Fusion Moves for Correlation Clustering

Anonymous CVPR submission

Paper ID 1881

Abstract

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.

1. Introduction

Correlation clustering [8], also known as the multicut problem [12] is a basic primitive in computer vision [3, 4, 35, 2] and data mining [5, 31, 10, 11]. See Sec. 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 [10, 11]. Correlation clustering can also be used to cluster query refinements in web search [31]. Because social and web-related networks are often huge, heuristic methods, *e.g.* the PIVOT-algorithm [1], are popular [11].

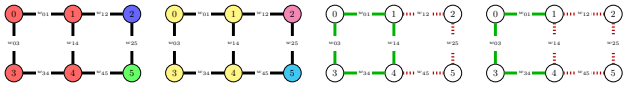
In computer vision applications, unsupervised image segmentation algorithms often start with an oversegmentation into superpixels (superregions), which are then clustered into "perceptually meaningful" regions by correlation clustering. Such an approach has been shown to yield state-of-the-art results on the Berkeley Segmentation Database [3, 23, 35, 2].

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 [4] or social networks [25], are not tractable because reasonable solutions cannot be computed in acceptable time.

Contribution. In this work 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 [9]. Second, we improve the anytime performance of polyhedral multicut methods [21] by more efficient separation procedures. Third, we introduce cluster-fusion moves, which extend the original fusion moves [24] 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.

Related Work. A natural approach is to solve the integer linear program (ILP) directly 2. To this end, efficient separation procedures have been found [20, 21] 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 [21]. 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 [35].

Another line of work uses move making algorithms to optimize correlation clustering [22, 7, 9]. Starting with an initial segmentation, auxiliary max-cut problems are (approximately) solved, such that the segmentation is strictly improved. As shown in [9] only Cut, Glue & Cut (CGC) can deal with large scale problems, but can also suffer from very large auxiliary problems.



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

Figure 1: Representing a clustering by node labels is ambiguous. 1a and 1b encode the same partition. Edge labels as in 1c do not suffer from such ambiguities, but can have *dangling edges* as in 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.

Outside computer vision, greedy methods [33, 29, 16, 14, 1] have been suggested for correlation clustering problems, see [15] for an overview. The PIVOT Algorithm [1] 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) [16], which iteratively reassigns nodes to clusters.

For energy minimization problems fusion moves have become increasingly popular [24, 19]. For many large scale computer vision applications fusion moves lead to good approximations with state of the art anytime performance [19]. Due to the ambiguity of a node-labeling, classical fusion moves [24] cannot be applied directly for correlation clustering. We will show how to overcome this problem in Sec. 4.

Outline: In Sec. 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. 3 and our proposed correlation clustering fusion moves in Sec. 4. We evaluate the proposed methods in Sec. 5 and conclude in Sec. 6 and 7.

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. 1. An edge labeling is only consistent if it does not violate any cycle constraint [12]. 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 the *multicut polytope* $MC(G) = \text{conv}(P(G))$. By $l(y)$ we denote some node labeling for a segmentation given by y.

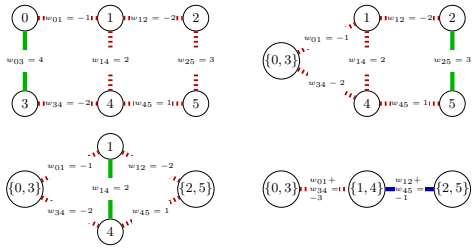


Figure 2: Energy-based Hierarchical clustering can be used to greedily optimize Eq. 2. In each step, the two nodes connected via the edge with the highest weight are merged by contracting this edge. (edge to be contracted is shown in green). Due to edge contraction parallel edges can occur, which are 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).

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 $l_i \in \mathbb{N}$

$$l^* = \arg\min_{l \in \mathbb{N}^{|V|}} \sum_{(i,j)\in E} w_{ij} \cdot [l_i \neq l_j], \tag{1}$$

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

$$y^* = \arg\min_{y \in P(G)} \sum_{(i,j)\in E} w_{ij} \cdot y_{ij}. \tag{2}$$

As shown in [21] both problems are equivalent, but formulation 1 suffers from ambiguities in the representation, *cf*. Fig. 1.

3. Energy Based Hierarchical Clustering

Agglomerative hierarchical clustering (HC) is widely used in graph / image segmentation [6]. In each step, the edge with the highest weight w is contracted (green edges in Fig. 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 [6].

Because we, contrary to [6], 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). 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 [9]. Excessive time in CGC is spent in the *cut phase* to solve the first two coloring on the complete graph. As shown in Sec. 5, allowing CGC to start from the EHC solution instead can improve performance drastically.

4. Correlation Clustering Fusion Moves

Fusion moves as defined in [24] work in the node domain and do not work properly for objective functions as Eq. 1 since the node coloring is ambiguous and has no semantic meaning, *cf*. Fig. 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''.

$$\breve{y}_{ij} = \max\{y'_{ij}, y''_{ij}\} \qquad \forall ij \in E \tag{3}$$

$$\mathcal{E}_0^{\breve{y}} = \{ij \in E \mid \breve{y}_{ij} = 0\} \tag{4}$$

The fusion move for correlation clustering is solving Eq. 2 with additional *must-link constraints* for all edges in $\mathcal{E}_0^{\breve{y}}$.

$$y^* = \arg\min_{y \in P(G)} \sum_{(i,j)\in E} w_{ij} \cdot y_{ij}. \tag{5}$$

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

By construction, solving Eq. 5 cannot increase the energy w.r.t. the proposals y' and y'', because y' and y'' are feasible solutions for problem 5.

As Lempitsky *et al*. [24], 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. 5.

4.1. Fast Optimization of CC-Fusion Moves

In general the auxiliary fusion problem 5 is, as for classical fusion [24], NP-hard. However, many variables have been fixed to be zero and we can reformulate 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*.

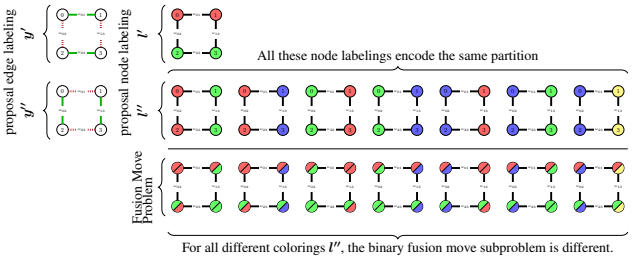


Figure 3: To fuse two edge labelings y' and y'' with fusion moves as defined by Lempitsky *et al*. [24] y' and y'' need to be transferred to the node domain. The mapping from edge labels to node labels is ambiguous and even for this small graph there are seven node labels which result in different binary fusion move problems. Enumerating all labelings for a graph of non trival size becomes intractable.

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

$$\tilde{y}_{ij} = \begin{cases} \bar{y}_{l_i(y)l_j(y)} & \text{if } l_i(y) \neq l_j(y) \\ 0 & \text{else} \end{cases} \qquad \forall uv \in E \tag{6}$$

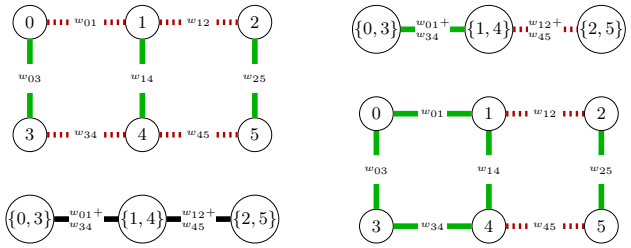
Theorem 1 (Equivalence)**.** *The back projection of the optimal segmentation $\bar{y}'$ of the contracted graph $G_y = (V_y, E_y, \bar{w})$ is an optimal solution of problem 5.*

Proof. Let y' be the back propagation of $\bar{y}'$, which is by definition feasible for 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 $e \in \mathcal{E}_0^{\breve{y}}$ we would have

$$\sum_{\bar{e}\in E_y} \bar{w}_{\bar{e}}\bar{y}'_{\bar{e}} = \sum_{e\in E\setminus\mathcal{E}_0^{\breve{y}}} w_e y'_e = \sum_{e\in E} w_e y'_e$$
$$> \sum_{e\in E} w_e y''_e = \sum_{e\in E\setminus\mathcal{E}_0^{\breve{y}}} w_e y''_e = \sum_{\bar{e}\in E_y} \bar{w}_{\bar{e}}\bar{y}''_{\bar{e}}$$

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 5 we can now solve problem 2 on the contracted graph $G_{\breve{y}}$. This is, depending on the intersection of the current and proposed solution, magnitudes smaller than G. The correlation clustering problem on $G_{\breve{y}}$



(a) Graph contraction (b) Back projection

Figure 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 nodes and 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.

can be solved by any correlation clustering solver. Since $G_{\bar{y}}$ is smaller, exact methods or good approximative methods like multicuts [21] or CGC [9] are very fast.

4.2. Polyhedral Interpretation

A polyhedral interpretation of fusion moves is shown in Fig. 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 [24] 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 [20, 21], which tighten an outer relaxation of the multicut polytope and contrary to our method do not operate in the feasible domain.

4.3. Proposal Generators

As discussed in [24], 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.

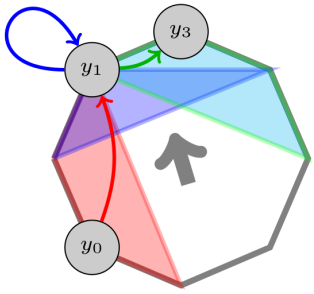


Figure 6: Each fusion move can be interpreted as an optimization of an inner polytope. Each inner polytope includes the current vertex. Starting with y_0 we optimize over the red polytope and find y_1 as optimum. Finally, when optimizing over the blue polytope we stay in y_1 as optimum, when optimizing over the green polytope we find y_2 which we will never leave again.

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. 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. 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 [13]. The edge weighted watershed algorithm [28] 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

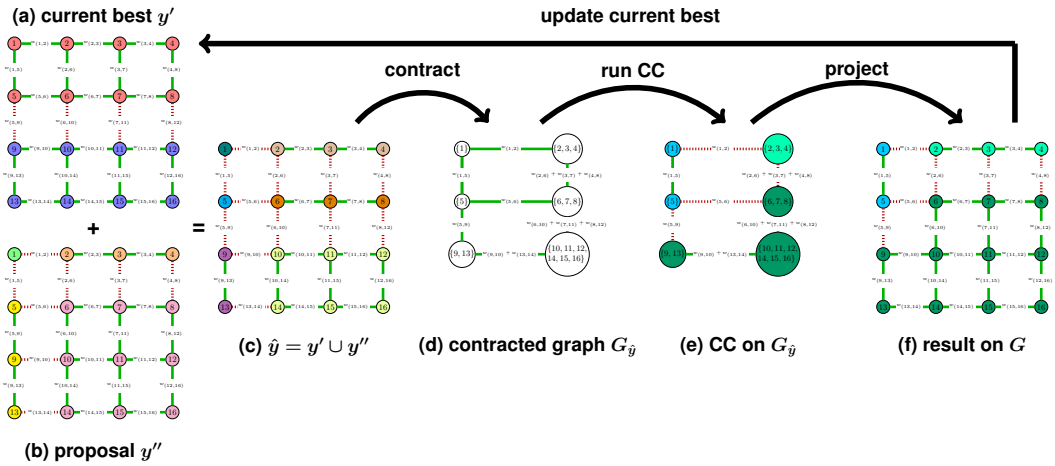


Figure 5: To fuse a current best segmentation y' (5a) with a proposal segmentation y'' (5b) we propose the following algorithm: $\hat{y}$ is defined as $y' + y''$ as in (5c). The contraction graph (5d) $G_{\hat{y}}$ is constructed by contracting all uncut edges in $\hat{y}$. The actual fusion move is solving (2) for $G_{\hat{y}}$ as in 5e and projecting the result back to G as in (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.

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 [34].

5. Experiments

In our experiments we compare to the following methods with publicly available implementation. For **CGC** [9] we used a branch of OpenGM[1] and for **KL** [22] the implementation in OpenGM[2]. For integer multicuts (**MC-I**) and relaxed multicuts (**MC-R**) [21] we modified OpenGM[2], as described in Sec. 5.2. From the field of data-mining we compare to the PIVOT-algorithm [1] followed by a round of BOEM [16] 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 [24] (**Fusion**) and select distinct labels for the two candidate segmentations. According to [9], CGC is faster and gives better energies than PlanarCC [35] and Expand & Explore [7]. 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. 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.

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 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

[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

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.

$$\sum_{i \to j \in E_{\rightarrow}} |w_{i \to j}| = 1 \qquad \forall i \in V, deg^{\text{out}}(i) \geq 1 \quad (7)$$

We define the following energy function

$$\begin{aligned} J(y) &= \sum_{i \to j \in E^+_{\rightarrow}} y_{ij} \cdot w_{i \to j} + \sum_{i \to j \in E^-_{\rightarrow}} (y_{ij} - 1) \cdot w_{i \to j} \\ &= \sum_{ij \in E} y_{ij} \cdot \underbrace{(w_{i \to j} + w_{j \to i})}_{w_{ij}} + \text{const} \end{aligned} \quad (8)$$

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 [17] 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 [3, 4].

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

$$J(y) = \sum_{ij \in E} y_{ij} \cdot \underbrace{log\left(\frac{p(y_{ij} = 0)}{p(y_{ij} = 1)}\right) + log\frac{1-\beta}{\beta}}_{w_{ij}} \quad (9)$$

where β is used as a prior [3].

We use the publicly available benchmark instances from [18, 17]. For 2D images from the Berkeley Segmentation Database [26], we took the segmentation problems called *image-seg* [3, 18]. For 3D volume segmentation we use the models *knott-3d-150*, *-300* and *-450* from [4, 17] as well as the large instance from the *3d-seg* model [3, 18]. 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 [4] who kindly provided us the dataset *knott-3d-550* with cube size 550^3.

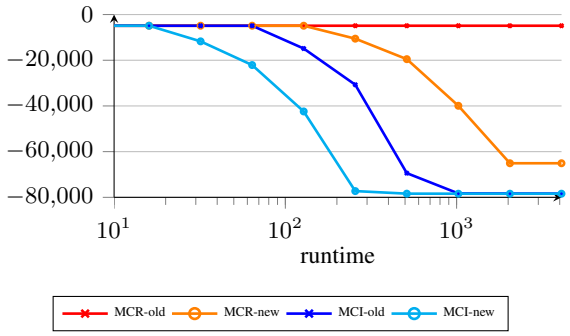


Figure 7: Comparison of the Multicut implementation of OpenGM and our modified implementation, which improves the runtime. However, for large scale problems it still does not scale.

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* [32] within the shortest path search by the Dykstra algorithm, which speeds up MC-R. Secondly, we follow [4] 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 [4] 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. 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.

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. 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. 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. 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.

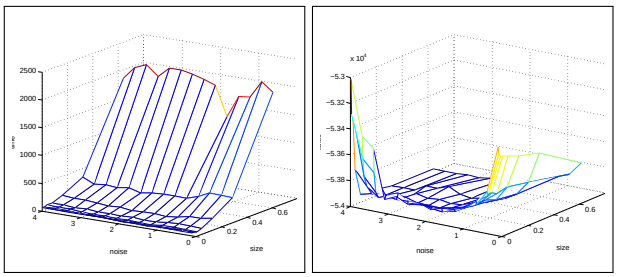


Figure 8: Empirical evaluation of the impact of noise used for proposal generation and the size of the proposals. Proposals with many segments cause longer runtime. Noise seemed not to be a critical parameter but should be selected large enough.

5.4. Evaluation

For the evaluation of the different methods on the datasets introduced in Sec. 5.1, we show zoomed anytime plots in Fig. 9 and variation of information (VOI) [27] and rand index (RI) [30] of the final solutions in Tab. 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. 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. 9(f).

Table 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**

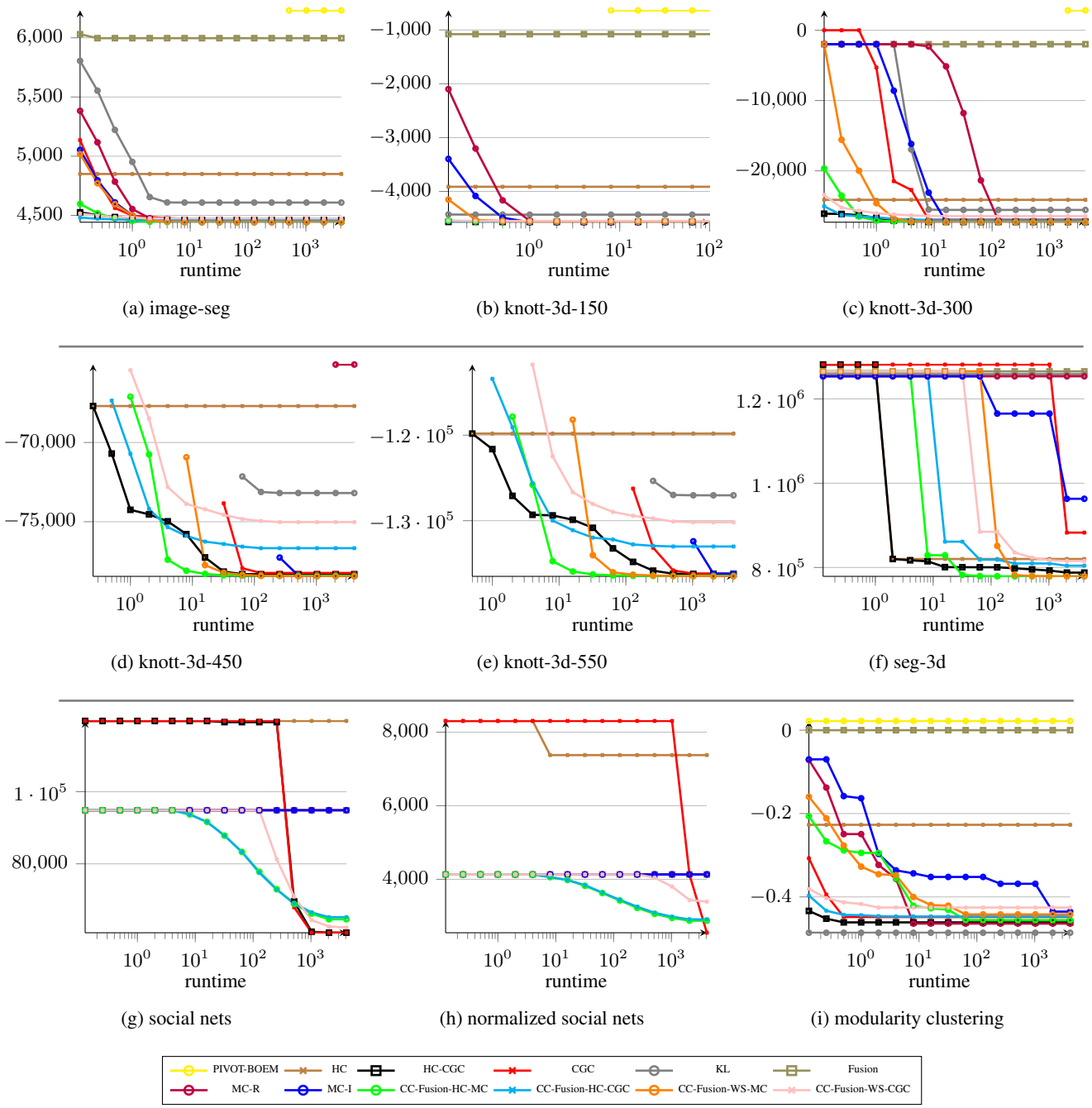
However, after a few minutes CC-Fusion-HC-MC and CC-Fusion-WS-MC give much better results and are also overall best in terms of energy, VOI, and RI. Pure EHC, Fusion and PIVOT-BOEM do not give useful results on any dataset.

6. Future Work

In future work we would like to investigate much larger problem instances and interactive correlation clustering, by enforcing updates only in local regions by appropriate proposals. Overall, a more specific selection of proposals, maybe conditioned on the current solution, should lead to better proposals and to faster progress of CC-Fusion. Furthermore, we can extend our approach to higher-order correlation clustering [23, 21] by using higher order subproblems.

7. 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 state-of-the-art 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 9: Among all proposed solvers, Fusion-HC-MC has the best overall anytime performance. With increasing problem size (9b-9e and 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. 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.

References

[1] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5):23:1–23:27, Nov. 2008. 1, 2, 5

[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. 1

[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, 6

[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, 6

[5] A. Arasu, C. Re, and D. Suciu. Large-scale deduplication with constraints using dedupalog. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009*. IEEE Computer Society, March 2009. 1

[6] P. 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, pages 182–, Washington, DC, USA, 2006. IEEE Computer Society. 2

[7] S. Bagon and M. Galun. Large scale correlation clustering optimization. *CoRR*, abs/1112.2903, 2011. 1, 5

[8] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. In *MACHINE LEARNING*, pages 238–247, 2002. 1

[9] T. Beier, T. Kroeger, J. H. Kappes, U. Koethe, and F. A. Hamprecht. Cut, Glue & Cut: A Fast, Approximate Solver for Multicut Partitioning. In *IEEE Conference on Computer Vision and Pattern Recognition 2014*, 2014. 1, 3, 4, 5

[10] Y. Chen, S. Sanghavi, and H. Xu. Clustering sparse graphs. In *NIPS*, pages 2213–2221, 2012. 1

[11] F. Chierichetti, N. N. Dalvi, and R. 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*, pages 641–650, 2014. 1

[12] S. Chopra and M. Rao. The partition problem. *Mathematical Programming*, 59(1-3):87–115, 1993. 1, 2

[13] C. Couprie, L. Grady, L. Najman, and H. Talbot. Power watershed: A unifying graph-based optimization framework. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(7):1384–1399, July 2011. 4

[14] M. Elsner and E. Charniak. You talking to me? A corpus and algorithm for conversation disentanglement. In *Proceedings of ACL-08: HLT*, pages 834–842, Columbus, Ohio, June 2008. Association for Computational Linguistics. 2

[15] M. Elsner and W. 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, pages 19–27, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. 2

[16] A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *ACM Trans. Knowl. Discov. Data*, 1(1), Mar. 2007. 2, 5

[17] J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, T. Kröger, J. Lellmann, N. Komodakis, B. Savchynskyy, and C. Rother. A comparative study of modern inference techniques for structured discrete energy minimization problems. *International Journal of Computer Vision*, pages 1–30, 2015. 6

[18] J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, J. Lellmann, N. Komodakis, et al. A comparative study of modern inference techniques for discrete energy minimization problems. In *CVPR*, 2013. 6

[19] 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. 2

[20] 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. 1, 4

[21] J. H. Kappes, M. Speth, G. Reinelt, and C. Schnörr. Higher-order segmentation via multicuts. *CoRR*, abs/1305.6387, 2013. 1, 2, 4, 5, 7

[22] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Sys. Tech. J.*, 49(2):291–308, 1970. 1, 5

[23] S. Kim, S. Nowozin, P. Kohli, and C. D. Yoo. Higher-order correlation clustering for image segmentation. In *NIPS*, pages 1530–1538, 2011. 1, 7

[24] V. Lempitsky, C. Rother, S. Roth, and A. Blake. Fusion moves for Markov random field optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1392–1405, aug 2010. 1, 2, 3, 4, 5

[25] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1361–1370, New York, NY, USA, 2010. ACM. 1

[26] 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 *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001. 6

[27] M. Meila. Comparing clusterings by the variation of information. In B. Schölkopf and M. K. Warmuth, editors, *Learning Theory and Kernel Machines*, volume 2777 of *Lecture Notes in Computer Science*, pages 173–187. Springer Berlin / Heidelberg, 2003. 7

[28] F. Meyer. Watersheds on edge or node weighted graphs "par l'exemple". *CoRR*, abs/1303.1829, 2013. 4

[29] V. Ng and C. Cardie. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 104–111, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. 2

[30] W. M. Rand. Objective criteria for the evaluation of clustering methods. *J. of the American Statistical Association*, 66(336):846–850, 1971. 7

[31] E. Sadikov, J. Madhavan, L. Wang, and A. Halevy. Clustering query refinements by user intent. In *Proceedings of the*

19th International Conference on World Wide Web, WWW '10, pages 841–850, New York, NY, USA, 2010. ACM. 1

[32] R. Sedgewick and K. Wayne. *Algorithms, 4th Edition.* Addison-Wesley, 2011. 6

[33] W. M. Soon, H. T. Ng, and D. C. Y. Lim. A machine learning approach to coreference resolution of noun phrases. *Comput. Linguist.*, 27(4):521–544, Dec. 2001. 2

[34] C. N. Straehle, U. Kthe, G. Knott, K. L. Briggman, W. Denk, and F. A. Hamprecht. Seeded watershed cut uncertainty estimators for guided interactive segmentation. In *CVPR'12*, pages 765–772, 2012. 5

[35] J. Yarkony, A. Ihler, and C. C. Fowlkes. Fast planar correlation clustering for image segmentation. In *ECCV*. Springer, 2012. 1, 5