Correlation Clustering with Dynamic Super-Nodes (DySNCC)

Anonymous CVPR submission

Paper ID ****

Notes

Abstract

We address the problem of partitioning a graph into a previously unknown number of clusters. Among all partitions, the one with the minimal sum of of cut edge weights is chosen. This problem is known as correlation clustering or multicut. We propose a general framework to find high quality approximate solutions for this NP-hard problem based on a move making algorithm: We use candidate solutions from a proposal generator to iteratively improve the best observed solution similar to fusion moves. The proposed solver outperforms any other solver w.r.t. to any time performance. The solutions found by this solver are close to global optimal solutions w.r.t. energy and problem specific measurements as VI for image segmentation problems.

1. Introduction

should there be a general introduction to multicuts?

Given an edge weighted graph, positive weights (*attractive*) encourage the adjacent nodes to be in the same connected component, while negative weights (*repulsive*) encourage the nodes to stay in different connect components. Therefore the sign of the weights encodes if two nodes should be merged or not and the magnitude of the weights encodes the certainty of this desire. The objective of correlation clustering / multicuts is finding the cut with a minimal sum of cut edge weights. The number of connected components / clusters is discovered from the weights (rewrite?!?). Correlation clustering / the multicut is NP-hard [**?**]. Despite the NP-hardness, correlation clustering has been successfully used for (i) partitioning a superpixel region adjacency graph [3, 4] (ii) with optional long range repulsive edges [5]. (iii) Alush and Globerger showed how to average multiple segmentations with the multicut objective [1]. (iv) Multicuts can also be used for interactive segmentation [6], (v) for co-segmentation [8] (vi) and to cluster fully connected graphs [**?**].

WHY WE NEED THIS NEW SOLVER?

Nunez-Iglesias *et al.* describe the multicut as "one-shot agglomeration of supervoxels" whose main drawback is the scalability [12].

BETTER MOTIVATE SUPERNODES

To address scalability, supernodes can be useful. To create supernodes, some edges will be contracted and their nodes are merged into super-nodes. Doing so, we can reduce the graph to a reasonable size, such that we can find global optimal multicut solutions within reasonable time. Any edge which is contracted is lost. If such an edge is cut in the global optimal solution, we cannot find this global optimal solutions. Therefore we avoid fixed decisions and use a dynamic energy aware contraction scheme. Different edge contractions lead to different super-node graphs. A trivial approach is the following. Create different super-node graphs, optimize them with multicuts, and take the result which leads to the lowest energy on the unmodified initial graph. Doing so, we throw away all results which do not have the lowest energy. Within this work we propose a framework which combine multiple *proposal* super-node graphs without any premature irreversible decisions. Furthermore, multiple super-node graphs are combined such that the information form all of them is used.

proposal super-node graphs sounds shitty

The proposed algorithm can be interpreted as a fusion moves algorithm for correlation clustering / multicut objective. We generate cheap and versatile proposals and *fuse* them with the current best solution with the guarantee that this cannot increase the energy.

1.1. Related Work

1.1.1 Multicut Objective

The multicut / correlation clustering objective can be formulated in different ways.

Edge Indicator Variables:

Method		Limitations	
Multicut [3, 10]	(I)LP	not scalable, slow	Cutting plane method. Global optimal but slow for huge problems. However, applicable for medium size superpixel RAG.
Expand And Explorer [6]	movemaker	gets stuck early	α-Expansion modification. Applicable to large scale problems but gets stuck in local minima(???).
Fast Planar CC [13]	dual-decompositon	planar graphs only, duality/integrality gab	Dual decompositon in planar subgraphs?? Works on planar graphs only. Fast (is there a word for fast to medium fast) for small to medium size graphs.
Cut Glue and Cut [7]	move-making	gets stuck in poor minima for non planar graphs	Greedy solver based on a series of 2 colorings. State of the art any time performance. Very fast for planar graphs. Gets stuck in local minima for non planar graphs.
THIS WORK	fusion move-making	no natural stopping condition	Fusion of a proposal segmentation with the current best segmentation. State of the art any time performance for (???)

Table 1: Overview of correlation clustering / multicut solvers

$$
\begin{aligned}
y^* &= \arg\min_{y} \sum_{e_{ij} \in E} w_{ij} \cdot y_{ij} && (1)\\
s.t.: &\quad y \in \textit{Multicut Polytope}
\end{aligned}
$$

Fully Connected Graph:

$$
\begin{aligned}
y^* &= \arg\min_{y} \sum_{i<j \in V} w_{ij} \cdot y_{ij} && (2)\\
s.t.: &\quad y_{ij} + y_{jk} < y_{i,k} \quad \forall i,j,k
\end{aligned}
$$

Node Coloring:

$$
\begin{aligned}
l^* &= \arg\min_{L} \sum_{e_{ij} \in E} w_{ij} \cdot [l_u \neq l_v] && (3)\\
y^*_{ij} &= [l_u \neq l_v] && (4)
\end{aligned}
$$

1.1.2 Solver for the Multicut Objective

- Multicut [10]
- Expand and Explorer [6]
- Fast Planar CC [13]
- Break and Conquer [2].
- Cut Glue And Cut [7]

1.1.3 Fusion Moves

Move making algorithms, in particular fusion moves, have become increasingly popular for energy minimization [?, 9]. For many large scale computer vision applications fusion moves lead to good approximations with state of the art any time performance [9].

1.2. Karger-Stein Algorithms

Use randomized procedure to reduce the number of nodes / edges to a reasonable number. On the smaller graph, more expensive solvers are used.

2. Intersection Fusion Cut

Global optimal solvers for multicut do not scale beyond ??? [?]. Good approximate solvers for planar graphs exist [7, 13] but have difficulties to find good solutions for non planar graphs [7]. To make multicuts scalable, it is desirable to replace the graph G with a graph $\hat{G}$, which is smaller in the number of nodes and edges. If $\hat{G}$ is small enough, it is feasible to use global optimal solvers. A trivial way to reduce the size of the graph is edge contraction. Until a desirable size is reached, one might contract the edge with the highest weight. A cut on the contracted graph $\hat{G}$ is always a valid cut on $\hat{G}$. On the other hand, if an edge e is contracted, it will never be cut. If e is part of the cut in a global optimal solution, we cannot find this cut. To overcome this problem we can randomize the edge contraction, and repeat this multiple times and remember the best solution. But this would throw away all the "not the best" solutions (rewrite me). Within this work we propose the following nifty trick. Instead of throwing away solutions, we intersect them with the best solution, and solve the multicut on the resulting graph. Therefore we fuse multiple solutions,

Algorithm 1 Fusion Based Algorithms

procedure INTERSECTION-BASED-INFERENCE(GEN,J,X)
  $x^0 \leftarrow$ initial state form X
  $n \leftarrow 0$ ▷ Number of moves
  $m \leftarrow 0$ ▷ Number of moves without progress
  while $m < m_{\max}$ and $n < n_{\max}$ **do**
    $n \leftarrow n + 1$
    $x' \leftarrow GEN(x^{n-1}, J, X)$ ▷ Generate proposal
    if $J(x^{n-1}) \leq J(x')$ **then**
      $x^n \leftarrow Fuse(x^{n-1}, x', J)$
    else
      $x^n \leftarrow Fuse(x', x^{n-1}, J)$
    end if
    if $J(x^n) \leq J(x^{n-1})$ **then**
      $m \leftarrow 0$ ▷ Reset counter
    else
      $m \leftarrow m + 1$ ▷ Increment counter
    end if
  end while
  return x^n
end procedure

Theorem 1 (NameMe)**.** *Solving ??? on the contracted graph is equivalent to solving ??? on the original graph with additional must link constraints*

Proof of theorem . todo □

Theorem 2 (NameMe)**.** *Let x_a and x_b be two proposal solutions and x_{new} the solution of* **??**. *x_{new} will never decrease the energy: $x_{new} \cdot w \leq \min(x_a \cdot w, x_b \cdot w)$*

Algorithm 2 Fusion Moves

Require: $J(x) \leq J(x')$
Ensure: $J(\hat{x}) \leq J(x)$

procedure FUSE(x, x', J)
  $\bar{x} \leftarrow$ **Intersect**(x, x') ▷ intersect uncut edges in x and x', return connected component labeling
  $\tilde{G} = (\tilde{E}, \tilde{V}), \tilde{W} \leftarrow$ **Contract**$(G, W, \bar{x})$ ▷ Contract all edges which are uncut in $\bar{x}$
  $\tilde{x}^* = \arg\min_{\tilde{x}} \sum_{e_{ij} \in \tilde{E}} \tilde{w}_{ij}[\tilde{x}_i \neq \tilde{x}_j]$ ▷ Solve the multicut objective on smaller graph $\tilde{G} = (\tilde{V}, \tilde{E})$
  $\hat{x} \leftarrow$ **ProjectBack**$(\tilde{x}^*)$ ▷ Translate the connected component labeling $\tilde{x}^*$ of $\tilde{G}$ to a connected component labeling of G
  return $\hat{x}$
end procedure

procedure FUSE$_{\text{BASE}}$(x, x', J)
  return $\arg\min_{\bar{x} \in \{x, x'\}} J(\bar{x})$
end procedure

Proof of theorem . todo □

Theorem 3 (NameMe)**.** *Let x_a and x_b be two proposal solutions and x_{opt} the global solution, X_a^c, X_b^c and X_{opt}^c the set of cut edges in x_a, x_b and x_{opt} optimal solution. Iff $X_{opt}^c \subseteq (X_a^c \cup X_b^c)$, the solution of* **??** *will be x_{opt}.*

Proof of of theorem. todo □

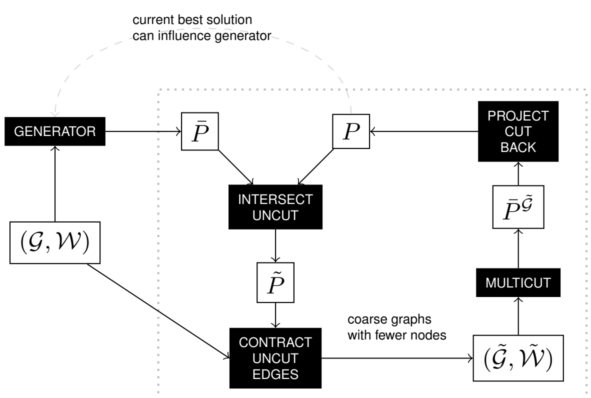


Figure 1: The proposed algorithm works in the following way: Given a graph $\mathcal{G}$, edge weights $\mathcal{W}$ and a proposal generator, the current best solution P is iteratively improved. A proposal generator generates different versatile proposal partitions $\bar{P}$. The proposal $\bar{P}$ is intersected with P which results in $\tilde{P}$. Contracting each each which is not cut in $\tilde{P}$ leads to a coarser graph $\tilde{\mathcal{G}} = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$ with new edge weights $\tilde{\mathcal{W}}$. If $\bar{P}$ and P have a small fraction of cut edges, $\tilde{\mathcal{G}}$ will be small ($|\tilde{\mathcal{V}}| << |\mathcal{V}|$ and $|\tilde{\mathcal{E}}| << |\mathcal{E}|$). The multicut objective on the smaller graph $\tilde{\mathcal{G}}$ can be optimized magnitudes faster than on $\tilde{\mathcal{G}}$. It is guaranteed that the optimal multicut partitioning $\bar{P}^{\tilde{\mathcal{G}}}$ on $\tilde{\mathcal{G}}$ projected back to $\mathcal{G}$ as a lower or equal energy than any of the two input partitions P and $\bar{P}$, therefore we store the result of fusion as new best state P and repeat the procedure.

2.1. Proposal Generators

Algorithm 3 Proposal Generators

procedure RAND2COLORING($x, G = (E, V), W$)
  $\tilde{W} \leftarrow$ **randomize**(W)
  $x' = \arg\min_{\bar{x}} \sum_{e_{ij} \in E} \tilde{w}_{ij}[\bar{x}_i \neq \bar{x}_j][x_i = x_j]$
    $s.t. \quad x_i \in \{0, 1\} \forall i$
  return x'
end procedure

procedure RANDEDGEWEIGHTEDWATERSHEDS($x, G = (E, V), W$)
  $\tilde{W} \leftarrow$ **randomize**(W)
  $getRandomSeeds$
  $runEdgeWeightedWs$
end procedure

procedure RANDHIERARCHICALCLUSTERING($x, G = (E, V), W$)
  $\tilde{W} \leftarrow$ **randomize**(W)
  $\hat{G} = (\hat{E}, \hat{V}) \leftarrow G = (E, V)$
  $\hat{W} \leftarrow W$
  while $|\hat{V}| > \gamma$ **and** $\max(\hat{W}) > \theta$ **do**
    $\hat{G} = (\hat{E}, \hat{V}), \hat{W} \leftarrow$ **contract**$(\hat{G}, \hat{W}, \arg\max_{\hat{E}}(\hat{W}))$
  end while
  $x' \leftarrow$ **getClusterResults**(???)
end procedure

2.1.1 Randomized HierarchicalClustering (RHC)

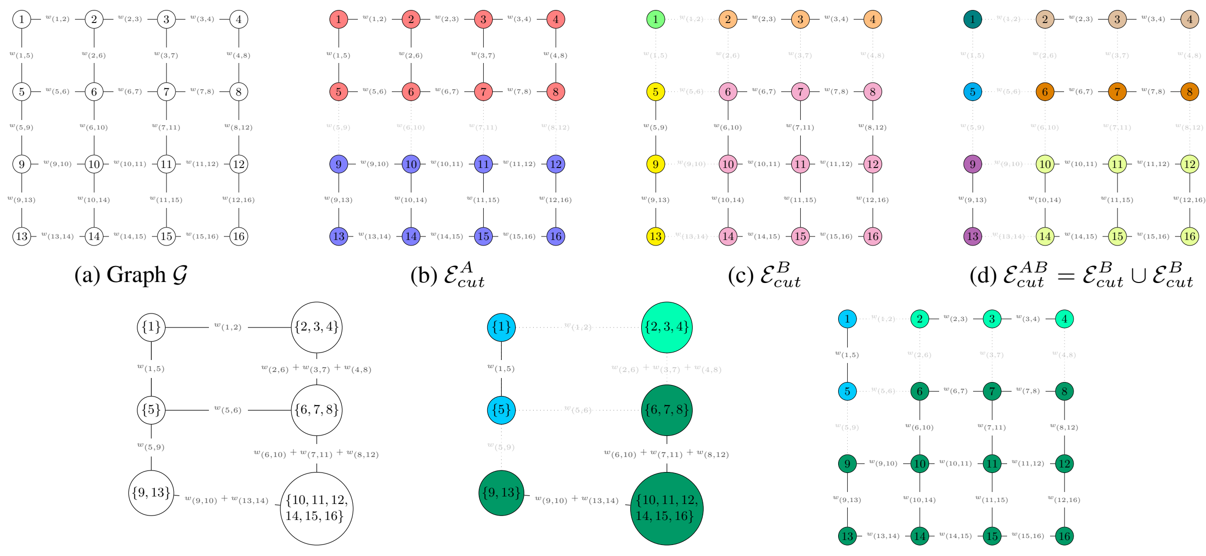
To generate proposals cut we can use bottom up hierarchical clustering. In each step we contract the edge with the highest weight. Parallel edges are merged into single edges by summing their weights. We stop when a certain number of nodes is reached, or the largest edge weight is smaller then a certain threshold. To get versatile proposals we either add noise to the edge weighs or permute a certain number of weights.

2.1.2 Randomized MaxCut (RMC)

To get energy aware proposals we can use max cuts. The max cut objective is the same as the multicut objective, but only two node colors are allowed. Therefore solutions of the max cut objective might be useful proposals. To create different proposals we set the weight of edges which are cut in the best solution to zero. In this way we favor solutions different form the current best. In addition we either add noise permute a certain number of weights.

2.1.3 Randomized Watersheds (RWS)

The edge weighted watershed algorithm [11] with random seeds can be used to get cheap proposals. Instead of n seeds distributed uniformly over all nodes we use the following. Draw n/k random edges only the negative edges, and give the two nodes of each random edge different seeds. Doing so, a random subset of negative edges is forced to be cut



(a) Graph $\mathcal{G}$ (b) $\mathcal{E}^{A}_{cut}$ (c) $\mathcal{E}^{B}_{cut}$ (d) $\mathcal{E}^{AB}_{cut} = \mathcal{E}^{B}_{cut} \cup \mathcal{E}^{B}_{cut}$

(e) $\bar{\mathcal{G}} = \textbf{contract}(\mathcal{E} \backslash \{\mathcal{E}^{AB}_{cut}\})$ (f) $\mathcal{E}^{\bar{\mathcal{G}}}_{cut}$ (g) $\mathcal{E}^{\mathcal{G}}_{cut} = \textbf{projectBack}(\mathcal{E}^{\bar{\mathcal{G}}}_{cut})$

Figure 2: Describe Method here

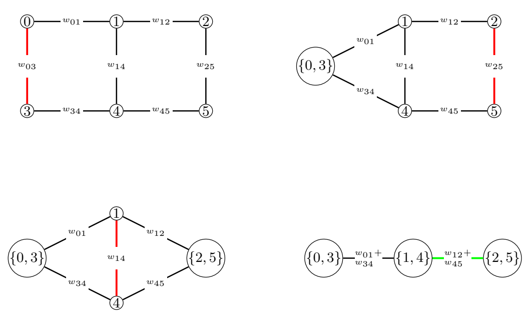


Figure 3: To use hierarchical clustering as an approximator for the multicut objective we contract the edge with the highest weight in each step (to be contracted edge is shown in red). Whenever multiple edges are merged into a single edge, we use the sum of the weight as a new weight, since we have to pay for all edges which are cut in the uncontracted graph. We stop when the highest edge weight is smaller or equal to zero (edge shown in green). Doing so, hierarchical clustering can be interpreted as an greedy approximator for the multicut objective.

within each proposal. For additional randomness, noise can be added to the edge weighs.

2.2. Fusion Move Solver

3. Experiments

3.1. Benchmark Models

Table 2: image-seg (100 instances)

algorithm	runtime	value	bound	mem	best	opt
ogm-CGC-planar*	0.28 sec	4445.22	4136.83	0.02 GB	28	0
ogm-mcfusion-HC-BASE*	0.12 sec	5462.60	$-\infty$	0.01 GB	0	0
ogm-mcfusion-HC-BASE-CGCF*	0.12 sec	5085.80	$-\infty$	0.01 GB	0	0
ogm-mcfusion-HC-CGC*	1.23 sec	4444.56	$-\infty$	0.01 GB	40	0
ogm-mcfusion-HC-CGC-CGCF*	1.22 sec	4444.75	$-\infty$	0.02 GB	40	0
ogm-mcfusion-HC-MC*	5.12 sec	4443.61	$-\infty$	0.04 GB	73	0
ogm-mcfusion-HC-MC-CGCF*	5.14 sec	4443.61	$-\infty$	0.05 GB	73	0
ogm-mcfusion-WS-BASE*	0.05 sec	6622.98	$-\infty$	0.01 GB	0	0
ogm-mcfusion-WS-BASE-CGCF*	1.77 sec	4460.12	$-\infty$	0.01 GB	3	0
ogm-mcfusion-WS-CGC*	1.44 sec	4445.57	$-\infty$	0.01 GB	16	0
ogm-mcfusion-WS-CGC-CGCF*	1.75 sec	4444.59	$-\infty$	0.01 GB	27	0
ogm-mcfusion-WS-MC*	6.00 sec	4444.58	$-\infty$	0.03 GB	22	0
ogm-mcfusion-WS-MC-CGCF*	6.31 sec	4443.79	$-\infty$	0.03 GB	46	0

Table 3: knott-3d-150 (8 instances)

algorithm	runtime	value	bound	mem	best	opt
ogm-CGC*	0.08 sec	−4566.41	−4855.18	0.01 GB	5	0
ogm-mcfusion-HC-BASE*	0.12 sec	0.00	$-\infty$	0.01 GB	0	0
ogm-mcfusion-HC-BASE-CGCF*	0.16 sec	−4160.88	$-\infty$	0.01 GB	0	0
ogm-mcfusion-HC-CGC*	0.36 sec	−4559.96	$-\infty$	0.01 GB	6	0
ogm-mcfusion-HC-CGC-CGCF*	0.36 sec	−4565.61	$-\infty$	0.01 GB	6	0
ogm-mcfusion-HC-MC*	0.88 sec	−4559.96	$-\infty$	0.03 GB	6	0
ogm-mcfusion-HC-MC-CGCF*	0.88 sec	−4565.61	$-\infty$	0.03 GB	6	0
ogm-mcfusion-WS-BASE*	0.03 sec	0.00	$-\infty$	0.01 GB	0	0
ogm-mcfusion-WS-BASE-CGCF*	0.12 sec	−4565.95	$-\infty$	0.01 GB	5	0
ogm-mcfusion-WS-CGC*	NaN sec	NaN	NaN	0.01 GB	4	0
ogm-mcfusion-WS-CGC-CGCF*	0.37 sec	−4571.22	$-\infty$	0.01 GB	7	0
ogm-mcfusion-WS-MC*	1.28 sec	−4570.83	$-\infty$	0.03 GB	5	0
ogm-mcfusion-WS-MC-CGCF*	1.31 sec	−4571.22	$-\infty$	0.03 GB	7	0

Table 4: knott-3d-300 (8 instances)

algorithm	runtime	value	bound	mem	best	opt
ogm-ICM	84.37 sec	−25196.51	$-\infty$	0.01 GB	0	0
ogm-KL	13.16 sec	−25556.93	$-\infty$	0.01 GB	0	0
ogm-LF-1	29.08 sec	−25243.76	$-\infty$	0.02 GB	0	0
MCR-CC	3423.65 sec	−26161.81	−27434.30	0.57 GB	1	1
MCR-CCFDB	1338.99 sec	−27276.12	−27307.22	0.15 GB	1	1
MCR-CCFDB-OWC	1367.03 sec	−27287.23	−27309.62	0.15 GB	6	6
MCI-CCFDB-CCIFD	1261.99 sec	−26826.57	−27308.19	0.37 GB	6	6
MCI-CCI	220.30 sec	−27302.78	−27305.02	0.28 GB	8	7
MCI-CCIFD	104.55 sec	−27302.78	−27302.78	0.16 GB	8	8
ogm-CGC*	4.53 sec	−27251.42	−28901.58	0.02 GB	0	0
ogm-mcfusion-HC-BASE*	1.19 sec	0.00	$-\infty$	0.02 GB	0	0
ogm-mcfusion-HC-BASE-CGCF*	2.38 sec	−24707.81	$-\infty$	0.04 GB	0	0
ogm-mcfusion-HC-CGC*	5.60 sec	−27282.58	$-\infty$	0.03 GB	2	0
ogm-mcfusion-HC-CGC-CGCF*	5.61 sec	−27283.82	$-\infty$	0.04 GB	3	0
ogm-mcfusion-HC-MC*	8.87 sec	−27283.36	$-\infty$	0.05 GB	3	0
ogm-mcfusion-HC-MC-CGCF*	8.92 sec	−27284.60	$-\infty$	0.06 GB	4	0
ogm-mcfusion-WS-BASE*	0.24 sec	0.00	$-\infty$	0.01 GB	0	0
ogm-mcfusion-WS-BASE-CGCF*	4.91 sec	−27253.30	$-\infty$	0.03 GB	0	0
ogm-mcfusion-WS-CGC*	22.99 sec	−27273.39	$-\infty$	0.01 GB	1	0
ogm-mcfusion-WS-CGC-CGCF*	23.91 sec	−27286.88	$-\infty$	0.03 GB	4	0
ogm-mcfusion-WS-MC*	36.85 sec	−27280.59	$-\infty$	0.05 GB	2	0
ogm-mcfusion-WS-MC-CGCF*	37.77 sec	−27293.33	$-\infty$	0.06 GB	6	0

Table 5: knott-3d-450 (8 instances)

algorithm	runtime	value	bound	mem	best	opt
ogm-ICM	883.63 sec	−72464.54	$-\infty$	0.03 GB	0	0
ogm-KL	186.89 sec	−73188.82	$-\infty$	0.03 GB	0	0
ogm-LF-1	298.07 sec	−72479.60	$-\infty$	0.04 GB	0	0
MCR-CC	9814.45 sec	−4892.36	−83272.85	0.39 GB	0	0
MCR-CCFDB	6404.34 sec	−4892.36	−83272.85	0.20 GB	0	0
MCR-CCFDB-OWC	6455.21 sec	−4892.36	−83272.85	0.19 GB	0	0
MCI-CCFDB-CCIFD	6404.14 sec	−4892.36	−83272.85	0.19 GB	0	0
MCI-CCI	1196.62 sec	−78135.34	−78518.55	0.83 GB	6	6
MCI-CCIFD	1379.90 sec	−78180.20	−78507.25	0.54 GB	6	6
ogm-CGC*	93.59 sec	−78234.10	−83272.85	0.06 GB	0	0
ogm-mcfusion-HC-BASE*	7.74 sec	0.00	$-\infty$	0.06 GB	0	0
ogm-mcfusion-HC-BASE-CGCF*	34.02 sec	−70750.35	$-\infty$	0.10 GB	0	0
ogm-mcfusion-HC-CGC*	68.29 sec	−78422.82	$-\infty$	0.10 GB	0	0
ogm-mcfusion-HC-CGC-CGCF*	67.88 sec	−78425.09	$-\infty$	0.13 GB	0	0
ogm-mcfusion-HC-MC*	102.12 sec	−78414.40	$-\infty$	0.12 GB	1	0
ogm-mcfusion-HC-MC-CGCF*	99.16 sec	−78418.71	$-\infty$	0.14 GB	1	0
ogm-mcfusion-WS-BASE*	1.20 sec	0.00	$-\infty$	0.03 GB	0	0
ogm-mcfusion-WS-BASE-CGCF*	99.82 sec	−78252.24	$-\infty$	0.07 GB	0	0
ogm-mcfusion-WS-CGC*	420.51 sec	−78434.90	$-\infty$	0.04 GB	0	0
ogm-mcfusion-WS-CGC-CGCF*	442.00 sec	−78455.68	$-\infty$	0.08 GB	0	0
ogm-mcfusion-WS-MC*	1456.53 sec	−78438.50	$-\infty$	0.12 GB	1	0
ogm-mcfusion-WS-MC-CGCF*	1492.79 sec	−78452.38	$-\infty$	0.14 GB	4	0

Table 6: knott-3d-550 (8 instances)

algorithm	runtime	value	bound	mem	best	opt
ogm-ICM	2787.19 sec	−126335.53	$-\infty$	0.06 GB	0	0
ogm-KL	604.50 sec	−127032.70	$-\infty$	0.06 GB	0	0
ogm-LF-1	987.28 sec	−126356.35	$-\infty$	0.07 GB	0	0
MCR-CC	68016.21 sec	−8187.14	−144703.64	1.20 GB	0	0
MCR-CCFDB	53527.56 sec	−8187.14	−144703.64	0.47 GB	0	0
MCR-CCFDB-OWC	53386.77 sec	−8187.14	−144703.64	0.47 GB	0	0
MCI-CCFDB-CCIFD	53404.59 sec	−8187.14	−144703.64	0.47 GB	0	0
MCI-CCI	3436.37 sec	−101592.17	−136714.73	1.53 GB	2	2
MCI-CCIFD	3299.31 sec	−120874.65	−136699.49	0.86 GB	2	2
ogm-CGC*	595.15 sec	−136188.55	−144703.64	0.11 GB	0	0
ogm-mcfusion-HC-BASE*	17.17 sec	0.00	$-\infty$	0.11 GB	0	0
ogm-mcfusion-HC-BASE-CGCF*	118.24 sec	−123617.74	$-\infty$	0.19 GB	0	0
ogm-mcfusion-HC-CGC*	208.48 sec	−136454.21	$-\infty$	0.21 GB	0	0
ogm-mcfusion-HC-CGC-CGCF*	205.22 sec	−136472.89	$-\infty$	0.27 GB	0	0
ogm-mcfusion-HC-MC*	300.08 sec	−136449.81	$-\infty$	0.22 GB	0	0
ogm-mcfusion-HC-MC-CGCF*	291.88 sec	−136468.45	$-\infty$	0.27 GB	1	0
ogm-mcfusion-WS-BASE*	2.32 sec	0.00	$-\infty$	0.05 GB	0	0
ogm-mcfusion-WS-BASE-CGCF*	568.33 sec	−136191.25	$-\infty$	0.14 GB	0	0
ogm-mcfusion-WS-CGC*	1694.84 sec	−136441.96	$-\infty$	0.07 GB	0	0
ogm-mcfusion-WS-CGC-CGCF*	1900.61 sec	−136491.63	$-\infty$	0.15 GB	0	0
ogm-mcfusion-WS-MC*	3617.78 sec	−136435.37	$-\infty$	0.23 GB	0	0
ogm-mcfusion-WS-MC-CGCF*	3773.77 sec	−136494.37	$-\infty$	0.28 GB	4	0

Table 7: seg-3d (2 instances)

algorithm	runtime	value	bound	mem	best	opt
ogm-ICM	1959.44 sec	618075.17	$-\infty$	0.22 GB	0	0
ogm-KL	2290.46 sec	441695.84	$-\infty$	0.20 GB	0	0
ogm-LF-1	1857.93 sec	552690.10	$-\infty$	0.27 GB	0	0
MCR-CC	NaN sec	NaN	NaN	0.11 GB	1	1
MCR-CCFDB	NaN sec	NaN	NaN	0.05 GB	1	1
MCR-CCFDB-OWC	NaN sec	NaN	NaN	0.05 GB	1	1
MCI-CCFDB-CCIFD	NaN sec	NaN	NaN	0.05 GB	1	1
MCI-CCI	1806.76 sec	414124.76	413523.92	4.43 GB	1	1
MCI-CCIFD	4201.35 sec	650888.58	401952.16	1.01 GB	1	1
ogm-CGC*	NaN sec	NaN	NaN	0.03 GB	0	0
ogm-mcfusion-HC-BASE*	27.97 sec	501728.60	$-\infty$	0.38 GB	0	0
ogm-mcfusion-HC-BASE-CGCF*	40.77 sec	471105.16	$-\infty$	0.55 GB	0	0
ogm-mcfusion-HC-CGC*	1293.80 sec	413546.15	$-\infty$	0.72 GB	1	0
ogm-mcfusion-HC-CGC-CGCF*	1291.94 sec	413545.50	$-\infty$	0.88 GB	2	0
ogm-mcfusion-HC-MC*	1171.51 sec	413552.74	$-\infty$	0.68 GB	1	0
ogm-mcfusion-HC-MC-CGCF*	1172.41 sec	413546.96	$-\infty$	0.85 GB	1	0
ogm-mcfusion-WS-BASE*	7.56 sec	748369.02	$-\infty$	0.19 GB	0	0
ogm-mcfusion-WS-BASE-CGCF*	6088.09 sec	413807.74	$-\infty$	0.40 GB	0	0
ogm-mcfusion-WS-CGC*	1838.10 sec	413760.67	$-\infty$	0.28 GB	0	0
ogm-mcfusion-WS-CGC-CGCF*	2484.01 sec	413573.58	$-\infty$	0.44 GB	0	0
ogm-mcfusion-WS-MC*	32840.49 sec	427635.00	$-\infty$	10.53 GB	0	0
ogm-mcfusion-WS-MC-CGCF*	34279.76 sec	413814.27	$-\infty$	10.69 GB	1	0

Table 8: modularity-clustering (6 instances)

algorithm	runtime	value	bound	mem	best	opt
ogm-ICM	0.09 sec	0.0000	$-\infty$	0.01 GB	0	0
ogm-KL	0.01 sec	−0.4860	$-\infty$	0.01 GB	3	0
ogm-LF-1	0.03 sec	0.0000	$-\infty$	0.01 GB	0	0
MCR-CC	97.63 sec	−0.4543	−0.5094	0.14 GB	2	1
MCR-CCFDB	1.81 sec	−0.4543	−0.5094	0.03 GB	1	1
MCR-CCFDB-OWC	602.50 sec	−0.4652	−0.4962	0.03 GB	5	5
MCI-CCFDB-CCIFD	601.28 sec	−0.4537	−0.5021	1.85 GB	5	5
MCI-CCI	1206.55 sec	−0.4312	−0.5158	2.84 GB	4	4
MCI-CCIFD	1203.92 sec	−0.4399	−0.5176	3.16 GB	4	4

3.2. Pixel-wise Multicuts

4. Conclusion

References

[1] A. Alush and J. Goldberger. Ensemble segmentation using efficient integer linear programming. *Pattern Analysis and Machine Intelligence*, 34(10):1966–1977, 2012. 1

[2] A. Alush and J. Goldberger. Break and conquer: Efficient correlation clustering for image segmentation. In *2nd International Workshop on Similarity-Based Pattern Analysis and Recognition*, 2013. 2

[3] B. Andres, J. H. Kappes, T. Beier, U. Köthe, and F. A. Hamprecht. Probabilistic image segmentation with closedness constraints. In *ICCV*, pages 2611–2618. IEEE, 2011. 1, 2

[4] 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 *ECCV*, pages 778–791. Springer, 2012. 1

[5] B. Andres, J. Yarkony, B. S. Manjunath, S. Kirchhoff, E. Turetken, C. C. Fowlkes, and H. Pfister. Segmenting Planar Superpixel Adjacency Graphs w.r.t.˜Non-planar Superpixel Affinity Graphs. In *EMMCVPR*, 2013. 1

[6] S. Bagon and M. Galun. Large scale correlation clustering optimization. *CoRR*, abs/1112.2903, 2011. 1, 2

[7] T. Beier, T. Kroeger, J. H. Kappes, U. Koethe, and F. Hamprecht. Cut, Glue & Cut: A Fast, Approximate Solver for Multicut Partitioning. In *IEEE Conference on Computer Vision and Pattern Recognition 2014*, 2014. 2

[8] D. Glasner, S. N. P. Vitaladevuni, and R. Basri. Contour-based joint clustering of multiple segmentations. In *CVPR*, pages 2385–2392. IEEE, 2011. 1

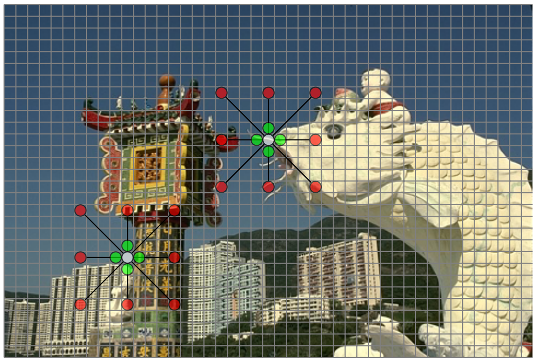


Figure 4: Pixel Level Multicut: every pixel (white nodes) is connected to its 4 *local* neighbors (edges between white and green nodes). Furthermore each pixel is connected to some *non-local* neighbors within a certain radius (edges between white and green nodes). The local neighbors are connected with a *positive* edge weight. If the edge indicator (as gradient magnitude) is very high, the local edge weight should be close to zero. If there is no evidence for a cut (low gradient magnitude for example) the local edge weight should be high. The *non-local edge weights* are *negative* to encourage label transitions. The weight of the non-local edge weights can be the negative value of the maximum gradient magnitude along a line between the red and white node. If there is evidence for a cut between red and white, the weight should be strongly(?) negative.

[9] J. H. Kappes, T. Beier, and C. Schnörr. Map-inference on large scale higher-order discrete graphical models by fusion moves. In *International Workshop on Graphical Models in Computer Vision*, 2014. Oral. 2

[10] J. H. Kappes, M. Speth, B. Andres, G. Reinelt, and C. Schnörr. Globally optimal image partitioning by multicuts. In *EMMCVPR*, pages 31–44. Springer, 2011. 2

[11] F. Meyer. Watersheds on edge or node weighted graphs ”par l’exemple”. *CoRR*, abs/1303.1829, 2013. 3

[12] J. Nunez-Iglesias, R. Kennedy, T. Parag, J. Shi, and D. B. Chklovskii. Machine learning of hierarchical clustering to segment n-dimensional images. *CoRR*, abs/1303.6163, 2013. 1

[13] J. Yarkony, A. Ihler, and C. C. Fowlkes. Fast planar correlation clustering for image segmentation. In *ECCV*. Springer, 2012. 2