 2015. URL: http://www.andres.sc/graph.html (visited on 11/18/2017).

[12] Björn Andres, Thorsten Beier, and Jörg H. Kappes. "OpenGM: A C++ Library for Discrete Graphical Models." In: *CoRR* abs/1206.0111 (2012).

[13] Björn Andres, Ullrich Köthe, Moritz Helmstaedter, Winfried Denk, and Fred A Hamprecht. "Segmentation of SBFSEM volume data of neural tissue by hierarchical classification." In: *Pattern Recogn.* (2008), pp. 142–152.

[14] Björn Andres, Ullrich Köthe, Thorben Kroeger, Moritz Helmstaedter, Kevin L. Briggman, Winfried Denk, and Fred A. Hamprecht. "3D segmentation of SBFSEM images of neuropil by a graphical model over supervoxel boundaries." In: *Medical Image Analysis* 16.4 (2012), pp. 796–805. ISSN: 1361-8415. DOI: 10.1016/j.media.2011.11.004. URL: http://www.sciencedirect.com/science/article/pii/S1361841511001666.

[15] Jesús Angulo and Dominique Jeulin. "Stochastic watershed segmentation." In: *Proc. ISMM Rio de Janeiro* 1 (2007), pp. 265–276.

[16] Arvind Arasu, Christopher Re, and Dan Suciu. "Large-Scale Deduplication with Constraints using Dedupalog." In: *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009*. IEEE Computer Society, 2009. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=77606.

[17] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. "Contour detection and hierarchical image segmentation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33 (2011), pp. 898–916.

[18] Pablo Arbelaez. "Boundary Extraction in Natural Images Using Ultrametric Contour Maps." In: *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*. CVPRW '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 182–. ISBN: 0-7695-2646-2. DOI: 10.1109/CVPRW.2006.48. URL: http://dx.doi.org/10.1109/CVPRW.2006.48.

[19] Ignacio Arganda-Carreras, Sebastian Seung, Ashwin Vishwanathan, and Daniel R Berger. *SNEMI3D: 3D Segmentation of neurites in EM images*. URL: http://brainiac2.mit.edu/SNEMI3D/ (visited on 04/04/2016).

[20] Ignacio Arganda-Carreras, Sebastian Seung, Ashwin Vishwanathan, and Daniel R Berger. *SNEMI3D: 3D Segmentation of neurites in EM images*. URL: http://brainiac2.mit.edu/SNEMI3D/ (visited on 04/04/2016).

[21] Ignacio Arganda-Carreras et al. "Crowdsourcing the creation of image segmentation algorithms for connectomics." In: *Frontiers in Neuroanatomy* 9.142 (2015), pp. 1–13. ISSN: 1662-5129. DOI: 10.3389/fnana.2015.00142. URL: http://www.frontiersin.org/neuroanatomy/10.3389/fnana.2015.00142/abstract.

[22] Yoram Bachrach, Pushmeet Kohli, Vladimir Kolmogorov, and Morteza Zadimoghaddam. "Optimal Coalition Structure Generation in Cooperative Graph Games." In: *AAAI*. 2013. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6407/7071.

[23] Shai Bagon and Meirav Galun. "Large Scale Correlation Clustering Optimization." In: *CoRR* abs/1112.2903 (2011). http://arxiv.org/abs/1112.2903.

[24] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. "Correlation Clustering." In: *MACHINE LEARNING*. 2002, pp. 238–247.

[25] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. "Correlation Clustering." In: *Machine Learning* 56.1–3 (2004), pp. 89–113. DOI: 10.1023/B:MACH.0000033116.57574.95.

[26] T Beier et al. "Multicut brings automated neurite segmentation closer to human performance." In: *Nature Methods* 14 (2017), pp. 101–102. DOI: 10.1038/nmeth.4151. URL: http://rdcu.be/oVDQ.

[27] Thorsten Beier, Fred Hamprecht, and Joerg Kappes. "Fusion Moves for Correlation Clustering." In: *CVPR. Proceedings*. 1. 2015, pp. 3507–3516.

[28] Thorsten Beier and Constantin Pape. *Nifty. A nifty library for graph based image segmentation.* 2015. URL: https://github.com/DerThorsten/nifty (visited on 11/18/2017).

[29] Thorsten Beier, Thorben Kroeger, Jörg Hendrik Kappes, Ullrich Koethe, and Fred Hamprecht. "Cut, Glue & Cut: A Fast, Approximate Solver for Multicut Partitioning." In: *IEEE Conference on Computer Vision and Pattern Recognition 2014*. 2014.

[30] Thorsten Beier, Björn Andres, Ullrich Köthe, and Fred A. Hamprecht. "An Efficient Fusion Move Algorithm for the Minimum Cost Lifted Multicut Problem." In: *ECCV (2)*. Vol. 9906. Lecture Notes in Computer Science. Springer, 2016, pp. 715–730.

[31] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. "Theano: a CPU and GPU math expression compiler." In: *Proc. Python Sci. Comp. Conf.* 4 (2010), p. 3.

[32] Manuel Berning, Kevin M. Boergens, and Moritz Helmstaedter. "SegEM: Efficient Image Analysis for High-Resolution Connectomics." In: *Neuron* 87.6 (2015), pp. 1193–1206. ISSN: 10974199. DOI: 10.1016/j.neuron.2015.09.003. URL: http://dx.doi.org/10.1016/j.neuron.2015.09.003.

[33] Serge Beucher and Fernand Meyer. "The morphological approach to segmentation: the watershed transformation." In: *Optical Engineering* 34 (1992), pp. 433–433.

[34] Davi D. Bock, Wei-Chung Allen Lee, Aaron M. Kerlin, Mark L. Andermann, Greg Hood, Arthur W. Wetzel, Sergey Yurgenson, Edward R. Soucy, Hyon Suk Kim, and R. Clay Reid. "Network anatomy and in vivo physiology of visual cortical neurons." In: *Nature* 471.7337 (2011), pp. 177–182. ISSN: 0028-0836. DOI: 10.1038/nature09802. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3095821/.

[35] Kevin L Briggman and Davi D Bock. "Volume electron microscopy for neuronal circuit reconstruction." In: *Curr. Opin. Neurobiol.* 22.1 (2012), pp. 154–161.

[36] Randal Burns, Joshua?T Vogelstein, and Alexander?S Szalay. "From Cosmos to Connectomes: The Evolution of Data-Intensive Science." In: *Neuron* 83.6 (2014), pp. 1249–1252. ISSN: 0896-6273. URL: http://www.sciencedirect.com/science/article/pii/S0896627314007466.

[37] Albert Cardona, Stephan Saalfeld, Stephan Preibisch, Benjamin Schmid, Anchi Cheng, Jim Pulokas, Pavel Tomancak, and Volker Hartenstein. "An Integrated Micro- and Macroarchitectural Analysis of the Drosophila Brain by Computer-Assisted Serial Section Electron Microscopy." In: *PLoS Biol* 8.10 (Oct. 2010), pp. 1–17. DOI: 10.1371/journal.pbio.1000502. URL: http://dx.doi.org/10.1371%2Fjournal.pbio.1000502.

[38] Yudong Chen, Sujay Sanghavi, and Huan Xu. "Clustering Sparse Graphs." In: *NIPS*. 2012, pp. 2213–2221.

[39] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. "cudnn: Efficient primitives for deep learning." In: *arXiv preprint arXiv:1410.0759* (2014).

[40] Flavio Chierichetti, Nilesh N. Dalvi, and Ravi Kumar. "Correlation clustering in MapReduce." In: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. 2014, pp. 641–650. DOI: 10.1145/2623330.2623743. URL: http://doi.acm.org/10.1145/2623330.2623743.

[41] Sunil Chopra and M.R. Rao. "The partition problem." English. In: *Mathematical Programming* 59.1–3 (1993), pp. 87–115. ISSN: 0025-5610. DOI: 10.1007/BF01581239. URL: http://dx.doi.org/10.1007/BF01581239.

[42] Dan C Ciresan, Alessandro Giusti, Luca M Gambardella, and Jurgen Schmidhuber. "Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images." In: *Proc. NIPS* (2012).

[43] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)." In: *arXiv preprint arXiv:1511.07289* (2015).

[44] William Cook and Paul Seymour. "Tour Merging via Branch-Decomposition." In: *INFORMS J. on Computing* 15.3 (July 2003), pp. 233–248. ISSN: 1526-5528. DOI: 10.1287/ijoc.15.3.233.16078. URL: http://dx.doi.org/10.1287/ijoc.15.3.233.16078.

[45] Camille Couprie, Leo Grady, Laurent Najman, and Hugues Talbot. "Power Watershed: A Unifying Graph-Based Optimization Framework." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 33.7 (July 2011), pp. 1384–1399. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2010.200. URL: http://dx.doi.org/10.1109/TPAMI.2010.200.

[46] Erik D. Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. "Correlation clustering in general weighted graphs." In: *Theoretical Computer Science* 361.2–3 (2006), pp. 172–187. DOI: 10.1016/j.tcs.2006.05.008.

[47] Piotr Dollár and C. Lawrence Zitnick. "Fast Edge Detection using Structured Forests." In: *PAMI* (2015).

[48] Micha Elsner and Eugene Charniak. "You Talking to Me? A Corpus and Algorithm for Conversation Disentanglement." In: *Proceedings of ACL-08: HLT*. Columbus, Ohio: Association for Computational Linguistics, 2008, pp. 834–842. URL: http://www.aclweb.org/anthology/P/P08/P08-1095.

[49] Micha Elsner and Warren Schudy. "Bounding and Comparing Methods for Correlation Clustering Beyond ILP." In: *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*. ILP '09. Boulder, Colorado: Association for Computational Linguistics, 2009, pp. 19–27. ISBN: 978-1-932432-35-0.

[50] Ahmed Fakhry, Hanchuan Peng, and Shuiwang Ji. "Deep models for brain EM image segmentation : novel insights and improved performance." In: *Bioinformatics* (2016). DOI: 10.1093/bioinformatics/btw165. eprint: http://bioinformatics.oxfordjournals.org/content/early/2016/03/25/bioinformatics.btw165.full.pdf+html. URL: http://bioinformatics.oxfordjournals.org/content/early/2016/03/25/bioinformatics.btw165.abstract.

[51] Charless Fowlkes and Jitendra Malik. *How Much Does Globalization Help Segmentation?* Tech. rep. Division of Computer Science, University of California, Berkeley, July 2004. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.789.

[52] Jan Funke, Bjoern Andres, Fred A. Hamprecht, Albert Cardona, and Matthew Cook. "Efficient Automatic 3D-Reconstruction of Branching Neurons from EM Data." In: *CVPR*. Ieee, 2012, pp. 1004–1011. ISBN: 978-1-4673-1228-8. DOI: 10.1109/CVPR.2012.6247777. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6247777.

[53] Jan Funke, Fabian Tschopp, William Grisaitis, Arlo Sheridan, Chandan Singh, Stephan Saalfeld, and Srinivas C. Turaga. "A Deep Structured Learning Approach Towards Automating Connectome Reconstruction from 3D Electron Micrographs." In: *CoRR* abs/1709.02974 (2017).

[54] *GLPK. GLPK (GNU Linear Programming Kit)*. 2012. URL: https://www.gnu.org/software/glpk/ (visited on 11/18/2017).

[55] Aristides Gionis, Heikki Mannila, and Panayiotis Tsaparas. "Clustering Aggregation." In: *ACM Trans. Knowl. Discov. Data* 1.1 (Mar. 2007). ISSN: 1556-4681. DOI: 10.1145/1217299.1217303. URL: http://doi.acm.org/10.1145/1217299.1217303.

[56] M. Grötschel and Y. Wakabayashi. "A cutting plane algorithm for a clustering problem." In: *Mathematical Programming* 45.1 (1989), pp. 59–96. DOI: 10.1007/BF01589097.

[57] *Gurobi. Gurobi Optimizer*. 2017. URL: http://www.gurobi.com (visited on 11/18/2017).

[58] Moritz Helmstaedter. "Cellular-resolution connectomics: challenges of dense neural circuit reconstruction." In: *Nat. Methods* 10.6 (May 2013), pp. 501–507. ISSN: 1548-7091. DOI: 10.1038/nmeth.2476. URL: http://www.nature.com/doifinder/10.1038/nmeth.2476.

[59] *IBM ILOG CPLEX. High-performance mathematical programming solver for linear programming, mixed integer programming, and quadratic programming.* 2017. URL: https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/ (visited on 11/18/2017).

[60] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." In: *Proc. ICML* (2015), pp. 448–456.

[61] Viren Jain, Joseph F Murray, Fabian Roth, Srinivas Turaga, Valentin Zhigulin, Kevin L Briggman, Moritz N Helmstaedter, Winfried Denk, and H Sebastian Seung. "Supervised learning of image restoration with convolutional networks." In: *Proc. Intl. Conf. Comp. Vision* (2007), pp. 1–8.

[62] Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. *pybind11 - Seamless operability between C++11 and Python*. https://github.com/pybind/pybind11. 2016.

[63] Cory Jones, Ting Liu, Nathaniel Wood Cohan, Mark Ellisman, and Tolga Tasdizen. "Efficient semi-automatic 3D segmentation for neuron tracing in electron microscopy images." In: *J. Neurosci. Methods* 246 (2015), pp. 13–21. ISSN: 0165-0270. URL: http://www.sciencedirect.com/science/article/pii/S0165027015000874.

[64] Elizabeth Jurrus, Antonio RC Paiva, Shigeki Watanabe, James R Anderson, Bryan W Jones, Ross T Whitaker, Erik M Jorgensen, Robert E Marc, and Tolga Tasdizen. "Detection of neuron membranes in electron microscopy images using a serial neural network architecture." In: *Med. Image Anal.* 14.6 (2010), pp. 770–783.

[65] Joerg Hendrik Kappes, Markus Speth, Gerhard Reinelt, and Christoph Schnoerr. "Higher-order segmentation via multicuts." In: *Computer Vision and Image Understanding* 143 (2016). Inference and Learning of Graphical Models Theory and Applications in Computer Vision and Image Analysis, pp. 104 –119. ISSN: 1077-3142. DOI: http://dx.doi.org/10.1016/j.cviu.2015.

11.005. URL: http://www.sciencedirect.com/science/article/pii/S1077314215002490.

[66] Jörg H. Kappes, Thorsten Beier, and Christoph Schnörr. "MAP-Inference on Large Scale Higher-Order Discrete Graphical Models by Fusion Moves." In: *Computer Vision - ECCV 2014 Workshops - Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part II*. 2014. DOI: 10.1007/978-3-319-16181-5_37.

[67] Jörg H. Kappes, Bogdan Savszinskyy, and Christoph Schnörr. "A Bundle Approach To Efficient MAP-Inference by Lagrangian Relaxation." In: *CVPR*. 2012.

[68] Jörg H. Kappes, Markus Speth, Björn Andres, Gerhard Reinelt, and Christoph Schnörr. "Globally Optimal Image Partitioning by Multicuts." In: *EMMCVPR*. Springer. 2011, pp. 31–44.

[69] Jörg H. Kappes et al. "A Comparative Study of Modern Inference Techniques for Discrete Energy Minimization Problems." In: *CVPR*. 2013.

[70] Jörg H. Kappes et al. "A Comparative Study of Modern Inference Techniques for Structured Discrete Energy Minimization Problems." English. In: *International Journal of Computer Vision* (2015), pp. 1–30. ISSN: 0920-5691. DOI: 10.1007/s11263-015-0809-x. URL: http://dx.doi.org/10.1007/s11263-015-0809-x.

[71] Jörg Hendrik Kappes, Markus Speth, Gerhard Reinelt, and Christoph Schnörr. "Higher-order Segmentation via Multicuts." In: *CoRR* abs/1305.6387 (2015). to appear in CVIU in 2016.

[72] Jörg Hendrik Kappes, Markus Speth, Gerhard Reinelt, and Christoph Schnörr. "Higher-order segmentation via multicuts." In: *Computer Vision and Image Understanding* - (2015), pp. –. DOI: http://dx.doi.org/10.1016/j.cviu.2015.11.005.

[73] Narayanan Kasthuri, Kenneth Jeffrey Hayworth, Daniel Raimund Berger, Richard Lee Schalek, José Angel Conchello, Seymour Knowles-Barley, Dongil Lee, Amelio Vázquez-Reina, Verena Kaynig, Thouis Raymond Jones, et al. "Saturated reconstruction of a volume of neocortex." In: *Cell* 162.3 (2015), pp. 648–661.

[74] Verena Kaynig, Thomas Fuchs, and Joachim M Buhmann. "Neuron geometry extraction by perceptual grouping in sstem images." In: *Proc. CVPR* (2010), pp. 2902–2909.

[75] Verena Kaynig, Amelio Vazquez-Reina, Seymour Knowles-Barley, Mike Roberts, Thouis R. Jones, Narayanan Kasthuri, Eric Miller, Jeff Lichtman, and Hanspeter Pfister. "Large-scale automatic reconstruction of neuronal processes from electron microscopy images." In: *Med. Image Anal.* 22.1 (2015), pp. 77–88. ISSN: 13618423. DOI: 10.1016/j.media.2015.02.001. arXiv: 1303.7186. URL: http://dx.doi.org/10.1016/j.media.2015.02.001.

[76] B. W. Kernighan and S. Lin. "An Efficient Heuristic Procedure for Partitioning Graphs." In: *Bell Sys. Tech. J.* 49.2 (1970), pp. 291–308.

[77] Margret Keuper. "Higher-Order Minimum Cost Lifted Multicuts for Motion Segmentation." In: *CoRR* abs/1704.01811 (2017). arXiv: 1704.01811. URL: http://arxiv.org/abs/1704.01811.

[78] Margret Keuper, Evgeny Levinkov, Nicolas Bonneel, Guillaume Lavoué, Thomas Brox, and Bjoern Andres. "Efficient Decomposition of Image and Mesh Graphs by Lifted Multicuts." In: *ICCV*. 2015, pp. 1751–1759. DOI: 10.1109/ICCV.2015.204. arXiv: 1505.06973. URL: http://arxiv.org/abs/1505.06973.

[79] Jinseop S Kim et al. "Space-time wiring specificity supports direction selectivity in the retina." In: *Nature* 509.7500 (2014), pp. 331–336. ISSN: 1476-4687. DOI: 10.1038/nature13240. URL: http://www.ncbi.nlm.nih.gov/pubmed/24805243.

[80] Sungwoong Kim, Sebastian Nowozin, Pushmeet Kohli, and Chang D. Yoo. "Higher-Order Correlation Clustering for Image Segmentation." In: *NIPS*. 2011.

[81] Sungwoong Kim, Chang Yoo, Sebastian Nowozin, and Pushmeet Kohli. "Image Segmentation Using Higher-Order Correlation Clustering." In: *TPAMI* 36 (9 2014), pp. 1761–1774.

[82] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *Proc. ICLR* (2014).

[83] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. "InstanceCut: from Edges to Instances with Multicut." In: *CVPR*. 2017.

[84] P. Kohli, A. Shekhovtsov, C. Rother, V. Kolmogorov, and P. Torr. "On partial optimality in multi-label MRFs." In: *ICML*. 2008.

[85] Vladimir Kolmogorov and Ramin Zabih. "What Energy Functions Can Be Minimized via Graph Cuts?" In: *ECCV*. 2002. ISBN: 3-540-43746-0. URL: http://dl.acm.org/citation.cfm?id=645317.649315.

[86] Vladimir Kolmogorov and Ramin Zabih. "What energy functions can be minimized via graph cuts?" In: *Pattern Analysis and Machine Intelligence* 26.2 (2004), pp. 147–159.

[87] Ullrich Köthe. *Generische Programmierung für die Bildverarbeitung*. Hamburg, Germany: BoD – Books on Demand, 2000.

[88] Anna Kreshuk, Ullrich Köthe, E. Pax, D. D. Bock, and Fred A. Hamprecht. "Automated Detection of Synapses in Serial Section Transmission Electron Microscopy Image Stacks." In: *PLoS ONE* 9 (2014). 1, p. 2. DOI: 10.1371/journal.pone.0087351.

[89] Anna Kreshuk, Robert Walecki, Ullrich Koethe, Mortimer Gierthmuehlen, Dennis Plachta, Christel Genoud, Kristin Haastert-Talini, and Fred A. Hamprecht. "Automated Tracing of Myelinated Axons and Detection of the Nodes of Ranvier in Serial Images of Peripheral Nerves." In: *J. Microsc.* 259 (2) (2015), pp. 143–154. DOI: 10.1111/jmi.12266.

[90] *Who is talking to whom: synaptic partner detection in anisotropic volumes of insect brain*. Vol. LNCS 9349. Springer, 2015, pp. 661–668. DOI: 10.1007/978-3-319-24553-9_81.

[91] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Adv. Neural Inf. Process Syst.* (2012), pp. 1097–1105.

[92] Thorben Kroeger, Jörg Hendrik Kappes, Thorsten Beier, Ulrich Koethe, and Fred A. Hamprecht. "Asymmetric Cuts: Joint Image Labeling and Partitioning." In: *36th German Conference on Pattern Recognition*. 2014.

[93] T. Kröger, S. Mikula, W. Denk, U. Köthe, and F. A. Hamprecht. "Learning to Segment Neurons with Non-local Quality Measures." In: *MICCAI 2013. Proceedings, in press*. Springer, 2013, pp. 419–427.

[94] Thorben Kröger. "Learning-based Segmentation for Connectomics." PhD thesis. 2014.

[95] Wei-Chung Allen Lee, Vincent Bonin, Michael Reed, Brett J. Graham, Greg Hood, Katie Glattfelder, and R. Clay Reid. "Anatomy and function of an excitatory network in the visual cortex." In: *Nature* 532.7599 (2016), pp. 370–374. ISSN: 0028-0836. URL: http://dx.doi.org/10.1038/nature17192.

[96] Victor S. Lempitsky, Carsten Rother, Stefan Roth, and Andrew Blake. "Fusion Moves for Markov Random Field Optimization." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 32.8 (2010), pp. 1392–1405.

[97] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. "Signed Networks in Social Media." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. Atlanta, Georgia, USA: ACM, 2010, pp. 1361–1370. ISBN: 978-1-60558-929-9.

[98] Thomas K. Leung and Jitendra Malik. "Contour Continuity in Region Based Image Segmentation." In: *ECCV (1)*. Ed. by Hans Burkhardt and Bernd Neumann. Vol. 1406. Lecture Notes in Computer Science. Springer, 1998, pp. 544–559. ISBN: 3-540-64569-1. URL: http://dblp.uni-trier.de/db/conf/eccv/eccv1998-1.html#LeungM98.

[99] Evgeny Levinkov, Alexander Kirillov, and Bjoern Andres. "A Comparative Study of Local Search Algorithms for Correlation Clustering." In: *GCPR*. 2017.

[100] Jeff W. Lichtman, Hanspeter Pfister, and Nir Shavit. "The big data challenges of connectomics." In: *Nat. Neurosci.* 17.11 (2014). Perspective, pp. 1448–1454. ISSN: 1097-6256. URL: http://dx.doi.org/10.1038/nn.3837.

[101] Min Lin, Qiang Chen, and Shuicheng Yan. "Network In Network." In: *CoRR* abs/1312.4400 (2013). http://arxiv.org/abs/1312.4400.

[102] Chen Liu, Hang Yan, Pushmeet Kohli, and Yasutaka Furukawa. "Multi-way Particle Swarm Fusion." In: *arXiv preprint arXiv:1612.01234* (2016).

[103] Ting Liu, Cory Jones, Mojtaba Seyedhosseini, and Tolga Tasdizen. "A modular hierarchical approach to 3D electron microscopy image segmentation." In: *J. Neurosci. Methods* 226 (2014), pp. 88–102. ISSN: 1872-678X. DOI: 10.1016/j.jneumeth.2014.01.022. URL: http://www.ncbi.nlm.nih.gov/pubmed/24491638.

[104] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." In: *Proc. CVPR* (2015), pp. 3431–3440.

[105] *LpMp. Solving LPs with convergent message passing*. 2017. URL: https://github.com/pawelswoboda/LP_MP (visited on 11/18/2017).

[106] Michael Maire, Pablo Arbelaez, Charless C. Fowlkes, and Jitendra Malik. "Using Contours to Detect and Localize Junctions in Natural Images." In: *CVPR*. 2008.

[107] D. Martin, C. Fowlkes, D. Tal, and J. Malik. "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics." In: *ICCV*. Vol. 2. 2001, pp. 416–423.

[108] Marina Meilă. "Comparing clusterings by the variation of information." In: *Learning theory and kernel machines*. Ed. by Bernhard Schölkopf and Manfred K. Warmuth. Vol. 2777. Lecture Notes in Computer Science. Springer, 2003, pp. 173–187.

[109] Fernand Meyer. "Watersheds on edge or node weighted graphs "par l'exemple"." In: *CoRR* abs/1303.1829 (2013). URL: http://arxiv.org/abs/1303.1829.

[110] Vincent Ng and Claire Cardie. "Improving Machine Learning Approaches to Coreference Resolution." In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 104–111. DOI: 10.3115/1073083.1073102. URL: http://dx.doi.org/10.3115/1073083.1073102.

[111] Juan Nunez-Iglesias, Ryan Kennedy, Toufiq Parag, Jianbo Shi, and Dmitri B Chklovskii. "Machine learning of hierarchical clustering to segment 2D and 3D images." In: *PloS ONE* 8.8 (2013), e71715.

[112] Xinghao Pan, Dimitris Papailiopoulos, Samet Oymak, Benjamin Recht, Kannan Ramchandran, and Michael I. Jordan. "Parallel Correlation Clustering on Big Graphs." In: *CoRR* abs/1507.05086 (2015). URL: http://arxiv.org/abs/1507.05086.

[113] C Pape, T Beier, P Li, V Jain, D. D. Brock, and A. Kreshuk. "Solving Large Multicut Problems for Connectomics via Domain Decomposition." In: *Bioimage Computing Workshop. ICCV, in press* (2017).

[114] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: *J. Mach. Learn. Res.* 12 (2011), pp. 2825–2830.

[115] QuantStack-Team. *Scientific computing in a polyglot world with xtensor*. 2017. URL: http://quantstack.net/c++/2017/05/30/polyglot-scientific-computing-with-xtensor.html (visited on 11/18/2017).

[116] William M. Rand. "Objective Criteria for the Evaluation of Clustering Methods." In: *J. of the American Statistical Association* 66.336 (1971), pp. 846–850.

[117] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In: *Proc. MICCAI* (2015), pp. 234–241.

[118] Carsten Rother, Vladimir Kolmogorov, Victor S. Lempitsky, and Martin Szummer. "Optimizing Binary MRFs via Extended Roof Duality." In: *CVPR*. 2007.

[119] Stephan Saalfeld, Albert Cardona, Volker Hartenstein, and Pavel Tomančák. "CATMAID: collaborative annotation toolkit for massive amounts of image data." In: *Bioinformatics* 25.15 (2009), pp. 1984–1986. DOI: 10.1093/bioinformatics/btp266. eprint: http://bioinformatics.oxfordjournals.org/content/25/15/1984.full.pdf+html. URL: http://bioinformatics.oxfordjournals.org/content/25/15/1984.abstract.

[120] Eldar Sadikov, Jayant Madhavan, Lu Wang, and Alon Halevy. "Clustering Query Refinements by User Intent." In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. Raleigh, North Carolina, USA: ACM, 2010, pp. 841–850. ISBN: 978-1-60558-799-8. DOI: 10.1145/1772690.1772776. URL: http://doi.acm.org/10.1145/1772690.1772776.

[121] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. "Evolution Strategies as a Scalable Alternative to Reinforcement Learning." In: *CoRR* abs/1703.03864 (2017). arXiv: 1703.03864. URL: http://arxiv.org/abs/1703.03864.

[122] Robert Sedgewick and Kevin Wayne. *Algorithms, 4th Edition*. Addison-Wesley, 2011, pp. I–XII, 1–955. ISBN: 978-0-321-57351-3.

[123] Jianbo Shi and Jitendra Malik. "Normalized Cuts and Image Segmentation." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 22.8 (Aug. 2000), pp. 888–905. ISSN: 0162-8828. DOI: 10.1109/34.868688. URL: http://dx.doi.org/10.1109/34.868688.

[124] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In: *CoRR* abs/1409.1556 (2014).

[125] Christoph Sommer, Christoph Straehle, Ullrich Kothe, and Fred A Hamprecht. "Ilastik: Interactive learning and segmentation toolkit." In: *Proc. ISBI* 1 (2011), pp. 230–233. DOI: 10.1109/ISBI.2011.5872394. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5872394.

[126] Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. "A Machine Learning Approach to Coreference Resolution of Noun Phrases." In: *Comput. Linguist.* 27.4 (Dec. 2001), pp. 521–544. ISSN: 0891-2017. URL: http://dl.acm.org/citation.cfm?id=972597.972602.

[127] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A simple way to prevent neural networks from overfitting." In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1929–1958.

[128] Benedikt Staffler, Manuel Berning, Kevin M Boergens, Anjali Gour, Patrick van der Smagt, and Moritz Helmstaedter. "SynEM, automated synapse detection for connectomics." In: *eLife* 6 (2017).

[129] Marijn F Stollenga, Wonmin Byeon, Marcus Liwicki, and Juergen Schmidhuber. "Parallel Multi-Dimensional LSTM, With Application to Fast Biomedical Volumetric Image Segmentation." In: *Adv. Neural Inf. Process Syst.* (2015), pp. 2980–2988.

[130] Christoph N. Straehle, Ullrich Koethe, Graham Knott, Kevin L. Briggman, Winfried Denk, and Fred A. Hamprecht. "Seeded watershed cut uncertainty estimators for guided interactive segmentation." In: *CVPR'12*. 2012, pp. 765–772.

[131] Paul Swoboda and Bjoern Andres. "A Message Passing Algorithm for the Minimum Cost Multicut Problem." In: *CVPR*. IEEE Computer Society, 2017, pp. 4990–4999.

[132] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In: *Proc. CVPR* (2015), pp. 1–9.

[133] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In: *arXiv preprint arXiv:1512.00567* (2015).

[134] Shin-Ya Takemura et al. "Synaptic circuits and their variations within different columns in the visual system of Drosophila." In: *Proc. Natl. Acad. Sci. USA* 112.44 (2015). ISSN: 1091-6490. DOI: 10.1073/pnas.1509820112. URL: http://www.pnas.org/lookup/doi/10.1073/pnas.1509820112$\backslash$nhttp://www.ncbi.nlm.nih.gov/pubmed/26483464.

[135] Siyu Tang, Mykhaylo Andriluka, Bjoern Andres, and Bernt Schiele. "Multiple People Tracking by Lifted Multicut and Person Re-Identification." In: *CVPR*. 2017.

[136] Zhuowen Tu. "Auto-context and its application to high-level vision tasks." In: *Proc. CVPR* (2008), pp. 1–8.

[137] Srinivas C Turaga, Kevin L Briggman, Moritz Helmstaedter, Winfried Denk, and H Sebastian Seung. "Maximin affinity learning of image segmentation." In: *Proc. NIPS* (2009), pp. 1865–1873.

[138] Srinivas C Turaga, Kevin L Briggman, Moritz Helmstaedter, Winfried Denk, and H Sebastian Seung. "Maximin affinity learning of image segmentation." In: *Proc. NIPS* (2009), pp. 1865–1873.

[139] Mustafa Gokhan Uzunbas, Chao Chen, and Dimitris Metaxas. "An efficient conditional random field approach for automatic and interactive neuron segmentation." In: *Med. Image Anal.* 27 (2016), pp. 31–44. ISSN: 13618423. DOI: 10.1016/j.media.2015.06.003. URL: http://dx.doi.org/10.1016/j.media.2015.06.003.

[140] Amelio Vazquez-Reina, Daniel Huang, Michael Gelbart, Jeff Lichtman, Eric Miller, and Hanspeter Pfister. "Segmentation Fusion for Connectomics." In: *ECCV*. IEEE. 2011, pp. 177–184.

[141] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. "Graph Cut Based Image Segmentation with Connectivity Priors." In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2008, pp. 1–8. ISBN: 978-1-4244-2242-5. DOI: 10.1109/cvpr.2008.4587440. URL: http://dx.doi.org/10.1109/cvpr.2008.4587440.

[142] T. Voice, M. Polukarov, and N. R. Jennings. "Coalition Structure Generation over Graphs." In: *Journal of Artificial Intelligence Research* 45 (2012), pp. 165–196.

[143] Adrian A. Wanner, Christel Genoud, Tafheem Masudi, Lea Siksou, and Rainer W. Friedrich. "Dense EM-based reconstruction of the interglomerular projectome in the zebrafish olfactory bulb." In: *Nat. Neurosci.* 19.6 (2016). Article, pp. 816–825. ISSN: 1097-6256. URL: http://dx.doi.org/10.1038/nn.4290.

[144] S Wolf, L Schott, U Köthe, and F. A. Hamprecht. "Learned Watershed: End-to-End Learning of Seeded Segmentation." In: *ICCV, in press* (2017).

[145] Xingwei Yang, Lakshman Prasad, and Longin Jan Latecki. "Affinity Learning with Diffusion on Tensor Product Graph." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 35.1 (2013), pp. 28–38. URL: http://dblp.uni-trier.de/db/journals/pami/pami35.html#YangPL13.

[146] Julian Yarkony, Alexander Ihler, and Charless C. Fowlkes. "Fast Planar Correlation Clustering for Image Segmentation." In: *ECCV*. Springer, 2012.

[147] Julian Yarkony, Thorsten Beier, Pierre Baldi, and Fred A. Hamprecht. "Parallel Multicut Segmentation via Dual Decomposition." In: *New Frontiers in Mining Complex Patterns - Third International Workshop, NFMCP 2014, Held in Conjunction with ECML-PKDD 2014, Nancy, France, September 19, 2014, Revised Selected Papers*. 2014, pp. 56–68. DOI: 10.1007/978-3-319-17876-9_4. URL: http://dx.doi.org/10.1007/978-3-319-17876-9_4.

[148] Christopher Zach. "Generalized Fusion Moves for Continuous Label Optimization." In: *Asian Conference on Computer Vision*. Springer. 2016, pp. 67–81.

[149] Anthony M. Zador, Joshua Dubnau, Hassana K. Oyibo, Huiqing Zhan, Gang Cao, and Ian D. Peikon. "Sequencing the Connectome." In: *PLOS Biology* 10.10 (Oct. 2012), e1001411+. DOI: 10.1371/journal.pbio.1001411. URL: http://dx.doi.org/10.1371/journal.pbio.1001411.

Colophon

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX: