Solving large Multicut problems for connectomics via domain decomposition

Constantin Pape[1,2], Thorsten Beier[1], Peter Li[3], Viren Jain[3], Davi D. Bock[2], and Anna Kreshuk[*1,2]

[1]HCI/IWR, University Heidelberg, [2]HHMI Janelia, [3]Google

Abstract

In this contribution we demonstrate how a Multicut-based segmentation pipeline can be scaled up to datasets of hundreds of Gigabytes in size. Such datasets are prevalent in connectomics, where neuron segmentation needs to be performed across very large electron microscopy image volumes. We show the advantages of a hierarchical block-wise scheme over local stitching strategies and evaluate the performance of different Multicut solvers for the segmentation of the blocks in the hierarchy. We validate the accuracy of our algorithm on a small fully annotated dataset ($5 \times 5 \times 5$ µm) and demonstrate no significant loss in segmentation quality compared to solving the Multicut problem globally. We evaluate the scalability of the algorithm on a $95 \times 60 \times 60$ µm image volume and show that solving the Multicut problem is no longer the bottleneck of the segmentation pipeline.

1. Introduction

Connectomics is a domain of neuroscience which strives to understand structure-function relations in neural circuits from the directed graph of neural connections. The graph itself – the wiring diagram of a nervous system – is usually reconstructed from very large stacks of neural tissue images acquired by electron microscopy (EM) [12, 16]. Reconstruction of the graph consists of two major sub-problems: tracing of neurons through the image stack and detecting synapses which connect the neurons.

Unlike fluorescent labeling methods which typically reveal a sparse subset of neurons, the heavy metal stains used in EM label all cell membranes in a piece of brain tissue. Consequently, neuron tracing or segmentation have to be based on boundary evidence and have to be performed on images of very high resolution. At the same time, neural cells spread over very large volumes; hundreds of Gigabytes of images have to be analyzed to reconstruct a graph which

would be relevant for biological analysis. Figure 1 shows one image of such a dataset. Note how the segmented neurons on the left pass through the whole dataset, while the diameter of some processes in the inset on the right is so small they can only be traced at full resolution. Recently, the first complete brain of an adult fruit-fly has been imaged [39], the dataset measures 110 Terabytes. The on-going efforts to image the brains of small vertebrates are anticipated to produce datasets of equal or greater size, for example: [17, 28, 25].

Until very recently, most of the effort of the computer vision community has concentrated on solving the automated reconstruction problem correctly at small scale. Since the reconstruction accuracy has not been sufficiently good for direct biological analysis, neuroscientists have resorted to collective manual tracing [35, 11] or manual proof-reading of automatically generated segmentations [22, 20, 31]. The hurdle of fully automated segmentation, however, does not seem insurmountable anymore: Lee et al. [24] have demonstrated better-than-human performance on the SNEMI3D [4] challenge dataset, the gap to human performance on ISBI2012 [5] is shrinking, automatic segmentations of the new CREMI challenge [15] are also of extremely good quality. The question of scaling these methods up from tiny challenge datasets to Terabyte-sized real data without substantial loss in accuracy is now starting to be addressed [33, 32, 27]. So far, the scaling has been achieved by performing the segmentation block-wise and merging the sub-block solutions based on local evidence. In this contribution, we propose a different approach which optimizes a global objective function and allows to over-rule the local sub-block solutions if evidence at larger scale suggests it would lead to a better overall segmentation.

Our approach builds on the neuron segmentation pipeline of [10], which is currently state-of-the-art or close to it on all three popular connectomics challenges. The pipeline is based on the Multicut graph partitioning problem. While this problem is NP-hard [13], practical approximate solvers have recently been introduced. We extend the pipeline to perform the global superpixel graph construction

---

out-of-core and in parallel (subsection 3.2) and propose a hierarchical block-wise approximate solver for the Multicut problem (subsection 3.3). While the resulting solution is no longer globally optimal, we demonstrate sufficient accuracy on three datasets of the CREMI challenge (subsection 4.3) and excellent scaling behavior on a large cutout from the whole fruit-fly brain dataset of [39] (subsection 4.4).

While we focus on neuron reconstruction, the methods introduced in this paper are potentially relevant for a wider scope of problems that involve large-scale graph partitioning.

2. Related Work

Most neuron segmentation pipelines in use today [30, 14, 38] follow the same sequence of steps. First, a membrane probability map is computed by a convolutional neural network. Based on this map, over-segmentation of the volume into fragments, also known as superpixels or supervoxels, is performed to reduce the scale of the problem and to leverage more context information than available directly on pixel level. The region adjacency graph of the fragments is then constructed and the segmentation problem is solved as partitioning or clustering on that graph.

The existing approaches to solving the neuron segmentation problem at scale are based on the popular GALA algorithm for agglomerative clustering with learned edge weights ([30], [29]) . The volume is partitioned into blocks, potentially with overlap, and GALA is applied to each block independently and in parallel. The resulting block solutions are then stitched to form the global volume segmentation. In [33] this is achieved by stitching segments across blocks according to largest overlap, while [32] employs a similar strategy, but includes additional heuristics to prevent false merges. In [27] the blocks are stitched by re-applying the algorithm on the overlap of the blocks.

In contrast to all of the above, we propose to use Multicut for the segmentation of the blocks, as it has been shown to be superior to GALA in neuron segmentation challenges [10]. Also, we formulate the graph partitioning problem on the complete dataset and solve it hierarchically from the bottom up.

The Multicut has been applied in computer vision problems frequently [19, 2]. Andres et al. first applied it to connectomics in [3]. They also show that the Multicut can be formulated as an integer linear program (ILP) and introduce a solver based on the cutting planes approach. This solver is guaranteed to converge to the globally optimal solution. However, the problem being NP-hard, this approach is not suitable for inferring large problems, which call for an approximate solver.

Beier et al. [8] introduce two such solvers: the first one is based on the fusion moves algorithm. It iteratively generates diverse proposal solutions and merges them using a base Multicut solver until no further improvement of solutions can be achieved. The other is based on greedy additive edge contraction. The Kerninghan-Lin graph partitioning algorithm [21] can also be applied to the Multicut problem. We analyze the trade-offs of these algorithms for the connectomics use case in subsection 4.2 in more detail. Further approximate solvers have been introduced in [9, 6, 36, 1].

Block-wise solvers for similar problems have also been proposed in [18, 23]. In fact, our approach could be viewed as a generalization of [23] to work on real data: their solver is developed for structured loss minimization and has only been applied to very small blocks with very large overlap.

3. Methods

In the following we briefly describe the setup of the Multicut problem for the segmentation of neurons in EM datasets, as well as the block-wise solver we propose to solve this problem at sufficiently large scale.

3.1. Multicut

The Multicut [13] is a graph partitioning problem, where the number of partition elements is not known in advance. It is defined by a graph $G = (V, E)$ and costs $\mathbf{c}$ associated with the edges E. The objective of the problem is to partition the nodes V into clusters, minimizing the sum of costs associated with edges between the clusters (the "cut" edges). As Andres et al. showed in [3], it can be formulated as an optimization problem by associating binary variables y_e with edges:

$$\tilde{\mathbf{y}} = \arg\min_{\mathbf{y}} \sum_e y_e \cdot c_e \qquad \text{s.t.} \qquad (1)$$

$$\forall\, Y \in \text{cycles}(G) : \forall e \in Y : x_e \leq \sum_{\hat{e} \in Y \setminus e} x_{\hat{e}}. \qquad (2)$$

Here the first equation expresses the objective and the second defines the Multicut constraints necessary to guarantee a consistent solution without "dangling edges". In this formulation, edges with a positive weight are attractive (they vote for joining the associated nodes) and edges with a negative weight are repulsive. In general, finding the optimal solution for Equation 1 is NP-hard and even the existing approximate solvers can not directly handle a graph sufficiently large for a connectomics experiment. Our hierarchical approach introduced in subsection 3.3 allows to limit application of the solvers to much smaller graphs, corresponding to sub-blocks of the volume or, at higher levels of the hierarchy, to the reduced problem on the stitched sub-blocks.

3.2. Constructing Multicut Problem

The Multicut problem is set up similar to [10]. First, we use a deep convolutional neural network to predict the
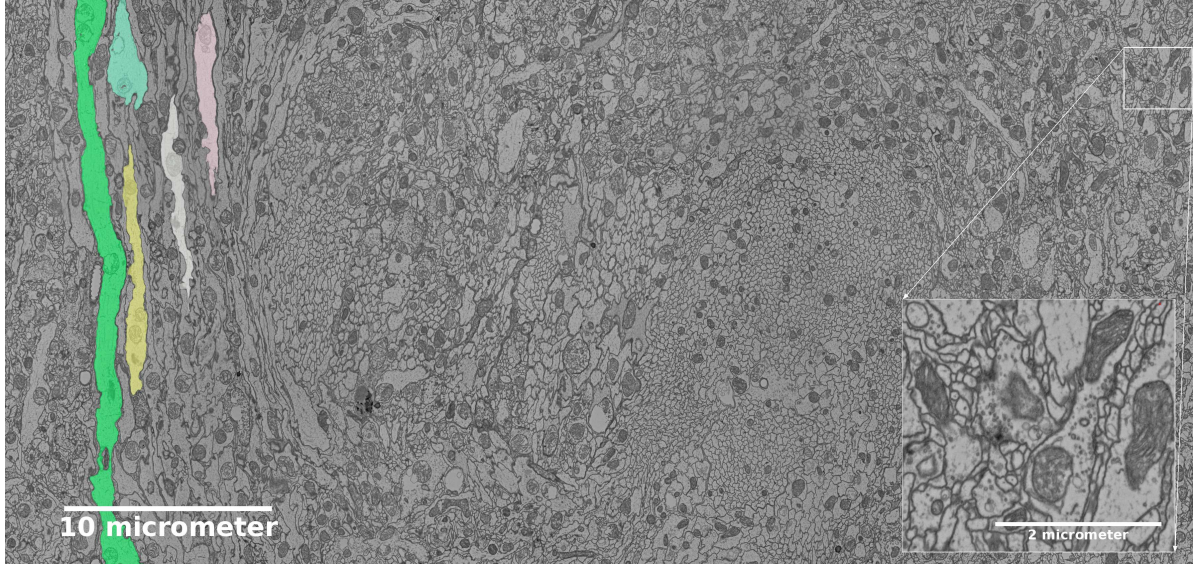
Figure 1: Part of an image from the whole fruit-fly brain dataset [39]. Highlighted neurons were segmented automatically with the proposed block-wise algorithm (some post-processing was applied to merge small falsely split fragments). The inset on the right shows some of the smaller process as well as a synaptic contact site, roughly in the middle.

affinity of a voxel to its neighbors. The network architecture used here is based on multiple 2-D Inception / GoogleLeNet towers [37], which share weights and are applied to neighboring slices before being fused for the output classification layer. It is trained to also predict longer range affinities in order to better resolve local ambiguities, similar to [24]. We obtain boundary probabilities by averaging over the local x- and y-channel of the predicted affinity map. Based on these probabilities, we generate a fragmentation with running a watershed on the distance transform of the boundary. The fragmentations are computed separately for each slice and then stacked along the (anisotropic) z-axis. A region adjacency graph is constructed for the fragments.

Here, we differentiate between two types of graph edges: the xy-edges that connect nodes (fragments) in the same z-slice and z-edges that connect nodes in different z-slices. In addition, slices that are affected by serious imaging defects or completely missing are excluded from the graph. We detect such slices automatically based on the observation that the number of fragments in those slices is very different from the number of fragments in slices not affected by a defect. To exclude them, we remove all edges that connect nodes in the defected slice z with nodes in the adjacent slices $z + 1$ and $z - 1$. Then we connect nodes in $z + 1$ and $z - 1$ via the so-called *skip* edges. This procedure is generalized to defects that occur in several consecutive slices.

Next we compute features for all edge types. These are based on appearance of the raw data and probability maps accumulated over the fragment boundaries and on the raw data appearance accumulated over the fragments. The fea-

tures are used to predict with Random Forests[1] (separately for xy-, z- and skip-edges) if an edge should be present in the final segmentation. The resulting probabilities **p** are then weighted according to the edge type and length and transformed to edge costs for the Multicut problem via:

$$c_e = \log \frac{1 - p_e}{p_e} + \log \frac{1 - \beta}{\beta}, \tag{3}$$

where the boundary bias β allows to tune the degree of over-segmentation.

All computations are implemented in parallel using multi-threading and out-of-core by leveraging an HDF5 data backend. The compute times for all the steps above are listed in the section 4.

3.3. Block-wise Multicut

Our overall goal is to solve the Multicut for very large problems such as the segmentation of connectomics datasets. We want to achieve this by minimizing a global objective because we postulate this strategy to be superior to the local approaches pursued in previous work (cf. section 2, subsection 4.3). To this end, we divide the global problem into sub-problems feasible for existing solvers and use the sub-solutions to reduce the global problem. The *block-wise Multicut solver* achieves this by extracting sub-problems from a covering of the volume with overlapping blocks. It then solves these sub-problems in parallel and reduces the global problem by merging the nodes which

---

[1]Trained on separate data with ground-truth annotations.

are unambiguously merged in the solutions of the sub-problems. These steps can be iterated until the reduced problem is small enough to be inferred with one of the approximate solvers. For a schematic overview and an example, see algorithm 1 and Figure 2.

In step 1 (Figure 2, algorithm 1) we construct the global problem (Figure 2, (b)) from the fragmentation (a) according to subsection 3.2. In step 2, we construct sub-problems from blocks as shown in (c).

The assignment of nodes to sub-problems is soft, i.e. a sub-problem contains all nodes, together with associated edges and costs, for which the corresponding fragments overlap with the sub-block. The example sub-graphs are shown in (c).

We solve the sub-problems in parallel (step 3) and retain whether the edges in the sub-graphs should be merged (dotted edges in (d)). In step 4, these sub-results are projected to the global graph (e). The state of an edge in this projection depends on its sub-result and its type, illustrated in (f):

- *Connecting edges* (green lines), for which the nodes belong to different sub-problems, are not merged.

- *Shared edges* (red lines), for which both nodes belong to multiple sub-problems, are merged if they are merged in all sub-problem solutions.

- *Unique edges* (blue lines), for which both nodes belong to a single sub-problem, are merged if they are merged in the sub-problem solution.

After projection, the graph is reduced according to the merge decisions and costs for the new edges are computed from the old costs by summation (step 4, (g)). This process can be iterated with increasing block sizes until the reduced problem is feasible (step 5). The result (h) is then projected (step 6) to the initial global problem to obtain a segmentation (i).

Note that we take a conservative approach, not merging connecting edges and only merging shared edges if all their sub-results vote for a merge, because merge decisions that are taken at a given hierarchy level cannot be undone later. Due to this fact, the solution of the block-wise Multicut is approximate, even if an exact solver is used for the sub-problems.

4. Results and Discussion

4.1. Setup

The accuracy and runtime of the block-wise algorithm was evaluated on the 3 training sets of the CREMI segmentation challenge. Each of these datasets consists of a $8{\times}12{\times}12$ μm block ($200{\times}3000{\times}3000$ pixels), including a $5{\times}5{\times}5$ μm ($125{\times}1250{\times}1250$ pixels) crop with pixel-wise ground-truth annotations. The CNN was pre-trained on data

Data: globalProblem, blockShape, overlap, nLevels
Result: nodeResult
1 problem = globalProblem;
 for *l in nLevels* **do**
2   subProblems = extractSubproblems(problem, blockShape, overlap);
3   subSolutions = solveMulticutsInParallel(subProblems);
4   problem = reduceProblem(problem, subProblems);
     blockShape *= 2;
 end
5 nodeResult = solveMulticut(problem);
6 nodeResult = projectToGlobalGraph(nodeResult, globalProblem);

Algorithm 1: Schematic overview of the block-wise Multicut solver. A given problem is reduced by solving sub-problems and merging the graph accordingly for a given number of iterations. The reduced problem is then solved and projected back to the global solution. Line numbers correspond to the numbered arrows in Figure 2.

from the ISBI-challenge, but only trained on the Sample A dataset. Note, that we perform the evaluation for each dataset individually and train the edge cost Random Forests on the two other datasets to ensure a complete train/test separation.

We compare the following algorithms for solving the Multicut problems in our pipeline: ILP-based solver *ilp* [3], the Kernighan-Lin algorithm *kl* [21], the greedy additive edge contraction *gaec* [8], and the fusion moves solver *fm-x*, where *x* denotes the base solver used in the fusion move [8]. The approximate solvers that profit from warm-starting are initialized with the solutions of cheaper solvers. That is, we warm-start kl with the solutions of gaec and fm-ilp, fm-kl with the solution of kl.

Next, we validate the proposed block-wise solver on the same three datasets, using the ground-truth crops for evaluation. We also compare to baseline stitching approaches and compare different parameter settings.

Finally, we evaluate the scalability of our approach with different Multicut solvers on a large cutout from the whole adult fruit-fly brain dataset, the so-called Sample D. The cutout covers a $95{\times}60{\times}60$ μm block which includes several regions of interest from the neuroscience point of view. The CREMI challenge datasets are contained in this block, but no other parts of the block have ground-truth annotations.

All experiments were performed on a single workstation with 20 physical cores and 256 GB of RAM.
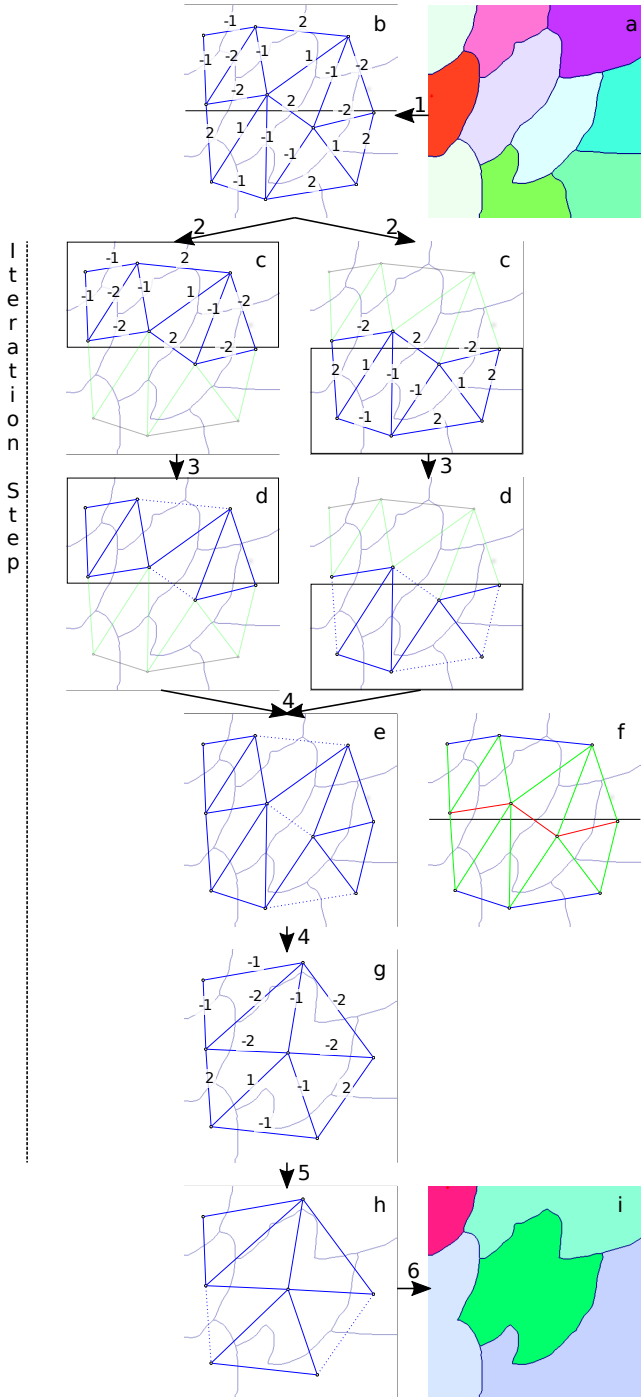
Figure 2: Block-wise Multicut example: from the global problem (a, b) sub-problems are extracted from blocks (c) and solved in parallel (d). The sub-solutions are projected to the global problem (e, g), blue and red edges are merged if needed (f). After a fixed number of iterations, the reduced problem is solved (h) and projected to a segmentation (i).

4.2. Solver performance at smaller scale

We apply the solvers kl, fm-kl and fm-ilp to the three problems created from the large CREMI blocks. See Figure 3 for the anytime performance on the different samples. For all three samples, the kl-solver converges after a few minutes. The fusion-move solvers can improve on its results, but converge much slower. For Sample A fm-ilp shows a slightly better anytime performance, while for Sample B and C fm-kl performs better. Both solvers do not converge in the set time limit of one hour.

These experiments show that the kl-solver is well suited for inference of large problems. It can be combined with fusion moves to converge to even lower energies. In contrast, the ilp-solver is not suited for inference of such large problems, as it yields inferior results as base solver for the fusion move.

4.3. Segmentation accuracy evaluation

Since ground-truth annotations are only available for the small datasets, we crop the segmentations obtained for larger datasets to the annotated regions and perform validation on those. To avoid worse scores from processes that are joined outside the annotated region we run connected components on the cropped segmentations. We report the same score as used in the CREMI challenge – the mean of Rand-F1-score, VI-split and VI-merge [5, 26]. For this score, lower values correspond to better performance. Note, that with the block size we chose for experiments in Table 1, the validation region is stitched from 27 sub-blocks.

The first three rows of Table 1 contain the comparison of the proposed block-wise algorithm with the global Multicut results. To ensure the correct baseline, we compare both to the Multicut solution for the small dataset (solved with fm-ilp) and to the cropped Multicut solution for the larger dataset (solved with fm-kl). In both cases the global Multicut is run until convergence.

Besides the global solution, we compare to other popular block stitching approaches. Starting from the same sub-block solutions as the block-wise Multicut, we perform stitching by largest overlap as in [33, 32] and, as an alternative, greedily stitch the sub-blocks by merging all edges between sub-block solutions (connecting edges as described in subsection 3.3) that are attractive. We also compare to an approach where we resolve the Multicut for all problems arising from the block-overlaps and stitch based on these solutions, similar to the approach in [27].

The results in Table 1 show much better scores for the block-wise Multicut as compared to other stitching approaches. In fact, it achieves scores slightly better than that of the global Multicut solver, though we believe this effect not to be systematic. Its runtime is over an order of magnitude smaller than the global solver. It is also interesting to note that the results cropped from larger datasets have bet-
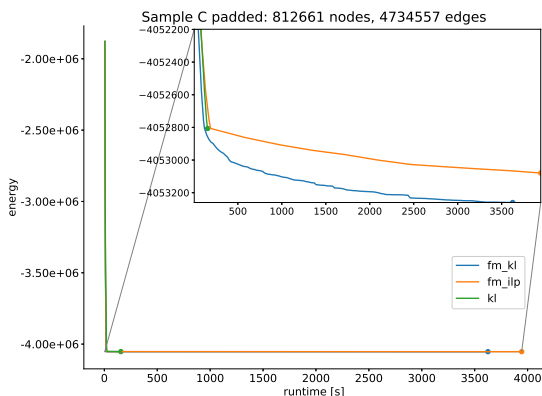
ter scores than the results on the small annotated datasets. This effect can probably be explained by additional context
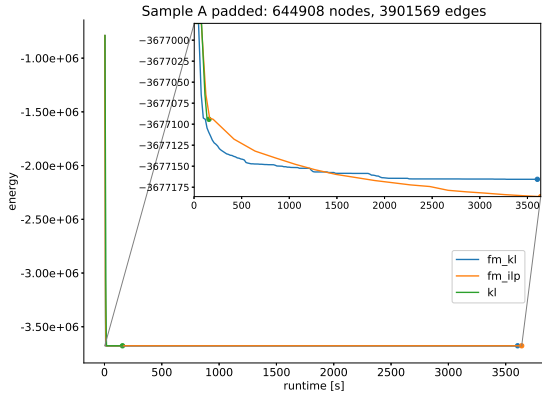
(a) Sample A
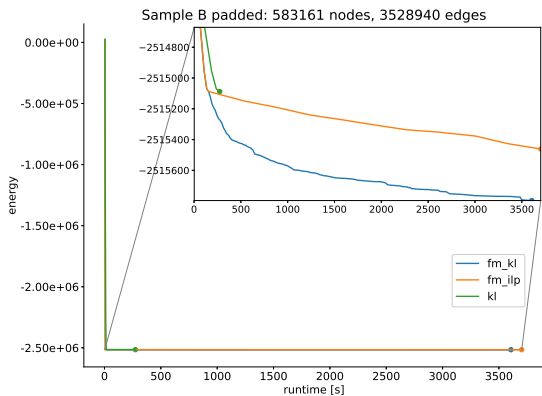
(b) Sample B

(c) Sample C

Figure 3: Anytime performances of different solvers for the CREMI datasets with time limit of 1 hour.

at the edges of the dataset which is available for the larger volumes. In addition, we compare the energies Equation 1 of the solutions, which are lowest for the global Multicut solution, but not significantly different from our block-wise solver. For Sample C, stitching based on the local Multicut results yields the lowest energy. This is surprising and we do not have a satisfactory explanation yet.

Note that the best scores reported here are not competitive with the current state-of-the-art in the CREMI challenge leaderboard[2], although the top challenge results were obtained with the Multicut pipeline on which our approach is based. Several factors may contribute to this performance gap: first the leading submission (MALAMC) is based on a network that is itself placed fairly high in the leaderboard (MALA). Second, the MALA network, as well as the leading submission, were generated on realigned volumes which make 3D continuity of the processes much more reliable. Finally, the network used here, as well as the edge cost Random Forest, were only trained on parts of the ground-truth.

Now that the advantage of using block-wise Multicut over local stitching has been established, we evaluate the influence of the algorithm parameters on accuracy and runtime. In Table 2a, we solve the problem with different numbers of levels in the hierarchy. One level corresponds to running the the algorithm on sub-blocks, stitching the results and re-solving the reduced problem. Increasing the number of levels reduces the problem size for each individual Multicut, but also makes the overall problem less similar to the global Multicut. Still, as we can observe in Table 2a, using 3 levels does not significantly affect the accuracy for datasets A and C.

Table 2b illustrates the algorithm performance for different shapes of the sub-blocks. Setting this parameter to $50 \times 512 \times 512$ seems to provide the best trade-off between context and runtime. However, if smaller block size needs to be selected due to limitations in computing resources, the decline in accuracy is still tolerable.

The overlap between sub-blocks is analyzed in Table 2c. Larger overlaps increase the problem size and thus the runtime of the algorithm, but in our experiments they do not to provide a significant accuracy boost.

Finally, Table 2d analyzes the performance of different solvers in the sub- and reduced problems. Here, as in Figure 3, we can appreciate just how fast the Kernighan-Lin algorithm is, without a big loss in the segmentation score. Based on these results a viable segmentation strategy can be constructed, where the problem is first solved with the kl-solver, coarse-grained analysis is performed on the segmentation and the blocks with most interesting structures are re-segmented with a more accurate solver.

---

[2]https://cremi.org/

	Sample A			Sample B			Sample C		
Solver	Score	Time	Energy	Score	Time	Energy	Score	Time	Energy
Multicut (small)	0.3200	-	-	0.7432	-	-	0.7197	-	-
Multicut	0.2690	1874	**-537332**	0.6921	9066	**-2117459**	0.6893	13580	-90414
Block-wise Multicut	**0.2653**	175	-536943	**0.6811**	702	-2103525	**0.6874**	483	-90335
Overlap Stitching	0.4319	462	-495487	0.8509	572	-2000648	0.7904	690	-84919
Greedy Stitching	0.4329	77	-527632	0.9479	184	-1967663	1.5115	235	-35957
Multicut Stitching	0.3748	104	-533473	0.8242	227	-2110191	0.6951	279	**-90621**

Table 1: Comparison of different solvers on the three large problems extracted from CREMI. The global Multicut is inferred with fm-kl (large problem) / fm-ilp (small problem). The block-wise Multicut is run with a single hierarchy level. We use a block shape of (50, 512, 512) and a block-overlap of (5, 50, 50). We use the solver fm-ilp for the sub-problems and fm-kl for the reduced global problem. The baseline solvers use the same blocking. All runtimes are reported in seconds and exclude the problem construction, which is the same in every case.

	Sample A			Sample B			Sample C		
Level	Score	Time	Energy	Score	Time	Energy	Score	Time	Energy
1	0.2653	175	**-536943**	**0.6811**	702	**-2103525**	0.6874	483	**-90335**
2	0.2636	149	-536925	0.7218	**288**	-2103023	0.6905	390	-90310
3	**0.2635**	**148**	-536925	0.7315	303	-2102854	0.6894	**376**	-90303

(a) Block-wise Multicut results with increasing number of solver levels.

Block	Sample A			Sample B			Sample C		
Shape	Score	Time	Energy	Score	Time	Energy	Score	Time	Energy
25, 256, 256	0.2795	363	-536929	0.6872	2519	**-2103877**	0.7013	1190	**-90351**
50, 512, 512	**0.2653**	**175**	**-536943**	**0.6811**	**702**	-2103525	**0.6874**	**483**	-90335
100, 1024, 1024	0.2682	299	-536938	0.7322	1387	-2103116	0.6940	1891	-90341

(b) Block-wise Multicut results with increasing block shapes.

Block	Sample A			Sample B			Sample C		
Overlap	Score	Time	Energy	Score	Time	Energy	Score	Time	Energy
1, 1, 1	0.2748	180	-536855	0.6996	**354**	-2102547	0.6951	**369**	-90262
2, 20, 20	0.2674	**172**	-536899	**0.6546**	436	-2103286	**0.6780**	575	-90297
5, 50, 50	**0.2653**	175	-536943	0.6811	702	-2103525	0.6874	483	-90335
10, 100, 100	0.2686	241	**-536965**	0.6817	737	**-2103878**	0.6907	742	**-90355**

(c) Block-wise Multicut results with increasing block overlaps.

Solver	Sample A			Sample B			Sample C		
Sub / Reduced	Score	Time	Energy	Score	Time	Energy	Score	Time	Energy
kl / kl	0.2682	**12**	-536928	**0.6546**	**16**	-2103049	0.6879	**20**	-90323
fm-kl / kl	0.2692	28	-536939	0.6819	42	-2103314	**0.6869**	61	-90331
fm-ilp / fm-kl	0.2653	175	-536943	0.6811	702	-2103525	0.6874	483	**-90335**
ilp / fm-ilp	**0.2585**	824	**-536951**	0.6693	2265	**-2103947**	-	-	-

(d) Block-wise Multicut results with different solvers. Inference with ilp / fm-kl did not converge for one of the sub-blocks.

Table 2: Lesion study for the block-wise Multicut solver. In each table, only a single parameter is varied, the others are set to the values as in Table 1. The row corresponding to the result of Table 1 is marked italic. Times are reported in seconds.

4.4. Solver performance on a larger scale

Here we report the results from running the block-wise algorithm on a 150 GB image volume (Sample D). Since no ground-truth annotations are available except for the small cutout used for our smaller scale experiments, we can only compare the energies achieved by different solvers. Similar to Table 2d, Table 3 shows that inferring with the cheaper solver combination yields comparable energies, while being significantly faster. It has to be noted that lower energies do not directly translate to lower segmentation scores (see, for example, the results on Sample A in Table 1). While the relationship is true in principle, a lot of noise is introduced by the projection to superpixels and especially by the errors of the classifier which estimates the energies of individual edges.

For the remaining parts of the pipeline the runtimes are as follows: generating fragments – 5 hours, graph construction and feature extraction – 27 hours, saving of the segmentation result to disk – 2.5 hours. We don't have an exact estimate for inference of probability maps, but this step was the clear bottleneck, as the network used here is not fully convolutional and hence inefficient in inference.

Level	Subsolver	Inference	Total	Energy
1	1.07	-	-	-
2	0.13	10.34	11.54	-151057.2
3	0.15	10.67	12.03	-150930.2
4	0.28	10.22	11.86	-150871.1

(a) Results with solvers fm-ilp / fm-kl.

Level	Subsolver	Inference	Total	Energy
1	0.69	22.80	23.50	**-151239.2**
2	0.11	2.34	3.14	-151055.7
3	0.08	0.65	1.52	-150931.3
4	0.13	0.41	**1.41**	-150873.3

(b) Results with solvers kl / kl.

Table 3: Results of the block-wise Multicut solver on the Sample D problem. We compare two different solver combinations which are both inferred up to hierarchy level 4 and report the time it took for solving the sub-problems (Subsolver), the reduced problem (Inference) and total runtime (excluding problem construction) as well as the global energy. All times are reported in hours. We use initial block-shape (50, 512, 512) and block overlaps (2, 20, 20). For fm-ilp / fm-kl reduced inference for level 1 was not feasible. For this solver combination reduced inference was performed with a time-limit of 10 hours.

5. Conclusion

We have introduced a hierarchical block-wise solver for the Multicut problem and, by evaluation on a neuron segmentation challenge as well as on a large experimental dataset, demonstrated excellent scalability without a significant loss in performance. Solving the Multicut can no longer be considered the bottleneck of the segmentation pipeline of [10].

Meirovitch et al. in [27] introduce a coarse/fine segmentation approach, which uses a very fast algorithm to quickly scan over the volume for interesting structures and then performs a more accurate segmentation at the regions of interest. A similar scheme could be adapted to the pipeline we propose, running the Kernighan-Lin algorithm first – it can segment a 150 GB volume in just 1.4 hours – and then fine-tuning with a fusion moves solver or the more expensive lifted Multicut problem [7, 10] on a selection of blocks.

The core idea of our algorithm – gradually reducing the global graph partitioning problem instead of greedily stitching the sub-graph partitionings – is not restricted to the Multicut problem. It can also be applied to agglomerative clustering, although, considering the performance of the Kernighan-Lin algorithm, it is not obvious that a significant speedup can be gained this way.

To further improve the scalability and performance of our segmentation pipeline we plan to upgrade our CNN to a fully-convolutional architecture based on a U-Net [34], aiming for more efficient inference and higher quality of probability maps. Better membrane probability maps not only create better over-segmentations, but also make for more reliable features. This will allow to compute less features for the edge cost Random Forest without loss of accuracy, thus addressing the next big bottleneck of the Multicut pipeline. Should the block-wise Multicut solver at some point become the bottleneck again, inference of the reduced problem can be sped up by using a parallel implementation of the Kernighan-Lin solver.

Our hierarchical block-wise approach also opens the door to incorporation of higher-level priors: while biologically improbable segments are not distinguishable on the level of individual fragments, they become much easier to find on the sub-block level.

6. Acknowledgments

References

[1] A. Alush and J. Goldberger. Ensemble segmentation using efficient integer linear programming. *IEEE transactions on pattern analysis and machine intelligence*, 34(10):1966–1977, 2012.

[2] B. Andres, J. H. Kappes, T. Beier, U. Kothe, and F. A. Hamprecht. Probabilistic image segmentation with closedness constraints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2611–2618. IEEE, 2011.

[3] B. Andres, T. Kroeger, K. L. Briggman, W. Denk, N. Korogod, G. Knott, U. Koethe, and F. A. Hamprecht. Globally optimal closed-surface segmentation for connectomics. In *European Conference on Computer Vision*, pages 778–791. Springer, 2012.

[4] I. Arganda-Carreras, S. H. Seung, A. Vishwanathan, and D. R. Berger. SNEMI 3D: 3D Segmentation of Neurites in EM Images, 2013.

[5] I. Arganda-Carreras, S. C. Turaga, D. R. Berger, D. Cireşan, A. Giusti, L. M. Gambardella, J. Schmidhuber, D. Laptev, S. Dwivedi, J. M. Buhmann, et al. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in neuroanatomy*, 9, 2015.

[6] S. Bagon and M. Galun. Optimizing large scale correlation clustering. *Electr. Prepr*, 2011.

[7] T. Beier, B. Andres, U. Köthe, and F. A. Hamprecht. An efficient fusion move algorithm for the minimum cost lifted multicut problem. In *European Conference on Computer Vision*, pages 715–730. Springer, 2016.

[8] T. Beier, F. A. Hamprecht, and J. H. Kappes. Fusion moves for correlation clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3507–3516, 2015.

[9] T. Beier, T. Kroeger, J. H. Kappes, U. Kothe, and F. A. Hamprecht. Cut, glue & cut: A fast, approximate solver for multicut partitioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 73–80, 2014.

[10] T. Beier, C. Pape, N. Rahaman, T. Prange, S. Berg, D. D. Bock, A. Cardona, G. W. Knott, S. M. Plaza, L. K. Scheffer, et al. Multicut brings automated neurite segmentation closer to human performance. *Nature Methods*, 14(2):101–102, 2017.

[11] K. M. Boergens, M. Berning, T. Bocklisch, D. Bräunlein, F. Drawitsch, J. Frohnhofen, T. Herold, P. Otto, N. Rzepka, T. Werkmeister, et al. webknossos: efficient online 3d data annotation for connectomics. *Nature Methods*, 14(7):691–694, 2017.

[12] K. L. Briggman and D. D. Bock. Volume electron microscopy for neuronal circuit reconstruction. *Current opinion in neurobiology*, 22(1):154–161, 2012.

[13] S. Chopra and M. R. Rao. The partition problem. *Mathematical Programming*, 59(1-3):87–115, 1993.

[14] J. Funke, B. Andres, F. A. Hamprecht, A. Cardona, and M. Cook. Efficient automatic 3d-reconstruction of branching neurons from em data. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1004–1011. IEEE, 2012.

[15] J. Funke, S. Saalfeld, D. Bock, S. Turaga, and E. Perlman. CREMI - MICCAI Challenge on Circuit Reconstruction from Electron Microscopy Images., 2016.

[16] M. Helmstaedter, K. L. Briggman, S. C. Turaga, V. Jain, H. S. Seung, and W. Denk. Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature*, 500(7461):168–174, 2013.

[17] D. G. C. Hildebrand, M. Cicconet, R. M. Torres, W. Choi, T. M. Quan, J. Moon, A. W. Wetzel, A. S. Champion, B. J. Graham, O. Randlett, et al. Whole-brain serial-section electron microscopy in larval zebrafish. *bioRxiv*, page 134882, 2017.

[18] K. Jung, P. Kohli, and D. Shah. Local rules for global map: When do they work ? In *Advances in Neural Information Processing Systems 22*, pages 871–879. Curran Associates, Inc., 2009.

[19] J. Kappes, M. Speth, B. Andres, G. Reinelt, and C. Schn. Globally optimal image partitioning by multicuts. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 31–44. Springer, 2011.

[20] N. Kasthuri, K. J. Hayworth, D. R. Berger, R. L. Schalek, J. A. Conchello, S. Knowles-Barley, D. Lee, A. Vázquez-Reina, V. Kaynig, T. R. Jones, et al. Saturated reconstruction of a volume of neocortex. *Cell*, 162(3):648–661, 2015.

[21] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970.

[22] J. S. Kim, M. J. Greene, A. Zlateski, K. Lee, M. Richardson, S. C. Turaga, M. Purcaro, M. Balkam, A. Robinson, B. F. Behabadi, et al. Space-time wiring specificity supports direction selectivity in the retina. *Nature*, 509(7500):331, 2014.

[23] T. Kroeger, S. Mikula, W. Denk, U. Koethe, and F. A. Hamprecht. Learning to segment neurons with non-local quality measures. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 419–427. Springer, 2013.

[24] K. Lee, J. Zung, P. Li, V. Jain, and H. S. Seung. Superhuman accuracy on the snemi3d connectomics challenge. *arXiv preprint arXiv:1706.00120*, 2017.

[25] W. Lee, V. Bonin, M. Reed, B. J. Graham, G. Hood, K. Glattfelder, and R. C. Reid. Anatomy and function of an excitatory network in the visual cortex. *Nature*, 532(7599):370–374, 2016.

[26] M. Meilă. Comparing clusteringsan information based distance. *Journal of multivariate analysis*, 98(5):873–895, 2007.

[27] Y. Meirovitch, A. Matveev, H. Saribekyan, D. Budden, D. Rolnick, G. Odor, S. K.-B. T. R. Jones, H. Pfister, J. W. Lichtman, and N. Shavit. A multi-pass approach to large-scale connectomics. *arXiv preprint arXiv:1612.02120*, 2016.

[28] J. L. Morgan, D. R. Berger, A. W. Wetzel, and J. W. Lichtman. The fuzzy logic of network connectivity in mouse visual thalamus. *Cell*, 165(1):192–206, 2016.

[29] J. Nunez-Iglesias, R. Kennedy, T. Parag, J. Shi, and D. B. Chklovskii. Machine learning of hierarchical clustering to segment 2d and 3d images. *PloS one*, 8(8):e71715, 2013.

[30] J. Nunez-Iglesias, R. Kennedy, S. M. Plaza, A. Chakraborty, and W. T. Katz. Graph-based active learning of agglomeration (gala): a python library to segment 2d and 3d neuroimages. *Frontiers in neuroinformatics*, 8:34, 2014.

[31] S. M. Plaza. Focused proofreading: efficiently extracting connectomes from segmented em images. *arXiv preprint arXiv:1409.1199*, 2014.

[32] S. M. Plaza and S. E. Berg. Large-scale electron microscopy image segmentation in spark. *arXiv preprint arXiv:1604.00385*, 2016.

[33] W. R. G. Roncal, D. M. Kleissas, J. T. Vogelstein, P. Manavalan, K. Lillaney, M. Pekala, R. Burns, R. J. Vogelstein, C. E. Priebe, M. A. Chevillet, et al. An automated images-to-graphs framework for high resolution connectomics. *Frontiers in neuroinformatics*, 9, 2015.

[34] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[35] S. Saalfeld, A. Cardona, V. Hartenstein, and P. Tomančák. Catmaid: collaborative annotation toolkit for massive amounts of image data. *Bioinformatics*, 25(15):1984–1986, 2009.

[36] P. Swoboda and B. Andres. A message passing algorithm for the minimum cost multicut problem. *arXiv preprint arXiv:1612.05441*, 2016.

[37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[38] M. G. Uzunbas, C. Chen, and D. Metaxas. An efficient conditional random field approach for automatic and interactive neuron segmentation. *Medical image analysis*, 27:31–44, 2016.

[39] Z. Zheng, J. S. Lauritzen, E. Perlman, C. G. Robinson, M. Nichols, D. Milkie, O. Torrens, J. Price, C. B. Fisher, N. Sharifi, S. A. Calle-Schuler, L. Kmecova, I. J. Ali, B. Karsh, E. T. Trautman, J. Bogovic, P. Hanslovsky, G. S. X. E. Jefferis, M. Kazhdan, K. Khairy, S. Saalfeld, R. D. Fetter, and D. D. Bock. A complete electron microscopy volume of the brain of adult drosophila melanogaster. *bioRxiv*, 2017.