7 APPENDIX

7.1 Overview of problem set-up

	Normal Edges	Dense Lifted Edges	Sparse Lifted Edges
Drosophila Neural Tissue	Mean boundary evidence	-	False merge oracle predictions
Murine Neural Tissue	RF based on edge features	RF based on region/ clustering features	RF based on axon/ dendrite attribution
Sponge Choanocytes	Mean boundary evidence	-	semantic segmentation of small structures
Arabidopsis Roots	Mean boundary evidence	-	instance segmentation of nuclei

Table 5. Overview of the three problem set-ups. RF abbreviates random forest.

7.2 Hierarchical Lifted Multicut Solver

```
Data: graph $G$, edge weights $W_E$, lifted edges and weights $F$ and $W_F$, nLevels, blockShape
Result: node partition $P$
$\hat{G}, \hat{F}, \hat{W}_E, \hat{W}_F = G, F, W_E, W_F$;
for n in nLevels do
1   blocks = getBlocks(blockShape);
    subPartitions = [];
    /* this for-loop can be parallelized */
    for block in blocks do
2       $G_{sub}, W_E^{sub}$ = getSubproblem($\hat{G}$, $\hat{W}_E$, block);
3       $F_{sub}, W_F^{sub}$ = getLiftedEdges($G_{sub}$, $\hat{F}$, $\hat{W}_F$);
4       $P_{sub}$ = solveLiftedMulticut($G_{sub}, W_E^{sub}, F_{sub}, W_F^{sub}$);
        subPartitions.append($P_{sub}$);
    end
5   $\hat{G}, \hat{F}, \hat{W}_E, \hat{W}_F$ = reduceProblem($\hat{G}, \hat{F}, \hat{W}_E, \hat{W}_F$, subPartitions);
    blockShape *= 2;
end
$P$ = solveLiftedMulticut($\hat{G}, \hat{F}, \hat{W}_E, \hat{W}_F$);
$P$ = projectToInitialGraph($G$, $P$);
```

Algorithm 1: Hierarchical lifted multicut algorithm based on the approximate multicut solver of (Pape et al., 2017). (1): getBlocks tiles the volume with blocks of blockShape. (2): getSubproblem extracts the sub-graph and weights of edges in this graph from the given block coordinates. (3): getLiftedEdges extracts the lifted edges that connect nodes which are *both* part of the sub-graph as well as the corresponding weights. (4): solveLiftedMulticut solves the lifted multicut problem using one of the two approximate solvers (Beier et al., 2017; Keuper et al., 2015). (5): reduceProblem: reduces the graph by contracting nodes according to the sub-partition results. Also updates edge weights as well as lifted edges and their weights accordingly.

	Energy	Time [s]
Greedy-Additive (Beier et al., 2016)	-1585593.5	2.03
Kernighan-Lin (Keuper et al., 2015)	-1645876.7	174.69
Fusion-Moves (Beier et al., 2016)	-1645876.7	181.48
Hierarchical (Ours)	-1630274.3	3.29

Table 6. Evaluating our proposed hierarchical solver and other multicut solvers. In order to run this experiment, we have constructed a smaller lifted multicut problem from the Drosophila neural tissue dataset by cutting out a $1 \times 10 \times 10$ micron block from its center, computing graph and local edge weights as described in section 4, introducing dense lifted edges within a graph neighborhood of 2 and setting their costs to the most repulsive edge cost along the weighted shorted path between the edge's terminal nodes. The problem at hand contained approximately 34,000 nodes, 244,000 normal edges and 2,384,000 lifted edges. The evaluation shows that the proposed solver yields energies comparable to Kernighan-Lin or Fusion-Moves, but its runtime is two orders of magnitude smaller and comparable to Greedy-Additive (which yields inferior energies). Kernighan-Lin was warm-started with the results of Greedy-Additive and Fusion-Moves with the results of Kernighan-Lin. Hierarchical has used Kernighan-Lin (warm-started with the solution of Greedy-Additive) for the sub-problems. While we only compare the solvers for a single problem size, we have observed very good scalability of our solver, which has solved much larger problems in section 4.