Journal of Machine Learning Research 18 (2017) 1-35 Submitted 3/15; Revised 8/16; Published 1/17

Local algorithms for interactive clustering

Pranjal Awasthi PRANJAL.AWASTHI@RUTGERS.EDU
Department of Computer Science
Rutgers University

Maria Florina Balcan NINAMF@CS.CMU.EDU
School of Computer Science
Carnegie Mellon University

Konstantin Voevodski KVODSKI@GOOGLE.COM
Google, NY, USA

Editor: Le Song

Abstract

We study the design of interactive clustering algorithms. The user supervision that we consider is in the form of cluster split/merge requests; such feedback is easy for users to provide because it only requires a high-level understanding of the clusters. Our algorithms start with any initial clustering and only make local changes in each step; both are desirable properties in many applications. Local changes are desirable because in practice edits of other parts of the clustering are considered churn - changes that are perceived as quality-neutral or quality-negative. We show that in this framework we can still design provably correct algorithms given that our data satisfies natural separability properties. We also show that our framework works well in practice.

1. Introduction

Clustering is usually studied in an unsupervised learning scenario where the goal is to partition the data given pairwise similarity information. Designing provably-good clustering algorithms is challenging because given a similarity function there may be many possible clusterings of the data. Traditional approaches resolve this ambiguity by making assumptions on the data-generation process. For example, there is a large body of work on clustering data that is generated by a mixture of Gaussians (Achlioptas and McSherry, 2005; Kannan et al., 2005; Dasgupta, 1999; Arora and Kannan, 2001; Brubaker and Vempala, 2008; Kalai et al., 2010; Moitra and Valiant, 2010; Belkin and Sinha, 2010), and finding clusters with certain density (Rinaldo and Wasserman, 2010; Chaudhuri and Dasgupta, 2010). But instead of making such assumptions and trying to set the corresponding hyperparameters, we can use limited user (expert) supervision to help the algorithm reach the correct answer.

Interactive clustering algorithms have been facilitated by the availability of cheap crowd-sourcing tools in recent years, which enable collection of relevant user input. In certain applications such as search and document classification, where users are willing to help a clustering algorithm arrive at their own desired answer with a small amount of additional feedback, interactive clustering algorithms are very useful. The works of Balcan and Blum (2008) and Awasthi and Zadeh (2010) provide some initial theoretical results in this new and exciting research area, but their models are not very practical.

We observe that in many practical settings we already start with a fairly good clustering computed with semi-automated techniques. For example, consider a massive online news portal that maintains a large collection of news articles. Suppose that the news articles are clustered on the "back-end," and are used to serve several "front-end" applications such as recommendations and article profiles. For such a system, we do not have the freedom to compute arbitrary clusterings and present them to the user, which has been suggested in prior interactive clustering work (Balcan and Blum, 2008; Awasthi and Zadeh, 2010). But it is still feasible to get specific feedback on the current proposed clustering and *locally* edit this clustering. In particular, we may only want to change the "bad" part of the clustering that is revealed by the feedback without changing the rest of the clustering. Our intuition for only considering *local* edits is that in practice changes to other parts of the clustering are considered *churn* - changes that are perceived as quality-neutral or quality-negative. This observation is especially true for large clustering systems that serve many users, which operate on the assumption that if some of the data is broken, it will be pointed out by a user. For such applications it is undesirable to change what the user is not complaining about. Motivated by these observations, in this paper we study the problem of designing *local* algorithms for interactive clustering.

We propose a theoretical interactive clustering model and provide strong experimental evidence supporting its utility in practice. In our model we start with some initial clustering of the data. The algorithm then interacts with the user in stages. In each stage the user provides limited feedback on the proposed clustering in the form of *split* and *merge* requests. The algorithm then makes a *local* edit to the clustering that is consistent with user feedback. Such edits are aimed at improving the problematic part of the clustering pointed out by the user. The goal of the algorithm is to quickly converge (using as few requests as possible) to a clustering that the user is happy with - we call this clustering the target (ground truth) clustering.

In our model the user may request a certain cluster to be *split* if it is *overclustered* (intersects two or more clusters in the target clustering). The user may also request to *merge* two given clusters if they are *underclustered* (both intersect the same target cluster). Note that the user does not tell the algorithm how to perform the split or the merge; such input is unrealistic because it requires a manual analysis of the data points in the corresponding clusters. We also restrict the algorithm to only make *local* changes at each step - in response we may change only the cluster assignments of the points in the corresponding clusters. If the user requests to split a cluster C_i, we may change only the cluster assignments of the points in C_i, and if the user requests to merge C_i and C_j, we may only reassign the points in C_i and C_j.

The split and merge requests in our model are a natural form of feedback. It is easy for users to spot over/underclustering errors and request the corresponding splits/merges (without having to provide any additional information about how to perform the edit). For our model to be practically applicable, we also need to account for noise in the user requests. In particular, if the user requests a merge, only a fraction or a constant number of the points in the two clusters may belong to the same target cluster. Our model (see Section 2) allows for such noisy user responses.

We study the complexity of algorithms in this framework (the number of edits requests needed to find the target clustering) as a function of the error of the initial clustering. We define clustering error in terms of *underclustering* error δ_u and *overclustering* error δ_o (see Section 2). Given that the initial error is often fairly small [1], we would like to develop algorithms whose complexity depends

1. Given 2 different k-clusterings, δ_u and δ_o is at most k^2.

polynomially on δ_u, δ_o and only logarithmically on n, the number of data points. We show that this is indeed possible given that the target clustering satisfies a natural *stability* property (see Section 2). In addition, we develop provably correct algorithms for the well-known correlation-clustering objective (Bansal et al., 2004), which considers pairs of points that are clustered inconsistently with respect to the target clustering (see Section 2).

1.1 Our Results

Our local interactive clustering model is summarized in Section 2.4. This model is then instantiated with specific split/merge algorithms in Section 3 and Section 4. In Section 3 we study the η-merge model. Here we assume that the user may request to split a cluster C_i only if C_i contains points from several ground-truth clusters. The user may request to merge C_i and C_j only if an η-fraction of the points in each C_i and C_j are from the same ground-truth cluster. Note that these restrictions are on the user requests, and not the clustering we are editing. In particular, there may be pairs of clusters C_i and C_j where a smaller fraction of the points are from the same target cluster. But we assume that the user will not request to merge C_i and C_j in such cases because there is not enough evidence to request this merge. Instead, we assume that the user will ask for a split of C_i and/or C_j first, or ask for another merge involving C_i or C_j (where there is more evidence that they need to be merged). This is a realistic restriction because we assume that the merge requests come from high-level observations about the clusters, for example from observing the cluster profiles/summaries.

For this model for $\eta > 0.5$, given an initial clustering with overclustering error δ_o and underclustering error δ_u, we present interactive clustering algorithms that require δ_o split requests and $2(\delta_u + k)\log_{\frac{1}{1-\eta}} n$ merge requests to find the target clustering, where n is the number of points in the data set. For $\eta > 2/3$, given an initial clustering with correlation-clustering error δ_{cc}, we present algorithms that require at most δ_{cc} edit requests to find the target clustering.

In Section 4 we relax the condition on the merges and allow the user to request a merge even if C_i and C_j only have a single point from the same target cluster. We call this the *unrestricted-merge* model. Here the requirement on the accuracy of the user response is much weaker and we need to make further assumptions about the nature of the requests. In particular, we assume that each merge request is chosen uniformly at random from the set of possible merge requests. Under this assumption we present algorithms that with probability at least $1 - \epsilon$ require δ_o split requests and $O(\log \frac{k}{\epsilon} \delta_u^2)$ merge requests to find the target clustering.

Our interactive clustering algorithms take either global or local pairwise similarity data as input. Our most general algorithms use the global average-linkage tree T_{glob} to compute local clustering edits. This tree is constructed from all the data points in the clustering, but it is too large to be directly pruned by users. Still, we can use this global tree to compute accurate local clustering edits. Our split algorithm finds the node in T_{glob} where the corresponding points are first split in two. It is more challenging to design a correct merge procedure, given that we allow "impure" merges, where the clusters in the merge request intersect more than one ground-truth cluster. To perform such merges, in the η-merge model we design an algorithm to extract the "pure" subsets of the two clusters, which must only contain points from the same target cluster. Our algorithm searches for the deepest node in T_{glob} that has enough points from both clusters. In the unrestricted-merge model, we develop another merge algorithm that either merges the two clusters or merges them and splits them. This algorithm always makes progress if the requested merge is "impure," and makes progress on average if it is "pure" (both clusters are subset of the same target cluster).

When the data satisfies stronger assumptions, we present more efficient split and merge algorithms that do not require global pairwise similarity information. These procedures compute the edit by only considering the similarities between the points in the user request.

In Section 5 we demonstrate the effectiveness of our algorithms on real data. We show that for the purposes of splitting known overclustering instances, our split algorithm performs better than well-known algorithms in unsupervised split/merge clustering literature, such as spectral clustering and k-means. We also test our entire interactive clustering framework on the 20 Newsgroup data set, which is known to very challenging for unsupervised (Telgarsky and Dasgupta, 2012; Heller and Ghahramani, 2005; Dasgupta and Hsu, 2008; Dai et al., 2010; Boulis and Ostendorf, 2004; Zhong, 2005) and semi-supervised clustering methods (Basu et al., 2002, 2004). We find that in many scenarios our framework is able find the target Newsgroup clustering after a limited number of edit requests.

1.2 Related work

In this section we give an overview of the related work in the literature.

Interactive Clustering: Interactive clustering models in previous works (Balcan and Blum, 2008; Awasthi and Zadeh, 2010) were inspired by an analogous model for learning under feedback (Angluin, 1998). In this model, the algorithm can propose a hypothesis to the user (in this case, a clustering of the data) and get some feedback regarding the correctness of the current hypothesis. As in our model, the feedback in Balcan and Blum (2008); Awasthi and Zadeh (2010) is in the form of split and merge requests. The goal is to design efficient algorithms that require few user requests. A major limitation in the models of Balcan and Blum (2008) and Awasthi and Zadeh (2010) is that the algorithm is able to choose any arbitrary clustering as the starting point, and can make arbitrary changes to the clustering in each step. Hence these algorithms may propose a series of "bad" clusterings to the user to quickly prune the search space and reach the target clustering. Our interactive clustering model is in the context of an initial clustering; we are restricted to only making local changes to this clustering to correct the errors pointed out by the user. This model is well-motivated by several applications, including the Google application described in Section 5.1.

Active Clustering: Other active clustering frameworks have been proposed, where *active* either refers to selecting which pairwise similarities to consider (not studied here), or requesting user supervision with respect to the ground-truth clustering. For the former problem, Erikkson et al. (2011) study minimizing the number of pairwise similarities needed for an algorithm to compute an accurate clustering. They propose an adaptive algorithm that selects which pairwise similarities to consider. The assumption on the similarity function that they study, which they call *tight clustering condition*, is equivalent to the *strict separation* property studied here. Krishnamurthy et al. (2012) propose a different framework to adaptively select which pairwise similarities to consider. They also study an assumption on the similarity function, which is a generalization of the *strict separation* property that considers the expected pairwise similarities. Their framework recursively splits clusters; they propose a spectral-clustering algorithm to perform the split. They also suggest using the k-means algorithm to perform the split (albeit with no provable guarantees). In our experimental section we compare our proposed split procedures with a similar spectral clustering algorithm and the k-means algorithm (see Section 5.1).

For the latter semi-supervised clustering problem, Nie et al. (2012) develop a clustering algorithm that iteratively extends class labels. It may be used with an empty set of initial labels (unsupervised setting), or with a non-empty set of initial labels (semi-supervised setting). The algorithm minimizes an estimate of Bayes error with respect to the unlabeled instances.

Basu et al. (2002) study the same semi-supervised clustering problem in the context of the k-means objective. Their algorithm has access to the class labels of some seed fraction f of the points in the data set. They propose two variations of a supervised k-means algorithm: one which only uses the known labels to compute the initial clustering (termed *Seeded K-Means*), and another that also enforces that the known labels do not change while the centers/clusters are updated (termed *Constrained K-Means*). They perform experiments on the same 20 Newsgroups data sets that we consider in Section 5.2. The experiments of Basu et al. (2002) show that even though the supervision improves accuracy, even for $f > 0.5$ the output clustering still has many mistakes. They also show that the accuracy depends on how difficult the data set is w.r.t. the separability of the ground-truth clusters, and show improved accuracy on an easier instance (termed *Different-3 Newsgroups*) when compared to a harder one (termed *Same-3 Newsgroups*). In a related work, Basu et al. (2004) study the same problem but with supervision in the form of pairwise *must-link* and *cannot-link* constraints. They propose another supervised k-means algorithm and perform experiments on the same 20 Newsgroups data sets. They are still unable to come close to recovering the ground-truth for a difficult Newsgroups data set (termed *News-sim3*), but have more success with an easier Newsgroup data set (termed *News-diff3*). Ashtiani et al. (2016) also study the k-means objective with supervision in the form of pairwise *must-link* and *cannot-link* constraints, which are modeled as oracle queries. For center-based clustering instances (specified by a Voroni decomposition around a set of centers) that satisfy a separability condition with respect to the cluster centers, they propose an accurate algorithm that requires a limited number of queries; they do not provide any experimental results.

In our experimental section we give a comparison of our results on the 20 Newsgroups data sets with those of Basu et al. (2002) and Basu et al. (2004) (see Section 5.2.4). Our comparison considers the amount of required supervision and accuracy of the final clustering output. We show that only our framework is able to fully recover the ground-truth clustering, albeit we can do this consistently only with some restrictions on the requested merges. On the other hand, Basu et al. (2002) and Basu et al. (2004) do not come close to finding the ground-truth for the harder Newsgroups data sets even when they use a large amount of supervision.

We also note that the supervision considered by Nie et al. (2012), Basu et al. (2002), Basu et al. (2004) and Ashtiani et al. (2016) is harder for the user to provide. Such supervision requires an understanding of the individual data points on the part of the user. The user must study the individual data points to provide instance class labels as in Nie et al. (2012) and Basu et al. (2002), or study pairwise relationships between individual data points to provide must-link and cannot-link constraints as in Basu et al. (2004) and Ashtiani et al. (2016). The supervision in the form of cluster split/merge requests that we consider here is a more realistic form of interaction - it only requires the user to understand the high-level properties of the clusters.

Split/Merge Techniques: Several unsupervised split/merge frameworks have also been proposed (Ding and He, 2002; Lee et al., 2012; Chaudhuri et al., 1992). They focus on designing split/merge algorithms but do not consider any user feedback. Ding and He (2002) propose to use a spectral clustering algorithm to perform the splits. To perform merges, they only consider returning the union of the points in the two clusters. In our experiments, we compare the effectiveness of our split procedures with splits given by spectral clustering (see Section 5.1). With respect to merges, our merge algorithms for $\eta = 1$ are equivalent to the merge proposed by Ding and He (2002), but in our framework merges may only be initiated by user requests, while Ding and He (2002) propose automatically selecting the next clusters to merge using the max-min-cut criterion (without any user supervision), which they show works well in practice. However, the experiments of Ding and He (2002) also assume that we know the target number of clusters, while our model does not make such assumptions.

Lee et al. (2012) propose a different unsupervised split/merge framework based on optimizing Bayes error. They propose spectral clustering and k-means for computing the splits, which we compare with in our experimental section (see Section 5.1). To perform merges, they again only consider returning the union of the two clusters; the next two clusters to merge are selected by optimizing Bayes error. Unlike Ding and He (2002), they do not assume that we know the number of target clusters. They show good experimental results for video segmentation, but their unsupervised approach may not work for challenging data sets like 20 Newsgroups, where it may not be possible to find the ground truth without user feedback. Chaudhuri et al. (1992) propose another unsupervised split/merge framework for image-segmentation applications. Their split procedure is application-specific because it considers gray-scale values (we cannot compare with it), while their merge is based on edge density.

Data Separability Properties: The data separability (stability) property that we consider in this work is a natural generalization of the "stable marriage" property (see Definition 2), which has been studied in a variety of previous works (Balcan et al., 2008; Bryant and Berry, 2001). It is the weakest among the stability properties that have been studied recently such as *strict separation* and *strict threshold separation* (Balcan et al., 2008; Erikkson et al., 2011; Ackerman et al., 2012; Ackerman and Dasgupta, 2014; Balcan et al., 2014). This property is also known to hold for real-world data. In particular, Voevodski et al. (2012) observed that this property holds for protein sequence data, where similarities are computed with sequence alignment and ground truth clusters correspond to evolutionary-related proteins. Other stronger separability properties have also been considered in the literature (Awasthi and Balcan, 2015).

2. Notation and Preliminaries

Given a data set X of n points we define $\mathcal{C} = \{C_1, C_2, \dots C_k\}$ to be a k-clustering of X where the C_i's represent the individual clusters. Given two clusterings $\mathcal{C}$ and $\mathcal{C}'$, we next define several notions of clustering error, which are used in our theoretical analysis.

2.1 Clustering Error

Given two clusterings $\mathcal{C}$ and $\mathcal{C}'$, we define the distance between a cluster $C_i \in \mathcal{C}$ and the clustering $\mathcal{C}'$ as:

$$\operatorname{dist}(C_i, \mathcal{C}') = |\{C'_j \in \mathcal{C}' : C'_j \cap C_i \neq \emptyset\}| - 1.$$

This distance is the number of *additional* clusters in $\mathcal{C}'$ that contain points from C_i; it evaluates to 0 when all points in C_i are contained in a single cluster in $\mathcal{C}'$. Naturally, we can then define the distance between $\mathcal{C}$ and $\mathcal{C}'$ as:

$$\operatorname{dist}(\mathcal{C}, \mathcal{C}') = \sum_{C_i \in \mathcal{C}} \operatorname{dist}(C_i, \mathcal{C}').$$

Notice that this notion of clustering distance is asymmetric: $\mathrm{dist}(\mathcal{C}, \mathcal{C}') \neq \mathrm{dist}(\mathcal{C}', \mathcal{C})$. Also note that $\mathrm{dist}(\mathcal{C}, \mathcal{C}') = 0$ if and only if $\mathcal{C}$ refines $\mathcal{C}'$. Observe that if $\mathcal{C}$ is the ground-truth clustering, and $\mathcal{C}'$ is a proposed clustering, then $\mathrm{dist}(\mathcal{C}, \mathcal{C}')$ can be considered an *underclustering error*, and $\mathrm{dist}(\mathcal{C}', \mathcal{C})$ an *overclustering error*.

An underclustering error is an instance of several clusters in a proposed clustering containing points from the same ground-truth cluster; this ground-truth cluster is said to be *underclustered*. Conversely, an overclustering error is an instance of points from several ground-truth clusters contained in the same cluster in a proposed clustering; this proposed cluster is said to be *overclustered*. In the following sections we use $\mathcal{C}^* = \{C_1^*, C_2^*, \ldots C_k^*\}$ to refer to the ground-truth clustering, and use $\mathcal{C}$ to refer to a proposed clustering. We use δ_u to refer to the underclustering error of a proposed clustering, and δ_o to refer to the overclustering error. In other words, we have $\delta_u = \mathrm{dist}(\mathcal{C}^*, \mathcal{C})$ and $\delta_o = \mathrm{dist}(\mathcal{C}, \mathcal{C}^*)$. We use δ to denote the sum of the two errors: $\delta = \delta_u + \delta_o$. We call δ the *under/overclustering error*, and use $\delta(\mathcal{C}, \mathcal{C}^*)$ to refer to the error of $\mathcal{C}$ with respect to $\mathcal{C}^*$.

We observe that the *under/overclustering error* $\delta(\mathcal{C}, \mathcal{C}^*)$ may also be restated in terms of the *bipartite cluster-intersection graph* of Xiang et al. (2012). This bipartite graph $G = (\mathcal{C}, \mathcal{C}^*, E)$ describes the relationships between the clusters of $\mathcal{C}$ and $\mathcal{C}^*$, with nodes on one side corresponding to the clusters of $\mathcal{C}$ and nodes on the other side corresponding to the clusters of $\mathcal{C}^*$. The set of (unweighted) edges E corresponds to the intersections between the two sets of clusters: there is an edge if the corresponding clusters intersect on at least one data point (Xiang et al., 2012). Then we can express the *under/overclustering error* $\delta(\mathcal{C}, \mathcal{C}^*)$ using vol(G), the sum of the degrees of the nodes of G:

$$\delta(\mathcal{C}, \mathcal{C}^*) = \mathrm{vol}(G) - |\mathcal{C}| - |\mathcal{C}^*|.$$

In addition to the *under/overclustering* error defined above, in parts of our analysis we define the distance between two clusterings using the *correlation-clustering* objective function. Given a proposed clustering $\mathcal{C}$, and a ground-truth clustering $\mathcal{C}^*$, we define the correlation-clustering error δ_{cc} as the number of (ordered) pairs of points that are clustered *inconsistently* with $\mathcal{C}^*$:

$$\delta_{cc} = |\{(u, v) \in X \times X : c(u, v) \neq c^*(u, v)\}|,$$

where $c(u, v) = 1$ if u and v are in the same cluster in $\mathcal{C}$, and 0 otherwise; $c^*(u, v) = 1$ if u and v are in the same cluster in $\mathcal{C}^*$, and 0 otherwise. In our analysis we also call each such pair of points a *pairwise correlation-clustering error*.

Note that as before we may divide the correlation-clustering error δ_{cc} into overclustering component δ_{cco} and underclustering component δ_{ccu}:

$$\delta_{cco} = |\{(u, v) \in X \times X : c(u, v) = 1 \text{ and } c^*(u, v) = 0\}|,$$
$$\delta_{ccu} = |\{(u, v) \in X \times X : c(u, v) = 0 \text{ and } c^*(u, v) = 1\}|.$$

Observe that by definition $\delta_{cc} = \delta_{cco} + \delta_{ccu}$.

2.2 Definitions

Our interactive clustering model concerns computing *local* clustering edits, which only change the part of the clustering that the user is complaining about. This intuition is captured by the following definition.

Definition 1 (Local algorithm) *We say that an interactive clustering algorithm is* local *if in each iteration only the cluster assignments of the points involved in the edit request may be changed. If the user requests to split C_i, the algorithm may only reassign the points in C_i. If the user requests to merge C_i and C_j, the algorithm may only reassign the points in $C_i \cup C_j$.*

We next define the properties of a clustering that we study in this work, which describe how separable the ground-truth clusters are.

Definition 2 (Stability) *Given a clustering $\mathcal{C} = \{C_1, C_2, \cdots C_k\}$ over a domain X and a similarly function $S : X \times X \mapsto \Re$, we say that $\mathcal{C}$ satisfies stability with respect to S if for all $i \neq j$, and for all $A \subset C_i$ and $A' \subseteq C_j$, $S(A, C_i \setminus A) > S(A, A')$, where for any two sets A, A', $S(A, A') = E_{x \in A, y \in A'} S(x, y)$.*

In addition to the *stability* property, we also study the stronger *strict separation* and *strict threshold separation* properties, which were first introduced in Balcan et al. (2008). They are defined below. Clearly, we can verify that *strict separation* and *strict threshold separation* imply *stability*.

Definition 3 (Strict separation) *Given a clustering $C = \{C_1, C_2, \cdots C_k\}$ over a domain X and a similarly function $S : X \times X \mapsto \Re$, we say that C satisfies strict separation with respect to S if for all $i \neq j$, $x, y \in C_i$ and $z \in C_j$, $S(x, y) > S(x, z)$.*

Definition 4 (Strict threshold separation) *Given a clustering $C = \{C_1, C_2, \cdots C_k\}$ over a domain X and a similarly function $S : X \times X \mapsto \Re$, we say that C satisfies strict threshold separation with respect to S if there exists a threshold t such that, for all i, $x, y \in C_i$, $S(x, y) > t$, and, for all $i \neq j$, $x \in C_i, y \in C_j$, $S(x, y) \leq t$.*

We model the user as an oracle that provides edit requests. In order for our algorithms to make progress, the oracle requests must be somewhat consistent with the target clustering, which is captured by the following definitions.

Definition 5 (η-merge model) *The oracle may request to split a cluster C_i only if C_i contains points from more than one target cluster. The oracle may request to merge two clusters C_i and C_j only if at least an η-fraction of the points in each C_i and C_j belong to the same target cluster.*

Definition 6 (unrestricted-merge model) *The oracle may request to split a cluster C_i only if C_i contains points from more than one target cluster. The oracle may request to merge two clusters C_i and C_j only if C_i and C_j both intersect the same target cluster.*

Note that the assumptions about the split requests are the same in both models. With respect to the merges, in the η-merge model the oracle may request to merge two clusters if both have a *constant fraction* of points from the same target cluster. In the unrestricted-merge model, the oracle may request to merge two clusters even if both only have *some* points from the same target cluster. In each model, we will refer to the split/merge requests that satisfy these definitions as *valid* edit requests.

Also note that in the η-merge model the restrictions on valid merge requests are with respect to the oracle requests, and not the clustering we are editing. In particular, there may be pairs of clusters C_i and C_j where a smaller fraction of the points are from the same target cluster. But we assume that the oracle will not request to merge C_i and C_j in such cases. Instead, we assume that the oracle will ask for a split of C_i and/or C_j first, or ask for a different (valid) merge involving C_i or C_j.

2.3 Generalized clustering error

We observe that the clustering errors defined in the previous section may be generalized by abstracting their common properties. We define the following properties of what we call *natural* clustering error, which is any integer-valued error that decreases when we locally improve the proposed clustering.

Definition 7 *We say that a clustering error is* natural *if it satisfies the following properties:*

- *If there exists a cluster C_i that contains points from C_j^* and some other ground-truth cluster(s), then splitting this cluster into two clusters $C_{i,1} = C_i \cap C_j^*$ and $C_{i,2} = C_i - C_{i,1}$ must decrease the error.*
- *If there exist two clusters that contain only points from the same target cluster, then merging them into one cluster must decrease the error.*
- *The error is integer-valued.*

We expect a lot of definitions of clustering error to satisfy the above criteria (especially the first two properties), in addition to other domain-specific criteria. Clearly, the under/overclustering error $\delta = \delta_u + \delta_o$ and the correlation-clustering error δ_{cc} are also *natural* clustering errors (Claim 8). Given a *natural* clustering error γ, a proposed clustering $\mathcal{C}$ and the target clustering $\mathcal{C}^*$, we will use $\gamma(\mathcal{C}, \mathcal{C}^*)$ to denote the magnitude of the error of $\mathcal{C}$ with respect to $\mathcal{C}^*$. We can also prove that the under/overclustering error defined in the previous section is the lower-bound on any *natural* clustering error (Theorem 9).

Claim 8 *The under/overclustering error and the correlation clustering error satisfy Definition 7 and hence are natural clustering errors.*

Theorem 9 *For any* natural *clustering error γ, any proposed clustering $\mathcal{C}$, and any target clustering $\mathcal{C}^*$, $\gamma(\mathcal{C}, \mathcal{C}^*) \geq \delta(\mathcal{C}, \mathcal{C}^*)$.*

Proof Given any proposed clustering $\mathcal{C}$, and any target clustering $\mathcal{C}^*$, we may transform $\mathcal{C}$ into $\mathcal{C}^*$ via the following sequence of edits. First, we split all overclustering instances using the following iterative procedure: while there exists a cluster C_i that contains points from C_j^* and some other ground-truth cluster(s), we split it into two clusters $C_{i,1} = C_i \cap C_j^*$ and $C_{i,2} = C_i - C_{i,1}$. Note that this iterative split procedure will require exactly δ_o split edits, where δ_o is the initial overclustering error. Then, when we are left with only "pure" clusters (each intersects exactly one target cluster), we merge all underclustering instances using the following iterative procedure: while there exist two clusters C_i and C_j that contain only points from the same target cluster, merge C_i and C_j. Note that this iterative merge procedure will require exactly δ_u merge edits, where δ_u is the initial underclustering error. Let us use γ to refer to any *natural* clustering error of $\mathcal{C}$ with respect to $\mathcal{C}^*$. By the first property of *natural* clustering error, each split must have decreased γ by at least one. By the second property, each merge must have decreased γ by at least one as well. Given that we performed exactly $\delta = \delta_o + \delta_u$ edits, it follows that initially $\gamma(\mathcal{C}, \mathcal{C}^*)$ must have been at least δ. ■

For additional discussion about comparing clusterings see Meilă (2007). Note that several criteria discussed in Meilă (2007) satisfy our first two properties (given that we replace "must decrease the error" with "must increase the similarity" in Definition 7). In addition, the Rand criteria and the Mirkin criteria discussed in Meilă (2007) are closely related to the correlation clustering error defined here (all three measures are a function of the number of pairs of points that are clustered incorrectly).

2.4 Local interactive clustering

We now give a general overview of our local interactive clustering model. This model is then instantiated by specific split/merge algorithms in Section 3 and Section 4 (depending on the assumptions about the oracle requests and the separability of the data).

Our approach is outlined in Figure 1 - we iteratively keep on making the local edits requested by the oracle. Note that the oracle does not provide the algorithm with any additional information about how to execute the split/merge. The description in Figure 1 also assumes that the requests come one at a time. Batch requests can be handled as well by queuing them in arbitrary order and then executing them one at a time (as long as the clusters in the request have not changed).

Given that the oracle requests must be somewhat consistent with the target clustering (see our assumptions about the validity of such requests in Section 2.2), and that our algorithms must therefore make progress with respect to finding the target clustering (see our analysis in Section 3 and Section 4), we expect the loop to terminate when we reach the target clustering.

Also note that the description in Figure 1 does not say anything about the sequence of oracle requests. The algorithms for the η-merge model in Section 3 are consistent with this description; they make no additional assumptions about this sequence. However, the algorithms in the unrestricted-merge model in Section 4 do make an additional assumption that each merge request is drawn uniformly at random from the set of valid merge requests.

Figure 1: Local interactive clustering model

```
while (there exists split/merge request from oracle)
    • perform corresponding local clustering edit
```

3. The η-merge model

In this section we describe and analyze our split/merge algorithms in the η-merge model. As a preprocessing step for all our split/merge algorithms, we first run the hierarchical average-linkage algorithm on all the points in the data set to compute the global average-linkage tree, which we denote by T_{glob}. The leaf nodes in this tree contain the individual points, and the root node contains all the points. The tree is computed in a bottom-up fashion: starting with the leafs in each iteration the two most similar nodes are merged, where the similarity between two nodes N_1 and N_2 is the average similarity between the points in N_1 and the points in N_2.

We assign a label "impure" to each cluster in the initial clustering; these labels are used by the merge procedure but are not visible to the user. Given a split or merge request from the oracle, a local clustering edit is computed from the global tree T_{glob} as described in Figure 2 and Figure 3.

To implement Step 1 in Figure 2, we start at the root of T_{glob} and "follow" the points in C_i down one of the branches until we find a node that splits them. In order to implement Step 2 in Figure 3, it suffices to perform a post-order traversal of T_{glob} and return the first node that has "enough" points from both clusters.

Figure 2: Split procedure

```
Algorithm: SPLIT PROCEDURE
Input: Cluster $C_i$, global average-linkage tree $T_{glob}$.
    1. Search $T_{glob}$ to find the node $N$ at which the set of points in $C_i$ are first split in two.
    2. Let $N_1$ and $N_2$ be the children of $N$. Set $C_{i,1} = N_1 \cap C_i$, $C_{i,2} = N_2 \cap C_i$.
    3. Delete $C_i$ and replace it with $C_{i,1}$ and $C_{i,2}$. Mark the two new clusters as "impure".
```

The split procedure is fairly intuitive: if the average-linkage tree is consistent with the target clustering, it suffices to find the node in the tree where the corresponding points are first split in two. It is more challenging to develop a correct merge procedure: note that Step 2 in Figure 3 is only correct if $\eta > 0.5$, which ensures that if two nodes in the tree have more than an η-fraction of the points from C_i and C_j, one must be an ancestor of the other. If the average-linkage tree is consistent with the ground-truth, then clearly the node equivalent to the corresponding target cluster (that C_i and C_j both intersect) will have enough points from C_i and C_j; therefore the node that we find in Step 2 must be this node or one of its descendants. In addition, because our merge procedure may replace two clusters with three, we require pure/impure labels for the merge requests to terminate: "pure" clusters may only have other points added to them, and retain this label throughout the execution of the algorithm.

Figure 3: Merge procedure

```
Algorithm: MERGE PROCEDURE
Input: Clusters $C_i$ and $C_j$, global average-linkage tree $T_{glob}$.
    1. If $C_i$ is marked as "pure" set $\eta_1 = 1$ else set $\eta_1 = \eta$. Similarly set $\eta_2$ for $C_j$.
    2. Search $T_{glob}$ for a node of maximal depth $N$ that contains enough points from $C_i$ and
       $C_j$: $|N \cap C_i| \geq \eta_1 |C_i|$ and $|N \cap C_j| \geq \eta_2 |C_j|$.
    3. Replace $C_i$ by $C_i \setminus N$, replace $C_j$ by $C_j \setminus N$.
    4. Add a new cluster containing $N \cap (C_i \cup C_j)$, mark it as "pure".
```

We now state the performance guarantee for these split and merge algorithms.

Theorem 10 *Suppose the target clustering satisfies stability, and the initial clustering has overclustering error δ_o and underclustering error δ_u. In the η-merge model, for any $\eta > 0.5$, the algorithms* *in Figure 2 and Figure 3 require at most δ_o split requests and $2(\delta_u + k)\log_{\frac{1}{1-\eta}} n$ merge requests to find the target clustering.*

In order to prove the theorem, we must do some preliminary analysis. First, we observe that if the target clustering satisfies stability, then every node of the average-linkage tree must be *laminar* (consistent) with respect to the target clustering. Informally, each node in a hierarchical clustering tree T is said to be *laminar* (consistent) with respect to the clustering $\mathcal{C}$ if for each cluster $C_i \in \mathcal{C}$, the points in C_i are first grouped together in T before they are grouped with points from any other cluster $C_{j \neq i}$. We formally state and prove these observations next.

Definition 11 (Laminar) *A node N is laminar with respect to a clustering $\mathcal{C}$ if for each cluster $C_i \in \mathcal{C}$ we have either $N \cap C_i = \emptyset$, $N \subseteq C_i$, or $C_i \subset N$.*

Lemma 12 *Suppose the ground-truth clustering $\mathcal{C}^*$ over a domain X satisfies stability with respect to a similarity function S. Let T be the average-linkage tree for X constructed with S. Then every node in T is laminar w.r.t. $\mathcal{C}^*$.*

Proof The proof of this statement can be found in Balcan et al. (2008). The intuition is that if there is a node in T that is not laminar w.r.t. C^*, then the average-linkage algorithm, at some step, must have merged $A \subseteq C_i^*$, with $B \subset C_j^*$ for some $i \neq j$. However, this will contradict the stability property for the sets A and/or B. ■

It follows that the split computed by the algorithm in Figure 2 must also be consistent with the target clustering; we call such splits *clean*.

Definition 13 (Clean split) *A partition (split) of a cluster C_i into clusters $C_{i,1}$ and $C_{i,2}$ is said to be* clean *if $C_{i,1}$ and $C_{i,2}$ are non-empty, and for each ground-truth cluster C_j^* such that $C_j^* \cap C_i \neq \emptyset$, either $C_j^* \cap C_i = C_j^* \cap C_{i,1}$ or $C_j^* \cap C_i = C_j^* \cap C_{i,2}$.*

We now prove the following properties regarding the correctness of the split/merge procedures; these properties are then used in the proof of Theorem 10.

Lemma 14 *If the ground-truth clustering satisfies stability and $\eta > 0.5$ then,*

a. *The split procedure in Figure 2 always produces a clean split.*

b. *The new cluster added by the merge procedure in Figure 3 must be "pure", i.e., it must contain points from a single ground-truth cluster.*

Proof a. For purposes of contradiction, suppose the returned split is not clean: $C_{i,1}$ and $C_{i,2}$ contain points from the same ground-truth cluster C_j^*. It must be the case that C_i contains points from several ground-truth clusters, which implies that w.l.o.g. $C_{i,1}$ contains points from some other ground-truth cluster $C_{l \neq j}^*$. This implies that N_1 is not laminar w.r.t. $\mathcal{C}^*$, which contradicts Lemma 12. **b.** By our assumption, at least $\eta|C_i|$ points from C_i and $\eta|C_j|$ points from C_j are from the same ground-truth cluster C_l^*. Clearly, the node N' in T_{glob} that is equivalent to C_l^* (which contains all the points in C_l^* and no other points) must contain *enough* points from C_i and C_j, and only ascendants and descendants of N' may contain more than an $\eta > 1/2$ fraction of points from both clusters. Therefore, the node N that we find in Step 2 in Figure 3 must be N' or one of its descendants, and will only contain points from C_l^*. ■

We also prove an additional property of the split algorithm - it always makes progress with respect to reducing overclustering error. This property is then used in the proof of Theorem 10.

Lemma 15 *If the ground-truth clustering satisfies stability, then the split algorithm in Figure 2 must reduce overclustering error by exactly 1.*

Proof Suppose we execute Split(C_1), and call the resulting clusters C_2 and C_3. Call δ_1 the overclustering error before the split, and δ_2 the overclustering error after the split. Let's use k_1 to refer to the number of ground-truth clusters that intersect C_1, and define k_2 and k_3 in the same manner. Due to the *clean split* property, no ground-truth cluster can intersect both C_2 and C_3, therefore it must be the case that $k_2 + k_3 = k_1$. Also, clearly $k_2, k_3 > 0$. Therefore we have:

$$\begin{aligned}\delta_2 &= \delta_1 - (k_1 - 1) + (k_2 - 1) + (k_3 - 1) \\ &= \delta_1 - k_1 + (k_2 + k_3) - 1 \\ &= \delta_1 - 1.\end{aligned}$$

■

Using Lemma 14 and Lemma 15, we can now prove the bounds on the number of split and merge requests stated in Theorem 10.

Proof [Proof of Theorem 10] We first give a bound on the number of splits. Lemma 15 shows that each split reduces the overclustering error by exactly 1. It is easy to verify that merges cannot increase overclustering error. Therefore the total number of splits may be at most δ_o.

We next give the argument about the number of *impure* and *pure* merges. We call a merge *pure* if both clusters have the label "pure", and call it *impure* otherwise. We first argue that we cannot have too many impure merges before each cluster in C is marked pure. Consider the clustering $P = \{C_i \cap C_j^* \mid C_i \text{ is marked "impure" and } C_i \cap C_j^* \neq \emptyset\}$. Clearly, at the start $|P| = \delta_u + k$. A merge does not increase the number of clusters in P, and the splits do not change P at all (because of the *clean split* property). Moreover, each impure merge *depletes* some $P_i \in P$ by moving at least $\eta|P_i|$ of its points to a pure cluster. Clearly, we can then have at most $\log_{1/(1-\eta)} n$ merges depleting each P_i. Since each impure merge must deplete some P_i, it must be the case that we can have at most $(\delta_u + k) \log_{1/(1-\eta)} n$ impure merges in total.

Notice that a pure cluster can only be created by an impure merge, and there can be at most one pure cluster created by each impure merge. Clearly, a pure merge removes exactly one pure cluster. Therefore the number of pure merges may be at most the total number of pure clusters that are created, which is at most the total number of impure merges. Therefore the total number of merges must be less than $2(\delta_u + k) \log_{1/(1-\eta)} n$. ■

We can also restate the run-time bound in Theorem 10 in terms of any *natural* clustering error γ. The following corollary follows from Theorem 10 and Theorem 9.

Corollary 16 *Suppose the target clustering satisfies stability, and the initial clustering has clustering error γ, where γ is any* natural *clustering error as defined in Definition 7. In the η-merge model, for any $\eta > 0.5$, the algorithms in Figure 2 and Figure 3 require at most $O((\gamma + k)\log_{\frac{1}{1-\eta}} n)$ edit requests to find the target clustering.*

3.1 Algorithms for correlation-clustering error

To efficiently find the target clustering with respect to the correlation clustering error, we propose a different merge procedure, which is described in Figure 4.

Figure 4: Merge procedure for the *correlation-clustering* objective

```
Algorithm: MERGE PROCEDURE
Input: Clusters $C_i$ and $C_j$, global average-linkage tree $T_{glob}$
  Search $T_{glob}$ for a node of maximal depth $N$ that contains enough points from $C_i$ and $C_j$:
  $|N \cap C_i| \geq \eta|C_i|$ and $|N \cap C_j| \geq \eta|C_j|$
  if $|C_i| \geq |C_j|$ then
    Replace $C_i$ by $C_i \cup (N \cap C_j)$
    Replace $C_j$ by $C_j \setminus N$
  else
    Replace $C_i$ by $C_i \setminus N$
    replace $C_j$ by $C_j \cup (N \cap C_i)$
  end if
```

Here instead of creating a new "pure" cluster, we add the corresponding points to the larger of the two clusters in the merge request. Notice that this algorithm is much simpler than the merge algorithm in Figure 3, and does not require pure/impure labels. Using this merge procedure and the split procedure presented earlier gives the following performance guarantee.

Theorem 17 *Suppose the target clustering satisfies stability, and the initial clustering has correlation-clustering error δ_{cc}. In the η-merge model, for any $\eta > 2/3$, using the split and merge procedures in Figures 2 and 4 requires at most δ_{cc} edit requests to find the target clustering.*

Proof Recall that we may express the correlation clustering error δ_{cc} as a summation of overclustering error δ_{cco} and underclustering error δ_{ccu} (see Section 2.1). Consider the contributions of individual points to δ_{cco} and δ_{ccu}, which are defined as:

$$\delta_{cco}(u) = |\{v \in X : c(u,v) = 1 \text{ and } c^*(u,v) = 0\}|,$$
$$\delta_{ccu}(u) = |\{v \in X : c(u,v) = 0 \text{ and } c^*(u,v) = 1\}|.$$

We first argue that a split of a cluster C_i must reduce δ_{cc}. Given that the split is *clean*, it is easy to verify that the outcome may not increase $\delta_{ccu}(u)$ for any $u \in C_i$. We can also verify that for each $u \in C_i$, $\delta_{cco}(u)$ must decrease by at least 1. This completes the argument, given that for all other points $w \notin C_i$, $\delta_{cco}(w)$ and $\delta_{ccu}(w)$ remain unchanged.

We now argue that if $\eta > 2/3$, each merge of C_i and C_j must reduce δ_{cc}. Without loss of generality, suppose that $|C_i| \geq |C_j|$, and let us use P to refer to the "pure" subset of C_j that is moved to C_i. We observe that the outcome must remove at least δ_1 *pairwise correlation-clustering errors* (see Section 2.1), where δ_1 satisfies $\delta_1 \geq 2|P|(\eta|C_i|)$. Similarly, we observe that the outcome may add at most δ_2 pairwise correlation-clustering errors, where δ_2 satisfies:

$$\delta_2 \leq 2|P|((1-\eta)|C_i|) + 2|P|((1-\eta)|C_j|) \leq 4|P|((1-\eta)|C_i|).$$

It follows that for $\eta > 2/3$, δ_1 must exceed δ_2; therefore the number of pairwise correlation-clustering errors must decrease, giving a lower correlation-clustering error. ■

Observe that the runtime bound in Theorem 17 is tight up to a constant: in some instances any *local* algorithm requires at least $\delta_{cc}/2$ edits to find the target clustering. To verify this, suppose the target clustering is composed of n singleton clusters, and the initial clustering contains $n/2$ clusters of size 2. In this instance, the initial correlation clustering error $\delta_{cc} = n$, and the oracle must issue at least $n/2$ split requests before we reach the target clustering (one for each of the $n/2$ initial clusters).

3.2 Algorithms under stronger assumptions

When the data satisfies stronger separability properties we may simplify the presented algorithms and/or obtain better performance guarantees. In particular, if the data satisfies the *strict separation* property (see Definition 3), we may change the split and merge algorithms to use the local average-linkage tree, which is constructed from only the points in the edit request. In addition, if the data satisfies *strict threshold separation* (see Definition 4), we may remove the restriction on η and use a different merge procedure that is correct for any $\eta > 0$.

Theorem 18 *Suppose the target clustering satisfies strict separation, and the initial clustering has overclustering error δ_o and underclustering error δ_u. In the η-merge model, for any $\eta > 0.5$, the algorithms in Figure 5 and Figure 6 require at most δ_o split requests and $2(\delta_u + k)\log_{\frac{1}{1-\eta}} n$ merge requests to find the target clustering.*

Proof Let us use $\mathcal{L}^*$ to refer to the ground-truth clustering of the points in the split/merge request. If the target clustering satisfies strict separation, it is easy to verify that every node in the local average-linkage tree T_{loc} must be laminar (consistent) w.r.t. $\mathcal{L}^*$. We can then use this observation to prove the equivalent of Lemma 14 for strict separation for the split procedure in Figure 5 and the merge procedure in Figure 6. Similarly, we can prove the equivalent of Lemma 15 for strict separation for the split procedure in Figure 5. The analysis in Theorem 10 remains unchanged. ■

Theorem 19 *Suppose the target clustering satisfies strict threshold separation, and the initial clustering has overclustering error δ_o and underclustering error δ_u. In the η-merge model, for any $\eta > 0$, the algorithms in Figure 5 and Figure 7 require at most δ_o split requests and $2(\delta_u + k)\log_{\frac{1}{1-\eta}} n$ merge requests to find the target clustering.*

Proof If the target clustering satisfies strict threshold separation, we can verify that for any $\eta > 0$, the split procedure in Figure 5 and the merge procedure in Figure 7 still have the properties stated in Lemma 14 and Lemma 15. The analysis in Theorem 10 remains unchanged.

To verify that the split procedure always produces a clean split, again let us use $\mathcal{L}^*$ to refer to the ground-truth clustering of the points in the split request. If the target clustering satisfies strict threshold separation, we can again verify that every node in the local average-linkage tree T_{loc} must be laminar (consistent) w.r.t. $\mathcal{L}^*$. It follows that the split procedure always produces a clean split and must reduce overclustering error by exactly 1. Note that clearly this argument does not depend on the setting of η.

We now verify that the new cluster added by the merge procedure in Figure 7 must be "pure" (must contain points from a single target cluster). Observe that due to the strict threshold separation property, in the graph G in Figure 7, all pairs of points from the same target cluster must be connected before any pairs of points from different target clusters. It follows that the first component that contains at least an η-fraction of points from C_i and C_j must be "pure". Note that this argument applies for any $\eta > 0$. ■

Figure 5: Split procedure under stronger assumptions

```
Algorithm: SPLIT PROCEDURE
Input: Cluster $C_i$, local average-linkage tree $T_{loc}$.

  1. Let $C_{i,1}$ and $C_{i,2}$ be the children of the root in $T_{loc}$.

  2. Delete $C_i$ and replace it with $C_{i,1}$ and $C_{i,2}$. Mark the two new clusters as "impure".
```

Figure 6: Merge procedure under strict separation

```
Algorithm: MERGE PROCEDURE
Input: Clusters $C_i$ and $C_j$, local average-linkage tree $T_{loc}$.

  1. If $C_i$ is marked as "pure" set $\eta_1 = 1$ else set $\eta_1 = \eta$. Similarly set $\eta_2$ for $C_j$.

  2. Search $T_{loc}$ for a node of maximal depth $N$ that contains enough points from $C_i$ and $C_j$:
     $|N \cap C_i| \geq \eta_1 |C_i|$ and $|N \cap C_j| \geq \eta_2 |C_j|$.

  3. Replace $C_i$ by $C_i \setminus N$, replace $C_j$ by $C_j \setminus N$.

  4. Add a new cluster containing $N \cap (C_i \cup C_j)$, mark it as "pure".
```

As in Corollary 16, we may also restate the run-time bounds in Theorem 18 and Theorem 19 in terms of any natural clustering error γ. The following corollaries follow from Theorem 18, Theorem 19 and Theorem 9.

Corollary 20 *Suppose the target clustering satisfies strict separation, and the initial clustering has clustering error γ, where γ is any* natural *clustering error as defined in Definition 7. In the η-merge model, for any $\eta > 0.5$, the algorithms in Figure 5 and Figure 6 require at most $O((\gamma + k) \log_{\frac{1}{1-\eta}} n)$ edit requests to find the target clustering.*

Figure 7: Merge procedure under strict threshold separation

```
**Algorithm**: MERGE PROCEDURE
**Input**: Clusters $C_i$ and $C_j$.

1. If $C_i$ is marked as "pure" set $\eta_1 = 1$ else set $\eta_1 = \eta$. Similarly set $\eta_2$ for $C_j$.

2. Let $G = (V, E)$ be a graph where $V = C_i \cup C_j$ and $E = \emptyset$. Set $N = \emptyset$.

3. While true:
     Connect the next-closest pair of points in G;
     Let $\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_m$ be the connected components of $G$;
     if there exists $\hat{C}_l$ such that $|\hat{C}_l \cap C_i| \geq \eta|C_i|$ and $|\hat{C}_l \cap C_j| \geq \eta|C_j|$ then
        $N = \hat{C}_l$;
        break;
     end if

4. Replace $C_i$ by $C_i \setminus N$, replace $C_j$ by $C_j \setminus N$.

5. Add a new cluster containing $N$, mark it as "pure".
```

Corollary 21 *Suppose the target clustering satisfies strict threshold separation, and the initial clustering has clustering error γ, where γ is any* natural *clustering error as defined in Definition 7. In the η-merge model, for any $\eta > 0$, the algorithms in Figure 5 and Figure 7 require at most $O((\gamma + k) \log_{\frac{1}{1-\eta}} n)$ edit requests to find the target clustering.*

4. The unrestricted-merge model

In this section we further relax the assumptions about the nature of the oracle requests. As before, the oracle may request to split a cluster if it contains points from two or more target clusters. For merges, now the oracle may request to merge C_i and C_j if both clusters contain only a single point from the same ground-truth cluster. We note that this is a minimal set of assumptions for a local algorithm to make progress, otherwise the oracle may request irrelevant splits or merges that prevent the algorithm from finding the target clustering. For this model we propose the merge algorithm described in Figure 8. The split algorithm remains the same as in Figure 2.

To provably find the ground-truth clustering in this setting we require that each merge request must be chosen uniformly at random from the set of *valid* merge requests. This assumption is consistent with the observation in Awasthi and Zadeh (2010) that in the unrestricted-merge model with arbitrary request sequences, even very simple clustering instances require a prohibitively large number of requests. We do not make additional assumptions about the nature of the split requests; in each iteration any *valid* split may be proposed by the oracle. In this setting our algorithms have the following performance guarantee.

Theorem 22 *Suppose the target clustering satisfies stability, and the initial clustering has overclustering error δ_o and underclustering error δ_u. In the unrestricted-merge model, assuming that each*

Figure 8: Merge procedure for the unrestricted-merge model

```
Algorithm: MERGE PROCEDURE
Input: Clusters $C_i$ and $C_j$, global average-linkage tree $T_{avg}$.

    1. Let $C_i'$, $C_j'$ = Split($C_i \cup C_j$), where the split is performed as in Figure 2.
    2. Delete $C_i$ and $C_j$.
    3. If the sets $C_i'$ and $C_j'$ are the same as $C_i$ and $C_j$, then add $C_i \cup C_j$, otherwise add $C_i'$ and $C_j'$.
```

merge request is chosen uniformly at random from the set of valid merge requests, with probability at least $1 - \epsilon$, the algorithms in Figure 2 and Figure 8 require δ_o split requests and $O(\log \frac{k}{\epsilon}\delta_u^2)$ merge requests to find the target clustering.

Theorem 22 is proved in the following lemmas. We first state a lemma regarding the correctness of the Algorithm in Figure 8. We argue that if the algorithm merges C_i and C_j, it must be the case that both C_i and C_j only contain points from the same ground-truth cluster.

Lemma 23 *If the algorithm in Figure 8 merges C_i and C_j in Step 3, it must be the case that $C_i \subset C_l^*$ and $C_j \subset C_l^*$ for some ground-truth cluster C_l^*.*

Proof We prove the contrapositive. Suppose C_i and C_j both contain points from C_l^*, and in addition $C_i \cup C_j$ contains points from some other ground-truth cluster. Let us define $S_1 = C_l^* \cap C_i$ and $S_2 = C_l^* \cap C_j$. Because the clusters C_i', C_j' result from a *clean* split, it follows that $S_1, S_2 \subset C_i'$ or $S_1, S_2 \subset C_j'$. Without loss of generality, assume $S_1, S_2 \subset C_i'$. Then clearly $C_i' \neq C_i$ and $C_i' \neq C_j$, so C_i and C_j are not merged. ■

The δ_o bound on the number of split requests follows from the observation that each split reduces the overclustering error by exactly 1 (see Lemma 15), and the fact that the merge procedure does not increase overclustering error (see Lemma 24).

Lemma 24 *The merge algorithm in Figure 8 does not increase overclustering error.*

Proof Suppose $C_i \cup C_j$ intersects several ground-truth clusters, and hence we obtain two new clusters C_i', C_j'. Let us call δ_1 the overclustering error before the merge, and δ_2 the overclustering error after the merge. Let's use k_1 to refer to the number of ground-truth clusters that intersect C_i, k_2 to refer to the number of ground-truth clusters that intersect C_j, and define k_1' and k_2' in the same manner. The new clusters C_i' and C_j' result from a "clean" split, therefore no ground-truth cluster may intersect both of them. It follows that $k_1' + k_2' < k_1 + k_2$. Therefore we now have:

$$\begin{aligned}\delta_2 &= \delta_1 - (k_1 - 1) - (k_2 - 1) + (k_1' - 1) + (k_2' - 1)\\ &= \delta_1 - (k_1 + k_2) + (k_1' + k_2') < \delta_1.\end{aligned}$$

If C_i and C_j are both subset of the same ground-truth cluster, then clearly the merge operation has no effect on the overclustering error. ■

Finally, Lemma 25 and Lemma 26 bound the number of impure and pure merges. Here we call a proposed merge *pure* if both clusters are subset of the same ground-truth cluster, and *impure* otherwise.

Lemma 25 *The merge algorithm in Figure 8 requires at most δ_u impure merge requests.*

Proof We argue that the result of each impure merge request must reduce the underclustering error δ_u by at least 1. Recall from Section 2.1 that we may express δ_u as a summation of underclustering error with respect to each target cluster. Suppose the oracle requests to merge C_i and C_j, and C'_i and C'_j are the resulting clusters. Clearly, the local edit has no effect on the underclustering error with respect to target clusters that do not intersect C_i or C_j. In addition, because the new clusters C'_i and C'_j result from a *clean* split, for target clusters that intersect exactly one of C_i, C_j, the underclustering error must stay the same. For target clusters that intersect both C_i and C_j, the underclustering error must decrease by exactly one; the number of such target clusters is at least one. ■

Lemma 26 *Assuming that each merge request is chosen uniformly at random from the set of valid merge requests, the probability that the algorithm in Figure 8 requires more than $O(\log \frac{k}{\epsilon}\delta_u^2)$ pure merge requests is less than ϵ.*

Proof We first consider the pure merge requests involving points from some ground-truth cluster C_i^*, the total number of pure merge requests (involving any ground-truth cluster) can then be bounded with a union-bound. To facilitate our argument, let us assign an identifier to each cluster containing points from C_i^* in the following manner:

1. Maintain a *cluster-id* variable, which is initialized to 1. To assign a "new" identifier to a cluster, set its identifier to *cluster-id*, and increment *cluster-id*.

2. In the initial clustering, assign a *new* identifier to each cluster containing points from C_i^*.

3. When we split a cluster containing points from C_i^*, assign its identifier to the newly-formed cluster containing points from C_i^*.

4. When we merge two clusters that are not both subset of the same target cluster, if one of the clusters contains points from C_i^*, assign its identifier to the newly-formed cluster containing points from C_i^*. If both clusters contain points from C_i^*, assign a *new* identifier to the newly-formed cluster containing points from C_i^*.

5. When we merge two clusters C_1 and C_2, and both contain only points from C_i^*, if the outcome is one new cluster, assign it a *new* identifier. If the outcome is two new clusters, assign them the identifiers of C_1 and C_2.

Observe that when clusters containing points from C_i^* are assigned identifiers in this manner, the maximum value of *cluster-id* is bounded by $O(\delta_i)$, where δ_i denotes the underclustering error of the initial clustering with respect to C_i^*: $\delta_i = \mathrm{dist}(C_i^*, C)$. To verify this, consider that we assign exactly $\delta_i + 1$ new identifiers in Step 2, and each time we assign a new identifier in Steps 4 and 5, δ_i decreases by one.

We say that a *pure* merge request involving points from C_i^* is *original* if the user has never asked us to merge clusters with the given identifiers, otherwise we say that this merge request is *repeated*. Given that the maximum value of *cluster-id* is bounded by $O(\delta_i)$, the total number of *original* merge requests must be $O(\delta_i^2)$.

We now argue that if a merge request is not original, we can lower bound the probability that it will result in the merging of the two clusters. For a repeated merge request $M_i = \text{Merge}(C_1, C_2)$, let X_i be a random variable defined as follows:

$$X_i = \begin{cases} 1 & \text{if neither } C_1 \text{ nor } C_2 \text{ have been involved in a pure merge request since} \\ & \text{the last time a merge of clusters with these identifiers was proposed.} \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, when $X_i = 1$ it must be the case that C_1 and C_2 are merged. Assuming that each merge request is chosen uniformly at random from the set of valid merge requests, we observe that $\Pr[X_i = 1] > \frac{1}{2\delta_i + 1}$. To verify this, observe that in each merge request the probability that the user requests to merge C_1 and C_2 is $\frac{1}{m}$, and the probability that the user requests a pure merge involving C_1 or C_2 and some other cluster is less than $\frac{2\delta_i}{m}$, where m is the total number of valid merge requests; we can then bound the probability that the former happens before the latter.

We can then use a Chernoff bound to argue that after $t = O(\log \frac{k}{\epsilon}\delta_i^2)$ *repeated* merge requests, the probability that $\sum_{i=1}^{t} X_i < \delta_i$ (which must be true if we need more *repeated* merge requests) is less than ϵ/k. Therefore, the probability that we need more than $O(\log \frac{k}{\epsilon}\delta_i^2)$ *repeated* merge requests is less than ϵ/k.

By the union-bound, the probability that we need more than $O(\log \frac{k}{\epsilon}\delta_i^2)$ *repeated* merge requests for *any* ground-truth cluster C_i^* is less than $k \cdot \epsilon/k = \epsilon$. Therefore with probability at least $1 - \epsilon$ for all ground-truth clusters we need $\sum_i O(\log \frac{k}{\epsilon}\delta_i^2) = O(\log \frac{k}{\epsilon}\sum_i \delta_i^2) = O(\log \frac{k}{\epsilon}\delta_u^2)$ *repeated* merge requests, where δ_u is the underclustering error of the initial clustering. Also recall that for all ground-truth clusters we need $\sum_i O(\delta_i^2) = O(\delta_u^2)$ *original* merge requests. Adding the two terms together, it follows that with probability at least $1 - \epsilon$ we need a total of $O(\log \frac{k}{\epsilon}\delta_u^2)$ pure merge requests. ■

As in the previous section, we also restate the run-time bound in Theorem 22 in terms of any *natural* clustering error γ. The following corollary follows from Theorem 22 and Theorem 9.

Corollary 27 *Suppose the target clustering satisfies stability, and the initial clustering has clustering error γ, where γ is any* natural *clustering error as defined in Definition 7. In the unrestricted-merge model, assuming that each merge request is chosen uniformly at random from the set of valid merge requests, with probability at least $1 - \epsilon$, the algorithms in Figure 2 and Figure 8 require $O(\log \frac{k}{\epsilon}\gamma^2)$ edit requests to find the target clustering.*

4.1 Algorithms under stronger assumptions

As in Section 3.2, if the data satisfies *strict separation*, then instead of the split procedure in Figure 2 we can use the procedure in Figure 5, which uses the local average-linkage tree (constructed from only the points in the user request). Similarly, we can modify the merge procedure in Figure 8 to perform the split as in Figure 5. We can then prove the equivalent of Theorem 22 and Corollary 27 for strict separation for these split/merge algorithms.

5. Experimental Results

We perform two sets of experiments. We first test our proposed split algorithm on the clustering of business listings maintained by Google. We then test the proposed model in its entirety on the 20 Newsgroups data set.

5.1 Clustering business listings

Google maintains a large collection of data records representing businesses, which as of 2013 contained more than 100 millions records. These records are initially clustered using a similarity function. Each cluster is expected to contain records about the same distinct business. Clusters are then summarized; these summaries are served to users online via various front-end applications. Users report bugs such as "you are displaying the name of one business, but the address of another" (caused by overclustering), or "a particular business is shown multiple times" (caused by underclustering). The clusters involved in these requests are often quite large and usually contain records about several businesses. Therefore this setting closely matches our interactive clustering model - users provide high-level feedback requesting cluster merges/splits, where the corresponding data points intersect several ground-truth clusters.

Here we evaluate the effectiveness of our proposed split algorithm in such cases. Correctness of a split is evaluated with respect to the ground-truth labels, which are obtained through manual inspection of the listings in the cluster. For example, from manual inspection it is very easy to see that two Starbucks records with different addresses are in fact different businesses (have different ground-truth labels) even if, say, they have the same corporate phone number. But this observation is not necessarily captured by the noisy similarity function, causing the listings to incorrectly cluster together.

We consider two criteria to determine whether a split is correct or not, and show experimental results for both. We first consider a binary split correct if the two resulting sub-clusters are "clean" according to Definition 13. Note that a clean split is sufficient and necessary for reducing the under/overclustering error in our model. To compute the splits, we use the algorithm in Figure 5, which we refer to as *Clean-Split*. This algorithm is easier to implement and run than the algorithm in Figure 2 because it only requires local pairwise similarities (for the listings in the cluster). But this algorithm is still provably correct under stronger assumptions on the data (see Theorem 18 and Theorem 19), which we believe are still satisfied in this application.

We compare performance with two well-known split algorithms from split/merge clustering literature: the optimal 2-median clustering (*2-Median*), and a "sweep" of the second-smallest eigenvector of the corresponding graph Laplacian matrix. With respect to the spectral split algorithm, let $\{v_1, \ldots, v_n\}$ be the order of the vertices when sorted by their eigenvector entries. We compute the partition $\{v_1, \ldots, v_i\}$ and $\{v_{i+1}, \ldots, v_n\}$ such that its conductance is smallest (*Spectral-Balanced*), and a partition such that the similarity between v_i and v_{i+1} is smallest (*Spectral-Gap*).

We compare the split algorithms on overclustering instances that were discovered during a manual clustering quality evaluation. We consider 20 such instances to ensure that our results are consistent, given that there may be differences in the difficulty of each instance. We have anonymized

Table 1: Number of correct (clean) splits on the Google business listings data sets.

Clean-Split	2-Median	Spectral-Gap	Spectral-Balanced
19	13	12	3

these data-sets and made them publicly available. [2] The results are presented in Table 1. We observe that the *Clean-Split* algorithm works best, giving a correct split in 19 out of the 20 cases. The well-known *Spectral-Balanced* technique usually does not give correct splits in this application. The balance constraint causes it to put records about the same business on both sides of the partition (especially when all the "clean" splits are not well-balanced), which increases clustering error. As expected, the *Spectral-Gap* technique improves on this limitation (because it does not have a balance constraint), but the result is often still incorrect. The *2-Median* algorithm performs fairly well, but it is still a limited approach in this application: the optimal centers may correspond to listings about the same business, and even if they represent distinct businesses, the resulting split is still sometimes incorrect.

In addition to using the clean-split criterion, we also evaluate the computed splits using the correlation-clustering (cc) error. We find that using this criterion *Clean-Split* and *2-Median* compute the best splits, while the other two algorithms perform significantly worse. The results for *Clean-Split* and *2-Median* are presented in Table 2 - we see that *Clean-Split* is more reliable because it fails to reduce the cc-error only once, compared to 4 such failures for *2-Median*. Table 2 also reinforces our theoretical understanding of the effectiveness of our split algorithm. We know that a clean split is sufficient to reduce the cc-error, but it is not necessary. Our experiments illustrate these observations: *Clean-Split* makes progress in reducing the cc-error in 19 out of 20 cases (when the resulting split is clean), while we have also confirmed that *2-Median* sometimes still reduces the cc-error even when the resulting split is not clean.

5.2 Clustering Newsgroup documents

In order to test our entire interactive clustering model, we perform experiments on the 20 Newsgroups data set.[3] The points in the data set are posts to twenty different online forums (called newsgroups). We sample these data to generate 5 data sets of manageable size (containing 276-301 points), which are labeled A through E in the figures. Each data set contains some documents from every newsgroup. The purpose of these 5 data sets is to check the consistency of our results, given that there may be considerable variation in how separable the ground-truth clusters are in each case. For each data set, the ground-truth is determined by the newsgroups that the documents belong to.

Each Newsgroup document is represented by a term frequency - inverse document frequency (tf-idf) vector (Salton and Buckley, 1988). We use cosine similarity to compare these vectors, which gives a similarity score in $[0, 1]$. We compute an initial clustering by using the following procedure to perturb the ground-truth: for each document we keep its ground-truth cluster assignment with probability 0.5, and otherwise reassign it to one of the other clusters, which is chosen uniformly at random. This procedure generates initial clusterings with overclustering error of about 100, underclustering error of about 100, and correlation-clustering error of about 5000.

2. The anonymized Google business listings data sets are available here.
3. http://people.csail.mit.edu/jrennie/20Newsgroups/

Table 2: Change in correlation-clustering error for splits on the Google business listings data sets. A negative value corresponds to a decrease in error.

Data Set	Clean-Split	2-Median
1	-14	-14
2	-5	-5
3	-11	-11
4	-117	-117
5	-42	+90
6	-4	-4
7	-12	-30
8	-27	-27
9	-6	-6
10	-6	-6
11	+6	-8
12	-10	+14
13	-6	-6
14	-12	-22
15	-6	-6
16	-10	+14
17	-11	-27
18	-10	-10
19	-11	-5
20	-10	-10

To simulate user feedback, we compute the set of all valid split and merge requests. Consistent with our model, a split of a cluster is considered valid if it contains points from 2 or more ground-truth clusters, and a merge is considered valid if at least an η- fraction of the points in each cluster are from the same ground-truth cluster. We then choose one of the valid splits/merges uniformly at random, and ask the algorithm to compute the corresponding split/merge. We continue to iterate until we find the ground-truth clustering or reach a limit on the number of edit requests (set to 20000).

The Newsgroup data is know to be very challenging for unsupervised (Telgarsky and Dasgupta, 2012; Heller and Ghahramani, 2005) and semi-supervised clustering algorithms (Basu et al., 2002, 2004). In particular, on harder Newsgroup clustering instances Basu et al. (2002) and Basu et al. (2004) report that they are unable to come close to finding the ground-truth clustering even with considerable supervision.

To study how our algorithms perform on easier clustering instances, we also slightly prune our data sets. Our heuristic iteratively finds and removes the outlier in each target cluster, where the outlier is the point with minimum sum-similarity to the other points in the target cluster. For each data set, we perform experiments with the original (unpruned) data set, a pruned data set with 2 points removed per target cluster, and a pruned data set with 4 points removed per target cluster, which prunes 40 and 80 points, respectively. Our pruning heuristic generates clustering instances where the ground-truth clusters are more separable, similar to the Different-3 data set of Basu et al. (2002) and the News-diff3 data set of Basu et al. (2004).

5.2.1 EXPERIMENTS IN THE η-MERGE MODEL

We first experiment with algorithms in the η-merge model. Here we use the algorithm in Figure 2 to perform the splits, and the algorithm in Figure 3 to perform the merges. We show the results for data set A in Figure 9; the complete experimental results are in Appendix B. We find that for larger settings of η, the amount of supervision (number of split/merge requests needed to find the target clustering) is quite low and consistent with our theoretical analysis. The results are better for pruned data sets, where we get very good performance regardless of the setting of η. The results for the algorithms in Figure 2 and Figure 4 (for the correlation-clustering objective) are very favorable as well.

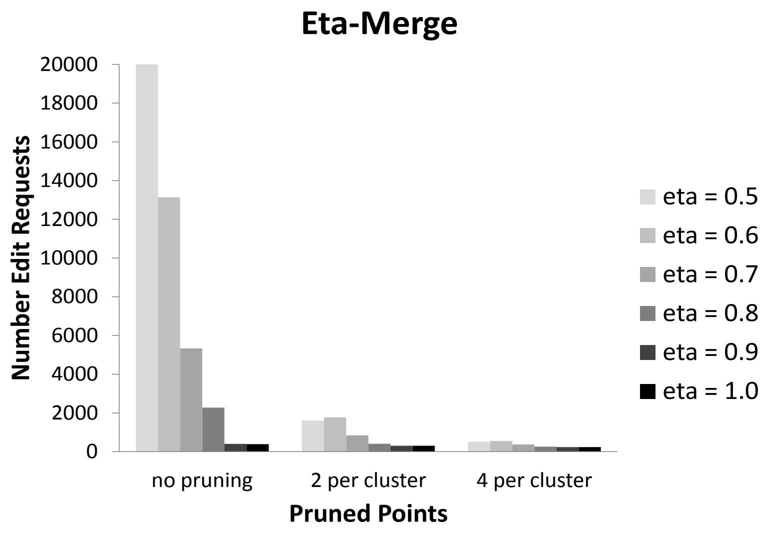
5.2.2 EXPERIMENTS IN THE UNRESTRICTED-MERGE MODEL

We also experiment with algorithms in the unrestricted merge model. Here we use the same algorithm to perform the splits, and use the algorithm in Figure 8 to perform the merges. We show the results on data set A in Figure 10; the complete experimental results are in Appendix B. We find that again for larger settings of η the amount of supervision is quite low and consistent with our theoretic analysis (we only show results for $\eta \geq 0.5$), and performance improves further for pruned data sets. Our investigations show that for unpruned data sets and smaller settings of η, we are still able to quickly achieve very small clustering error (results not shown), but the algorithms are unable to converge to the ground-truth clustering due to inconsistencies in the average-linkage tree. We can address some of these inconsistencies by constructing the tree in a more robust way, which indeed gives improved performance (see Appendix C).

5.2.3 EXPERIMENTS WITH SMALL INITIAL ERROR

We also consider a setting where the initial clustering is already very accurate. In order to simulate this scenario, when we compute the initial clustering, for each document we keep its ground-truth cluster assignment with probability 0.95, and otherwise reassign it to one of the other clusters, which is chosen uniformly at random. This procedure usually gives us initial clusterings with over-clustering and underclustering error between 5 and 20, and correlation-clustering error between 500 and 1000. Note that this setting also closely matches the scenario where the initial clustering is computed using a good clustering algorithm, which is quite realistic in some applications.

As expected, in this setting our interactive algorithms perform much better, especially on pruned data sets. Figure 11 displays the results; we can see that in these scenarios it often takes less than one hundred edit requests to find the target clustering in both models.



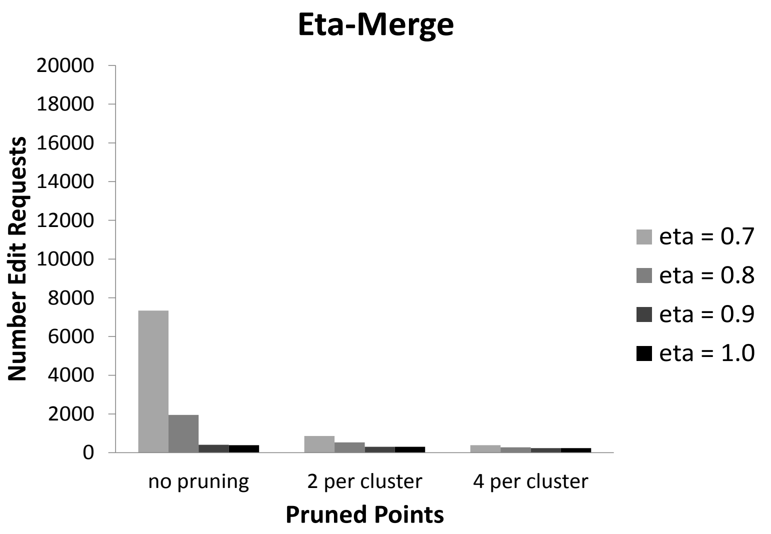


Figure 9: Experimental results in the η-merge model for data set A. The second chart corresponds to algorithms for correlation clustering error.

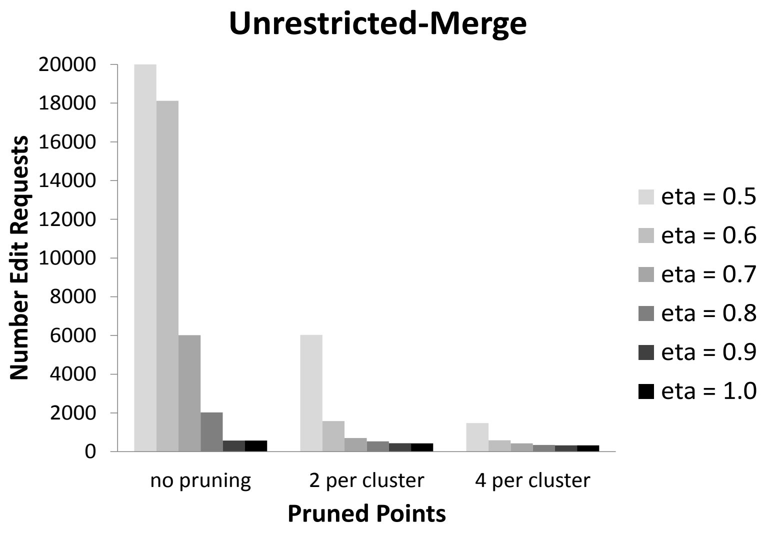


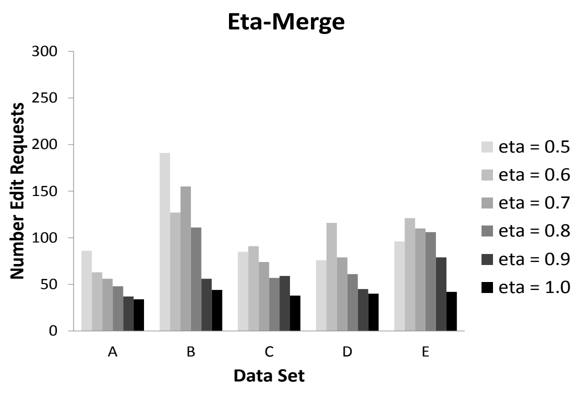
Figure 10: Experimental results in the unrestricted merge model for data set A.

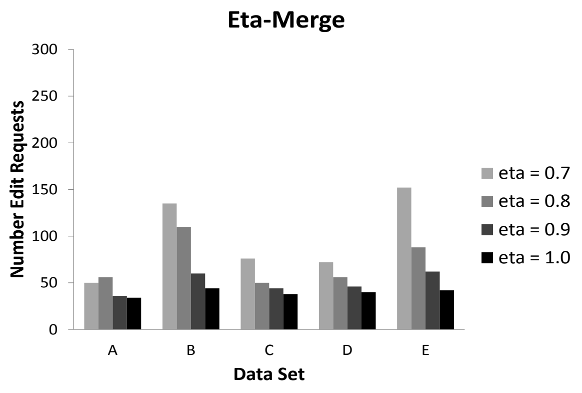
5.2.4 COMPARISON WITH OTHER SEMI-SUPERVISED CLUSTERING MODELS

In this section we give a performance comparison on the 20 Newsgroups data sets with the semi-supervised clustering models of Basu et al. (2002) and Basu et al. (2004). The accuracy of the output clustering is stated in terms of *normalized mutual information* (NMI) (see Strehl et al., 2000).

Table 3 shows a comparison on a harder Newsgroup clustering instance; table 4 shows a comparison on an easier instance. The performance of our approach is with respect to the η-merge model, using the algorithm in Figure 2 to perform the splits, and the algorithm in Figure 3 to perform the merges; we focus on a single setting of $\eta = 0.7$. To compare the amount of supervision used by each model, consider that n instance class labels in the framework of Basu et al. (2002) are equivalent to $n(n-1)/2$ pairwise labels in the framework of Basu et al. (2004). Let us also oversimplify and assume that the cost of a pairwise label in the framework of Basu et al. (2004) is the same as the cost of one/split merge request in our framework. Also, we can adjust for differences in data set size by assuming that the proportion of supervision does not change. In particular, 1500 instance class labels on a data set of 3000 documents correspond to 150 instance class labels on a data set of 301 documents. These 150 instance class labels are then equivalent to 150 * 149 / 2 = 11175 pairwise labels. Similarly, 1500 instance class labels on a data set of 3000 documents correspond to 138 instance class labels on a data set of 276 documents, which are then equivalent to 138 * 137 / 2 = 9453 pairwise labels.

Then for a harder clustering instance we observe that we need less supervision than Basu et al. (2002) and roughly 7.5 times more supervision than Basu et al. (2004) to achieve a completely correct clustering (NMI = 1.0) as opposed to a clustering that is only half-correct (NMI = 0.44, 0.55). For an easier clustering instance, we observe that we need roughly 2.7 times less supervision than





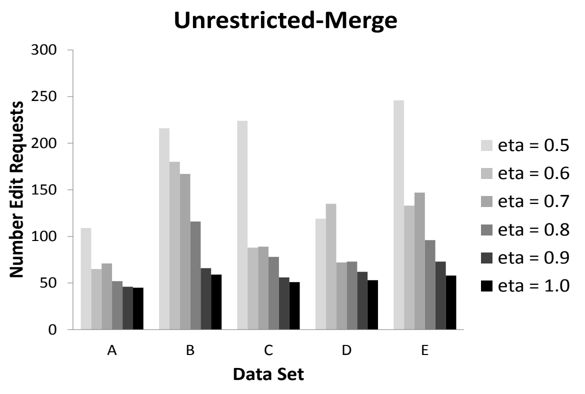


Figure 11: Experimental results for initial clusterings with small error. Results presented for pruned data sets (4 points per cluster). The second chart corresponds to algorithms for correlation clustering error.

Basu et al. (2002), and 3.9 times more supervision than Basu et al. (2004) to achieve a completely correct clustering (NMI = 1.0) as opposed to a mostly-correct clustering (NMI = 0.88).

These are favorable comparisons for our model - we are able to compute more accurate clusterings while using a comparable amount of supervision. We also note that for Basu et al. (2002) and Basu et al. (2004), additional supervision does not improve accuracy - the stated performance is the best accuracy they can achieve. Our framework is the only approach that has demonstrated a complete recovery of the ground truth on the Newsgroup data.

Table 3: Framework comparison on harder Newsgroup clustering instance.

Framework	Data Set Name	Data Set Size	Amount of Supervision	Output Accuracy
Basu et al. (2002)	Same-3	3000	1500 labeled data points	NMI = 0.44
Basu et al. (2004)	News-sim3	300	1000 pairwise constraints	NMI = 0.55
Interactive clustering	Data Set E	301	7573 split/merge requests	NMI = 1.0

Table 4: Framework comparison on easier Newsgroup clustering instance.

Framework	Data Set Name	Data Set Size	Amount of Supervision	Output Accuracy
Basu et al. (2002)	Different-3	3000	1500 labeled data points	NMI = 0.88
Basu et al. (2004)	News-diff3	300	1000 pairwise constraints	NMI = 0.88
Interactive clustering	Data Set D	276	3548 split/merge requests	NMI = 1.0

6. Discussion

In this work we motivated and studied a new framework and algorithms for interactive clustering. Our framework models practical constraints on the algorithms: we start with an initial clustering that we cannot modify arbitrarily, and are only allowed to make local edits consistent with user requests. In this setting, we develop several simple, yet effective algorithms under different assumptions about the nature of the edit requests and the structure of the data. We present theoretical analysis that shows that our algorithms converge to the target clustering after a limited number of edit requests. We also present experimental evidence that shows that our algorithms work well in practice.

Several directions come out of this work. It would be interesting to relax the condition on η in the η-merge model, and the assumption about the request sequences in the unrestricted-merge model. It is also important to study additional properties of an interactive clustering algorithm. In particular, it is often desirable that the algorithm never increase the error of the current clustering. Our algorithms in Figures 2, 4 and 8 have this property, but the algorithm in Figure 3 does not.

Other user feedback models have also been proposed (see Appendix A for how they relate to the split/merge feedback considered here). It is valuable to further study the practical applicability of each kind of feedback and understand under what conditions they are equivalent.

7. Acknowledgments

This work was supported in part by grants NSF CCF-0953192, NSF-CCF 1535967, NSF CCF-1422910, NSF IIS1618714, ONR grant N00014-09-1-0751, AFOSR grant FA9550-09-1-0538, a Google Research Award, and a Microsoft Research Fellowship.

References

Dimitris Achlioptas and Frank McSherry. On spectral learning of mixtures of distributions. In *COLT*, 2005.

Margareta Ackerman and Sajoy Dasgupta. Incremental clustering: The case for extra clusters. In *NIPS*, 2014.

Margareta Ackerman, Shai Ben-David, Simina Brânzei, and David Loker. Weighted clustering. In *AAAI*, 2012.

Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1998.

Sanjeev Arora and Ravi Kannan. Learning mixtures of arbitrary gaussians. In *STOC*, 2001.

Hassan Ashtiani, Shrinu Kushagra, and Shai Ben-David. Clustering with same-cluster queries. *CoRR*, abs/1606.02404, 2016.

Pranjal Awasthi and Maria-Florina Balcan. Center based clustering: A foundational perspective. In Christian Hennig, Marina Meila, Fionn Murtagh, and Roberto Rocci, editors, *Handbook of Cluster Analysis*. CRC Press, 2015.

Pranjal Awasthi and Reza Bosagh Zadeh. Supervised clustering. In *NIPS*, 2010.

Maria-Florina Balcan and Avrim Blum. Clustering with interactive feedback. In *ALT*, 2008.

Maria-Florina Balcan and Pramod Gupta. Robust hierarchical clustering. In *COLT*, 2010.

Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. A discriminative framework for clustering via similarity functions. In *STOC*, 2008.

Maria-Florina Balcan, Yingyu Liang, and Pramod Gupta. Robust hierarchical clustering. *Journal of Machine Learning Research*, 15:3831–3871, 2014.

Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56 (1-3):89–113, 2004.

Sugato Basu, Arindam Banerjee, and Raymond Mooney. Semi-supervised clustering by seeding. In *ICML*, 2002.

Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. Active semi-supervision for pairwise constrained clustering. In *SDM*, 2004.

Mikhail Belkin and Kaushik Sinha. Polynomial learning of distribution families. In *FOCS*, 2010.

Constantinos Boulis and Mari Ostendorf. Combining multiple clustering systems. In *PKDD*, 2004.

S. Charles Brubaker and Santosh Vempala. Isotropic PCA and affine-invariant clustering. *CoRR*, abs/0804.3575, 2008.

David Bryant and Vincent Berry. A structured family of clustering and tree construction methods. *Advances in Applied Mathematics*, 27(4):705–732, 2001.

D. Chaudhuri, B. B. Chaudhuri, and C. A. Murthy. A new split-and-merge clustering technique. *Pattern Recognition Letters*, 13(6):399–409, 1992.

Kamalika Chaudhuri and Sanjoy Dasgupta. Rates of convergence for the cluster tree. In *NIPS*, 2010.

Bo Dai, Baogang Hu, and Gang Niu. Bayesian maximum margin clustering. In *ICDM*, 2010.

Sanjoy Dasgupta. Learning mixtures of gaussians. In *FOCS*, 1999.

Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *ICML*, 2008.

Chris Ding and Xiaofeng He. Cluster merging and splitting in hierarchical clustering algorithms. In *ICDM*, 2002.

Brian Erikkson, Gautam Dasarathy, Aarti Singh, and Robert Nowak. Active clustering: robust and efficient hierarchical clustering using adaptively selected similarities. In *AISTATS*, 2011.

Katherine A. Heller and Zoubin Ghahramani. Bayesian hierarchical clustering. In *ICML*, 2005.

Adam Tauman Kalai, Ankur Moitra, and Gregory Valiant. Efficiently learning mixtures of two Gaussians. In *STOC*, 2010.

Ravindran Kannan, Hadi Salmasian, and Santosh Vempala. The spectral method for general mixture models. In *COLT*, 2005.

Akshay Krishnamurthy, Sivaraman Balakrishnan, Min Xu, and Aarti Singh. Efficient active algorithms for hierarchical clustering. In *ICML*, 2012.

Juho Lee, Suha Kwak, Bohyung Han, and Seungjin Choi. Online video segmentation by bayesian split-merge clustering. In *ECCV*, 2012.

Marina Meilă. Comparing clusterings - an information based distance. *Journal of Multivariate Analysis*, 98(5):873–895, 2007.

Ankur Moitra and Gregory Valiant. Settling the polynomial learnability of mixtures of gaussians. In *FOCS*, 2010.

Feiping Nie, Dong Xu, and Xuelong Li. Initialization independent clustering with actively self-training method. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 42(1):17–27, 2012.

Alessandro Rinaldo and Larry Wasserman. Generalized density clustering. *The Annals of Statistics*, 38(5):2678–2722, 2010.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing and management*, 24(5):513–523, 1988.

Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. Impact of similarity measures on web-page clustering. In *AAAI*, 2000.

Matus Telgarsky and Sanjoy Dasgupta. Agglomerative bregman clustering. In *ICML*, 2012.

Konstantin Voevodski, Maria-Florina Balcan, Heiko Röglin, Shang-Hua Teng, and Yu Xia. Active clustering of biological sequences. *Journal of Machine Learning Research*, 13:203–225, 2012.

Qiaoliang Xiang, Qi Mao, Kian Ming A. Chai, Hai Leong Chieu, Ivor Wai-Hung Tsang, and Zhendong Zhao. A split-merge framework for comparing clusterings. In *ICML*, 2012.

Shi Zhong. Generative model-based document clustering: a comparative study. *Knowledge and Information Systems*, 8:374–384, 2005.

Appendix A. Other feedback models

Besides the cluster split/merge feedback considered here, two other kinds of feedback have also been studied in the literature: *cluster-assignment* feedback (Basu et al., 2002; Nie et al., 2012) and *must-link/cannot-link* feedback (Basu et al., 2004; Ashtiani et al., 2016). The former reveals the ground-truth cluster assignment of a single data point; the latter reveals whether or not a pair of points are in the same ground-truth cluster.

Claim 28 *Cluster-assignment feedback and must-link/cannot link feedback can be converted to cluster split/merge feedback in the unrestricted-merge model.*

Proof We first observe that cluster-assignment feedback can be converted to must-link/cannot-link feedback as follows. For every two known cluster assignments for points x and y s.t. x and y belong to the same ground-truth cluster, output a must-link feedback for (x, y). For every two known cluster assignments for points x and y s.t. x and y belong to different ground-truth clusters, output a cannot-link feedback for (x, y).

We next observe that must-link/cannot-link feedback can be converted to cluster split/merge feedback in the unrestricted merge model as follows. For every must-link feedback for points x and y, if x and y belong to different clusters in the proposed clustering, request to merge these two clusters. For every cannot-link feedback for points x and y, if x and y belong to the same cluster in the proposed clustering, request to split this cluster. Observe that by definition, the output split/merge requests satisfy the assumptions of the unrestricted-merge model. ■

However, the conversion in Claim 28 is sometimes vacuous: cluster split/merge feedback is only output when the cluster-assignment or must-link/cannot-link feedback disagrees with the current clustering. Still, it is reasonable to assume that while the proposed clustering is not equivalent to the ground truth, a constant fraction of such feedback will disagree with the proposed clustering.

Appendix B. Complete experimental results

The following figures show the complete experimental results for all the algorithms. Figure 12 shows the results in the η-merge model. Figure 13 shows the results in the η-merge model for the algorithms in Figure 2 and Figure 4 (for the correlation-clustering objective). Figure 14 shows the results in the unrestricted-merge model.

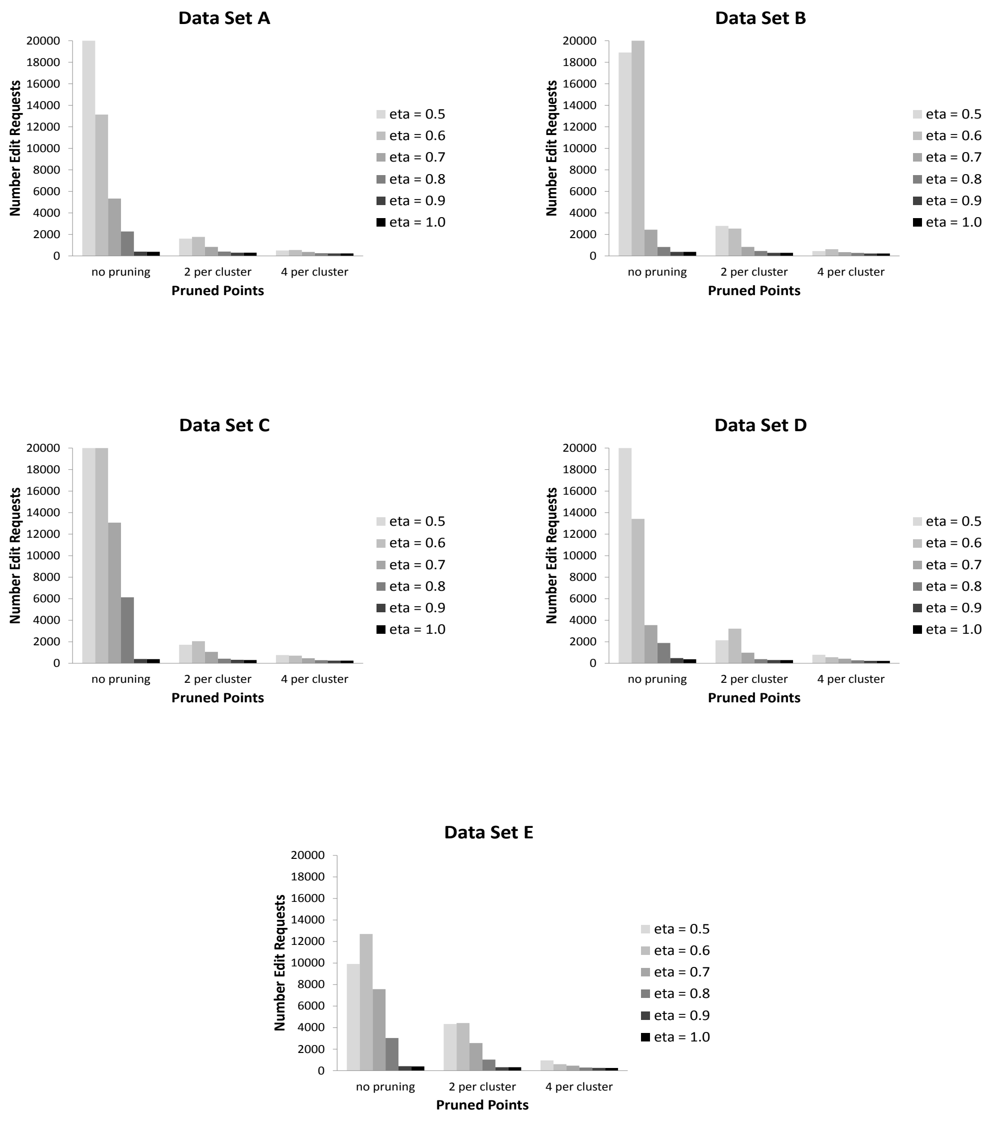


Figure 12: Experimental results in the η-merge model.

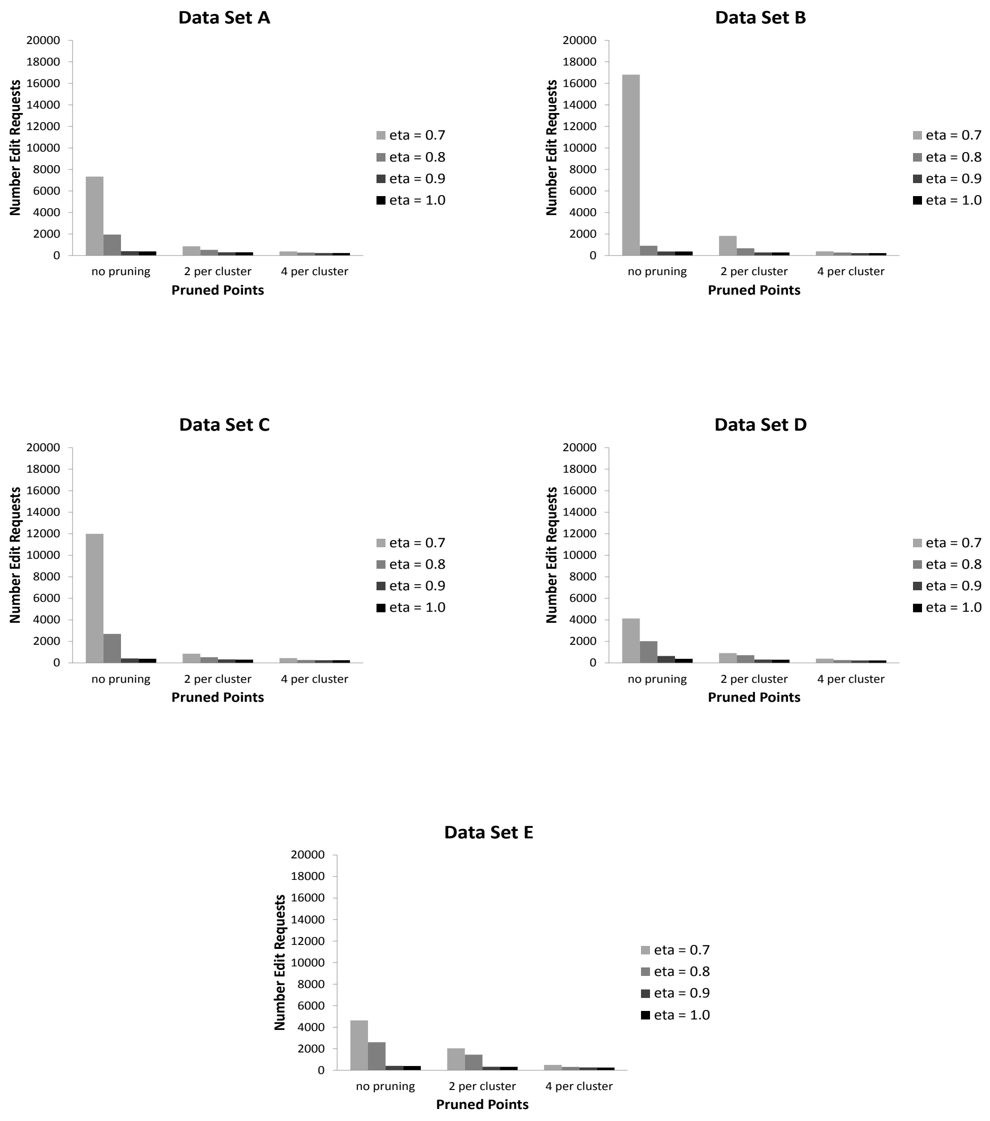


Figure 13: Experimental results in the η-merge model for algorithms for correlation-clustering objective.

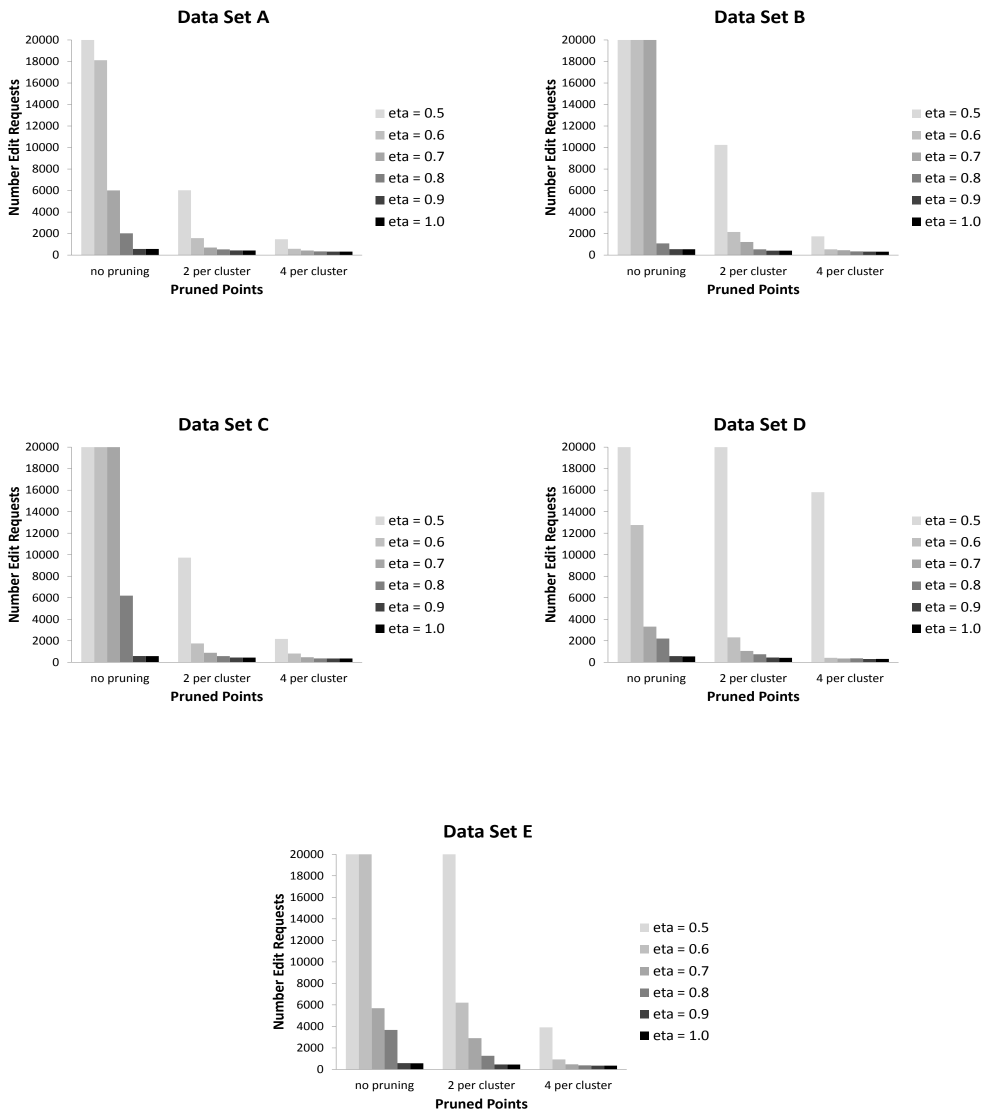


Figure 14: Experimental results in the unrestricted-merge model.

Appendix C. Experiments with robust average-linkage tree

When we investigate instances where our algorithms are unable to find the target clustering, we observe that there are outlier points that are attached near the root of the average-linkage tree, which are incorrectly split off and re-merged by the algorithm without making any progress towards finding the target clustering.

We can address these outliers by constructing the average-linkage tree in a more robust way: first find groups ("blobs") of similar points of some minimum size, compute an average-linkage tree for each group, and then combine these trees using average-linkage. The tree constructed in such fashion may then be used by our algorithms.

We tried this approach, using Algorithm 2 from Balcan and Gupta (2010) to compute the blobs. We find that using the robust average-linkage tree gives better performance for the unpruned data sets, but gives no gains for the pruned data sets, as expected. Figure 15 displays the comparison in the unrestricted merge model for the five unpruned data sets. For the pruned data sets, we expect the robust tree to be very similar to the standard tree, which explains why there is little difference in performance (results not shown).

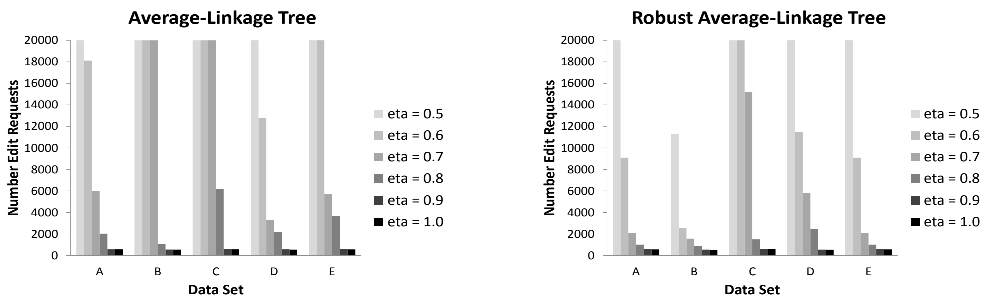


Figure 15: Experimental results in the unrestricted-merge model using a standard versus robust average-linkage tree. Results presented for unpruned data sets.