Solution Stability in Linear Programming Relaxations: Graph Partitioning and Unsupervised Learning

Sebastian Nowozin SEBASTIAN.NOWOZIN@TUEBINGEN.MPG.DE
Stefanie Jegelka STEFANIE.JEGELKA@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics, Spemannstrasse 38, 72076 Tübingen, Germany

Abstract

We propose a new method to quantify the solution stability of a large class of combinatorial optimization problems arising in machine learning. As practical example we apply the method to correlation clustering, clustering aggregation, modularity clustering, and relative performance significance clustering. Our method is extensively motivated by the idea of linear programming relaxations. We prove that when a *relaxation* is used to solve the original clustering problem, then the solution stability calculated by our method is conservative, that is, it never overestimates the solution stability of the true, unrelaxed problem. We also demonstrate how our method can be used to compute the entire *path of optimal solutions* as the optimization problem is increasingly perturbed. Experimentally, our method is shown to perform well on a number of benchmark problems.

1. Introduction

Several fundamental problems in machine learning can be expressed as the combinatorial optimization task

P1 $$\boldsymbol{z}^* := \operatorname{argmin}_{\boldsymbol{z}\in\mathcal{B}} \boldsymbol{w}^\top \boldsymbol{z},$$

where $\mathcal{B} \subseteq \{0,1\}^n$ is a specific set of indicator vectors of length n. For example, clustering problems can be posed naturally by means of binary variables indicating whether two samples are in the same cluster.

The formulation P1 is general and powerful. However, depending on the problem parameter $\boldsymbol{w}$, an optimal solution $\boldsymbol{z}^*$ might not be unique, or it might be *unstable*, i.e., a small perturbation to $\boldsymbol{w}$ will make another $\boldsymbol{z} \neq \boldsymbol{z}^*$ optimal. To ensure a reliable and principled use of P1 it is important to analyze the *stability* of $\boldsymbol{z}^*$, especially because the lack of stability can indicate serious modeling problems.

In machine learning, the value of $\boldsymbol{w}$ usually depends on the data, and possibly on a modeling parameter. Both these dependencies often introduce uncertainty. Real data commonly originates from noisy measurements or is assumed to be sampled from an underlying distribution. This data induces one $\boldsymbol{w}$ and thus one optimal solution, e.g. clustering, $\boldsymbol{z}_1^*$. If a slight perturbation to the data completely changes the solution to $\boldsymbol{z}_2^*$, then $\boldsymbol{z}_1^*$ must be treated with care. The preference of $\boldsymbol{z}_1^*$ over $\boldsymbol{z}_2^*$ could merely be due to noise. To account for uncertainty in the data, one commonly strives for stable solutions with respect to perturbations or re-sampling.

Modeling parameters are another source of uncertainty, for their "correct" value is usually unknown, and thus estimated or heuristically set. A stability analysis gives insight into how the parameter influences the solution on the given instance of data. Here too stability can indicate reliability.

In addition, a stability analysis can reveal characteristics of the data itself, as we illustrate in two examples. We can compute the path of all solutions as the perturbation increases systematically. Depending on the perturbation, this path may indicate structural information or help to analyze a modeling parameter.

In this paper, we present a *new general* method to quantify the solution stability of Problem P1 and compute the solution path along a parametric perturbation. In particular, we overcome the inability of existing approaches to handle a basic characteristic of linear programming relaxations to P1, namely, that only few constraints are known at a time. Owing to our formulation, two close variants of the same algorithm will suffice to solve both the nominal Problem P1 and the stability analysis.

A running example for P1 makes the general discussion concrete: the Graph Partitioning Problem (GPP), which unifies a number of popular clustering tasks. Our stability analysis for GPP hence allows a more thoughtful analysis of these clusterings.

1.1. Graph Partitioning Problem

In many unsupervised learning problems, we only have information about pairwise relations of objects, and not about features of individuals. Examples include

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

co-authorship and citations, or protein interactions. In this case, exemplar- or *centroid*-based approaches are inapplicable, and we directly use the graph of relations or similarities. Clustering corresponds to finding an appropriate partitioning of this graph.

Problem 1 (Graph Partitioning Problem) *Given an undirected, connected graph $G = (V, E)$, edge weights $\boldsymbol{w} : E \to \mathbb{R}$, partition the vertices into nonempty subsets so that the total weight of the edges with end points in different subsets is minimized.*

Note that, in contrast to common graph cut problems such as Minimum Cut or Normalized Cut, GPP does *not* pre-specify the number of clusters. To describe a partitioning of G, we will use indicator variables $z_{i,j} \in \{0, 1\}$ for each edge $(i, j) \in E$, where $z_{i,j} = 1$ if i and j are in different partitions, and $z_{i,j} = 0$ otherwise. Figure 1 shows an example. Let $\mathcal{Z}(G) = \{\boldsymbol{z} \in \{0,1\}^{|E|} \mid \exists \pi : V \to \mathbb{N} : \forall (i,j) \in E : z_{i,j} = [\![\pi(i) \neq \pi(j)]\!]\}$ be the set of all possible partitionings, where $[\![\cdot]\!]$ is the indicator function.

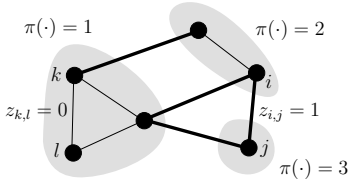

Figure 1. An example partitioning $\boldsymbol{z}$. Bold edges have $z_{i,j} = 1$, while others have $z_{k,l} = 0$.

Using this notation, we can formalize GPP as a special case of P1 with $\mathcal{B} = \mathcal{Z}(G)$:

$$\textbf{P2} \qquad \min_{\boldsymbol{z}} \sum_{(i,j)\in E} w(i,j)\, z_{i,j} \qquad \text{sb.t.} \quad \boldsymbol{z} \in \mathcal{Z}(G)$$

P2 encompasses a wide range of clustering problems if we set the weights $\boldsymbol{w}$ accordingly. Table 1 summarizes $\boldsymbol{w}$ for a number of popular clustering problems, and also for two biases: one favoring clusters of equal sizes, and one penalizing large clusters.

The information contained in a single weight $w(i, j)$ is often enough to make local decisions about i and j being in the same cluster. Global agreement of these local decisions is enforced by $\boldsymbol{z}$ being a valid partitioning. Exactly this global constraint $\boldsymbol{z} \in \mathcal{Z}(G)$ makes GPP difficult to solve.

In general, P1 is an integer linear program (ILP) and NP-hard. A common approach to solving P1 is to use a linear *relaxation* of the constraint $\boldsymbol{z} \in \mathcal{B}$.

Linear Relaxations. In general, the point set $\mathcal{B} \subseteq \{0, 1\}^n$ is finite but exponentially large in n and usually intractable. It is known from combinatorial optimization (Schrijver, 1998) that relaxing the set $\mathcal{B}$ to its convex hull $\text{conv}(\mathcal{B})$ will not change the minimizer $\boldsymbol{z}^*$ of P1. The set $\text{conv}(\mathcal{B})$ is by construction a bounded polyhedron – a so-called *polytope* – and at least one minimizer of a linear function over a polytope is a vertex. Therefore, at least one optimal solution of the relaxation will be integral, that means it is in $\mathcal{B}$ and thus an optimal solution of the exact problem. For GPP, the convex hull $\text{conv}(\mathcal{Z}(G))$ is the *Multicut Polytope*.

The convex hull is defined in terms of vertices $\boldsymbol{z} \in \{0, 1\}^n$. We can alternatively describe it in terms of intersecting halfspaces (Schrijver, 1998), i.e., linear inequalities. The minimal set of such inequalities to characterize the polytope exactly is the set of all *facet-defining inequalities*. Knowing these inequalities, we can derive a linear program *equivalent* to P2.

But often only a subset of the facet-defining inequalities is known, some are difficult to check and all are too many to handle efficiently. Therefore, one commonly replaces $\text{conv}(\mathcal{B})$ by an approximation $\widehat{\mathcal{B}} \supseteq \text{conv}(\mathcal{B}) \supset \mathcal{B}$ represented by a tractable subset of the facet-defining inequalities.

We will use such relaxations to derive a method for quantifying the stability of the optimal solution $\boldsymbol{z}^*$ with respect to perturbations in $\boldsymbol{w}$. In Section 2, we first introduce our notion of stability analysis and then show how to overcome the difficulties of existing approaches. Details about solving the formulated problems follow in Section 3. We describe the general cutting-plane algorithm for both P1 and the stability analysis in Section 3.1, while Section 3.2 specifies algorithmic details for GPP, i.e., a relaxation of the multicut polytope that is tighter than previous approximations for the problems in Table 1. Experiments in Section 4 evaluate our method.

2. Stability Analysis

We first detail our notion of stability and then develop our approach. The method is based on local polyhedral approximations to the feasible set of the combinatorial problem and efficiently identifies solution break points for parametric perturbations of $\boldsymbol{w}$.

We perturb the weight vector $\boldsymbol{w} \in \mathbb{R}^n$ by a vector $\boldsymbol{d} \in \mathbb{R}^n$. The resulting weights are then $\boldsymbol{w}'(\theta) = \boldsymbol{w} + \theta\boldsymbol{d}$ for a perturbation level θ. *Stability analysis* asks for the range of θ for which the optimal solution does not change, i.e., the *stability range*.

Definition 1 (Stability Range) *Let the feasible set $\mathcal{B} \subseteq \{0, 1\}^n$, a weight vector $\boldsymbol{w} \in \mathbb{R}^n$ and the optimal solution $\boldsymbol{z}^* := \text{argmin}_{\boldsymbol{z}\in\mathcal{B}}\, \boldsymbol{w}^\top \boldsymbol{z}$ be given. For*

Table 1. Graph partitioning formulations of clustering problems for a set of objects V or graph $G = (V, E)$, and $\lambda > 0$.

Problem	Description	Weights
Correlation Clustering	Given pairwise positive and negative similarity ratings $v(i,j) \in \mathbb{R}$ for samples i, j, find a partitioning that agrees as much as possible with these ratings (Bansal et al., 2002; Demaine et al., 2006; Joachims & Hopcroft, 2005).	$w(i,j) = v(i,j),\ \forall (i,j) \in E$
Clustering Aggregation, Consensus Clustering	Also known as *clustering ensemble* and *clustering combination*. Find a single clustering that agrees as much as possible with a given set of m clusterings (Gionis et al., 2007).	$w(i,j) = \frac{1}{m}\sum_{k=1}^{m}\left(1 - 2r_{i,j}^k\right),\ \forall (i,j) \in V \times V$, where $\mathbf{r}^k$ represents clustering k analogous to $\boldsymbol{z}$.
Modularity Clustering	Maximize *modularity*, i.e., the difference between the achieved and expected fraction of intra-cluster edges. Originally for unweighted graphs (Newman & Girvan, 2004; Brandes et al., 2008), it is straightforward to extend to weighted graphs (Newman, 2004; Gaertler et al., 2007), and so are the weights on the right.	$w(i,j) = \frac{1}{2\lvert E\rvert}\left(\eta_{i,j} - \frac{\deg(i)\deg(j)}{2\lvert E\rvert}\right),\ \forall (i,j) \in V \times V$, with $\eta_{i,j} = [\![(i,j) \in E]\!]$, and deg denoting the degree of a node.
Relative Performance Significance Clustering	Maximize the achieved versus expected performance, i.e., fraction of edges within clusters and of missing edges between clusters (Gaertler et al., 2007).	$w(i,j) = \frac{1}{n(n-1)}\left(2\eta_{i,j} - \frac{\deg(i)\deg(j)}{\lvert E\rvert}\right),\ \forall (i,j) \in V \times V$
Bias: Squared Differences of Cluster Sizes	The criterion $\lambda\sum_{k,l=1}^{K}(\lvert C_k\rvert - \lvert C_l\rvert)^2$ favors clusters of equal sizes.	$\Delta w(i,j) = -2\lambda,\ \forall (i,j) \in V \times V$
Bias: Squared Cluster Sizes	A penalty for large clusters is $\lambda\sum_{k=1}^{K}\lvert C_k\rvert^2 = \lambda\sum_{k=1}^{K}\sum_{i,j\in V} 2\lambda\lvert V\rvert^2 - \lambda\sum_{i,j\in V} z_{i,j}$.	$\Delta w(i,j) = -\lambda,\ \forall (i,j) \in V \times V$

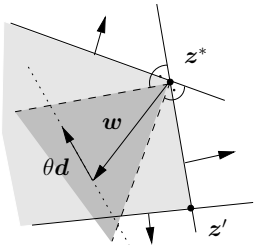

Figure 2. Geometry of Stability Analysis in a Polytope

a perturbation vector $\boldsymbol{d} \in \mathbb{R}^n$ *and modified weights* $\boldsymbol{w}'(\theta) = \boldsymbol{w} + \theta\boldsymbol{d}$*, the stability range is the interval* $[\rho_{\boldsymbol{d},-}, \rho_{\boldsymbol{d},+}] \in (\{-\infty, \infty\} \cup \mathbb{R})^2$ *of* θ *values for which* $\boldsymbol{z}^*$ *is optimal for the perturbed problem* $min_{\boldsymbol{z}\in\mathcal{B}}\ \boldsymbol{w}'(\theta)^\top \boldsymbol{z}$.

The geometry of stability ranges in the polytope $\text{conv}(\mathcal{B})$ is illustrated in Figure 2. The polytope is lightly shaded and bounded by lines representing the inequalities that define $\text{conv}(\mathcal{B})$. We know that $\boldsymbol{z}^*$ is optimal for $\boldsymbol{w}'(\theta) = \boldsymbol{w} + \theta\boldsymbol{d}$ for $\theta = 0$. The point $\boldsymbol{z}^*$ is a vertex of the polytope. Two of the inequalities are binding (satisfied with "="), indicated by two boundary lines touching $\boldsymbol{z}^*$. The negative normal vectors of the inequalities span a cone (shaded dark). As long as $\boldsymbol{w}'(\theta)$ lies in this cone, $\boldsymbol{z}^*$ is optimal. If $\boldsymbol{w}'(\theta)$ leaves the cone, say for a large enough $\theta > 0$, then we can improve over $\boldsymbol{z}^*$ by sliding along an edge of the polytope to another vertex $\boldsymbol{z}' \in \mathcal{B}$ whose associated cone now contains the new vector $\boldsymbol{w} + \theta\boldsymbol{d}$. Formally, if $\boldsymbol{w}'(\theta)$ is outside the cone, then a descent direction at an obtuse angle to $\boldsymbol{w}$ will be in $\mathcal{B}$. Moving $\boldsymbol{z}$ along this direction improves the value $\boldsymbol{w}'(\theta)^\top \boldsymbol{z}$.

We aim to find the value of θ where $\boldsymbol{w}'(\theta)$ leaves the cone. If we know all inequalities defining the polytope, then we have an explicit description of the cone. Common approaches to compute stability ranges (Schrijver, 1998; Jansen et al., 1997) rely on this knowledge and use the simplex *basis matrix* (Bertsimas & Tsitsiklis, 1997). But the inequalities for the multicut polytope (and $\text{conv}(\mathcal{B})$ in general) are not explicitly known, since the polytope is defined as the convex hull of a complicated set. Even for relaxations $\widehat{\mathcal{B}}$, the set of constraints is too large to be handled as a whole, and just a few local constraints are known to the solver at a time. With such a small subset, the normal cone is only partially known and the basis matrix approach grossly underestimates the stability range, making it useless for anything but trivial instances.

In an online setting, (Kılınç-Karzan et al., 2007) use axis-aligned perturbations for the cost vector to obtain both an inner and outer polyhedral approximation to the *stability region*, the region where changes to $\boldsymbol{w}$ re-

main without effect. In contrast, we aim for an exact stability range for a given perturbation direction.

We will now present a method to compute stability ranges even without explicit knowledge of all constraints at all times. Owing to the formulation, two close variants of the same algorithm will suffice to solve both the original problem and the stability analysis. We will also relate the stability range obtained from relaxations to the stability range of the exact problem.

2.1. Linear Programming Stability Analysis using Separation Oracles

To avoid use of the basis matrix, we adopt a lesser known idea of (Jansen et al., 1997, Section 4.2): at optimality, the primal and dual optimal values are equal. Hence, $\boldsymbol{z}^*$ is optimal (and $\boldsymbol{w}'(\theta)$ in the cone) as long as the optimal value of the perturbed dual equals $\boldsymbol{w}'(\theta)^\top \boldsymbol{z}^*$. Jansen et al. (1997) implement this idea in an LP derived from the dual of the original problem. With our implicit constraints, a dual-based approach is inapplicable. Therefore, we revert to the primal to construct a pair of *auxiliary linear programs* that search within the cone of *all possible* constraints defining $\text{conv}(\mathcal{B})$ around $\boldsymbol{z}^*$.

The resulting formulation is similar to the original Problem P1, so we can use a similar solution procedure to take into account *all* implicit constraints — a point we elaborate in Section 3.1. The following program yields the stability range for a given optimal solution $\boldsymbol{z}^*$ and perturbation direction $\boldsymbol{d}$.

$$\textbf{P3} \qquad \min_{\alpha\in\mathbb{R},\boldsymbol{z}\in\mathbb{R}^n} \quad \boldsymbol{w}^\top\boldsymbol{z} + \alpha\boldsymbol{w}^\top\boldsymbol{z}^*$$

$$\text{sb.t.} \qquad (\frac{1}{\alpha}\boldsymbol{z}) \in \text{conv}(\mathcal{B}) \tag{1}$$

$$(\boldsymbol{d}^\top\boldsymbol{z}^*)\alpha - \boldsymbol{d}^\top\boldsymbol{z} = t \tag{2}$$

$$0 \le \boldsymbol{z}_i \le \alpha, \quad i = 1,\dots,n.$$

Constraint (1) is still linear, because it corresponds to $A(\frac{1}{\alpha}\boldsymbol{z}) \le b$, or $A\boldsymbol{z} - \alpha b \le 0$. The constant $t \in \{-1, 1\}$ in (2) determines whether we search for the left interval boundary $\rho_{\boldsymbol{d},-}$ or right interval boundary $\rho_{\boldsymbol{d},+}$ of the stability range $[\rho_{\boldsymbol{d},-}; \rho_{\boldsymbol{d},+}]$. At the optimum, the Lagrange multiplier for Constraint (2) equals the boundary $\rho_{\boldsymbol{d},-}$ or $\rho_{\boldsymbol{d},+}$, depending on t. Problem P3 is primal infeasible if and only if $\rho_{\boldsymbol{d},-} = -\infty$ for the left boundary ($t = -1$) or $\rho_{\boldsymbol{d},+} = \infty$ for the right boundary ($t = 1$).

The stability range could also be found approximately by probing various values of θ, similar to a *line search* in continuous optimization. In contrast, our method finds the breakpoint exactly by solving one optimization per search direction. It is guaranteed not to miss any breakpoints, a property that is hard to ensure for an iterative point-wise testing procedure.

The hardness of P3, like that of the nominal problem P1, depends on the tractability of $\text{conv}(\mathcal{B})$. That means we are forced to replace $\text{conv}(\mathcal{B})$ by a tractable approximation $\widehat{\mathcal{B}}$ to solve P3 efficiently. We will outline the relaxation for GPP in Section 3.2. But if we use $\widehat{\mathcal{B}}$, then the stability range only refers to the relaxation, i.e., for $\theta \notin [\rho_{\boldsymbol{d},-}, \rho_{\boldsymbol{d},+}]$, the optimal solution *of the relaxation* is guaranteed to change. Theorem 1 relates this stability range of the relaxation to the stability range of the exact problem.

Theorem 1 (Stability Inclusion) *Let $\boldsymbol{z}^*$ be the optimal solution of P1 for a given $\mathcal{B} \subseteq \{0,1\}^n$ and weights $\boldsymbol{w} \in \mathbb{R}^n$. For a perturbation $\boldsymbol{d} \in \mathbb{R}^n$, let $[\xi_{\boldsymbol{d},-}, \xi_{\boldsymbol{d},+}]$ be the true stability range for θ on conv($\mathcal{B}$). If $\widehat{\mathcal{B}} \supseteq$ conv($\mathcal{B}$) is a polyhedral relaxation of $\mathcal{B}$ using only facet-defining inequalities and if $\boldsymbol{z}^*$ is a vertex of $\widehat{\mathcal{B}}$, then the stability range $[\rho_{\boldsymbol{d},-}, \rho_{\boldsymbol{d},+}]$ on $\widehat{\mathcal{B}}$, i.e., for the relaxation $\min_{\boldsymbol{z}\in\widehat{\mathcal{B}}} \boldsymbol{w}^\top\boldsymbol{z}$, is included in the true range: $[\rho_{\boldsymbol{d},-}, \rho_{\boldsymbol{d},+}] \subseteq [\xi_{\boldsymbol{d},-}, \xi_{\boldsymbol{d},+}]$.*

Proof. Let $S_{\mathcal{B}}$ be the set of all constraints defining $\text{conv}(\mathcal{B})$ at $\boldsymbol{z}^*$ and $S_{\widehat{\mathcal{B}}}$ the set of all facet-defining constraints for $\widehat{\mathcal{B}}$ at $\boldsymbol{z}^*$. As $S_{\widehat{\mathcal{B}}}$ contains only facet-defining constraints, we have $S_{\widehat{\mathcal{B}}} \subseteq S_{\mathcal{B}}$. As a result, the cone spanned by the negative constraint normals in $S_{\mathcal{B}}$ contains the cone spanned by the negative constraint normals in $S_{\widehat{\mathcal{B}}}$, and thus $[\rho_{\boldsymbol{d},-}, \rho_{\boldsymbol{d},+}] \subseteq [\xi_{\boldsymbol{d},-}, \xi_{\boldsymbol{d},+}]$. □

Theorem 1 and P3 suggest that with a tight enough relaxation $\widehat{\mathcal{B}}$, we can efficiently compute a good approximation of the stability range by essentially the same algorithm that we apply to P1. Besides quantifying the robustness of a solution with respect to parametric perturbations, stability ranges help to recover an entire path of solutions, as we will show next.

2.2. Efficiently Tracing the Solution Path

As we increase the perturbation level θ, the optimal solution changes at certain breakpoints, the boundary points of the current stability range. That means we can trace the path of all optimal solutions along the weight path $w + \theta\boldsymbol{d}$ for $\theta \in [-\infty, \infty]$ by repeatedly jumping to the solution at the breakpoint and computing the stability range to find the next breakpoint.

The interpretation of the path of solutions depends on the choice of weights and the perturbation. For GPP, we will use weights derived from similarity matrices and obtain all clustering solutions on a path defined by shifting a linear bias term. This amounts to computing all clusterings between the extremes "one big cluster" and "each sample is its own cluster".

3. Implementation

In the previous sections we formalized the nominal problem P1 and the stability analysis P3. Now we describe how to actually solve them. We first present a general algorithm and then specify details for GPP, mainly a suitable relaxation of the multicut polytope.

3.1. Cutting Plane Algorithm

The cutting plane method (Wolsey, 1998, chapter 11) shown in Algorithm 1 applies to both P1 and P3. Cutting plane algorithms provide a polynomial-time method to solve (appropriate) relaxations of ILPs.

The algorithm works with a small set of constraints that defines a loose relaxation $\mathcal{S}$ to the feasible set $\mathcal{B}$. It iteratively tightens $\mathcal{S}$ by means of *violated inequalities*. In Line 11, we solve the current LP relaxation. Having identified a minimizer $\boldsymbol{z}$, we search for a violated inequality in the set of *all* constraints (Line 12). If we find a violated inequality, we add it to the current constraint set to reduce $\mathcal{S}$ (Line 16) and re-solve with the tightened relaxation. Otherwise, $\boldsymbol{z}^* = \boldsymbol{z}$ is optimal with all constraints.

Algorithm 1 Cutting Plane Algorithm

```
1: (z*, f, optimal) = CuttingPlane(B, w)
2: Input:
3:     Set B ⊆ {0,1}^n, weights w ∈ R^n
4: Output:
5:     Optimal solution z* ∈ [0,1]^n,
6:     Lower bound on the objective f ∈ R,
7:     Optimality flag optimal ∈ {true, false}.
8: Algorithm:
9: S ← [0,1]^n {Initial feasible set}
10: loop
11:    z ← argmin_{z∈S} w^⊤ z {Solve LP relaxation}
12:    S_violated ← SeparateInequalities(B, z)
13:    if no violated inequality found then
14:       break
15:    end if
16:    S ← S ∩ S_violated {Cut z from feasible set}
17: end loop
18: optimal ← (z ∈ {0,1}^n) {Integrality check}
19: (f, z*) ← (w^⊤ z, z)
```

The search for a violated inequality is the *separation oracle*. It depends on the particular set $\mathcal{B}$ of the combinatorial problem at hand and the description of the relaxation $\widehat{\mathcal{B}}$. The separation oracle is decisive for the runtime. If it runs in polynomial time, then the entire algorithm runs in polynomial time (Wolsey, 1998, chapter 11). Hence, polynomial-time separability is an important criterion for the relaxation $\widehat{\mathcal{B}}$. The next section addresses such a relaxation for GPP.

3.2. Relaxations of the Multicut Polytope

Solving GPP over $\mathcal{Z}(G)$ or $\operatorname{conv}(\mathcal{Z}(G))$, the multicut polytope, is NP-hard (Deza & Laurent, 1997; Chopra & Rao, 1993). To relax $\operatorname{conv}(\mathcal{Z}(G))$ for an efficient optimization, we need facet-defining inequalities that describe an approximation to $\operatorname{conv}(\mathcal{Z}(G))$ and are separable in polynomial time. In addition, the tighter the relaxation is, i.e., the more inequalities we use, the more accurate the stability analysis becomes.

The multicut polytope $\operatorname{conv}(\mathcal{Z}(G))$ and variations have been researched in the late eighties and early nineties (Grötschel & Wakabayashi, 1989; Grötschel & Wakabayashi, 1990; Chopra & Rao, 1993; Deza et al., 1992; Deza & Laurent, 1997). We now discuss two subsets of the set of facet-defining inequalities for the multicut polytope that we use, *cycle inequalities* and *odd-wheel inequalities*. Both are polynomial-time separable, so we can tell efficiently whether a point satisfies all inequalities and if it does not, we can find a violated inequality.

Cycle Inequalities. The cycle inequalities are generalizations of the triangle inequality. Any valid graph partitioning $\boldsymbol{z}$ satisfies a *transitivity* relation: there is no all-zero path between any two adjacent vertices i, j that are in different subsets of the partition, i.e., for which $z_{i,j} = 1$. Formally, this property is described by the *cycle inequalities* (Chopra & Rao, 1993) that are facet-defining for chord-free cycles $((i,j),p)$, $p \in \operatorname{Path}(i,j)$, where $\operatorname{Path}(i,j)$ is the set of paths between i and j.

$$z_{i,j} \leq \sum_{(s,t)\in p} z_{s,t}, \qquad (i,j) \in E, \quad p \in \operatorname{Path}(i,j). \quad (3)$$

In complete graphs, all cycles longer than three edges contain chords. Hence, for complete graphs we can simplify the cycle inequalities to a polynomial number of triangle inequalities, as done in Grötschel and Wakabayashi (1989); Chopra and Rao (1993); and Brandes et al. (2008). The separation procedure for (3) is a simple series of shortest path problems, one for each edge. We omit it for reasons of space, details can be found in Chopra and Rao (1993).

Previous LP relaxations for correlation and modularity clustering (Emanuel & Fiat, 2003; Finley & Joachims, 2005; Demaine et al., 2006; Brandes et al., 2008) limit their approximation of the multicut polytope to cycle inequalities only. We call these equivalent relaxations LP-C relaxation. Our experiments will show that the LP-C relaxation is not very tight, and additional *odd-wheel inequalities* (Deza et al., 1992; Chopra & Rao, 1993) improve the approximation.

Odd-Wheel Inequalities. Let a *q-wheel* be a connected subgraph $S = (V_s, E_s)$ with a central vertex $j \in V_s$ and a cycle of the q vertices in $C = V_s \setminus \{j\}$. For each $i \in C$ there exists an edge $(i, j) \in E_s$. For every q-wheel, a valid partitioning $\boldsymbol{z}$ satisfies

$$\sum_{(s,t)\in E(C)} z_{s,t} - \sum_{i\in C} z_{i,j} \leq \lfloor \frac{1}{2} q \rfloor, \tag{4}$$

where $E(C)$ denotes the set of all edges in the outer cycle C. Deza et al. (1992) prove that the odd-wheel inequalities (4) are facet-defining for every odd $q \geq 3$. These inequalities are polynomially separable. The separation procedure is still rather involved and we omit it here; details can be found in Deza and Laurent (1997).

The inequalities (3) and (4) together describe a tight polynomial-time solvable relaxation to $\text{conv}(\mathcal{Z}(G))$ that we will call LP-CO relaxation.

4. Experiments and Results

The first part of the experiments addresses properties of our algorithm and relaxation. We compare our solution method to a popular heuristic and demonstrate the gain of tightening the relaxation to LP-CO. Experiment 4.2 relates optimality and runtime to properties of the data. The second part illustrates example applications: critical edges for modularity clustering and an analysis of the solution path for similarity data.

4.1. Tightness and comparison to a heuristic

In the Introduction, we show how to solve modularity clustering via GPP. Here we examine solution qualities of our LP relaxation and the Kernighan-Lin (KL) heuristic (Kernighan & Lin, 1970). The latter uses a greedy local search and generally converges fast. Contrary to the LP relaxation, where integrality indicates optimality, KL provides no guarantees.

We compare KL to two variants of relaxation: LP-C, which is limited to cycle-inequalities, and the tightened LP-CO, which also includes odd-wheel inequalities. Note that all previous LP relaxations of correlation and modularity clustering (Finley & Joachims, 2005; Brandes et al., 2008; Demaine et al., 2006; Emanuel & Fiat, 2003)correspond to LP-C.

The solution produced by the KL heuristic is always feasible but possibly suboptimal, and LP-C and LP-CO are weak and tight relaxations, respectively. Hence the *maximized* modularity always satisfies KL $\leq$ **OPT** $\leq$ LP-CO $\leq$ LP-C, where **OPT** is the true optimum.

We evaluate solutions on five networks described in Brandes et al. (2008); and Newman and Girvan (2004): `dolphins`, `karate`, `polbooks`, `lesmis` and `att180` (62, 34, 105, 77 and 180 nodes, respectively).

Table 2 shows the achieved modularity and the runtime. For all data sets, the LP-CO solutions are optimal (OPT=LP-CO) and all modularity scores agree with the best modularity in the literature.[1]

The Kernighan-Lin heuristic is always the fastest method and its solutions are close to optimal, as the upper bound provided by LP-C and LP-CO shows. KL itself does not give hints about closeness to optimality. The LP-C relaxation is in general very weak and obtains the optimal solution only on the smallest data set (`karate`). All it yields otherwise is an upper bound on the optimal modularity. So the effort of a tighter approximation (LP-CO) does improve the quality of the solution already on small examples.

Table 2. Modularity and runtimes on standard small network datasets. Fractional solutions are bracketed, optimal solutions are in boldface.

	Kernighan-Lin		LP-C		LP-CO	
dolphins	0.5268	0.4s	(0.5315)	4.2s	**0.5285**	9.1s
karate	**0.4198**	0.1s	**0.4198**	0.2s	**0.4198**	0.2s
polbooks	0.5226	7.0s	(0.5276)	147.4s	**0.5272**	148.5s
lesmis	0.5491	1.5s	(0.5609)	6.9s	**0.5600**	11.7s
att180	0.6559	14.5s	(0.6633)	302.3s	**0.6595**	1119.6s

4.2. LP-CO Scaling Behavior

After investigating the gain of the tighter relaxation, we now examine the scaling behavior of LP-CO with respect to edge density, problem difficulty and noise.

We sample a total of 100 vertices and uniformly assign one out of three "latent" class labels to each vertex. For a given edge density $d \in \{0.1, 0.15, \ldots, 1.0\}$ we sample a set E of $\frac{100\cdot 99}{2} d$ non-duplicate edges from the complete graph. To each edge $e \in E$ we assign with probability $n \in \{0, 0.05, \ldots, 0.5\}$ a "noisy" weight uniformly at random from the interval $[-1, 1]$. To all other edges we assign a "true" weight from either $[-1, 0]$ if the latent class label of the adjacent vertices are different, or from $[0, 1]$ if the latent class labels are equal. For each pair (d, n) we create ten graphs with the above properties and solve GPP on each instance. Figures 3 to 5 show where integrality was achieved, the average runtime and Rand index to the underlying labels. The index is 1 if the partitioning is identical to the latent classes. The expected Rand index of a

[1] Except for the `karate` data set which differs from the optimal modularity of 0.431 reported in (Brandes et al., 2008). We contacted the authors who discovered a corruption in their data set and confirmed our value of 0.4198.

random partitioning is $\frac{2}{3}$ (Rand, 1971).

The figures suggest two relations between properties of the data and the algorithm. First, integrality of the LP-CO solution (gray region in Figure 3) mostly coincides with the optimal solution being close to the "latent" labels, i.e., cases where the Rand index in Figure 4 is 1. Second, the runtime depends more on the noise level than on edge density. To save space, we do not illustrate corresponding results for the weaker LP-C relaxation. It generates 12% fewer integral solutions and smaller corresponding Rand indices, but runs faster when there is lots of noise.

4.3. Example applications of stability

We now apply stability analysis to investigate the properties of clustering solutions in two applications.

"Critical edges" in Modularity Clustering Modularity clustering is a popular tool to analyze networks. But which edges are *critical* for the partition at hand, i.e., their removal will change the optimal solution? To test whether an edge e is critical, we compute the stability range for the perturbation $\boldsymbol{d} = w_M(V, E \setminus \{e\}) - w_M(V, E)$, where w_M computes the modularity edge weights from the original undirected, unweighted graph. For $\theta = 1$, the GPP weights will correspond to $E \setminus \{e\}$, so e is critical if and only if $1 \notin [\rho_{\boldsymbol{d},-}, \rho_{\boldsymbol{d},+}]$. Figure 6 illustrates the critical edges on top of the partitioning of the `karate` network, an example for a social network.

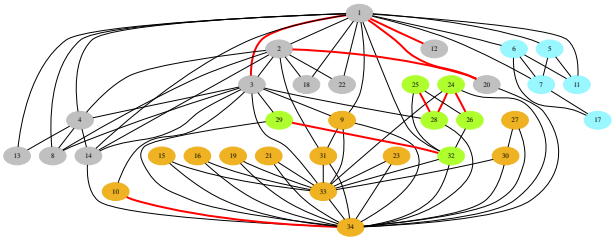

Figure 6. Critical edges in Zarachy's karate club network with four groups. A removal of any red edge would change the current (best) partitioning. All other edges can be removed individually without changing the solution. (Figure best viewed in color.)

Clustering Solution Path The solution path can reveal more information about a data set than one partition alone. Our data (courtesy of Frank Jäkel) contains pairwise similarities of 26 types of leaves in the form of human confusion rates. To investigate groups of leaves induced by those similarities, we solve GPP on a similarity graph with edge weights equal to the symmetrized confusion rates. This corresponds to weighted correlation clustering, where negative weights indicate dissimilarity.

We make low similarities negative by adding a threshold $\theta < 0$ from each edge ($\boldsymbol{d} = \mathbf{1}$). It is not obvious how to set θ; a higher θ will result in few clusters. Hence, we trace the solution path for $\theta = 0$ to the point when each node is a cluster.

Figure 7 illustrates how the stability ranges of the solutions vary along the path. Figure 8 shows some stable solutions. At change points of the path, the optimal solution often changes only little, as indicated by the Rand index (Rand, 1971). This means that many solutions are very similar and might represent the same underlying clustering. Indeed, the path reveals structural characteristics of the data: low-density areas in the graph will be cut first, whereas some leaves remain together throughout almost the entire path and form dense sub-communities. Leaves that are fluctuating between groups are not clearly categorized and likely to be at the boundary between two clusters.

In general, the solution path provides richer information than one single clustering and permits a more careful analysis of the data, in particular if the value of a decisive model parameter is uncertain.

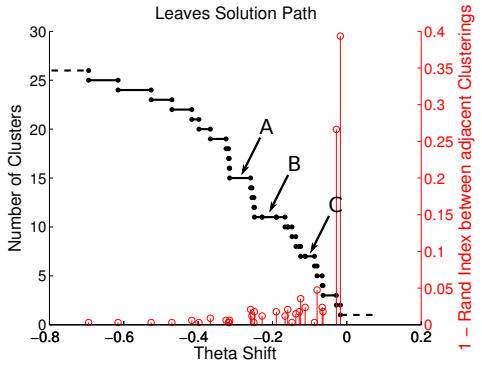

Figure 7. Clustering solution path for the leaves dataset. The red stems show the difference of adjacent clusterings.

5. Conclusions

We have shown a new general method to compute stability ranges for combinatorial problems. Applied to a unifying formulation, GPP, this method opens up new ways to carefully analyze graph partitioning problems. The experiments illustrate examples for GPP and an analysis of the method. A useful extension will be to find the perturbation to which the solution is most sensitive, rather than specifying the direction beforehand. Given the generality of the method developed in this work, where else could the analysis of solution stability lead to further insights? Examples may be other learning settings, algorithms that make use of combinatorial optimization, or theoretical analyses.

All code is made available as open-source at `http://www.kyb.mpg.de/bs/people/nowozin/lpstability/`.

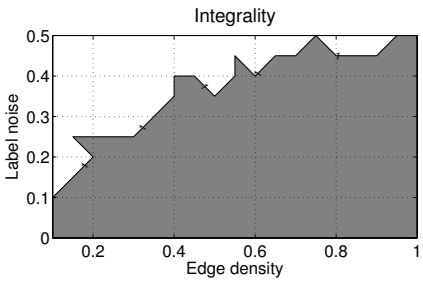

Figure 3. Parameters for which solutions were integral (gray).

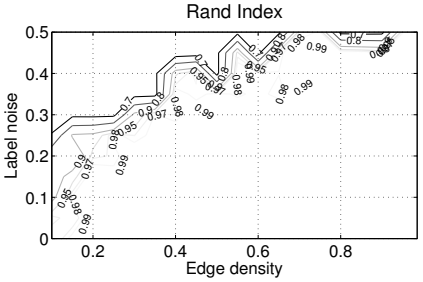

Figure 4. Mean Rand index of the partitioning vs. latent classes.

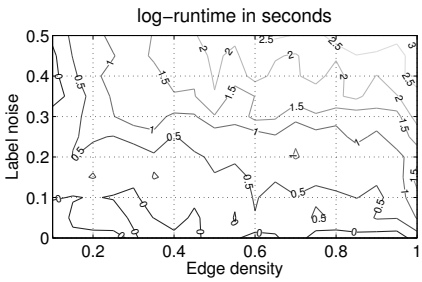

Figure 5. Log_{10}-runtime in seconds, averaged over ten runs.

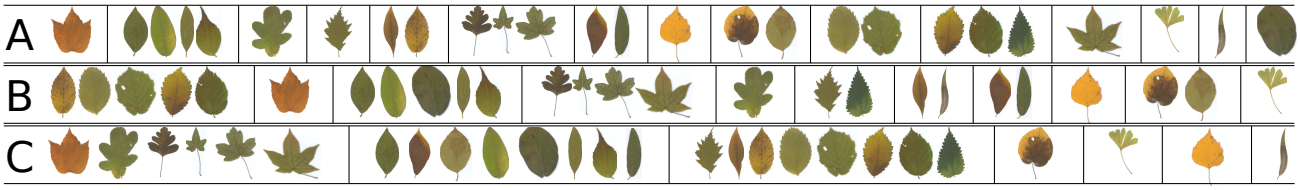

Figure 8. Stable solutions (A) 15 clusters, (B) 11 clusters, (C) 7 clusters. Grouped leaves are in the same cluster.

Acknowledgments. We thank Suvrit Sra, Christoph Lampert and Peter Gehler for discussions. The first author was funded by the EC project CLASS IST 027978 and the PASCAL2 network.

References

Bansal, N., Blum, A., & Chawla, S. (2002). Correlation clustering. *Foundations of Computer Science (FOCS'2002)* (pp. 238–247). IEEE.

Bertsimas, D., & Tsitsiklis, J. N. (1997). *Introduction to linear optimization.* Athena Scientific, Massachusetts.

Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z., & Wagner, D. (2008). On modularity clustering. *IEEE Trans. Knowl. Data Eng*, *20*, 172–188.

Chopra, S., & Rao, M. R. (1993). The partition problem. *Math. Program*, *59*, 87–115.

Demaine, E. D., Emanuel, D., Fiat, A., & Immorlica, N. (2006). Correlation clustering in general weighted graphs. *Theor. Comput. Sci*, *361*, 172–187.

Deza, M. M., Grötschel, M., & Laurent, M. (1992). Clique-web facets for multicut polytopes. *Mathematics of Operations Research*, *17*, 981–1000.

Deza, M. M., & Laurent, M. (1997). *Geometry of cuts and metrics.* Springer.

Emanuel, D., & Fiat, A. (2003). Correlation clustering – minimizing disagreements on arbitrary weighted graphs. *Eur. Symp. Alg.* (pp. 208–220). Springer.

Finley, T., & Joachims, T. (2005). Supervised clustering with support vector machines. *Intl. Conf. Mach. Learn.* (pp. 217–224). ACM.

Gaertler, M., Görke, R., & Wagner, D. (2007). Significance-driven graph clustering. *Algorithmic Aspects in Information and Management.* Springer.

Gionis, A., Mannila, H., & Tsaparas, P. (2007). Clustering aggregation. *Trans. on Know. Discovery from Data*, *1*.

Grötschel, M., & Wakabayashi, Y. (1989). A cutting plane algorithm for a clustering problem. *Math. Prog.*, *45*, 59–96.

Grötschel, M., & Wakabayashi, Y. (1990). Facets of the clique partitioning polytope. *Math. Prog.*, *47*.

Jansen, B., Jong, J., Roos, C., & Terlaky, T. (1997). Sensitivity analysis in linear programming: Just be careful! *European Journal of Operational Research*, *101*, 15–28.

Joachims, T., & Hopcroft, J. E. (2005). Error bounds for correlation clustering. *Intl. Conf. Mach. Learn.* (pp. 385–392). ACM.

Kernighan, B. W., & Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 291–307.

Kılınç-Karzan, F., Toriello, A., Ahmed, S., Nemhauser, G., & Savelsbergh, M. (2007). *Approximating the stability region for binary mixed-integer programs* (Technical Report). Gatech.

Newman, M. (2004). *Analysis of weighted networks* (Technical Report). Cornell University.

Newman, M. E. J., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, *69*.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *American Statistical Association Journal*, *66*, 846–850.

Schrijver, A. (1998). *Theory of linear and integer programming.* John Wiley & Sons.

Wolsey, L. A. (1998). *Integer programming.* John Wiley.