Cut, Glue & Cut: A Fast, Approximate Solver for Multicut Partitioning

Thorsten Beier* Thorben Kroeger* Jörg H. Kappes Ullrich Köthe Fred A. Hamprecht
firstname.lastname@iwr.uni-heidelberg.de, kappes@math.uni-heidelberg.de

Abstract

Recently, unsupervised image segmentation has become increasingly popular. Starting from a superpixel segmentation, an edge-weighted region adjacency graph is constructed. Amongst all segmentations of the graph, the one which best conforms to the given image evidence, as measured by the sum of cut edge weights, is chosen.

Since this problem is NP-hard, we propose a new approximate solver based on the move-making paradigm: first, the graph is recursively partitioned into small regions (cut phase). Then, for any two adjacent regions, we consider alternative cuts of these two regions defining possible moves (glue & cut phase). For planar problems, the optimal move can be found, whereas for non-planar problems, efficient approximations exist.

We evaluate our algorithm on published and new benchmark datasets, which we make available here. The proposed algorithm finds segmentations that, as measured by a loss function, are as close to the ground-truth as the global optimum found by exact solvers. It does so significantly faster then existing approximate methods, which is important for large-scale problems.

1. Introduction

Segmentation is an important problem in computer vision as a first step towards understanding an image. Many algorithms start with an over-segmentation into superpixels, which are then clustered into "perceptually meaningful" regions. Usually, the number of these regions is not known beforehand.

Recently, the multicut formulation [9] (sometimes called *correlation clustering*, [8]) has become increasingly popular for unsupervised image segmentation. Given an edge-weighted region adjacency graph, the problem is to find the segmentation which minimizes the cost of the cut edges. Such an approach has been shown to yield state-of-the-art results on the Berkeley Segmentation Database [4, 23, 2].

Unfortunately, solving the multicut problem is in general NP-hard. Any exact solver will therefore be plagued by scalability issues. However, segmentation of images with ever finer superpixel partitionings and of large volume images in computational neuroscience [5] demand solutions to large scale multicut problems. In this regime of large-scale problems, existing solvers either fail to find any solution after a reasonable time, rely on suitable edge weights (such that the problem decomposes naturally into independent subproblems) or yield an approximate solution possibly far away from the optimum.

*Authors contributed equally.

Contribution. In this work we consider (i) a new perspective on solving the multicut problem by *local* move making methods together with (ii) a new approximate multicut solver called *Cut, Glue & Cut (CGC)*. Furthermore, (iii) our method avoids re-solving the same moves by tracking their "dirtyness", which decreases the runtime. (iv) An extensive evaluation on existing and new benchmark datasets shows that CGC provides results close to optimality and equal application performance significantly faster than all its competitors, both exact and approximative methods, and gives new insights concerning the applicability of competing methods. (v) Our C++ implementation of CGC is part of the open-source OpenGM inference library [3].

Organization. In Sec. 2 we review two formulations of the multicut problem. After discussing related work (Sec. 3), we review the max-cut problem (Sec. 4). Based on a general formulation of local partition moves on segmentations in Sec. 5, we describe our new Cut, Glue & Cut algorithm in Sec. 6. Extensive experiments on benchmark datasets are presented in Sec. 7. Finally, we discuss our findings.

2. Problem Formulation

Let $G = (V, E, w)$ be a weighted region adjacency graph of nodes V, representing superpixels, and edges E. The function $w : E \to \mathbb{R}$ assigns a weight to each edge. 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 different regions.

A *subgraph* $G_A = \{A, E_A, w\}$ consists of nodes $A \subseteq V$ and edges $E_A := E \cap (A \times A)$. We call a connected component of nodes V a *region* $R \subseteq V$. The edges between A and B form the set $E_{A-B} := E \cap (A \times B)$. We use the shorthand $\overline{A} := V \setminus A$ for the *complement* of A.

Each node i is assigned a label $\mathbf{L}_i \in \{0, \ldots, |V|-1\}$. Using the indicator function $\delta(\cdot)$, the *multicut problem* can be written as a node labeling problem [7]:

$$\underset{\mathbf{L}}{\operatorname{argmin}} \left\{ \sum_{e=(i,j)\in E} w(e) \cdot \delta(\mathbf{L}_i \neq \mathbf{L}_j) \right\}, \qquad (1)$$

where $\delta(a) = 1$ if a is true and 0 else. While (1) looks like a common Potts energy without unary terms, there are some crucial differences:
First, any weight $w(e \in E)$ can be positive or negative.
Second, the lack of any unary data terms renders the problem much harder and introduces ambiguity.
Third, the label space is large. In general we have to set $|\mathbf{L}| = |V|$ to allow for the solution where every supervoxel becomes its own region without having to solve the graph coloring problem implicitly. For planar graphs, it would be sufficient that $\mathbf{L}_i \in \{0, ..., 3\}$ because any planar-graph is 4-colorable [6].

An alternative formulation of problem (1) is in terms of binary edge indicator variables $\boldsymbol{y} \in \{0,1\}^{|E|}$:

$$\underset{\boldsymbol{y}}{\operatorname{argmin}} \underbrace{\left\{ \sum_{e=(i,j)\in E} w(e) \cdot \boldsymbol{y}_e \right\}}_{\text{CUT}_G(\boldsymbol{y})} \quad \text{s.t.} \quad \boldsymbol{y} \in \text{MC}_G. \qquad (2)$$

A segmentation $\boldsymbol{y}$ has an associated *energy* which is denoted by $\text{CUT}_G(\boldsymbol{y})$. MC_G is the set of all valid multicuts of G; its convex hull forms the so called *multicut polytope* [9]. Restricting $\boldsymbol{y}$ to MC_G ensures that the labeling is *consistent*: if $\boldsymbol{y}_{ij} = 1$ then nodes i and j should be in separate regions even after connected component labeling. Intuitively, dangling line segments are forbidden in two dimensional images, and punctured walls or faces are ruled out in three dimensional images.

While any segmentation $\bigcup_i R_i \equiv V$ is represented by exactly one edge labeling $\boldsymbol{y} \in \text{MC}_G$, many labelings $\mathbf{L}$ represent the same segmentation. Therefore, a multicut representation (2) is preferrable to avoid large multiplicities of local optima. However, an efficient handling of the multicut polytope is essential.

3. Related Work

Applying state-of-the-art solvers for the labeling problem (1) is challenging since any permutation of the labelings transforms an optimal solution into another optimal solution, see [15] for more details and a recent review.

In general, $\boldsymbol{y} \in \text{MC}_G$ can be enforced by an exponential number of constraints [9], but in practice – for a given objective function – a small subset of those are sufficient. Therefore, a major branch of research has focused on cutting plane approaches, either by solving a relaxation of (2) by a sequence of linear programs [17, 18, 15, 11] or by solving problem (2) exactly by a sequence of integer linear programs [14, 4, 5, 15, 1, 2].

For the latter, one can avoid the cutting plane procedure by lifting to the fully connected graph at the expense of problem size [1]. For many image segmentation problems, however, the size of the complete graph is prohibitively large.

Alush and Goldberger [1, 2] suggest to use partial optimality in order to decompose the problem into its positively connected components, defined as the connected components of the graph $G' = (V, E')$ with $E' = \{e \in E | w_e > 0\}$. These problems are then separately solved to global optimality. While this recovers the true solution, it depends on the weights w whether a significant reduction of the problem is possible.

Yarkony et al. [23] suggest a dual column-generating method, called PlanarCC, which solves a relaxed linear program by iteratively solving weighted two-coloring problems. Because these sub-problems require planarity to be tractable, PlanarCC is restricted to problems with planar structure.

Another branch of research has focused on specialized move-making methods [16, 24, 7] that take the degeneracy of the solution, due to label permutations in (1), into account. Move-making algorithms maintain a valid solution (segmentation defined via $\mathbf{L}$ or, equivalently, $\boldsymbol{y} \in \text{MC}$) throughout the optimization procedure. In each step, a set of possible moves transforming the current segmentation is considered, and the move which realizes the maximal energy decrease is chosen. Therefore, the energy decreases monotonically until a (local) optimum is found. The Kerninghan-Lin method [16], used in circuit-layout design, applies a sequence of greedy local moves to neighboring regions. Bagon [7] recently proposed an extension to the α-expansion move-making algorithm for solving multicut problems, which handles the large label space and label ambiguity in (1) by using dynamic label sets and additional label fixing, respectively.

4. Max-Cut

The max-cut problem [10] (also known as the weighted 2-coloring problem) is a basic subproblem which has to be solved in the Kerninghan-Lin method [16], Expand-and-Explore [7], PlanarCC [23] and also our method.

The max-cut problem[1] is the specialization of (1) for the case of binary node labels $\mathbf{L}_i = \{0, 1\}$, given by

$$\underset{\mathbf{L}\in\{\mathbf{0},\mathbf{1}\}^{|\mathbf{E}|}}{\operatorname{argmin}} \left\{ \sum_{e=(i,j)\in E} w(e) \cdot \delta(\mathbf{L}_i \neq \mathbf{L}_j) \right\}. \qquad (3)$$

[1]For reasons of consistency, we consider wlog. minimization instead of maximization.

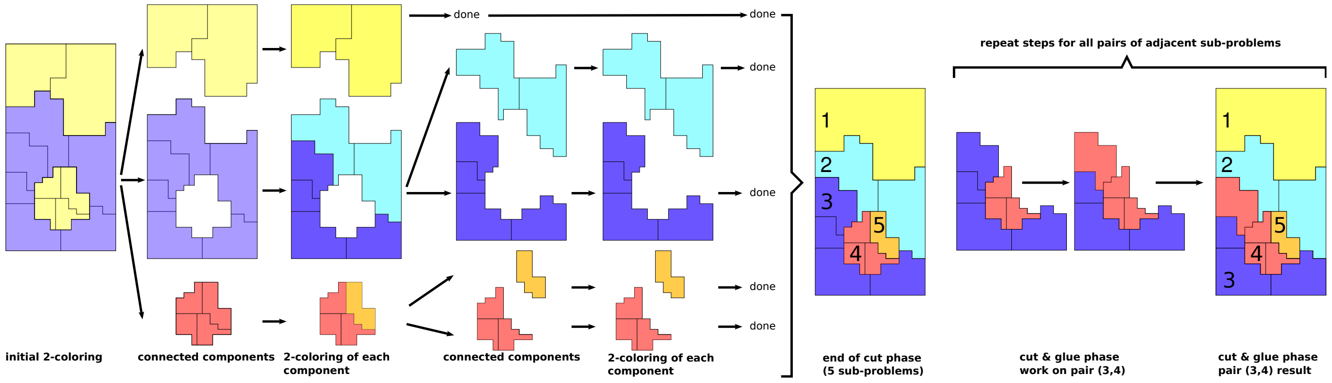

done
done
done
done
done
done
repeat steps for all pairs of adjacent sub-problems
1
2
3
5
4
1
2
5
4
3
initial 2-coloring
connected components
2-coloring of each component
connected components
2-coloring of each component
end of cut phase (5 sub-problems)
cut & glue phase work on pair (3,4)
cut & glue phase pair (3,4) result

Figure 1: *Left*: Illustration of the *cut phase*. The steps are visualized as a tree. First, a weighted 2-coloring problem (3) is solved on the initial superpixel adjacency graph of an image. A connected-component labeling yields the child nodes. For each node, the sequence of solving (3) and connected component labeling are repeated. Whenever a weighted 2-coloring yields only a single component, the energy cannot be decreased any further and the node has no children ("done"). *Right*: Illustration of the *glue&cut phase*. In the illustrated move, the regions labeled 3 and 4 are first merged and then a new, better partition is sought, which leads to a better global segmentation. This step is repeated until no more improvement is possible.

This should not be confused with the min-cut problem, which refers to sub-modular "graphcut problems" with non-negative edge weights w, for which a max-flow problem can be formulated [19]. Intuitively, restricting the label space to two labels (0 and 1) simplifies the problem. For planar graphs, (3) can be solved by the Blossom algorithm [22] in polynomial time. In general this problem is still NP-hard. In response, Kerninghan and Lin have suggested a greedy method for (3) used in the KL algorithm [16]. Recently, Bagon [7] suggested to use QPBO-I [21] for (3). While this LP relaxation often gives non-integral solutions for many nodes, the improving extension (-I) [21] iteratively fixes such nodes and therefore also deals with the ambiguity of the problem.

In our experiments, we found that using QPBO-I performs better than the greedy method used in the KL algorithm, but slightly worse than optimal solvers (Sec. 7).

5. Partition Moves

In this section we define a class of moves which can be used to iteratively improve segmentations. The Partition Move Theorem shows that under some technical conditions, the improvement of the local sub-problem leads to monotonous improvement of the global energy.

Theorem: *Partition Moves.* Let $\boldsymbol{y}^t \in \mathrm{MC}_G$ represent the segmentation of G at step t. Local partition moves are defined over a subset $S \subset V$ for which all edges crossing its borders have to be labeled one: $\forall\, e \in E_{S-\overline{S}} : \boldsymbol{y}^t_e = 1$.
Let G_S be the respective subgraph of G and $\hat{\boldsymbol{y}} \in \mathrm{MC}_{G_S}$ represent a given valid segmentation of S. For the combined segmentation

$$\boldsymbol{y}^{t+1} := \begin{cases} \hat{\boldsymbol{y}}_e & e \in E_S \\ \boldsymbol{y}^t_e & \text{else} \end{cases} \tag{4}$$

the following holds:
(i) $\boldsymbol{y}^{t+1} \in \mathrm{MC}_G$.
(ii) $\mathrm{CUT}_{G_S}(\hat{\boldsymbol{y}}_{E_S}) \le \mathrm{CUT}_{G_S}(\boldsymbol{y}^t_{E_S}) \Rightarrow \mathrm{CUT}_G(\boldsymbol{y}^{t+1}) \le \mathrm{CUT}_G(\boldsymbol{y}^t)$.

Proof. (i) Let us extend the local segmentation $\hat{\boldsymbol{y}}$ to G with

$$\hat{\boldsymbol{y}}_G := \begin{cases} \hat{\boldsymbol{y}}_e & \forall e \in E_S \\ 1 & \forall e \in E_{S-\overline{S}} \\ 0 & \forall e \in E_{\overline{S}} \end{cases}$$

Since (a) $\hat{\boldsymbol{y}}_G \in \mathrm{MC}_G$, (b) $\boldsymbol{y}^{t+1} = \mathrm{OR}(\hat{\boldsymbol{y}}_G, \boldsymbol{y}^t)$, and (c) the set of multicuts is closed under the OR operation, (i) holds.
(ii) For $\mathrm{CUT}_{G_S}(\hat{\boldsymbol{y}}_{E_S}) \le \mathrm{CUT}_{G_S}(\boldsymbol{y}^t_{E_S})$, the energy after the move can be written as:

$$\begin{aligned} \mathrm{CUT}_G(\boldsymbol{y}^{t+1}) &= \sum_{e\in E_S} w(e)\boldsymbol{y}^{t+1}_e + \sum_{e\in E\setminus E_S} w(e)\boldsymbol{y}^{t+1}_e \\ &\le \sum_{e\in E_S} w(e)\boldsymbol{y}^t_e + \sum_{e\in E\setminus E_S} w(e)\boldsymbol{y}^{t+1}_e \\ &= \sum_{e\in E_S} w(e)\boldsymbol{y}^t_e + \sum_{e\in E\setminus E_S} w(e)\boldsymbol{y}^t_e = \mathrm{CUT}_G(\boldsymbol{y}^t) \quad \square \end{aligned}$$

To obtain an alternative segmentation $\hat{\boldsymbol{y}}$ on G_S, one can either (a) solve the multicut problem over G_S (which is smaller than G) or (b) restrict the subset of possible segmentations, e.g. to all two-colorable segmentations of G_S. If we solve the problem to optimality and the current segmentation is in the feasible set of the move, it is guaranteed that $\boldsymbol{y}^{t+1}$ never increases the energy. For approximate solutions this is not the case; here the move should only be accepted

Algorithm 1: Cut, Glue & Cut algorithm

```
Input: weighted graph $G=(V,E,w)$
Output: approx. solution $Q$ to (1) for $G$
1 $Q^0 \leftarrow$ segmentation of $G$ into positively connected
  components
2 for $n = 1 \dots n_{iter}$ do
3     $Q^n \leftarrow$ cut_phase$(G, Q^{n-1})$    //Alg. 2
4     $Q^n \leftarrow$ glue_cut_phase$(G, Q^n)$    //Alg. 3
5     if $Q^n = Q^{n-1}$ then
6         exit
```

in case of energy decrease. While (b) restricts the set of possible moves, the problems become easier and in most cases, a specific energy-decreasing sequence of two-coloring moves (b) can generate the same segmentation as one single N-coloring move (a) as sketched in Fig. 2. However, even if an energy-decreasing sequence exists, finding this sequence itself is a hard combinatorial optimization problem, which is why we choose a greedy optimization approach.

6. Cut, Glue & Cut Algorithm

Overview The CGC algorithm (Alg. 1) always maintains a valid partitioning of the graph, which is iteratively improved by *local* moves which act on single or neighboring regions. It works in two distinct phases: (i) recursive *cut phase* (ii) *glue & cut phase*.

The cut phase recursively splits regions by solving max-cut problems (3), finally yielding a finer, lower energy segmentation driven by the problem's weights. In the glue & cut phase, any two neighboring segments are first merged and then a new cut is sought between them by solving (3). These two phases are repeated and the process stops when the energy cannot decrease any further.

Cut Phase In the *cut phase*, the regions are recursively split into smaller and smaller regions tuned to the weights w until a local optimum is found. As *cut moves* we consider the max-cut solution (3) for a single given region R to find

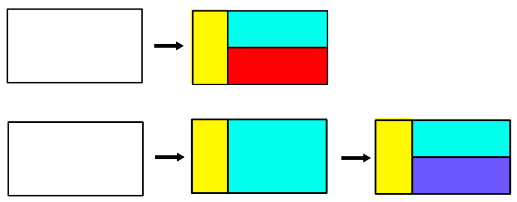


Figure 2: *Top:* Solving the labeling problem (1) with more than two colors allows for T-junctions. *Bottom:* A sequence of two colorings can model the same T-junction and is often energy-decreasing.

a better segmentation of the region $\hat{y} \in \text{MC}_{G_R}$, and accept the local move if the energy could be decreased compared to the previous solution $\text{CUT}_{G_R}(\boldsymbol{y}_R) = 0$. The Partition Move Theorem then guarantees a monotonically decreasing global energy for the segmentation.

The *cut phase* is illustrated in the left part of Fig. 1, and is given as Algorithm 2. All regions are first inserted into a queue Q. While Q is not empty, take a region $R \in Q$, then solve the max-cut problem (3) for G_R. Finally, a connected component labeling of the solution defines a new set of regions[2]. If the suggested cut reduces the local energy, we add the induced regions to Q. Otherwise, the energy of region R cannot further be reduced by this type of move; region R is marked as "done" and added to a list Q'.

Algorithm 2: Cut phase

```
Input: weighted graph $G = (V, E, w)$, segmentation
       into regions given as queue $Q$
Output: segmentation into smaller regions $Q'$
1 $Q' \leftarrow \emptyset$
2 while $Q \neq \emptyset$ do
3     $R \leftarrow$ queue_pop$(Q)$
4     $\boldsymbol{y}_{E_R} \leftarrow$ solve max-cut (3) for $G_R$
5     if $CUT_{G_R}(\boldsymbol{y}_{E_R}) < 0$ then
6         $Q \leftarrow Q \cup$ connected_components$(\boldsymbol{y}_{E_R})$
7     else
8         $Q' \leftarrow Q' \cup R$
```

Glue & Cut Phase In this phase, we consider *Glue & Cut moves* of pairs of adjacent segments until the energy cannot be decreased any further, yielding a better solution.

Intuitively, we expect that two common local operations can decrease the energy: (i) merging two segments or (ii) moving the boundary between two adjacent segments. This motivates our algorithm:

We again apply the Partition Move Theorem from Sec. 5. Given two adjacent regions R_1, R_2 in the current segmentation, we consider the *merged region* $R = R_1 \cup R_2$ (*glue*), and find a new segmentation $\hat{\boldsymbol{y}} \in \text{MC}_{G_R}$ (*cut*) by solving the max-cut problem (3). The local move is accepted if the energy $\text{CUT}_{G_R}(\hat{\boldsymbol{y}})$ is lower than the energy of the previous cut. The Partition Move Theorem then guarantees a monotonically decreasing energy for the segmentation.

Formally, (Alg. 3 and Fig. 1, right), the glue & cut phase starts from a given segmentation Q. We first obtain its edge labeling $\bar{\boldsymbol{y}}$. Initially, all edges $e \in E$ are marked "dirty".

[2]Note that in general there may be more than two connected regions, cf. Fig. 1, initial 2-coloring.

[3]Note that by using E' instead of E, we consider only one representative edge for each boundary between two regions for performance reasons.

Algorithm 3: Glue & Cut phase

```
**Input**: weighted graph $G=(V,E,w)$, segmentation $Q$
**Output**: improved segmentation $Q$ wrt. (1)
1  mark all edges $e \in E$ as dirty
2  $\bar{\boldsymbol{y}} \leftarrow$ edge_labeling($Q$)
3  while true do
4      $c \leftarrow 0$
5      $E' \leftarrow \{e \in E_{R_1-R_2} | R_1, R_2 \in Q$ adjacent$\}$[3]
6      for $e=(i,j) \in E'$ do
7          if $\bar{\boldsymbol{y}}_e = 0$ or $e$ is clean then
8              continue
9          find regions $R_1, R_2 \in Q$, s.t. $i \in R_1$, $j \in R_2$
10         $S \leftarrow R_1 \cup R_2$    //glue
11         $\boldsymbol{y}_{E_S} \leftarrow$ solve (3) for $G_S$    //cut
12         mark edges $E_S$ clean    $(\star)$
13         if $CUT_{G_S}(\boldsymbol{y}_{E_S}) < CUT_{G_S}(\bar{\boldsymbol{y}}_{E_S})$ then
14             $c \leftarrow c+1$
15             mark edges $E_{S-\overline{S}}$ dirty    $(\star)$
16             CC $\leftarrow$ connected_components($\boldsymbol{y}_{E_S}$)
17             if $|CC| > 2$ then
18                 mark edges $E_S$ dirty    $(\star)$
19             $Q \leftarrow Q \setminus \{R_1, R_2\} \cup$ CC
20             $\bar{\boldsymbol{y}} \leftarrow$ edge_labeling($Q$)
21     if $c = 0$ then
22         break
```

The following is repeated (line 3): For each pair (R_1, R_2) of adjacent regions we pick a single representative edge from the shared boundary $E \cap (R_1 \times R_2)$ to form the set E' (line 5). Then, for each such representative $e=(i,j)$ with $\boldsymbol{y}_e = 1$ and which is marked dirty, we find the best glue & cut move as described above. If, after processing all $e \in E'$, no move could be performed, we break out of the outer loop (line 22).

Book-keeping of modified boundaries. In order to avoid re-solving the same problem multiple times, the algorithm marks edges as "dirty" or "clean". If two adjacent regions are only separated by clean edges, a glue & cut move is *not* considered for this pair. In Alg. 3, statements related to book-keeping are marked with $(\star)$.

Imagine an accepted glue & cut move on region R which yields exactly two regions R_1 and R_2. We mark the boundary between R_1 and R_2, $E \cap (R_1 \times R_2)$, as clean, since re-solving (3) for $R_1 \cup R_2$ would again lead to the same two-coloring into R_1 and R_2. However, if an adjacent pair (R_1, R_3) is chosen subsequently, a glue & cut move could alter region R_1 into R_1'. Therefore, a move between R_1' and R_2 could improve the energy again. To allow this move it is necessary to mark the boundary of $R_1' \cup R_2$ as dirty. Note that for the case where an accepted move yields *more than two regions*, bookkeeping as above would not be correct and the internal edges of R have to be marked dirty.

7. Experiments

We evaluate the performance of our Cut, Glue & Cut algorithm on two different 2D segmentation benchmarks as well as on a new 3D volume segmentation benchmark

Algorithms. For our CGC algorithm, we consider three variants: CGC-$\overline{B}$ does not do book-keeping, CGC-$\overline{P}$ does not use the globally optimal Blossom solver for planar max-cut problems, but rather uses the approximate QPBO-I. CGC-$\overline{PB}$ does not do book-keeping while using QPBO-I as max-cut solver.

We compare against the following algorithms: Kerninghan-Lin (KL, [16]), Planar Correlation Clustering (PlanarCC, [23]), Expand-and-Explore [7], LP-based cutting plane method of a relaxed problem (MC-R), integer linear program based cutting plane method (MC-I), which always finds the globally optimal solution. MC-R and MC-I use facet-defining separation procedures and bounding techniques as described in [15]. We use the publicly available C++ implementation in OpenGM 2.1.1 [3] for KL, MC-R and MC-I. For Expand-and-Explore, we use the publicly available code[4] of the corresponding authors. For PlanarCC, we kindly obtained the implementation by the authors of [23]. While both are implemented in MATLAB, all computation-heavy parts are delegated to C++ functions via MEX wrappers, such that our comparison is fair.

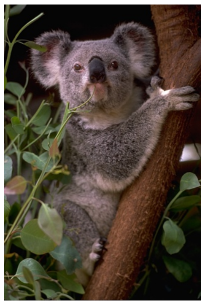
For PlanarCC, we follow the suggestion of the authors to stop the algorithm after 40 iterations for better runtime. Without this, for several instances, PlanarCC does not converge after one hour. With more iterations, the energy of the solutions improves slightly but at the expense of significantly longer runtime.

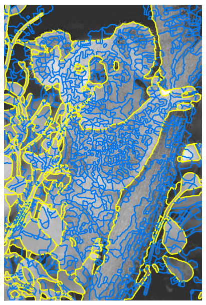
Note that in the Expand-and-Explore algorithm, each binary sub-problem includes by construction unary terms, which leads to non-planar sub-problems even when the original problem is planar.

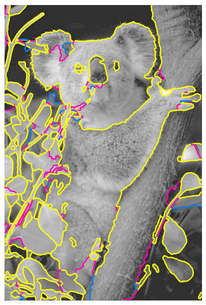
2D segmentation We consider planar 2D segmentation problems derived from the Berkeley segmentation database (BSD300): (i) models from [4] (test data, 100 instances) and (ii) models from [23] (training data, 200 instances). While the former uses local edge likelihoods learned by a Random Forest, the latter uses global probability of boundary (gPb). Furthermore, they obtain edge weights w from the edge probabilities $\mathbf{p}_e$ using different parameterizations. In [4],

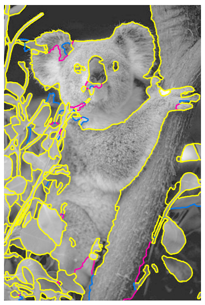
$$w_e = \log\left(\frac{\mathbf{p}(\boldsymbol{y}_e = 0)}{\mathbf{p}(\boldsymbol{y}_e = 1)}\right) + \log\frac{1-\beta}{\beta} \qquad (5)$$

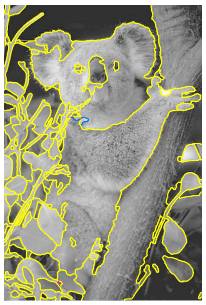
[4]http://www.wisdom.weizmann.ac.il/~bagon

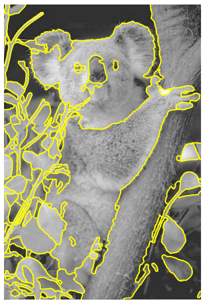

(a) original image


(b) superpixels


(c) first two coloring

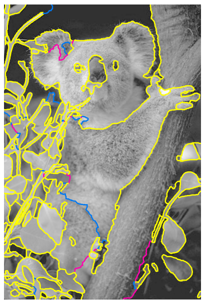

(d) end of cut phase


(e) CGC


(f) MC-I (global opt.)

(g) MC-R

(h) PlanarCC


(i) Expand & Explore

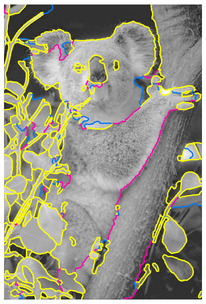

(j) Kerninghan-Lin

Figure 3: Comparison of the different multicut algorithms for a model from [4], based on the superpixel segmentation in (b). The colored boundaries indicate true positives (yellow), false negatives (red), false positives (blue) and true negatives (invisible) with respect to the globally optimal solution (f). **Top:** The image (a) is partitioned into superpixels (b). A single two-coloring leads to (c) and the cut phase ends with (d). The final output of the CGC algorithm is (e). **Bottom:** Results of various competitive methods.

is chosen while [23] use

$$w_e = \log\left(\frac{1-\mathrm{gPb}_e}{\mathrm{gPb}_e}\right) + \gamma. \tag{6}$$

Results for both datasets and different values of β and γ are shown in Fig. 4b and 4a. MC-I finds the global optimum for all instances. The left plots show the average energy distance (mean gap) to the optimum. Among all approximate methods, CGC performs best. Concerning runtimes, CGC has robustly low runtime for a wide range of parameters β, γ.

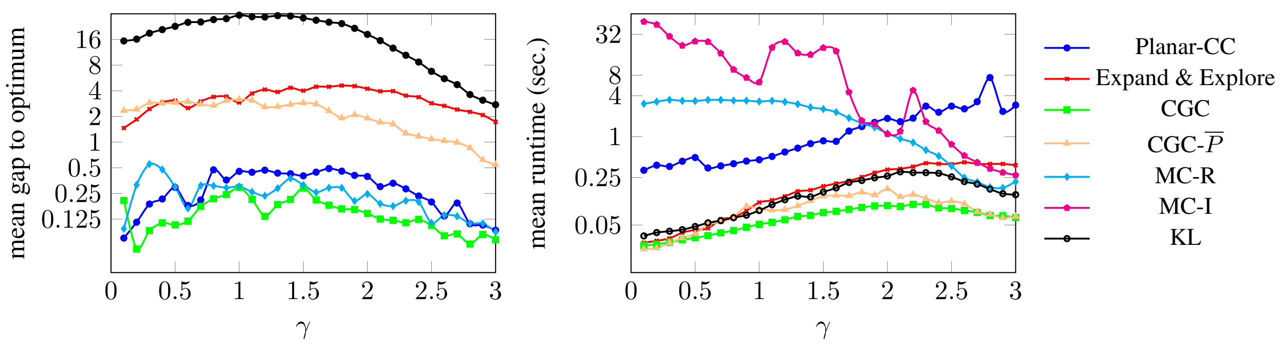
A more detailed comparison for $\beta = 0.33$, as used in [13], and $\gamma = 0.3$ can be found in Tab. 2, left and right, respectively. These tables additionally show how often the methods find the global optimum (best), and how often they were able to verify the optimality by themselves (ver. opt). For Tab. 2, left, groundtruth segmentations and superpixels were available, such that we can calculate the Variation of Information (VI, [20]). Interestingly, the globally optimal solution does not necessarily give the best segmentation as measured by the VI (but approximately, VI distance gets larger with increasing energy as expected). This means that in practice, it is sufficient to run the much faster CGC algorithm, see also Fig. 3.

Table 1: Performance of various multicut solvers on instances derived from [5] (8 instances, cube length $N = 400$ voxels). See Tab. 2 for a column legend. ver. opt. is seven for MC-I, else zero.

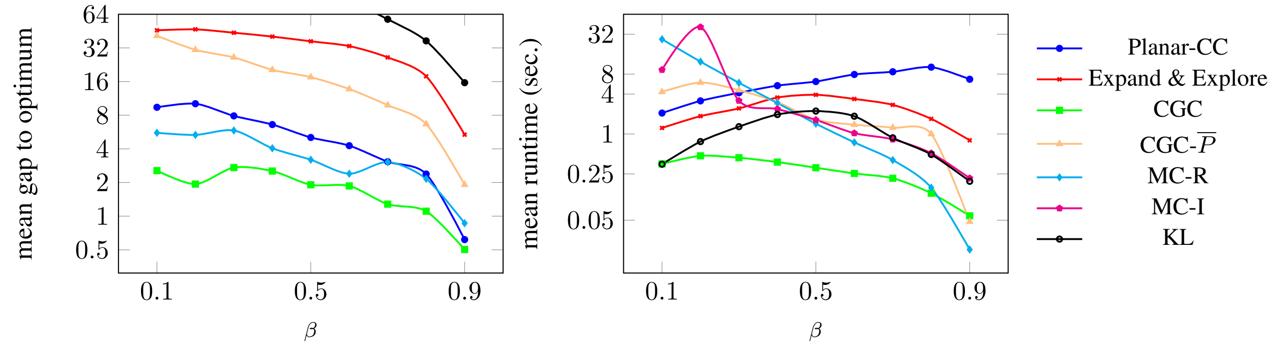
algorithm	runtime	value	bound	best	VI
ogm-KL	85.06 sec	-53476.75	$-\infty$	0	4.6930
Expand&Explore	1087.84 sec	-57054.25	$-\infty$	0	3.1228
CGC$-\overline{P}$	**64.95** sec	-57206.18	$-\infty$	1	2.1514
MC-R	4121.08 sec	-20111.68	-58774.97	0	4.0600
MC-I	745.53 sec	-57319.41	$\mathbf{-57386.73}$	**7**	**1.8539**

3D segmentation For 3D segmentation, recently Kappes et al. [13] released a benchmark dataset. However, this only includes one small and one huge instance. Therefore, we derive a more comprehensive set of instances with volumes of 30^3 up to 450^3 voxels from the models in [5], available as part of the OpenGM benchmark [12]. Results are shown in Fig. 4c and for volumes of 400^3 are detailed in Tab. 1.

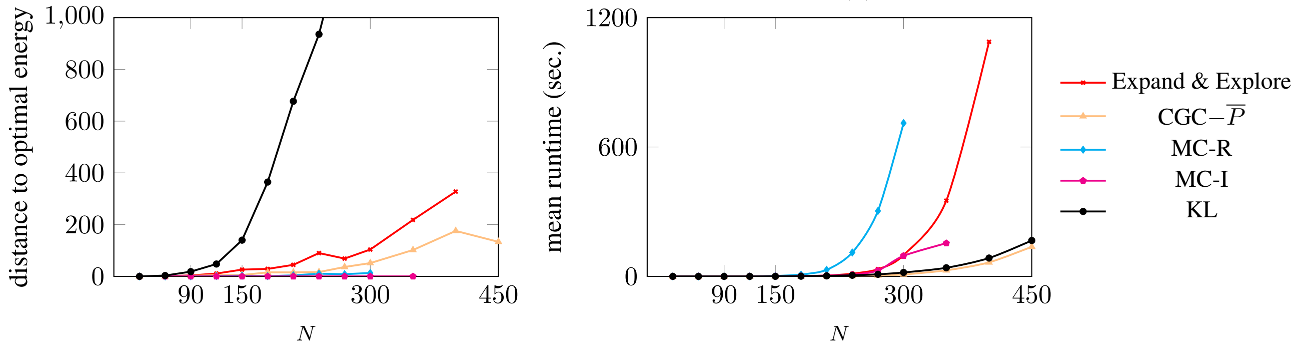
Based on the energy difference of CGC and CGC-$\overline{P}$ for the planar models, we expect reduced performance for the non-planar case of 3D volume image segmentation (Fig. 4c). Still, with a runtime as fast as KL (the fastest algorithm



(a) Evaluation on the planar model from [23] for different boundary penalties γ, averaged over 200 instances



(b) Evaluation on the planar models from [4] for different boundary penalties (β), averaged over 100 instances



(c) Evaluation on a new 3D segmentation benchmark for different volume sizes N^3 averaged over 8 instances

Figure 4: Evaluation on various benchmark datasets. **Left:** Gap to the global optimal energy (MC-I) averaged over all instances. **Right:** Mean runtime of the methods. For both 2D and 3D images, CGC outperforms all competitors in terms of runtime. On 2D images, CGC gives better results in terms of energy than all competitive approximative methods. Only Integer Multicut (MC-I) gives better partitions in terms of energy but is significantly slower. For 3D volume image segmentation, the approximate MC-R method beats CGC$-\overline{P}$, though at significant runtime cost. CGC is not applicable due to lack of planarity.

considered), CGC-$\overline{P}$ is able to obtain much lower-energy solutions and lies between MC-I, MC-R and Expand & Explore. For instances with 400^3 voxels, MC-I was only able to find for 7 out of 8 instances the global optimum within one hour (Tab. 1).

8. Conclusion

We have presented a new approximate solver, called Cut, Glue & Cut (CGC) for planar and non-planar multicut problems based on the move-making paradigm. It works in two phases: in the Cut phase, a low energy segmentation tuned to the problem's weights is created by recursively solving 2-coloring problems, either using the Blossom method (for planar problems) or QPBO-I. In the Glue & Cut phase, two adjacent sub-problems are first merged (glue) and then a possibly better cut is sought between them (cut). This process is repeated until no energy improvement is possible anymore. The experimental evaluation shows that CGC is considerably faster than existing methods while able to match an exact solver in quality, as measured by the Variation of Information on 2D images.

Table 2: Mean runtime, energy and bound (if available) for different multicut solvers. Also shown is how often each method finds the global optimum ("best"), and how often a method is able to verify the optimality by itself ("ver. opt"). Execution was aborted after one hour. Best values are marked in bold. **Left:** Summary over the models from [4], 100 instances. The VI column reports the Variation of Information [20]. **Right:** Summary over the models from [23], 200 instances. Without superpixel maps available, VI could not be calculated.

algorithm	runtime	value	bound	best	ver. opt	VI
KL	4.96 sec	4608.57	$-\infty$	0	0	2.6431
Expand & Explore	2.90 sec	4486.57	$-\infty$	1	0	2.9153
CGC-$\overline{PB}$	6.35 sec	4466.80	$-\infty$	1	0	**2.5247**
CGC-$\overline{P}$	5.35 sec	4466.80	$-\infty$	1	0	**2.5247**
CGC-$\overline{B}$	0.63 sec	4445.06	$-\infty$	23	0	2.5355
CGC	**0.42** sec	4445.06	$-\infty$	23	0	2.5355
MC-R	5.16 sec	4447.47	4442.34	35	35	2.5490
Planar-CC	5.20 sec	4450.73	4437.29	9	8	2.5603
MC-I	2.20 sec	**4442.64**	**4442.64**	**100**	**100**	2.5363

algorithm	runtime	value	bound	best	ver. opt
KL	0.04 sec	-73.41	$-\infty$	45	0
Expand & Explore	**0.03** sec	-89.90	$-\infty$	130	0
CGC-$\overline{PB}$	**0.03** sec	-89.45	$-\infty$	104	0
CGC-$\overline{P}$	**0.03** sec	-89.45	$-\infty$	104	0
CGC-$\overline{B}$	**0.03** sec	-92.25	$-\infty$	185	0
CGC	**0.03** sec	-92.25	$-\infty$	185	0
MC-R	3.48 sec	-91.70	-92.39	181	180
Planar-CC	0.36 sec	-92.16	-92.39	184	174
MC-I	29.80 sec	$\mathbf{-92.35}$	$\mathbf{-92.35}$	**200**	**200**

References

[1] A. Alush and J. Goldberger. Ensemble segmentation using efficient integer linear programming. *Pattern Analysis and Machine Intelligence*, 34(10):1966–1977, 2012. 2

[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, 2

[3] B. Andres, T. Beier, and J. H. Kappes. OpenGM: A C++ library for Discrete Graphical Models. *CoRR*, abs/1206.0111, 2012. `http://hci.iwr.uni-heidelberg.de/opengm2/`. 1, 5

[4] 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, 5, 6, 7, 8

[5] 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, 2, 6

[6] K. Appel and W. Haken. Every planar map is four colorable. *Illinois J. of Mathematics*, 21(3):429–490, 1977. 2

[7] S. Bagon and M. Galun. Large scale correlation clustering optimization. *CoRR*, abs/1112.2903, 2011. 2, 3, 5

[8] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004. 1

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

[10] M. M. Deza and M. Laurent. *Geometry of cuts and metrics*, volume 15. Springer, 1997. 2

[11] T. Finley and T. Joachims. Supervised clustering with support vector machines. In *ICML*, pages 217–224. ACM, 2005. 2

[12] 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. *CoRR*, abs/1404.0533, 2014. `http://hci.iwr.uni-heidelberg.de/opengm2/`. 6

[13] 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

[14] 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

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

[16] B. W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell system technical journal*, 49(1):291–307, 1970. 2, 3, 5

[17] S. Kim, S. Nowozin, P. Kohli, and C. D. Yoo. Higher-order correlation clustering for image segmentation. In *NIPS*, 2011. 2

[18] S. Kim, S. Nowozin, P. Kohli, and C. D. Yoo. Task-specific image partitioning. *Transactions on Image Processing*, 22(1-2):488–500, 2013. 2

[19] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004. 3

[20] M. Meilă. Comparing clusterings by the variation of information. In *Learning theory and kernel machines*, pages 173–187. Springer, 2003. 6, 8

[21] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary MRFs via extended roof duality. In *CVPR*, 2007. 3

[22] N. N. Schraudolph and D. Kamenetsky. Efficient exact inference in planar ising models. In *NIPS*, 2009. 3

[23] J. Yarkony, A. Ihler, and C. C. Fowlkes. Fast planar correlation clustering for image segmentation. In *ECCV*. Springer, 2012. 1, 2, 5, 6, 7, 8

[24] L. Zhao, H. Nagamochi, and T. Ibaraki. Greedy splitting algorithms for approximating multiway partition problems. *Mathematical Programming*, 102(1):167–183, 2005. 2