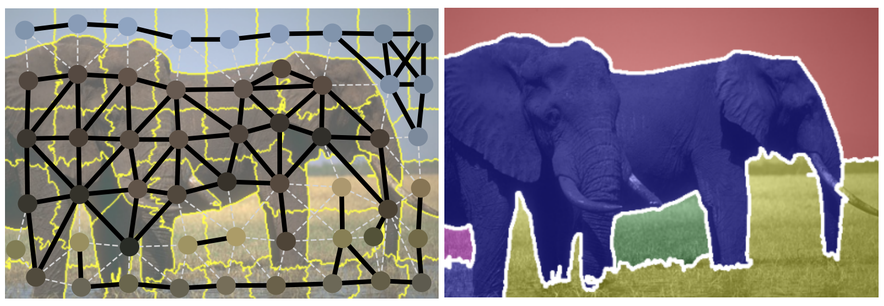
Learning to solve Minimum Cost Multicuts efficiently using Edge-Weighted Graph Convolutional Neural Networks

Steffen Jung, Margret Keuper

Motivation / Contribution

The minimum cost multicut problem is the NP-hard / APX-hard combinatorial optimization problem of partitioning a real-valued edge-weighted graph such as to minimize the total cost of the partition.



Our idea: Adapt graph convolutional neural networks for real-valued edge weights and use them to solve minimum cost multicuts.

Our contributions:

1. We formulate the minimum cost multicut problem as message-passing neural network.
2. We derive a loss to improve the feasibility of provided solutions.
3. We create synthetic training data to train our models efficiently.
4. We show in extensive experiments that our formulation is efficient and scalable.

Minimum Cost Multicut

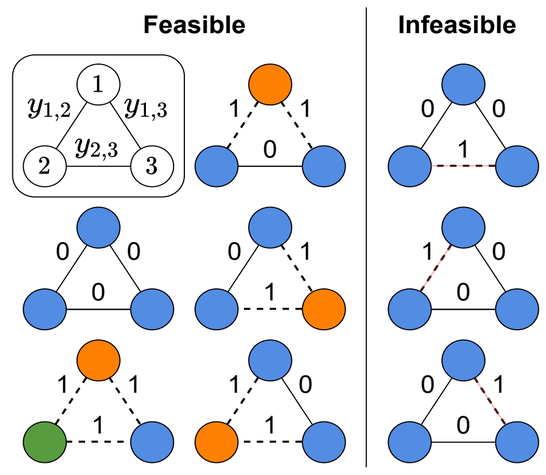
Definition: For a simple, connected graph $G = (V, E)$ and an associated cost function $w\colon E \to \mathbb{R}$, written below is an instance of the multicut problem

$$\min_{\mathbf{y}\in\{0,1\}^{|E|}} \quad c(\mathbf{y}) = \mathbf{y}^T\mathbf{w} = \sum_{e\in E} w_e y_e$$

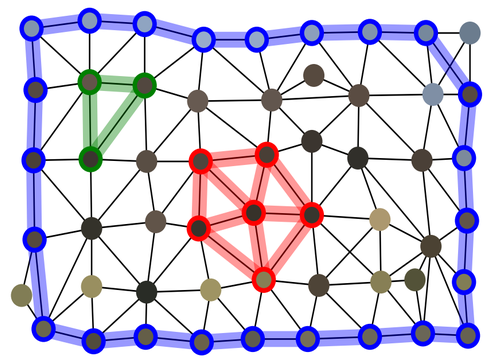
with y subject to the linear constraints

$$\forall C \in \text{cycles}(G), \forall e \in C\colon \; y_e \leq \sum_{e'\in C\setminus\{e\}} y_{e'},$$

where $\text{cycles}(\cdot)$ enumerates all cycles in graph G.



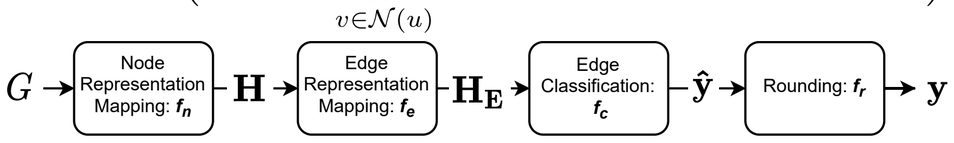
Each edge is assigned a binary edge label so that edges labeled with 1 are cut. The constraints (cycle consistency constraints) define the space of feasible edge label configurations by ensuring that there can be no cycle in which exactly one edge is cut. Deriving optimal solutions is oftentimes intractable for large problem instances, since the number of constraints can grow exponentially.



1) Multicut Neural Network Formulation

We enable negative-valued edge weights in formulation of Graph Convolutional Neural Network (GCN) by (i) incorporating edge weights and (ii) by replacing the normalized Laplacian with the signed normalized Laplacian.

$$\mathbf{h}_{\mathbf{u}}^{(t)} = g_\theta^{(t)}\left(\mathbf{h}_{\mathbf{u}}^{(t-1)} + \sum_{v\in\mathcal{N}(u)} \overset{\text{(i)}}{w_{v,u}} \cdot \left(\overset{\text{(ii)}}{\overline{\deg}(u)\overline{\deg}(v)}\right)^{-1/2} \mathbf{h}_{\mathbf{v}}^{(t-1)}\right)$$



The minimum cost multicut problem is cast as an edge classification problem. We can train our network via BCE given optimal solutions in the training data. In an ablation study we show that both adjustments (i) and (ii) are necessary to enable a GCN to learn good solutions.

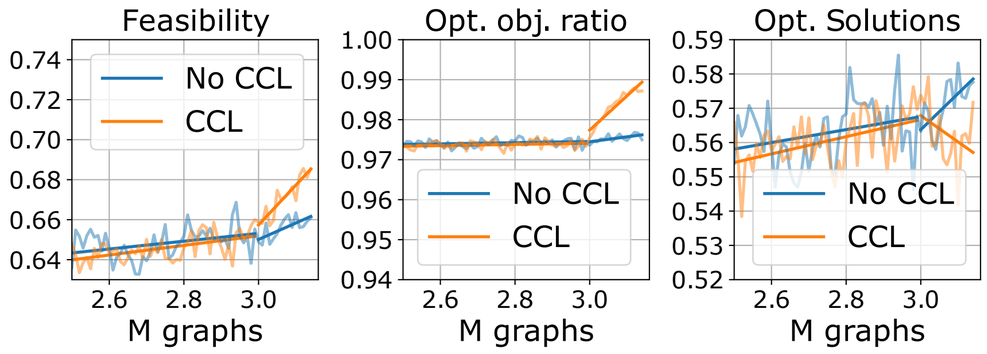
Variant	IrisMP	RandMP
GCN	Not applicable.	
- Laplacian	0.41	0.18
+ edge weights	0.95	0.18
+ signed norm.	**0.96**	**0.67**
= GCN_W	**0.96**	**0.67**
- edge weights	0.64	0.05

2) Cycle Consistency Loss

The BCE loss encodes feasibility only implicitly by comparison to the optimal solution. To explicitly learn feasible solutions, we formulate a feasibility loss that we denote Cycle Consistency Loss (CCL) as follows:

$$\mathcal{L}_{CCL} = \alpha \cdot \sum_{C\in cc(G,l)} \sum_{e\in C} \hat{y}_e \prod_{e'\in C\setminus\{e\}} (1 - \hat{y}_{e'})$$

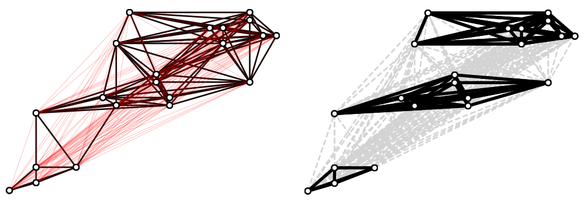
where *α* is a hyperparameter, balancing BCE and CCL, and *cc(G, l)* is a function that returns all chordless cycles in *G* of length at most *l*. CCL induces a trade-off between finding feasible and optimal solutions, where the model is forced to find feasible solutions to avoid the penalty, and as a consequence, settles for suboptimal relaxated solutions. However, the sum of objective values over multiple graphs after rounding improves.



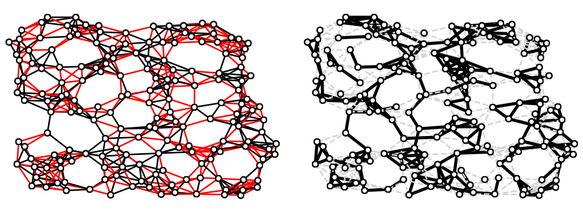
3) Training Data

The amount of available annotated problem instances is scarce. We generate two synthetic datasets to alleviate this problem.

IrisMP: Multicut instances on complete graphs based on the Iris flower dataset. Graphs have 20 nodes and 194 edges on average. Edge weights are computed as similarity measures via a Gaussian kernel of node distances and then converted via the logit function.

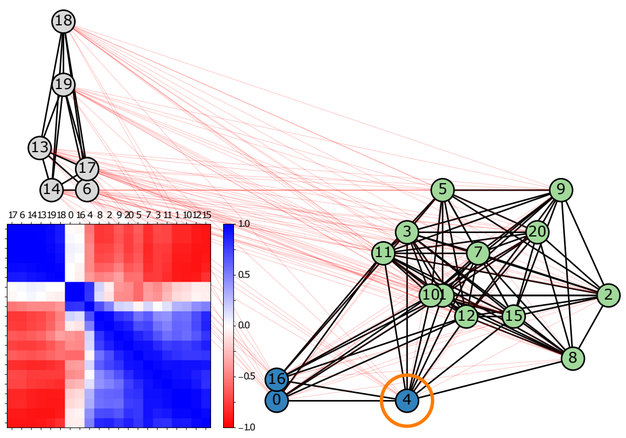


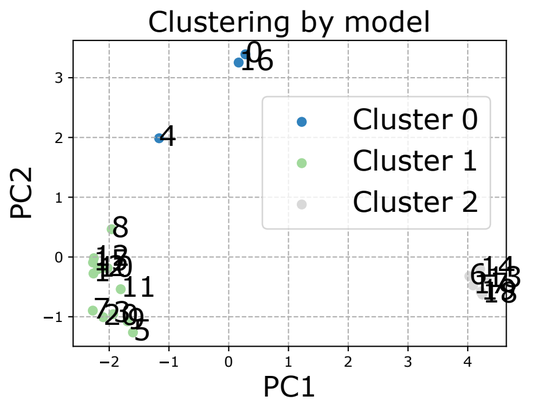
RandomMP: Randomly generated sparse *k*-nearest neighbor graphs via sampling coordinates on a 2D plane. Graphs have 180 nodes and 686 edges on average. Edge weights correspond to node distances whereas the median distance is subtracted from each edge weight resulting in ~50% negative weights.



4) Results

We show that our model achieves competitive results by comparing against an LP solver and a highly efficient heuristic called Greedy Additive Edge Contraction (GAEC). While GAEC achieves solutions with smaller costs, our model is more efficient and scalable. Moreover, our model provides gradients for possible down-stream tasks.





Nodes	GAEC [ms]	GAEC Objective	LP [ms]	LP Objective	ILP [ms]	ILP Objective	GCN_W_BN [ms]	GCN_W_BN Objective
10^1	**0**	−29	6	−24	11	−30	29	−29
10^2	**4**	−327	191	−246	273	−330	26	−276
10^3	**24**	−3051	6585	−2970	1299	−3093	29	−2643
10^4	228	−32 264	688 851	−31 531	18 604	−32 682	**78**	−27 552
10^5	2534	−323 189	OOT		2 171 134	−328 224	**557**	−269 122
10^6	35 181	−3 401 783	OOT		OOT		**8713**	−2 182 589