# Fusion Moves for Correlation Clustering

Thorsten Beier
thorsten.beier@iwr.uni-heidelberg.de

Fred A. Hamprecht
fred.hamprecht@iwr.uni-heidelberg.de

Jörg H. Kappes
kappes@math.uni-heidelberg.de

## 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

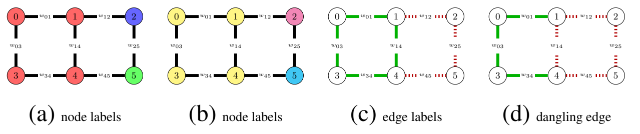


Figure 1: Representing a clustering by node labels is ambiguous. 1a and 1b encode the same partition. Edge labels as in 1c do not suffer from such ambiguities, but can have *dangling edges* as in 1d. Node 1 and 4 are in the same connected component, even tough $e_{14}$ is cut. This is not a valid partition,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

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.

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* $MC(G) = \mathrm{conv}(P(G))$. By $l(y)$ we denote some node labeling for a segmentation given by $y$.

Given a weighted graph $G = (V, E, w)$ we consider the problem of segmenting $G$ such that the costs of the edges between distinct segments is minimized. This can be formulated in the node domain by assigning each node $i$ a label $l_i \in \mathbb{N}$

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

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

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

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

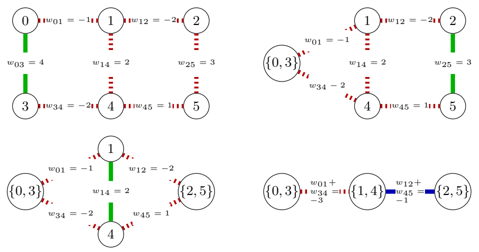


Figure 2: Energy-based Hierarchical clustering can be used to greedily optimize Eq. 2. In each step, the two nodes connected via the edge with the highest weight are merged by contracting this edge. (edge to be contracted is shown in green). Due to edge contraction parallel edges can occur, which are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

## 3. Energy Based Hierarchical Clustering

We now describe a fast heuristic for solving Eq. 2 which will be used later as a subroutine. 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 weighted mean of the edge weights is used [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 $y''$, $E_0^{\breve{y}}$ is the set of edges which are uncut in $y'$ and $y''$.

$$\breve{y}_{ij} = \max\{y'_{ij}, y''_{ij}\} \qquad \forall ij \in E \tag{3}$$
$$E_0^{\breve{y}} = \{ij \in E \mid \breve{y}_{ij} = 0\} \tag{4}$$

The fusion move for correlation clustering is solving Eq. 2 with additional *must-link constraints* for all edges in $E_0^{\breve{y}}$.

$$\begin{aligned} y^* = \arg\min_{y \in P(G)} & \sum_{(i,j)\in E} w_{ij} \cdot y_{ij}. \\ \text{s.t.} \quad & y_{ij} = 0 \qquad \forall (i,j) \in E_0^{\breve{y}} \end{aligned} \tag{5}$$

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, the proposed algorithm 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.

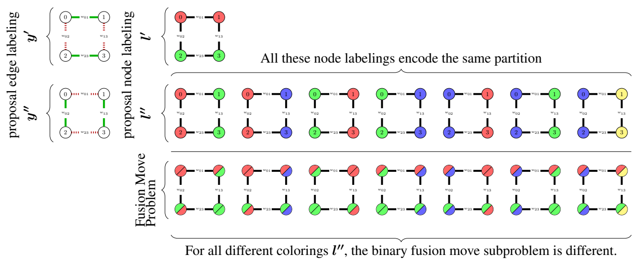

proposal edge labeling
$y'$
$y''$
proposal node labeling
$l'$
$l''$
All these node labelings encode the same partition
Fusion Move Problem
For all different colorings $l''$, the binary fusion move subproblem is different.

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.

### 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*.

**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

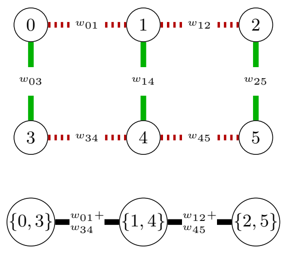
$$\tilde{y}_{ij} = \begin{cases} \bar{y}_{l_i(y)l_j(y)} & \text{if } l_i(y) \neq l_j(y) \\ 0 & \text{else} \end{cases} \qquad \forall uv \in E \tag{6}$$

**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.*

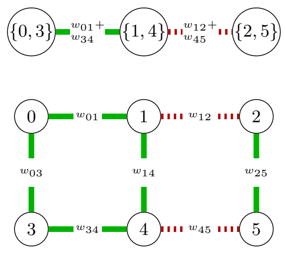
*Proof.* Let $y'$ be the back propagation of $\bar{y}'$, which is by definition feasible for 5. If $y'$ would not be an optimal solution, there must be a $y''$ with $\sum_{e\in E} w_e y'_e > \sum_{e\in E} w_e y''_e$. Since $y'_e$ and $y''_e$ are 0 for all $e \in E_0^{\breve{y}}$ we would have

$$\begin{aligned} \sum_{\bar{e}\in E_y} \bar{w}_{\bar{e}} \bar{y}'_{\bar{e}} &= \sum_{e\in E\setminus E_0^{\breve{y}}} w_e y'_e = \sum_{e\in E} w_e y'_e \\ &> \sum_{e\in E} w_e y''_e = \sum_{e\in E\setminus E_0^{\breve{y}}} w_e y''_e = \sum_{\bar{e}\in E_y} \bar{w}_{\bar{e}} \bar{y}''_{\bar{e}} \end{aligned}$$

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



(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.

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 $G_{\breve{y}}$ can be solved by any correlation clustering solver. Since $G_{\breve{y}}$ 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*.