RAMA: Rapid Algorithm for Multicut and Its Extension to Lifted Multicut

Paul Swoboda

11.3.2026

The Multicut Problem

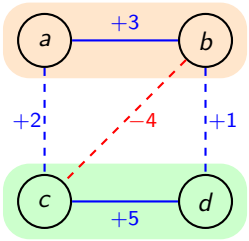
Given: Graph $G = (V, E)$ with edge costs $c : E \to \mathbb{R}$.

Goal: Find edge labeling $x \in \{0,1\}^E$ minimizing:

$$\min_{x \in \{0,1\}^E} \sum_{uv \in E} c_{uv}\, x_{uv} \quad \text{s.t. cycle inequalities}$$

- $x_{uv} = 0$: endpoints in **same** cluster
- $x_{uv} = 1$: endpoints in **different** clusters

NP-hard — equivalent to correlation clustering.



Solid = uncut, dashed = cut
Blue = attractive, red = repulsive

Optimal: $\{a, b\}$, $\{c, d\}$, cost $= -1$

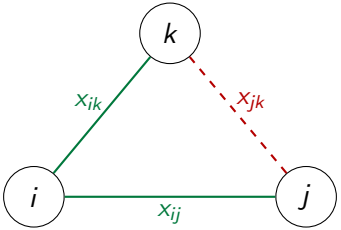
Cycle Inequalities

For any cycle C in G and any edge $e \in C$:

$$x_e \leq \sum_{e' \in C \setminus \{e\}} x_{e'}$$

Triangle inequality (most important case): For nodes i, j, k:

$$x_{ij} \leq x_{ik} + x_{jk}, \qquad x_{ik} \leq x_{ij} + x_{jk}, \qquad x_{jk} \leq x_{ij} + x_{ik}$$



Conflicted cycle: A cycle containing exactly one repulsive edge.

Dual Solver: Finding Conflicted Cycles

Conflicted cycles = cycles with exactly one repulsive edge.

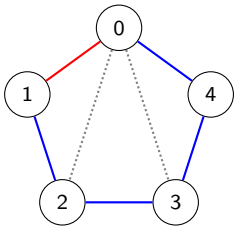
Detection:

1. **Short cycles** (length 3–5): Enumerate directly by intersecting neighbor lists — more efficient for tight constraints.
2. **Longer cycles**: For each repulsive edge (u, v), BFS in the positive subgraph finds the shortest path, closing a conflicted cycle.

Triangulation: All cycles are decomposed into triangles for uniform treatment.

- Quadrangle $\rightarrow$ 2 triangles
- Pentagon $\rightarrow$ 3 triangles

Pentagon $\rightarrow$ 3 triangles:

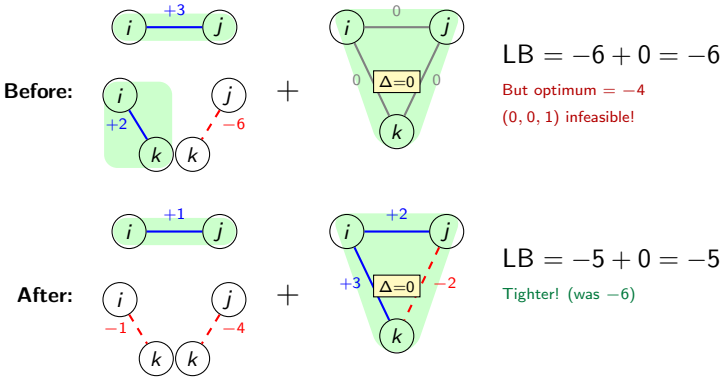


Red: repulsive edge
Blue: attractive edges
Dotted: triangulation diagonals

Δ_1=(0, 1, 2), Δ_2=(0, 2, 3), Δ_3=(0, 3, 4)

Dual Solver: Reparametrization

Reparametrization = shift cost between edges and triangle factors, preserving the objective but **tightening the bound**.



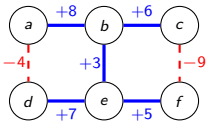
Iterate over all conflicted triangles $\Rightarrow$ LB $= \sum_e \min(0, c'_e) + \sum_\Delta \text{lb}(\Delta)$. Reparametrized costs c' guide the primal solver.

Primal Solver: Edge Contraction

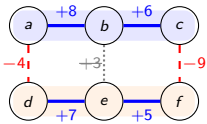
1. Build **max. spanning tree** on reparametrized costs
2. For each repulsive edge, find the **minimum-weight MST edge** on the path between its endpoints and remove it
3. **Contract** remaining MST edges
4. Repeat on coarsened graph

Blue = attractive
Red = repulsive
Bold = MST edge

1. Graph with MST

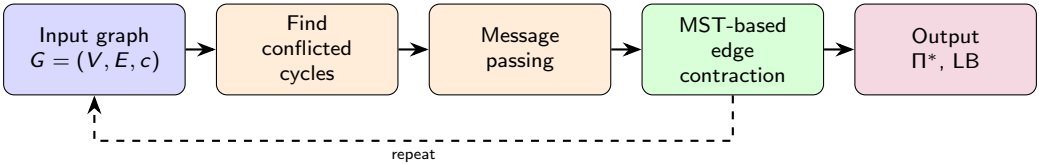


2. Remove bottleneck edge on repulsive paths



3. Contracted graph

RAMA Pipeline



The Lifted Multicut Problem

Given:

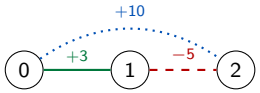
- Base graph $G_b = (V, E_b, c_b)$
- Lifted graph $G_l = (V, E_l, c_l)$

Lifted edges are long-range connections:

- Do **not** define graph connectivity
- Contribute to the objective function
- Their cut/join status is **determined** by base edge decisions

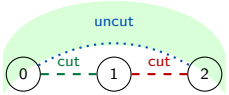
Key constraint:

$$y'_{uv} = \begin{cases} 0 & \text{if } u, v \text{ connected via uncut base edges} \\ 1 & \text{otherwise} \end{cases}$$



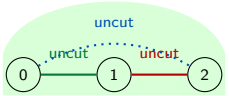
Solid: base+, dashed: base−, dotted: lifted

Union graph optimum: $\{0, 2\}, \{1\}$, cost $= -2$



Inadmissible! No base path 0–2

Optimal lifted multicut: $\{0, 1, 2\}$, cost $= 0$

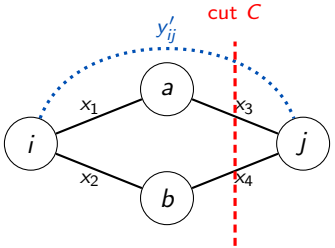


Lifted Cut Inequality

For a cut $C \subseteq E_b$ and a lifted edge $(i,j) \in E_l$ crossing the cut:

$$\sum_{e \in C} x_e - |C| + 1 \;\leq\; y'_{ij}$$

Meaning: If all base edges in C are cut ($x_e = 1$), then the lifted edge must also be cut ($y'_{ij} = 1$). Cutting fewer than $|C|$ base edges does not force the lifted edge to be cut.



$$x_3 + x_4 - 2 + 1 \leq y'_{ij} \qquad (\text{cut } C = \{(a,j),(b,j)\})$$

Finding Lifted Cut Constraints

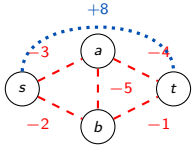
Goal: Find all-negative cuts crossed by positive lifted edges.

Preprocessing: Contract all positive base edges $\Rightarrow$ reduced graph with only repulsive edges.

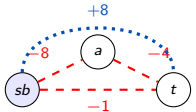
1. **Candidates:** Positive lifted edges (s, t) spanning different super-nodes
2. **Karger's contraction:**
   - Contract random edges in reduced graph
   - Stop when s, t separated
   - Multiple trials, keep smallest cut

Why Karger? Massively parallel (contract edges in parallel), easy control flow $\Rightarrow$ perfect for GPU.

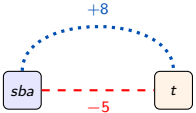
1. Reduced graph:



2. Contract s–b (s and t still separated):



3. Contract a–sb (s and t now separated!):

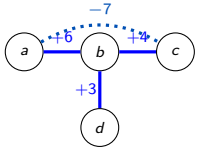


Primal Solver: Lifted-Aware Contraction

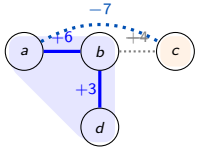
1. MST on **base edges** only
2. **Negative lifted edges** remove bottleneck MST edge on their path
3. Contract remaining MST edges
4. **Update lifted graph:** remove self-loops, sum parallel edges, absorb into base
5. Repeat on coarsened graph

Why only base edges? Lifted edges don't define connectivity ⇒ contracting them gives inadmissible solutions.

1. MST on base edges

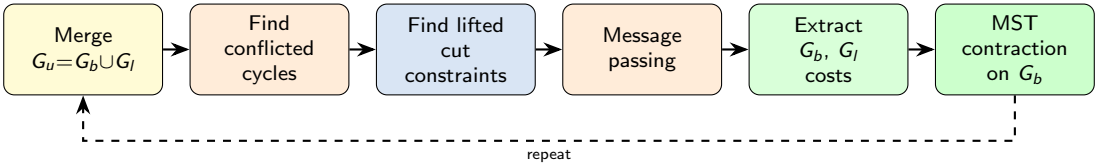


2. Remove bottleneck for lifted edge a–c



3. Contracted graph

Complete Lifted Multicut Pipeline



Implementation Details

Algorithm modes:

Mode	Dual	Primal	Use Case
`P`	×	✓	Fastest
`PD`	✓	✓	Best tradeoff
`PD+`	✓✓	✓	Best quality
`D`	✓	×	LB only

CPU/GPU backend:

- **GPU**: massively parallel, fast on large instances
- **CPU**: same code, for testing, slow

Interfaces: C++ API, Python, CLI

Code: `github.com/pawelswoboda/RAMA`
Branch: `lifted-multicut`

Applications of Lifted Multicut

Image Segmentation

- Pixel grid graph $\rightarrow$ base edges
- Long-range pixel similarity $\rightarrow$ lifted edges
- Evaluated on BSDS-500 and CREMI benchmarks

Mesh Decomposition

- 3D mesh faces $\rightarrow$ base edges
- Geometric feature similarity $\rightarrow$ lifted edges
- Evaluated on Princeton Segmentation Benchmark

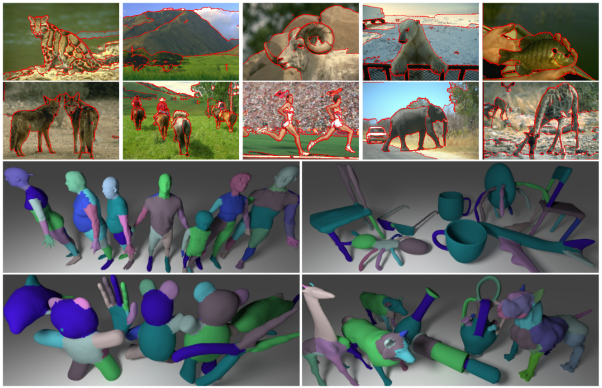


Image segmentation (top) and mesh decomposition (bottom) results using lifted multicuts.

Source: Keuper et al., ICCV 2015

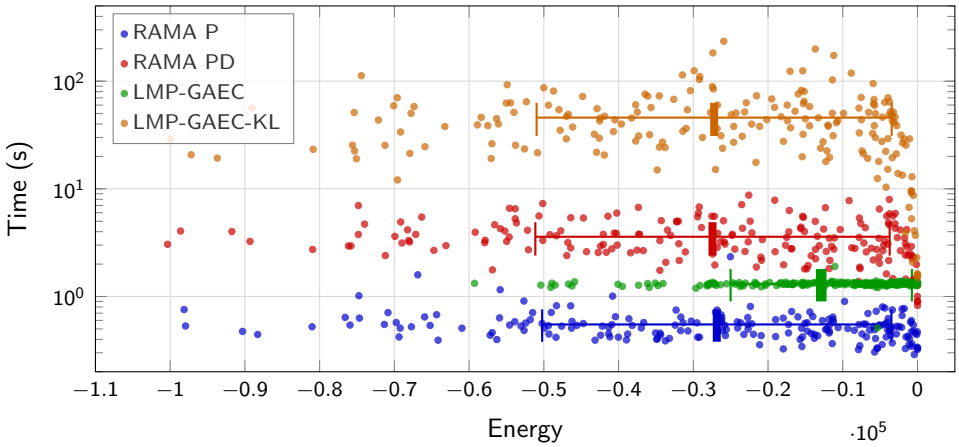
Experimental Results: Lifted Multicut

Dataset: 200 instances from Image segmentation.

Method	**Avg. Energy**	**Avg. Time (s)**	**Best (#)**
RAMA P (GPU)	−26 860	0.55	30
RAMA PD (GPU)	−27 431	3.59	60
LMP-GAEC	−12 883	1.31	0
LMP-GAEC-KL	−27 210	45.9	105

- RAMA PD achieves better energy than LMP-GAEC-KL at **13**× **speedup**
- RAMA P: **84**× **faster** than LMP-GAEC-KL, competitive energy
- LMP-GAEC without Kernighan–Lin refinement performs poorly
- RAMA PD lower bound: avg. −37 689
- Check what mesh segmentation will give.

Energy vs. Runtime



Concerns & To-Do

Concerns:

- Still large gaps between dual lower bound and primal solution (might be due to problem construction of considered experiments?)
- Lifted cut constraints do not improve the bound significantly

To-Do:

- Higher-order lifted multicut (constraints/costs beyond pairwise lifted edges)
  - Billoir fit between pairs of tracks might be handled to higher order costs in vertex reconstruction
- Extension to node labeling: Cluster vertices with additional classification, e.g. primary/secondary track?
- Other contraction strategies that exploit lifted multicut problem structure
- Integration into end-to-end learning pipelines
- Apply to primary/secondary vertex reconstruction problems
- Apply to clustering in event based camera PIV?
  - Clustering with multicut for multiple cameras at once also done for normal tracking