# Interactive Correlation Clustering with Existential Cluster Constraints

**Rico Angell** [1] **Nicholas Monath** [2] **Nishant Yadav** [1] **Andrew McCallum** [1]

## Abstract

We consider the problem of clustering with user feedback. Existing methods express constraints about the input data points, most commonly through must-link and cannot-link constraints on data point pairs. In this paper, we introduce *existential cluster constraints*: a new form of feedback where users indicate the features of desired clusters. Specifically, users make statements about the existence of a cluster having (and not having) particular features. Our approach has multiple advantages: (1) constraints on clusters can express user intent more efficiently than point pairs; (2) in cases where the users' mental model is of the desired clusters, it is more natural for users to express cluster-wise preferences; (3) it functions even when privacy restrictions prohibit users from seeing raw data. In addition to introducing existential cluster constraints, we provide an inference algorithm for incorporating our constraints into the output clustering. Finally, we demonstrate empirically that our proposed framework facilitates more accurate clustering with dramatically fewer user feedback inputs.

## 1. Introduction

Real-world deployed machine learning systems will inevitably be imperfect despite ever growing labeled training data and model sizes. Machine learning models will encounter novel inputs and patterns and noise unseen at training which will lead to incorrect predictions. This lack of precision is amplified in applications where training data is limited or the objective is misaligned with the task. Fortunately, human users interacting with machine learning systems over time have the ability to provide feedback to correct the model outputs and thus improve the precision and safety of the system.

[1]Manning College of Information and Computer Sciences, University of Massachusetts Amherst [2]Google Research. Correspondence to: Rico Angell <rangell@cs.umass.edu>.

*Proceedings of the $39^{th}$ International Conference on Machine Learning*, Baltimore, Maryland, USA, PMLR 162, 2022. 

Techniques and approaches for gathering, representing, and utilizing human feedback have been developed for tasks in both supervised and unsupervised machine learning. For classification, human-in-the-loop active labeling techniques have been employed to gather labels for problematic inputs and fine-tune the model (Settles, 2009). Weak supervision enables noisy label gathering by allowing users to write labeling rules for data where labeled data is scarce (Ratner et al., 2017). Integrating human feedback about relevant features instead of input example labels into classification models has been shown useful (Chang et al., 2007; Druck et al., 2009; Liang et al., 2009; Ganchev et al., 2010; Dasgupta et al., 2018). Human preferences about agent actions in complicated environments have been integrated into reinforcement learning systems to improve policy learning (Christiano et al., 2017; Griffith et al., 2013). Tasks with difficult to define objectives, such as learning to summarize text (Stiennon et al., 2020), have been shown to benefit from human feedback. Interactive models have been used for applications such as knowledge base construction (Kratzwald et al., 2020) and entity resolution (Kobren et al., 2019). Human user feedback has also been used to correct both flat and hierarchical clusterings (Balcan & Blum, 2008; Davidson & Basu, 2009; Vikram & Dasgupta, 2016; Mazumdar & Saha, 2017; Bibi et al., 2019; Nie et al., 2021).

Clustering is an area of machine learning that could particularly benefit from human feedback. Since clustering is often unsupervised and used for discovery of entities, patterns, and groups, it generally needs additional input or side information to guide predictions. In past work on clustering with human feedback, users interact with the predicted clustering and provide feedback in the form of pairwise point constraints (Wagstaff & Cardie, 2000), relative comparisons among data points (Frome et al., 2007), and cluster sizes (Xu et al., 2009). Existing approaches either directly integrate this feedback into existing clustering frameworks (Wagstaff et al., 2001; Shental et al., 2004; Klein et al., 2002) or adapt the similarity measure of the data to produce a more accurate predicted clustering (Kulis et al., 2009; Lu & Carreira-Perpinan, 2008; Li et al., 2008; 2009; Xing et al., 2002).

In this paper, we introduce a novel feedback mechanism for clustering in which users make statements about the *features* of desired clusters. More specifically, human users

view the predicted clustering and make statements about the existence of a cluster containing a specific subset of discrete features and not containing another disjoint subset of discrete features. We posit this enables users to convey cluster-wise preferences more naturally with respect to a user's mental model of the output clustering. Furthermore, existential cluster constraints are more efficient in terms of the number of human feedback inputs to reach an ideal output clustering both in theory and practice. Lastly, unlike prior forms of feedback, existential cluster constraints allow users to provide feedback when data privacy is a concern without viewing the raw data or cluster sizes.

We show theoretically that existential cluster constraints are able to express feedback with far fewer human inputs than pairwise point constraints. Unsurprisingly, we also show that it is *NP*-hard in general to find a clustering which satisfies all of the existential cluster constraints.

We introduce an inference algorithm for clustering with existential cluster constraints within the correlation clustering framework. The algorithm begins by solving a semidefinite programming (SDP) relaxation of an integer linear program which jointly clusters the data points and existential cluster constraints by combining the traditional correlation clustering optimization problem with extra optimization constraints formed from the existential cluster constraints. Next, we construct a DAG-shaped hierarchical clustering (i.e., sparse cluster trellis (Macaluso et al., 2021)) over the data points and existential cluster constraints using the solution of SDP relaxation as an affinity matrix. Finally, we extract the optimal flat clustering consistent with the trellis constructed in the previous step using a dynamic programming algorithm which memoizes optimal partial clusterings.

We evaluate the efficiency of existential cluster constraints in terms of human feedback inputs and the efficacy of our inference algorithm on publicly available author coreference datasets. Author coreference is a real world clustering task where the predictions are viewed by human users on websites such as Google Scholar[1], Semantic Scholar[2], and OpenReview[3]. We simulate the process of human feedback in a series of rounds in which an oracle generates one or more constraints to correct the previous round's predictions and run our inference algorithm on the running set of constraints. We compare existential cluster constraint directly against must-link and cannot-link constraints. On every dataset, we find existential cluster constraints require nearly half as many user feedback inputs than must-link and cannot-link constraints to reach the ground truth clustering.

[1]scholar.google.com
[2]semanticscholar.org
[3]openreview.net

### 1.1. Contributions

In summary, our contributions are as follows:

- **Novel Feedback Paradigm.** We introduce a novel form of feedback for interactive clustering where users make statements about the existence of a clustering containing (and not containing) particular discrete features. (Section 3)
- **Theoretical Analysis of Paradigm.** We theoretically analyze the efficiency and hardness of satisfying this novel form of feedback. (Sections 3.1-3.2)
- **Inference Algorithm.** We detail an inference algorithm for incorporating existential cluster constraints into the correlation clustering framework. (Section 4)
- **Empirical Evaluation.** We empirically evaluate the efficiency of existential cluster constraints and the efficacy of our inference algorithm. (Section 5)

## 2. Problem Setup

In this section, we will describe the problem setting that will be the foundation from which we will detail our contributions. We will define the correlation clustering objective used throughout the remainder of the paper, the domain of the vertices and cluster features used to express existential cluster constraints, and our assumptions about a latent ground truth clustering.

### 2.1. Correlation Clustering

Correlation clustering is a graph-based clustering framework that has several advantages over other clustering objectives (Bansal et al., 2004; Swamy, 2004; Demaine et al., 2006). For example, correlation clustering does not require the user to specify the number of clusters or a threshold for choosing a particular flat clustering. There are several correlation clustering objectives with different properties, but we will use a version of the max-agree correlation clustering objective.

In our setting, we are given a graph $G = (V, E)$ of $n$ vertices where each edge $e = (u, v)$ has a weight $w_e \in \mathbb{R}$ (sometimes written as $w_{uv}$) representing the affinity between $u$ and $v$. A *clustering* $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of vertices $V$ is a set of nonempty sets such that $C_i \subseteq V$, $C_i \cap C_j = \emptyset$ for all $i \neq j$, and $\bigcup_{C \in \mathcal{C}} C = V$. The goal of the correlation clustering problem (Bansal et al., 2004) is to find a clustering of the vertices which maximizes the sum of positive edge-weights within each cluster minus the negative edge-weights across the clusters. Let $E^+$ and $E^-$ be the subsets of edges with positive and negative edge weights, respectively. Formally, the max-agree correlation clustering

optimization problem can be written as follows:

$$\max \sum_{(u,v)\in E^+} w_{uv}\, x_u \cdot x_v - \sum_{(u,v)\in E^-} w_{uv}(1 - x_u \cdot x_v) \quad \text{s.t.} \quad x_v \in \{e_1, e_2, \ldots, e_n\}, \quad v \in V \tag{1}$$

where $e_i \in \mathbb{R}^n$ is the $i$th standard basis vector. A complete assignment to the decision variables $x_v$ defines a clustering of the vertices: $x_v = e_i$ implies vertex $v$ is assigned to cluster $i$. Conversely, any clustering of the vertices can be expressed in an assignment to the decision variables. Additionally, it is well known that Problem (1) is intractable to optimize exactly, unless *P = NP* (Bansal et al., 2004).

Alternatively, consider the following optimization problem:

$$\max \sum_{(u,v)\in E} w_{uv}\, x_u \cdot x_v \quad \text{s.t.} \quad x_v \in \{e_1, e_2, \ldots, e_n\}, \quad v \in V \tag{2}$$

This objective considers only the *intra*-cluster edges of a clustering regardless of the sign of the weights. By only considering intra-cluster edges, the partial objective computed over a subset of vertices $S \subset V$ is independent from the choice of partitioning the remaining set of vertices $V \setminus S$. This property makes Problem (2) more favorable than Problem (1) for optimization.

In general, clustering objectives can be viewed as a way to rank candidate solutions. Interestingly, Problem (1) and Problem (2) are *order-equivalent* (See the supplementary material of Greenberg et al. (2018) for the proof). In other words, these objectives generate the same complete ranking of candidate clusterings. We will use Problem (2) as our correlation clustering objective since it has desirable properties for optimization which we will utilize in Section 4.2.

### 2.2. Vertex and Cluster Features

In order to specify clustering constraints in terms of the features of desired clusters, we need to define the features of vertices and clusters we will consider in our setting. We assume that each of the $n$ vertices has an associated set of discrete features. Let $\Phi = \{\phi_1, \phi_2, \ldots\}$ be the set of possible features and $\Phi(v) \subseteq \Phi$ be the subset of features associated with vertex $v \in V$. Furthermore, for any subset of vertices $S \subseteq V$, we define the set of features associated with $S$ as $\Phi(S) = \bigcup_{v\in S} \Phi(v)$. We make no assumptions about the relationship between features and edge weights.

### 2.3. Ground Truth Clustering

We assume that there exists a latent ground truth clustering $\mathcal{C}^*$ which may be different from the clustering that optimizes the correlation clustering objective. Let $k$ be the number of ground truth clusters. We refer to this problem of achieving the latent ground truth clustering through interactive feedback as the *interactive correlation clustering problem with discrete features*.

## 3. Existential Cluster Constraints

Most commonly, feedback in clustering is provided in the form of must-link and cannot-link constraints where users specify whether pairs of points must or cannot be in the same cluster, respectively. We can easily integrate these pairwise point constraints into the correlation clustering framework by re-assigning the edge weight for each must-link and cannot-link constraint to $\infty$ and $-\infty$, respectively. Despite being easy to incorporate into the correlation clustering framework, pairwise point constraints have several practical downsides: (1) they can be inefficient in expressing large sweeping changes to whole clusters; (2) it might be unnatural for users to express desired attributes of output clusters when expressing feedback at the level of pairs of points; (3) we may not want users to interact with the raw data points directly.

In response, we propose $\exists$-*constraints* (pronounced *"there-exists constraints"*), a novel form of feedback which enables users to express statements about the existence of a cluster with and without particular cluster features. Formally, a $\exists$-constraint $\xi \subseteq \{+, -\} \times \Phi$ uniquely characterizes a constraint asserting there exists a cluster $C \in \mathcal{C}^*$ such that all of the positive features $\xi^+ := \{\phi \,|\, (+, \phi) \in \xi\}$ are contained in the features of $C$ (i.e. $\xi^+ \subseteq \Phi(C)$) and none of the negative features $\xi^- := \{\phi \,|\, (-, \phi) \in \xi\}$ are contained in the features of $C$ (i.e. $\xi^- \cap \Phi(C) = \emptyset$). At any given time, let the set of $\exists$-constraints be represented by $\Xi$. We say that a subset of nodes $S \subseteq V$ *satisfies* a $\exists$-constraint $\xi$ if $\xi^+ \subseteq \Phi(S)$ and $\xi^- \cap \Phi(S) = \emptyset$. Observe that it is perfectly feasible for more than one ground truth cluster to satisfy a $\exists$-constraint.

We say that a subset of vertices $S \subseteq V$ is *incompatible* with a $\exists$-constraint $\xi$ if $\Phi(S) \cap \xi^- \neq \emptyset$ and denote this as $S \perp \xi$. Similarly, two $\exists$-constraints $\xi_a, \xi_b$ are incompatible if $\xi_a^+ \cap \xi_b^- \neq \emptyset$ or $\xi_a^- \cap \xi_b^+ \neq \emptyset$, and is also denoted $\xi_a \perp \xi_b$. Furthermore, we say a subset of vertices $S \subseteq V$ and a $\exists$-constraint $\xi$ or two $\exists$-constraints $\xi_a, \xi_b$ are *compatible* if they are not incompatible and denote this as $S \not\perp \xi$ and $\xi_a \not\perp \xi_b$, respectively. Moreover, we use *compatible* to describe when a subset of vertices $S \subseteq V$ could be part of a cluster that *satisfies* a certain $\exists$-constraint $\xi$. This definition allows for $\Phi(S) \cap \xi^+$ to be empty, but still be $S$ is compatible with $\xi$ just as long as $\Phi(S) \cap \xi^-$ is empty.

**Example Use Case of $\exists$-constraints .** Consider the problem of author coreference where there is a database of papers, grants, and other professional information about au-

thors of scientific papers and we want to decide which of these belong to the same individual. This problem can be framed as a clustering problem where the output is some aggregation of the information (features) into clusters. Now consider the following example of how $\exists$-constraints could be used to correct errors made by a system to solve this problem. Suppose there are two different authors "Alice Smith" and "Alice R. Smith" where "Alice Smith" has a co-author "Bob Johnson" on multiple papers and "Alice R. Smith" never co-authored a paper with anyone named "Bob Johnson". Now suppose the current output of the clustering algorithm predicts that the records of "Alice Smith" and "Alice R. Smith" belong to the same individual. To correct the clustering error, a human could provide feedback in the form of a $\exists$-constraint as follows:

$$\begin{aligned}\xi = \{&(+, \texttt{first\_name} : \text{“Alice”}),\\ &(+, \texttt{middle\_initial} : \text{“R”}),\\ &(+, \texttt{last\_name} : \text{“Smith”}),\\ &(-, \texttt{coauthor\_name} : \text{“Bob Johnson”})\}.\end{aligned}$$

Incorporating this $\exists$-constraint will cause the cluster to be split such that "Alice R. Smith" and "Alice Smith" are no longer in the same cluster (there might still be other errors which need to be fixed, but these can be fixed with additional feedback from the user). Note how this could be much more efficient in terms of human feedback as compared to must-link and cannot-link constraints. If "Alice Smith" wrote many papers with "Bob Johnson", the user might have to provide many more cannot-link constraints in order to split "Alice Smith" and "Alice R. Smith".

### 3.1. Efficiency of Feedback

The purpose of interaction in clustering is to enable users to correct mistakes in the predicted clustering by providing guidance to the algorithm through feedback. When providing feedback, we assume users are able to view the cluster features of predicted clusters and then provide one or more $\exists$-constraints. To ease the burden on the user, the feedback mechanism should ideally be both natural for the user to specify given their mental model of the latent ground truth clustering as well as efficient in terms of the number of user feedback inputs they need to provide. We posit that $\exists$-constraints are both more efficient and natural to specify than previous feedback mechanisms.

Intuitively, specifying an existential cluster constraint is natural for a user. When viewing the features of a predicted cluster, a user has a mental model of the corresponding latent ground truth cluster in terms of the salient features that both belong and do not belong in the ground truth cluster. By better aligning the feedback mechanism with the user's mental model, we ease the burden on the user, and thus, make the process of providing feedback more efficient.

Existential cluster constraints are also more efficient along another dimension, namely, the number of human feedback inputs. Each existential cluster constraint is a compressed representation of a powerful logical statement about the output clustering. Theoretically, we provide a worst-case lower bound on how much more powerful $\exists$-constraints can be when compared to must-link and cannot-link constraints in our setting. Note the following proposition.

**Proposition 3.1.** *The interactive correlation clustering problem with discrete features will require $\Omega(k^2)$ more pairwise node constraints than total features required by $\exists$-constraints to achieve the latent ground truth clustering in the worst case, where $k$ is number of clusters.*

See Appendix A.1 for the proof. We will show empirical analysis of efficiency of $\exists$-constraints compared with must-link and cannot-link constraints in Section 5.4.

### 3.2. Hardness of Satisfaction

Given the representational power and potentially combinatorially many ways a $\exists$-constraint could be satisfied, it seems intractable to efficiently infer a clustering which satisfies all $\exists$-constraints $\Xi$ in general. In fact, we show that even without considering the clustering objective it is *NP*-hard to return a clustering of the vertices that satisfies all of the $\exists$-constraints $\Xi$. Note the following proposition.

**Proposition 3.2.** *Given a graph $G = (V, E)$, a discrete feature map $\Phi$, and a set of $\exists$-constraints $\Xi$, it is NP-hard to determine if there exists a clustering of the vertices $V$ which satisfies all of the $\exists$-constraints.*

See Appendix A.2 for the proof of reduction from SAT. This directly implies the following corollary.

**Corollary 3.3.** *There does not exist an exact polynomial time algorithm that can guarantee all valid $\exists$-constraints will be satisfied by the inferred clustering, unless P = NP.*

This follows from a simple contradiction argument. If there existed a polynomial time clustering algorithm that can guarantee all valid $\exists$-constraints will be satisfied by the inferred clustering, then we could use this algorithm to solve the decision problem determined to be *NP*-hard in Proposition 3.2.

## 4. Algorithm

In this section, we present our inference algorithm for correlation clustering with $\exists$-constraints. We start by integrating the $\exists$-constraints into the optimization problem presented in (2). Our general approach is to jointly cluster the vertices and the $\exists$-constraints, so we create additional variables for each of the $\exists$-constraints. Additionally,

we create constraints to ensure that all of the constraints are satisfied and none of the constraints are incompatible with any of the other members of the predicted clustering. Given a $\exists$-constraint $\xi$ and positive feature $\phi \in \xi^+$, let $\mathbb{S}_\phi^\xi := \{v \in V : \phi \in \Phi(v) \wedge \Phi(v) \cap \xi^- = \emptyset\}$ be the set of candidate vertices that could satisfy $\phi \in \xi^+$. The optimization problem with $\exists$-constraints integrated can be written as follows:

$$\begin{aligned} \max \sum_{(u,v)\in E} & w_{uv}\, x_u \cdot x_v \\ \text{s.t.} \quad & x_s \in \{e_1, e_2, \ldots, e_n\}, \quad s \in V \cup \Xi \\ & \sum_{v \in \mathbb{S}_\phi^\xi} x_v \cdot x_\xi \geq 1, \quad \phi \in \xi^+, \xi \in \Xi \\ & x_s \cdot x_\xi = 0, \quad s \in V \cup \Xi, \xi \in \Xi : s \perp \xi \end{aligned} \tag{3}$$

The second constraint ensures that all of the positive features in $\xi^+$ are contained in the same cluster as $\xi$ for all $\xi \in \Xi$. The last constraint ensures that $\xi$ is not in the same cluster as anything incompatible with it for all $\xi \in \Xi$. This optimization problem is intractable to optimize in general by Corollary 3.3. Our general approach is to relax Problem (3) to an SDP and then round the fractional solution from the SDP to obtain a predicted clustering of the vertices and the $\exists$-constraints.

### 4.1. SDP Relaxation

We relax (3) following the standard SDP relaxation used for max-agree correlation clustering. The decision variables in the optimization problem are changed from standard basis vectors whose dot products determine whether or not two elements (vertices or $\exists$-constraints) are in the same cluster to a positive semi-definite matrix where each entry represents the global affinity two elements have for one another (corresponding to the row and column of the entry). We constrain the diagonal of this positive semi-definite matrix to be all ones (representing each element's affinity with itself) and we enforce that all entries in the matrix are non-negative. The constraints ensuring $\exists$-constraint satisfaction are similar to (3).

Formally, the SDP relaxation is as follows:

$$\begin{aligned} \max \sum_{(u,v)\in E} & w_{uv}\, X_{uv} \\ \text{s.t.} \quad & X \succeq 0, \\ & X_{ss} = 1, \quad s \in V \cup \Xi \\ & X_{st} \geq 0, \quad s, t \in V \cup \Xi \\ & \sum_{v \in \mathbb{S}_\phi^\xi} X_{v\xi} \geq 1, \quad \phi \in \xi^+, \xi \in \Xi \\ & X_{s\xi} = 0, \quad s \in V \cup \Xi, \xi \in \Xi : s \perp \xi \end{aligned} \tag{4}$$

### 4.2. Trellis-based Rounding Algorithm

After we have solved the SDP relaxation, we need to round the fractional solution to maximize not only the number of $\exists$-constraints satisfied, but also the correlation clustering objective. We accomplish this by constructing a data structure over the potential (partial) clusterings of $V \cup \Xi$ and then selecting the global optimal clustering from that data structure using a dynamic programming algorithm.

#### 4.2.1. Sparse Cluster Trellis

A *sparse cluster trellis* (or just *trellis*, for short) $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ is a directed acyclic graph where the vertices $\mathcal{V} \subseteq \mathcal{P}(V \cup \Xi)$ (Greenberg et al., 2018; Macaluso et al., 2021). The set of vertices always includes the *leaves* $\{s\}_{s \in V \cup \Xi}$ and the *root* $V \cup \Xi$. There exists a directed edge from one vertex $S_\ell \in \mathcal{V}$ (the *child*) to another $S_p \in \mathcal{V}$ (the *parent*) if $S_\ell$ is a maximal subset of $S_p$. For every child $S_\ell$ of $S_p$ there also exists a *complimentary child* $S_r = S_p \setminus S_\ell \in \mathcal{V}$. We define the set of child, complementary child pairs for a vertex $S_p \in \mathcal{V}$ as $\texttt{splits}_\mathcal{T}(S_p) := \{(S_\ell, S_r) \in \mathcal{V}^2 : S_\ell \cup S_r = S_p \wedge S_\ell \cap S_r = \emptyset\}$. A sparse cluster trellis is a generalization of a hierarchical clustering tree which allows for more clusterings to be represented. In practice, we construct a sparse cluster trellis by first constructing one or more binary hierarchical clusterings of $V \cup \Xi$ guided by the solution to the SDP, $X^*$, and then combining them into a single data structure (Macaluso et al., 2021).

#### 4.2.2. Trellis-Consistent Partition

Given a trellis $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, a *trellis-consistent partition* is a clustering $\mathcal{C}$ of $V \cup \Xi$ such that $C \in \mathcal{V}$ for all $C \in \mathcal{C}$. A $\exists$-constraint $\xi$ is satisfied in a trellis-consistent partition $\mathcal{C}$ if $\xi$ is satisfied by $C \cap V$ for $\xi \in C \in \mathcal{C}$. The number of trellis-consistent partitions encoded by a trellis is exponential in $|\mathcal{V}|$. Since 3 only considers intra-cluster edges, we can extract the trellis-consistent partition which maximizes the objective using a dynamic programming algorithm which scales linearly in $|\mathcal{V}|$. We refer to the process of extracting a trellis-consistent partition as *cutting* the trellis.

#### 4.2.3. Trellis Cut Algorithm

Once the trellis is constructed, we need to efficiently extract the optimal trellis-consistent partition. This is a multi-objective optimization problem: we want to maximize both the number of $\exists$-constraints satisfied in the trellis-consistent partition (we know from Section 3.2 that it is intractable to satisfy all the $\exists$-constraints in general) and, secondarily, maximize the correlation clustering objective. We extract the optimal trellis-consistent partition using a dynamic programming algorithm where we memoize the best partial solution at each vertex in the trellis. To efficiently memoize the best partial solution at each vertex, we compute

**Algorithm 1** TRELLISCUT$(G, \Phi, \Xi, \mathcal{T})$

Initialize assignment map, $a[\,\cdot\,] : \mathcal{V} \to \mathcal{P}(\mathcal{V})$
Initialize constraint satisfaction map, $s[\,\cdot\,] : \mathcal{V} \to \mathbb{Z}_+$
Initialize objective value map, $o[\,\cdot\,] : \mathcal{V} \to \mathbb{R}$
**for** $S_p \in \texttt{toposort}_{\mathcal{T}}(\mathcal{V})$ **do**
  $a[S_p] \leftarrow \{S_p\}$
  $s[S_p] \leftarrow \sum_{\xi \in S_p \cap \Xi} \mathbb{1}\{S_p \cap V \text{ satisfies } \xi\}$
  $o[S_p] \leftarrow \sum_{u,v \in S_p \cap V} w_{uv}$
  **for** $(S_\ell, S_r) \in \texttt{splits}_{\mathcal{T}}(S_p)$ **do**
    **if** $(s[S_\ell] + s[S_r] > s[S_p]$
      **or** $(s[S_\ell] + s[S_r] = s[S_p]$
        **and** $o[S_\ell] + o[S_r] > o[S_p]))$ **then**
      $a[S_p] \leftarrow a[S_\ell] \cup a[S_r]$
      $s[S_p] \leftarrow s[S_\ell] + s[S_r]$
      $o[S_p] \leftarrow o[S_\ell] + o[S_r]$
**return** $a[V \cup \Xi]$

the partial solutions for each vertex in the trellis in topological order from leaves to root. Let $\texttt{toposort}_{\mathcal{T}}(\mathcal{V})$ be an ordered set of trellis vertices topologically ordered from leaves to root.

Algorithm 1 details the dynamic programming algorithm we use to optimally cut the trellis. The algorithm takes as input the graph $G = (V, E)$ including edge weights, a discrete feature map $\Phi$, a set of $\exists$-constraints $\Xi$, and a trellis $\mathcal{T} = (\mathcal{V}, \mathcal{E})$. For each vertex $S$ in the trellis, we initialize the following: an assignment map $a[\,\cdot\,] : \mathcal{V} \to \mathcal{P}(\mathcal{V})$ which stores the optimal partitioning of $S$, a constraint satisfaction map $s[\,\cdot\,] : \mathcal{V} \to \mathbb{Z}_+$ which stores the number of constraints satisfied by the optimal partitioning, and an objective value map $o[\,\cdot\,] : \mathcal{V} \to \mathbb{R}$ which stores the correlation clustering objective value for the optimal partitioning. For each vertex $S_p \in \mathcal{V}$ in topological order, we start by assigning the optimal partitioning to be all the vertices $s \in S_p$ to be in a single cluster and store the corresponding number of satisfied constraints and correlation clustering objective value in their respective maps. Then, for each child and complementary child pair $(S_\ell, S_r)$ we check if the union of optimal partitionings from $S_\ell$ and $S_r$ is more favorable than the previously stored partitioning for $S_p$. After iterating over all vertices in the trellis in topological order, the algorithm returns the optimal partitioning of the root.

**Proposition 4.1.** *Given a graph $G = (V, E)$, a discrete feature map $\Phi$, a set of $\exists$-constraints $\Xi$, and a trellis $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, Algorithm 1 returns the trellis-consistent partition which maximizes the number of $\exists$-constraints satisfied a priori, then maximizes the (3) value subject to maintaining the number of $\exists$-constraints satisfied.*

See Appendix A.3 for the proof. The critical idea is that both objectives, the number of $\exists$-constraints satisfied by a trellis-consistent partition and the correlation clustering objective, decompose over the structure of the trellis. This enables the dynamic programming approach to leverage optimal partial solutions to efficiently compute the globally optimal trellis-consistent partition.

Table 1: Author Coreference Data Test Set Statistics.

| | PubMed | QIAN | SCAD-zbMATH |
|---|---|---|---|
| # vertices | 315 | 410 | 1,196 |
| # clusters | 34 | 77 | 166 |
| # features | 14,093 | 10,366 | 8,203 |
| # blocks | 5 | 38 | 120 |
| min block size | 12 | 2 | 2 |
| mean block size | 63.0 | 10.8 | 31.5 |
| max block size | 142 | 100 | 133 |

## 5. Experiments

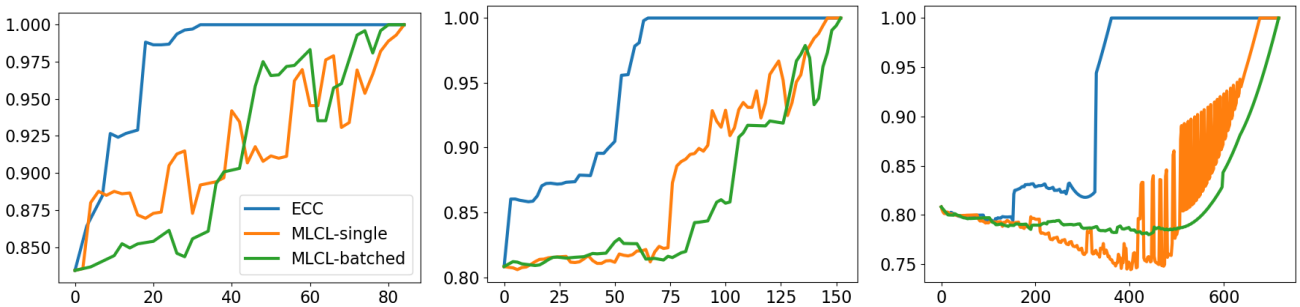
We empirically evaluate the effectiveness of $\exists$-constraints and our inference algorithm in achieving the latent ground truth clustering on several *author coreference* datasets (sometimes referred to as *author name disambiguation*). We directly compare the benefits of $\exists$-constraints with must-link and cannot-link constraints. We accomplish this by simulating a series of rounds of a user generating one or more constraints based on the predicted clustering produced in the previous round and integrating the new constraint(s) into the next prediction. The simulation terminates when the latent ground truth clustering is predicted.

### 5.1. Datasets

We evaluate our constraints and algorithm on several author coreference datasets. Each author coreference dataset is composed of *mentions* (or *records*) of authors. The goal is to cluster the mentions such that each cluster contains all of the mentions of exactly one individual author. The mentions are preprocessed into *blocks*, easy to partition mentions. Commonly, each block is uniquely defined by the first initial and last name of an author. If two author mentions have completely different first initials or last names, they are safely assumed to be from two distinct authors. We use three publicly available datasets from S2AND (Subramanian et al., 2021), a state-of-the-art author coreference system[4].

To produce the edge weights on the graph for each block, we train the pairwise S2AND LightGBM (Ke et al., 2017) classifier model using the pairwise features detailed in Subramanian et al. (2021). We use the default and recommended 80/10/10 train/val/test split for each dataset, tuning all of the hyperparameters on the validation set. For each test set, we construct the complete graph for each block and for

[4]https://github.com/allenai/S2AND




(a) PubMed (b) QIAN (c) SCAD-zbMATH

Figure 1: **CU vs. Adjusted Rand Index.** In each plot, the x-axis is the number of constraint units (CU) used by each feedback constraint method and the y-axis is the corresponding Adjusted Rand Index of the resulting clustering after incorporating the generated constraints. The blue line is existential cluster constraints (ECC), the orange line is must-link and cannot-link constraints generated one at a time (MLCL-single), and the green line is must-link and cannot link constraints generated in batches (MLCL-batched) to mimic ECC.

each edge assign edge weights based on the trained pairwise classifier output logits shifted by a threshold tuned on the validation set. If the classifier score is above (below) the threshold, the edge weight will be positive (negative). The resulting graph will be taken as input into the simulation of clustering with human feedback. Test set statistics for each dataset can be seen in Table 1.

We extract discrete features from each record given the text features for each field in the record. Features we extract include, but are not limited to, names, affiliation n-grams, email address n-grams, paper title n-grams, co-author names, co-author affiliation n-grams, etc. All of these discrete features are visible in the predicted cluster features and can be used by the constraint generator simulating human feedback.

### 5.2. Constraint Generation

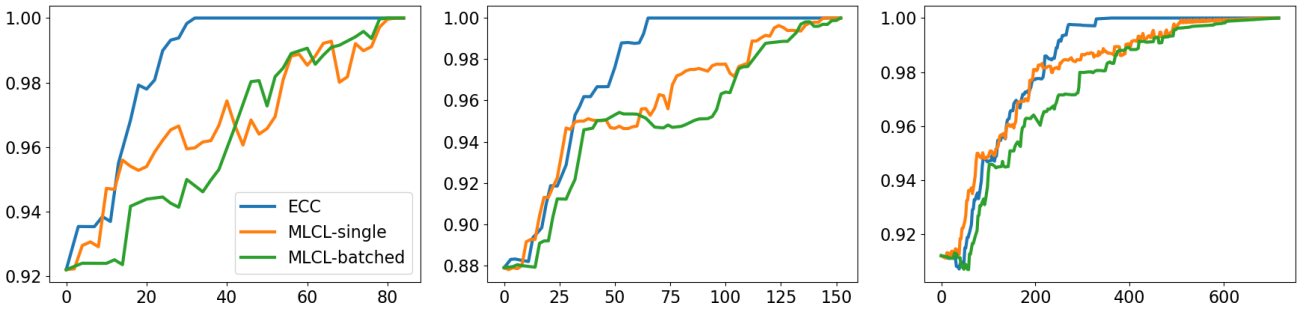
We implement an oracle constraint generator for both $\exists$-constraints and must-link and cannot-link constraints to efficiently and effectively simulate a human user providing feedback. At each round, the generator has access to latent ground truth clustering, the predicted clustering, and the discrete features for all of the mentions. To generate a constraint (or set of constraints), the generator first selects a predicted cluster and a target gold cluster. The generator selects the gold cluster that has the lowest similarity with its most similar predicted cluster (computed based on Jaccard similarity of the sets of vertices). This pair of gold and predicted clusters has the most potential for improving the clustering.

After choosing a pair of gold and predicted clusters, the generator constructs constraint(s) which aim to correct the predicted cluster to look more like the gold cluster. There are three sets of vertices in the pair of gold and predicted clusters: vertices in both the gold and predicted clusters (overlapping vertices), vertices in the gold cluster and not in the predicted cluster (positive vertices), and vertices in the predicted cluster and not in the gold cluster (negative vertices). To generate a $\exists$-constraint, the generator picks a salient feature from the overlapping vertices, a salient feature from each predicted cluster containing any of the positive vertices, and a salient feature from each gold cluster containing any of the negative vertices. We have two modes for generating must-link and cannot-link constraints: batched and single. To generate a batch of must-link and cannot link constraints, the generator picks a vertex from the overlapping vertices and creates a must-link constraint to a vertex in each predicted cluster containing any of the positive vertices, and creates a cannot-link constraint to a vertex in each gold cluster containing any of the negative vertices. To generate a single must-link or cannot link constraint, we generate a batch of must-link and cannot-link constraints and pick the constraint with the potential to correct the clustering of the highest number of vertices. Hence, the constraint generator will never generate a constraint (of any type) which is already satisfied.

### 5.3. Metrics

We use several metrics to compare $\exists$-constraints with must-link and cannot-link constraints. To directly compare the amount of human feedback input, we define *constraint units* (CU) to be two units for every must-link and cannot-link constraint and one unit for every feature in each $\exists$-constraint. The number of CUs is representative of how many selections users need to make to specify both types of constraints. We will use CU as the metric for comparing the efficiency of both types of constraints.

We use three different metrics to measure the quality of the




(a) PubMed (b) QIAN (c) SCAD-zbMATH

Figure 2: **CU vs. FMC.** In each plot, the x-axis is the number of constraint units (CU) used by each feedback constraint method and the y-axis is the corresponding feature match coefficient (FMC) of the resulting clustering after incorporating the generated constraints. The blue line is existential cluster constraints (ECC), the orange line is must-link and cannot-link constraints generated one at a time (MLCL-single), and the green line is must-link and cannot link constraints generated in batches (MLCL-batched) to mimic ECC.

Table 2: Initial Clustering Metrics and Constraint Quantities Utilized to Achieve Latent Ground Truth Clustering.

| | **PubMed** | **QIAN** | **SCAD-zbMATH** |
|---|---|---|---|
| Initial F1 | 95.6 | 96.0 | 96.8 |
| Initial Rand Index | 83.4 | 80.8 | 80.8 |
| Initial FMC | 92.2 | 87.9 | 91.2 |
| # ECC | 14 | 27 | 139 |
| # MLCL-single | 42 | 73 | 338 |
| # MLCL-batched | 40 | 76 | 358 |
| CU ECC | 32 | 65 | 361 |
| CU MLCL-single | 84 | 146 | 676 |
| CU MLCL-batched | 80 | 152 | 716 |

predicted clustering in each round. First, we want a measure of the accuracy of the features of the predicted clusters. We define the *feature match coefficient* (FMC) to be the average Jaccard similarity between each gold cluster features and its most similar (by Jaccard similarity) predicted cluster features. The two other metrics we use are standard clustering metrics for measuring the accuracy of the clustering of the vertices: F1 (sometimes referred to as V-measure in the clustering literature) and Adjusted Rand Index.

### 5.4. Results

In Table 2 we detail the initial clustering metrics (without any constraints) for each of the three author coreference datasets. We also show both the number of constraints and number of constraint units generated by the constraint generator before achieving the latent ground truth clustering. As can be seen in the table, existential cluster constraints required nearly half as many constraint units as must-link and cannot-link constraints to achieve the ground truth clustering. Figures 1&2 show the trajectories of both Adjusted Rand Index and FMC, respectively, with respect to the number of CUs as constraints were generated. As can be seen, we find that the existential clustering constraints provide for efficiency improvements in terms of the number of constraint units compared to the must-link and cannot-link constraints. These trends seem to hold across each of the metrics considered.

## 6. Related Work

Interactive clustering spans a wide range of work that aims to involve humans in the data clustering process and can be broadly characterized by (1) the goal of interactive clustering, (2) the method of interaction used, and (3) the mechanism to use feedback from users.

In this paper, the goal of interactive clustering, like majority of existing work, is to improve the quality of clustering using feedback from users. Other goals of interaction could be to help users understand the predicted clusters, and find outliers during exploratory data analysis (Amant & Cohen, 1998; Kwon et al., 2017), or to extract a flat clustering from hierarchical clustering (Vitale et al., 2019). However, the latter does not collect or use feedback from the users to obtain an improved clustering of the data.

The second important characteristic of interactive clustering methods is the mode of interaction. Must-link and cannot-link constraints is one of the most widely used modes of interaction where the user provide input on whether a pair of points should or should not be in the same clustering (Wagstaff & Cardie, 2000; Geerts & Ndindi, 2014). Other modes of interaction include feedback on cluster sizes (Xu et al., 2009), triplet constraints specifying relative distances orderings amongst datapoints (Chatziafratis et al., 2018), and explicitly merging and splitting clusters (Huang et al., 2000; Looney, 2002; Basu et al., 2010) or moving datapoints from one cluster to another (Coden et al., 2017). In this paper, we introduce a novel type of feedback that

enables users to provide feedback by making statements about existence of clusters with or without specific (meta-) features.

Finally, the feedback is incorporated in the clustering process by modifying the distance measure using the user feedback (Xing et al., 2002; Kulis et al., 2009; Lu & Carreira-Perpinan, 2008; Mitchell, 2016), or by explicitly using the feedback to merge and split clusters or move datapoints (Huang et al., 2000; Looney, 2002; Basu et al., 2010; Coden et al., 2017). In our work, we use the feedback as additional constraints during the clustering process.

## 7. Conclusion

In this work, we introduced a new feedback paradigm for interactive clustering in which users provide existential constraints on the features of clusters. We advocated for existential cluster constraints as a natural form of feedback that does not require users to inspect many individual data points, but rather use their mental model of what the output clustering should look like to define clustering constraints. We introduced an inference algorithm for incorporating existential cluster constraints into the correlation clustering framework. We demonstrated that our proposed feedback method provides efficiency improvements as compared to must-link and cannot-link constraints.

One could consider several possible directions for future work. In real-world settings, data is arriving incrementally all the time, so we need efficient methods for incorporating new data and existential cluster constraints as they arrive. Additionally, there will always be the possibility that the existential cluster constraints provided by a user are false (maliciously or not), so we need methods to balance this possibility with trying to incorporate that constraint to influence the output clustering. Lastly, future work could be done on novel interaction paradigms for other machine learning tasks and algorithms for incorporating them into the task.

## Acknowledgements

We thank Patrick Flaherty and Andrew McGregor as well as members of UMass IESL for helpful discussion and feedback. This work is funded in part by the Center for Data Science and the Center for Intelligent Information Retrieval, and in part by the National Science Foundation under Grants No. 1763618, and in part by the Chan Zuckerberg Initiative under the project Scientific Knowledge Base Construction, and in part by IBM Research AI through the AI Horizons Network. The work reported here was performed in part by the Center for Data Science and the Center for Intelligent Information Retrieval, and in part using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative. Rico Angell is supported by the National Science Foundation Graduate Research Fellowship under Grant No. 1938059. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor(s).

## References

Amant, R. S. and Cohen, P. R. Interaction with a mixed-initiative system for exploratory data analysis. *Knowledge-based systems*, 10(5):265–273, 1998.

Balcan, M.-F. and Blum, A. Clustering with Interactive Feedback. In *International Conference on Algorithmic Learning Theory*, 2008.

Bansal, N., Blum, A., and Chawla, S. Correlation Clustering. *Machine Learning*, 2004.

Basu, S., Fisher, D., Drucker, S., and Lu, H. Assisting users with clustering tasks by combining metric learning and classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.

Bibi, A., Wu, B., and Ghanem, B. Constrained k-means with general pairwise and cardinality constraints, 2019.

Chang, M.-W., Ratinov, L., and Roth, D. Guiding semi-supervision with constraint-driven learning. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, 2007.

Chatziafratis, V., Niazadeh, R., and Charikar, M. Hierarchical clustering with structural constraints. In *International conference on machine learning*, 2018.

Christiano, P., Leike, J., Brown, T. B., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *arXiv preprint arXiv:1706.03741*, 2017.

Coden, A., Danilevsky, M., Gruhl, D., Kato, L., and Nagarajan, M. A method to accelerate human in the loop clustering. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 237–245. SIAM, 2017.

Dasgupta, S., Dey, A., Roberts, N., and Sabato, S. Learning from discriminative feature feedback. In *NeurIPS*, pp. 3959–3967, 2018.

Davidson, I. and Basu, S. Clustering with constraints., 2009.

Demaine, E. D., Emanuel, D., Fiat, A., and Immorlica, N. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 2006.

Druck, G., Settles, B., and McCallum, A. Active learning by labeling features. In *Proceedings of the 2009 conference on Empirical methods in natural language processing*, 2009.

Frome, A., Singer, Y., Sha, F., and Malik, J. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *2007 IEEE 11th International Conference on Computer Vision*, 2007.

Ganchev, K., Graça, J., Gillenwater, J., and Taskar, B. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 2010.

Geerts, F. and Ndindi, R. Interactive correlation clustering. In *2014 International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 170–176. IEEE, 2014.

Greenberg, C., Monath, N., Kobren, A., Flaherty, P., McGregor, A., and McCallum, A. Compact Representation of Uncertainty in Clustering. In *Advances in Neural Information Processing Systems*, 2018.

Griffith, S., Subramanian, K., Scholz, J., Isbell, C. L., and Thomaz, A. L. Policy shaping: Integrating human feedback with reinforcement learning. Georgia Institute of Technology, 2013.

Huang, Z., Ng, M. K., Lin, T., and Cheung, D. An interactive approach to building classiffication models by clustering and cluster validation. In *International Conference on Intelligent Data Engineering and Automated Learning*, pp. 23–28. Springer, 2000.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems*, 2017.

Klein, D., Kamvar, S. D., and Manning, C. D. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. Technical report, 2002.

Kobren, A., Monath, N., and McCallum, A. Integrating user feedback under identity uncertainty in knowledge base construction. In *Automated Knowledge Base Construction (AKBC)*, 2019.

Kratzwald, B., Kunpeng, G., Feuerriegel, S., and Diefenbach, D. IntKB: A verifiable interactive framework for knowledge base completion. In *Proceedings of the 28th International Conference on Computational Linguistics*, 2020.

Kulis, B., Basu, S., Dhillon, I., and Mooney, R. Semi-supervised graph clustering: a kernel approach. *Machine learning*, 2009.

Kwon, B. C., Eysenbach, B., Verma, J., Ng, K., De Filippi, C., Stewart, W. F., and Perer, A. Clustervision: Visual supervision of unsupervised clustering. *IEEE transactions on visualization and computer graphics*, 24(1):142–151, 2017.

Li, Z., Liu, J., and Tang, X. Pairwise constraint propagation by semidefinite programming for semi-supervised classification. In *Proceedings of the 25th international conference on Machine learning*, 2008.

Li, Z., Liu, J., and Tang, X. Constrained clustering via spectral regularization. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Liang, P., Jordan, M. I., and Klein, D. Learning from measurements in exponential families. In *Proceedings of the 26th annual international conference on machine learning*, 2009.

Looney, C. G. Interactive clustering and merging with a new fuzzy expected value. *Pattern Recognition*, 35(11): 2413–2423, 2002.

Lu, Z. and Carreira-Perpinan, M. A. Constrained spectral clustering through affinity propagation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. IEEE, 2008.

Macaluso, S., Greenberg, C., Monath, N., Ah Lee, J., Flaherty, P., Cranmer, K., McGregor, A., and McCallum, A. Cluster Trellis: Data Structures & Algorithms for Exact Inference in Hierarchical Clustering. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, 2021.

Mazumdar, A. and Saha, B. Clustering with noisy queries, 2017.

Mitchell, L. A. *INCREMENT-Interactive Cluster Refinement*. Brigham Young University, 2016.

Nie, F., Zhang, H., Wang, R., and Li, X. Semi-supervised clustering via pairwise constrained optimal graph. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021.

O'Donoghue, B., Chu, E., Parikh, N., and Boyd, S. SCS: Splitting Conic Solver, version 3.1.1. https://github.com/cvxgrp/scs, November 2021.

Perret, B., Chierchia, G., Cousty, J., aes, S. F. G., Kenmochi, Y., and Najman, L. Higra: Hierarchical graph analysis. *SoftwareX*, 10:1–6, 2019. ISSN 2352-7110. doi: 10.1016/j.softx.2019.100335.

Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., and Ré, C. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*. NIH Public Access, 2017.

Settles, B. Active learning literature survey. 2009.

Shental, N., Bar-Hillel, A., Hertz, T., and Weinshall, D. Computing gaussian mixture models with em using equivalence constraints. *Advances in neural information processing systems*, 2004.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D. M., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. Learning to summarize from human feedback. *Advances in Neural Information Processing Systems*, 2020.

Subramanian, S., King, D., Downey, D., and Feldman, S. S2AND: A Benchmark and Evaluation System for Author Name Disambiguation. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2021*, 2021.

Swamy, C. Correlation clustering: maximizing agreements via semidefinite programming. In *SODA*, 2004.

Vikram, S. and Dasgupta, S. Interactive bayesian hierarchical clustering. In *International Conference on Machine Learning*, 2016.

Vitale, F., Rajagopalan, A., and Gentile, C. *Flattening a Hierarchical Clustering through Active Learning*. 2019.

Wagstaff, K. and Cardie, C. Clustering with instance-level constraints. *AAAI/IAAI*, 2000.

Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al. Constrained k-means clustering with background knowledge. In *Icml*, 2001.

Xing, E., Jordan, M., Russell, S. J., and Ng, A. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 2002.

Xu, L., Li, W., and Schuurmans, D. Fast normalized cut with linear constraints. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Yadav, N., Kobren, A., Monath, N., and Mccallum, A. Supervised Hierarchical Clustering with Exponential Linkage. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

# A. Proofs of Theoretical Statements

## A.1. Proof of Proposition 3.1

Consider a complete graph with $k$ ground truth clusters each with $\ell$ nodes where $v_{ij}$ denotes the $j$th node in the $i$th cluster. Let the total number of features be $k+\ell$ and the features for each node $v_{ij}$ be $\Phi(v_{ij}) = \{\phi_1, \ldots, \phi_{i-1}, \phi_{i+1}, \ldots, \phi_k, \phi_{k+j}\}$. Let the edge labels and weights be as follows:

$$\begin{cases} k^2\ell^2 & (v_{ij}, v_{i'j}) \\ 1 & (v_{ij}, v_{ij'}) \\ -(1+\varepsilon) & \text{o.w.} \end{cases},$$

where $\varepsilon > 0$. The optimal clustering with respect to the correlation clustering objective is $\ell$ clusters, each of the form $\{v_{1,j}, v_{2,j}, \ldots, v_{k,j}\}$. The cost of this clustering is $k\ell^2 - k\ell$, the sum of all of the ground truth within cluster edges, all of which are cut in the optimal predicted clustering. With clear reasoning, we can see why this is the optimal clustering. We never want to cut edges of the form $(v_{ij}, v_{i'j})$, since cutting one of these edges will incur a cost of $k^2\ell^2$, more than the sum of the weights of all other edges. Furthermore, we never want to add any edges of the form $(v_{ij}, v_{ij'})$ to the predicted clustering since doing so will incur a cost of at least $2\varepsilon$ for every one of these edges that we include.

We will now show that this construction requires at least $k^2 + k\ell - 2k$ pairwise node constraints to achieve the ground truth clustering. To accomplish this, we will show that to achieve the ground truth clustering using pairwise node constraints for each pair of ground truth clusters either we need to put cannot-link constraints on all the edges of the form $(v_{ij}, v_{i'j})$ or the pairwise node constraints need to fully specify the ground truth partitioning of that pair of ground truth clusters (i.e. there is exactly one partitioning which satisfies all of the pairwise node constraints and it is the ground truth partitioning).

For any arbitrary pair of ground truth clusters, assume the pairwise node constraints do not fully specify the ground truth partitioning and the optimal partitioning given the pairwise node constraints is the ground truth partitioning. For the sake of contradiction, suppose that the set of pairwise node constraints does not include cannot-link constraints for each edge of the form $(v_{ij}, v_{i'j})$. This means that there is a partitioning which satisfies all of the constraints which includes an edge of the form $(v_{ij}, v_{i'j})$. But, the partitioning which includes the edge of the form $(v_{ij}, v_{i'j})$ actually has a lower cost than the ground truth partitioning which does not contain that edge. Hence, the optimal partitioning is not the ground truth partitioning and we have a contradiction.

There are exactly $k^2\ell - k\ell$ edges of the form $(v_{ij}, v_{i'j})$. One set of pairwise node constraints which will clearly result in the ground truth clustering is placing a cannot-link constraint on all edges of the form $(v_{ij}, v_{i'j})$. But, we can actually achieve the ground truth clustering with fewer pairwise node constraints. We can place $\ell - 1$ must-link constraints in each cluster to force the ground truth clusters to be connected and place $k^2 - k$ across ground truth cluster cannot-link constraints. This is the smallest set of pairwise node constraints which fully separates the ground truth clusters. This results in a total of $k^2 + k\ell - 2k$ pairwise node constraints which for $\ell > 1$ is less than $k^2\ell - k\ell$.

Lastly, we will show that there is a set of $\exists$-constraints with a total of $k\ell + k - \ell - 1$ features which ensures the ground truth clustering will be returned by the correlation clustering objective. For each of $k-1$ of the ground truth clusters, create a $\exists$-constraint for cluster $i$ with the form $\{(-, \phi_i), (+, \phi_{k+1}), (+, \phi_{k+2}), \ldots, (+, \phi_{k+\ell})\}$. In order to satisfy this $\exists$-constraint, every node from the $i$th ground truth cluster will be required to be contained in a predicted cluster and none of the nodes from any other ground truth cluster will be contained in the same predicted cluster. The cluster we do not explicitly create a $\exists$-constraint for will be predicted since it is separated from all of the other clusters by the set of $\exists$-constraints. Observe that there are $\ell + 1$ features in each constraint, so the total number of features over all of the $\exists$-constraints is exactly $k\ell + k - \ell - 1$. The result follows.

## A.2. Proof of Proposition 3.2

We will show a polynomial time reduction from SAT-CNF. Suppose we have an algorithm that can determine if there exists a clustering of the vertices of a graph which satisfies all of some set of $\exists$-constraints. Additionally, suppose we have an arbitrary SAT-CNF formula $f$. For every disjunctive clause $c$ in $f$, create a feature $\phi_c$ and add it to the set of possible features. For every boolean variable $x$ in $f$, create two vertices in the graph, namely $v_x$ and $v_{\neg x}$. In addition, create a unique feature for each vertex in the graph (e.g. $\phi_x$ for $v_x$ and $\phi_{\neg x}$ for $v_{\neg x}$). For every clause $c$ in $f$ that contains $x$ (and similarly $\neg x$), add the corresponding clause feature $\phi_c$ to $\Phi(v_x)$ (and similarly $\Phi(v_{\neg x})$). Create the *all-clause* $\exists$-constraint $\{(+, \phi_c) : c \in f\}$;

this constraint is satisfied if and only if each disjunctive clause evaluates to `True`. Lastly, create the set of *valid-assignment* $\exists$-constraints $\{(+, \phi_x), (-, \phi_{\neg x})\}$ for every boolean variable $x$.

Clearly this is a polynomial time reduction. If there exists a clustering of the vertices which satisfies all of the $\exists$-constraints, then the set of vertices in the cluster which satisfies the *all-clause* $\exists$-constraint directly corresponds to an assignment to the boolean variables that will be a satisfying assignment to $f$. For every vertex $v$ in the *all-clause* cluster, if $v$ corresponds to a variable $x$, then the satisfying assignment includes setting $x = \texttt{True}$. Otherwise, $v$ corresponds to the negation of a variable $\neg x$, and the satisfying assignment includes setting $x = \texttt{False}$. The *valid-assignment* $\exists$-constraints ensure that the assignment will be valid since one of $v_x$ or $v_{\neg x}$ can be in the cluster satisfying the *all-clause* $\exists$-constraint , but not both. If there does not exist a clustering of the vertices which will satisfy both the *all-clause* $\exists$-constraint and *valid-assignment* $\exists$-constraints, then there exists at least one feature $\phi_c$ that cannot be satisfied in the *all-clause* cluster without violating the *valid-assignment* $\exists$-constraints. That means that clause $c$ cannot be satisfied in the formula $f$. Hence, there does not exist a valid assignment to the boolean variables for which $f$ is satisfied. The result follows.

### A.3. Proof of Proposition 4.1

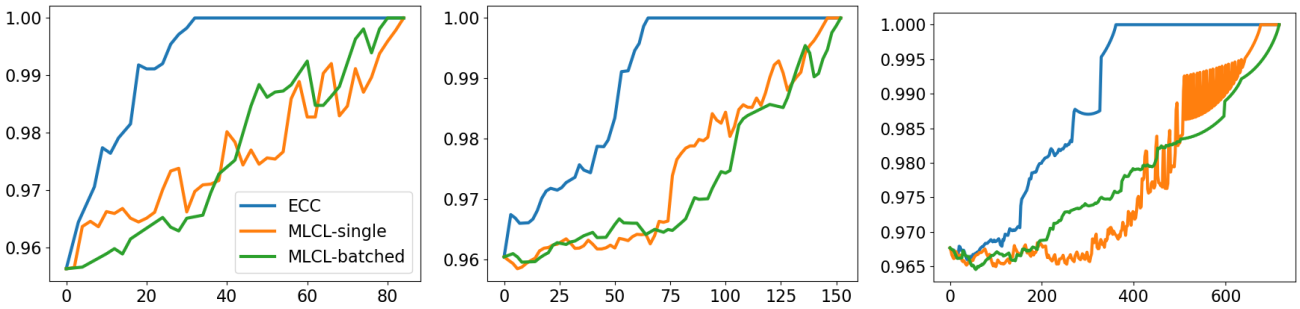
We proceed via induction over the trellis vertices in topological order from leaves to root. The base case is the children $S_\ell$ and $S_r$ are leaves and the parent $S_p$ represents the set of two vertices each contained as singletons in $S_\ell$ and $S_r$. There can be no $\exists$-constraints satisfied by the partial clustering $\{S_\ell, S_r\}$, so if any $\exists$-constraints are satisfied in $S_p$ (the case where one of $S_\ell$ or $S_r$ represents the set of a $\exists$-constraint and the other satisfies that $\exists$-constraint ), then $S_p$ will be chosen as the memoized clustering stored in $a[S_p]$. If no $\exists$-constraints are satisfied in $S_p$, then we just need to determine which partial clustering has the higher objective value, either $\{S_\ell, S_r\}$ or $\{S_p\}$. The algorithm will choose the partial clustering which has the higher objective values. The base case follows.

The inductive step is next. Without loss of generality, suppose $S_p$ only has pair of children $(S_\ell, S_r)$. By assumption, $a[S_\ell]$ and $a[S_r]$ memoize the best (according to our objective stated in the proposition) partial clusterings of $S_\ell$ and $S_r$, respectively. Algorithm 1 will choose either $\{S_p\}$ or $a[S_\ell] \cup a[S_r]$ as the partial clustering to memoize in $a[S_p]$. Suppose $\{S_p\}$ does not have more satisfied $\exists$-constraints or a higher objective value than $s[S_\ell] + s[S_r]$ and $o[S_\ell] + o[S_r]$, respectively. Algorithm 1 will choose $a[S_\ell] \cup a[S_r]$ as the partial clustering to memoize in $a[S_p]$. Out of all of the trellis-consistent partial clusterings of $S_p$ where no element $x_\ell \in S_\ell$ is contained in the same cluster as any element $x_r \in S_r$, the partial clustering $a[S_\ell] \cup a[S_r]$ will be the best, since the objective value and the number of $\exists$-constraints satisfied in a partial clustering decompose given this required bifurcation of the set of elements. Thus, the inductive hypothesis has been shown and the result follows.

## B. Implementation Details

We use SCS (O'Donoghue et al., 2021), a modern convex optimization solver, to solve the SDP relaxation shown in Objective (4). For must-link and cannot link constraints, we use the total sum of the edge weights on the graph to act as a proxy for $\infty$, since SCS cannot handle infinite edge weights. We use the default hyperparameters and set the maximum number of iterators to 50k. Given the output of the SDP solver, we use Higra (Perret et al., 2019) to construct the trellis by constructing greedy binary hierarchical clusterings of the vertices and the $\exists$-constraints using five different linkage functions: average, complete, single, exp($\alpha = -1$), and exp($\alpha = 1$) (Yadav et al., 2019). We substitute a large negative number into all entries in $X^*$ that represent two incompatible elements to safely avoid them from being merged by the hierarchical clustering algorithm. We construct the trellis by taking taking the union of all partial clusterings represented in the five binary hierarchical clustering trees.

## C. Additional Experimental Results




(a) PubMed (b) QIAN (c) SCAD-zbMATH

Figure 3: **CU vs. F1.** In each plot, the x-axis is the number of constraint units (CU) used by each feedback constraint method and the y-axis is the corresponding cluster F1 of the resulting clustering after incorporating the generated constraints. The blue line is existential cluster constraints (ECC), the orange line is must-link and cannot-link constraints generated one at a time (MLCL-single), and the green line is must-link and cannot link constraints generated in batches (MLCL-batched) to mimic ECC.