Modern businesses generate a tremendous amount of data which are stored in large databases. These datasets are then used to make many critical decisions. Therefore, ensuring the quality of these datasets becomes extremely important. As the data accumulates from multiple sources over time, many errors creep into the data. For example, many records end up having duplicate entries. Record de-duplication is a central task in managing large scale databases. The goal is to detect records in a database that correspond to the same real word entity.

The problem of data de-duplication can be viewed as a clustering task. Here, the goal is to put records corresponding to the same physical entity in the same cluster while separating the records corresponding to different entities into different clusters. Clustering for de-duplication has many characteristics which are different from standard clustering problems. Many popular clustering algorithms like k-means or k-median receive as input the value k, that is the number of clusters to output. This information is unknown in de-duplication applications. In any dataset, the number of different-cluster pairs (i.e. different entity pairs) is order of magnitude greater than the number of positive or same-cluster pairs (i.e. same entity pairs). Hence, common machine learning tools of classification prediction (learning a binary classifier over the set of pairs of instances) do not automatically transfer to this domain as the dataset is heavily skewed towards the negative pairs.

The framework of correlation clustering is very natural for modelling the problem of data de-duplication [**?**]. Here, de-duplication is viewed as an optimization problem over graphs. More formally, given a dataset X and a complete graph G over the set. The edges of the graph are labelled 0 or 1. An edge label of zero indicates that the corresponding vertices have been deemed to be in different cluster while an edge label of one indicates that the corresponding vertices should be in the same cluster. The motivation for edge labelling is the following. Often the practitioner can design a pairwise similarity function over the pairs of points. The pairs whose similarity is above a certain threshold are deemed as positive (or same-cluster) and the remaining pairs are deemed to be negative (or different-cluster). Sometimes, the similarity metric is also learned from training data.

Given the graph, the goal of correlation clustering is to find a clustering of the dataset which correlates 'as much as possible' with the given edges. In other words, find a clustering which minimizes the correlation loss w.r.t the given edges. Correlation loss is defined as the sum of the number of zero edges within a cluster plus the number of one edges across different clusters.

However, solving this optimization problem is NP-Hard [**?**].

In this chapter, we offer a formal modelling of such record de-duplication tasks. Our framework is the same as correlation clustering but with the added *promise* that the input graph edges E is 'close to' the optimal correlation clustering of the given dataset. We analyse the computational complexity of this problem and show that even under strong promise, correlation clustering is NP-Hard. Moreover, the problem remains NP-Hard (assuming the ETH hypothesis) even when we are allowed to make queries to a human expert (or an *oracle*) as long as the number of queries is not too large (sub-linear in the number of points in the dataset).

Given these negative results, we introduce the framework of restricted correlation clustering (RCC). This framework has two important differences from the standard (and promise) correlation clustering formulation. Firstly, the optimization problem is restricted over a given class $\mathcal{F}$ of clusterings. Secondly, the goal is to find a clustering which correlates as much as possible with the unknown target (or ground truth) clustering C^*. That is, C^* is the clustering where only records corresponding to same entity are in the same cluster. We offer an algorithmic approach (which uses the help of an oracle) with success guarantees for the restricted version. The 'success guarantee' depends on the complexity of the class $\mathcal{F}$ (measured by VC-Dim($\mathcal{F}$)) as well as the 'closeness' of the distance (or similarity) metric d to the target clustering.

1 Related Work

The most relevant work is the framework of correlation clustering developed by [**?**] that we discussed in the previous section. Other variations of correlation clustering have been considered. For example [**?**], consider a problem where the edges can be labelled by a real number instead of just 0 or 1. Edges with large positive weights encourage those vertices to be in the same cluster while edges with large negative weights encourage those points to be in different clusters. They showed that the problem is NP-Hard and gave a $O(\log n)$ approximation to the weighted correlation clustering problem. [**?**] made several contributions to the correlation clustering problem. For the problem of minimizing the correlation clustering loss (for unweighted complete graphs), they gave an algorithm with factor 4 approximation. They also proved that the minimization problem is APX-Hard.

More recently, [?] considered the problem of correlation clustering in the presence of an oracle. If the number of clusters k is known, they proposed an algorithm which makes $O(k^{14} \log n)$ queries to the oracle and finds a $(1+\epsilon)$-approximation to the correlation clustering problem. They showed that the problem is NP-Hard to approximate with $o\big(\frac{k}{poly \log k}\big)$ queries to an oracle. In this work, we obtain similar results for the promise correlation clustering problem.

Supervision in clustering has been addressed before. For example, [?, ?, ?] considered *link/don't-link* constraints. This is a form of non-interactive clustering where the algorithm gets as input a list of pairs which should be in the same cluster and a list pairs which should be in different clusters. [?] developed a framework of interactive clustering where the supervision is provided in the form of *split/merge* queries. The algorithm gives the current clustering to the oracle. The oracle responds by telling the which clusters to merge and which clusters to split.

In this work, we use the framework of same-cluster queries developed by [?]. At any given instant, the clustering algorithm asks the same-cluster oracle about two points in the dataset. The oracle replies by answering either 'yes' or 'no' depending upon whether the two points lie in the same or different clusters.

On de-duplication side, most prevalent are approaches that are based on designing a similarity measure (or distance) over the records, such that records that are highly similar according to that measure are likely to be duplicates and records that measure as significantly dissimilar are likely to represent different entities. For example, to handle duplicate records created due typographical mistakes, many character-based similarity metrics have been considered. Examples of such metrics include the edit or levenshtein distance [?], smith-waterman distance [?] and jaro distance metric [?]. Token-based similarity metrics try to handle rearrangement of words, for example [?] and [?]. Other techniques include phonetic-based metrics and numerical metrics (to handle numeric data). A nice overview of these methods can be found in [?].

While the above approaches relied on designing a good similarity metric, some works try to 'learn' the distance function from a labelled training dataset of pairs of records. Examples of such works include [?] and [?]. Clustering for de-duplication has been mostly addressed in application oriented works. [?] assumes that the duplicate records are transitive. The clustering problem now reduces to finding the connected components in a graph.

[?] carried out an extensive experimental evaluation of different graph-based clustering algorithms on a simulated dataset of strings.

2 Preliminaries

Given X, a clustering C of the set X partitions it into k disjoint subsets or clusters. The clustering C can also be viewed as a $\{0,1\}$-function over the domain $X^{[2]} := \{(x_1, x_2) : x_1 \neq x_2\}$. Here, $C(x_1, x_2) = 1$ iff x_1, x_2 belong to the same cluster according to C. Similarly, we also view the edges of a graph $G = (X, E)$ as a $\{0,1\}$-function over $X^{[2]}$.

We allow a clustering algorithm to make queries to a human oracle in the following way.

Definition 1 (Same-cluster oracle [?]). *Given a set X and an unknown target clustering C^*. A same-cluster C^*-oracle receives a pair $(x_1, x_2) \in X^{[2]}$ as input and outputs 1 if and only if x_1, x_2 belong to the same cluster according to C^*.*

From the perspective of de-duplication, a same-cluster oracle receives two records x_1 and x_2. The oracle returns 1 if x_1 and x_2 correspond to the same real-world entity. Otherwise, the oracle responds 0.

Definition 2 (Correlation loss[?]). *Given graph $G = (X, E)$ where X is the set of vertices (the given dataset to be clustered) and E is the set of edges. The correlation loss of a clustering C w.r.t the edges E is defined as*

$$\begin{aligned} corrL_E(C) &= corrN_E(C) + corrP_E(C), \text{ where} \\ corrN_E(C) &= |\{(x,y) : C(x,y) = 1 \text{ and } E(x,y) = 0\}|, \\ corrP_E(C) &= |\{(x,y) : C(x,y) = 0 \text{ and } E(x,y) = 1\}| \end{aligned} \tag{1}$$

A weighted version of the loss function places weights of w_1 and w_2 on the two terms and is defined as

$$corrL^w_E(C) = w\, corrN_E(C) + (1-w)\, corrP_E(C) \tag{2}$$

The goal of correlation clustering is to find a clustering which minimizes the (weighted) correlation loss. We also consider the normalized version of this loss function w.r.t a target clustering C^* (rather than the edges E).

Definition 3 (Normalized correlation loss)**.** *Given domain X and a target clustering C^*. The loss of a clustering C w.r.t the target C^* is defined as*

$$\begin{aligned} L_{C^*}(C) = \ \mu \ L_{P^+}(C) \ + \ (1-\mu) \ L_{P^-}(C), \ \textit{where} \\ L_{P^+}(C) = \mathop{\mathbf{P}}_{(x,y)\sim P^+}\big[C(x,y)=0\big], \\ L_{P^-}(C) = \mathop{\mathbf{P}}_{(x,y)\sim P^-}\big[C(x,y)=1\big] \end{aligned} \tag{3}$$

where P^+ is the uniform distribution over $X_+^{[2]} = \{(x,y) : C^*(x,y)=1\}$ and P^- is the uniform distribution over $X_-^{[2]} = \{(x,y) : C^*(x,y)=0\}$.

The normalized correlation loss measures two quantities for the clustering C. The first is the fraction of the true positive pairs that C gets wrong (or loss over the positive pairs). The second is the fraction of true negative pairs that C gets wrong (or the loss over the negative pairs). It then obtains a weighted sum of the two losses.

Lets observe the relation between Defns. 2 and 3. Define $\gamma_0 := \mathbf{P}[C^*(x,y)=1]$, that is the probability of true positive (or same-cluster pairs) in the dataset. Using the notation of Defn. 2, we see that $corrP_{C^*}(C) = \gamma_0 |X^{[2]}| L_{P^+}(C)$ and $corrN_{C^*}(C) = (1-\gamma_0)|X^{[2]}|L_{P^-}(C)$. Normalising by $|X^{[2]}|$ and choosing $\mu = \frac{w_2\gamma_0}{w_1(1-\gamma_0)+w_2\gamma_0}$ gives us the normalized version of the loss function.

Definition 4 (Informative metric [**?**])**.** *Given a metric space (X,d), a target clustering C^* and a parameter λ. We say that the metric d is (α,β)-informative w.r.t C^* and λ if*

$$\mathop{\mathbf{P}}_{(x,y)\sim U^2}\ \big[d(x,y) > \lambda \ \mid \ C^*(x,y)=1\big] \ \leq \ \alpha \tag{4}$$

$$\mathop{\mathbf{P}}_{(x,y)\sim U^2}\ \big[C^*(x,y)=1 \ \mid \ d(x,y)\leq\lambda\big] \ \geq \ \beta \tag{5}$$

Here U^2 is the uniform distribution over $X^{[2]}$.

In deduplication applications, often the distance function is such that pairs with distance within a certain threshold are likely to be in the same cluster. The definition of an informative metric formalizes this intuition. It says that most of the true positive pairs have a distance of atmost λ between them. Also, amongst all pairs with distance $\leq \lambda$, atleast a β fraction of them belong to the same cluster.

Definition 5 (γ-skewed)**.** *Given X and a target clustering C^*. We say that X is γ-skewed w.r.t C^* if*

$$\mathop{\mathbf{P}}_{(x,y)\sim U^2} \left[\; C^*(x,y) = 1 \right] \;\; \leq \;\; \gamma$$

In de-duplication applications, most of the pairs are negative (or belong to different clusters). The above definition states this property formally. In the next section, we introduce our framework of *promise correlation clustering* and discuss the computational complexity of the problem both in the absence and presence of an oracle.

3 Promise Correlation Clustering

Definition 6 (Promise correlation clustering (PCC))**.** *Given a clustering instance $G = (X, E)$. Let C^* be such that*

$$C^* = \arg\min_{C \in \mathcal{F}} \;\; corrL_E(C) \tag{6}$$

where $\mathcal{F}$ is the set of all possible clusterings C such that $m(C) \leq p$. Given that E is (α, β)-informative w.r.t C^*. Find the clustering C^*.

When the edges E correspond to a clustering C then $\beta = 1$ and $\alpha = 0$. We show in the subsequent sections that even for this 'restricted' class of clusterings (when the size of the maximum cluster is atmost a constant M) and given the prior knowledge, PCC is still NP-Hard. Furthermore, PCC is NP-Hard even when we are allowed to make $o(|X|)$ queries to a C^*-oracle.

3.1 PCC is NP-Hard

Theorem 7. *Finding the optimal solution to the Promise Correlation Clustering problem is NP-Hard for all $p \geq 3$ and for $\alpha = 0$ and $\beta = \frac{1}{2}$.*

To prove the result, we will use a reduction from exact cover by 3-sets problem which is known to be NP-Hard.

(X3C) Given a universe of elements $U = \{x_1, \ldots, x_{3q}\}$ and a collections of subsets $S = \{S_1, \ldots, S_m\}$. Each $S_i \subset U$ and contains exactly three elements. Does there exist $S' \subseteq S$ such that each element of U occurs exactly once in S'?

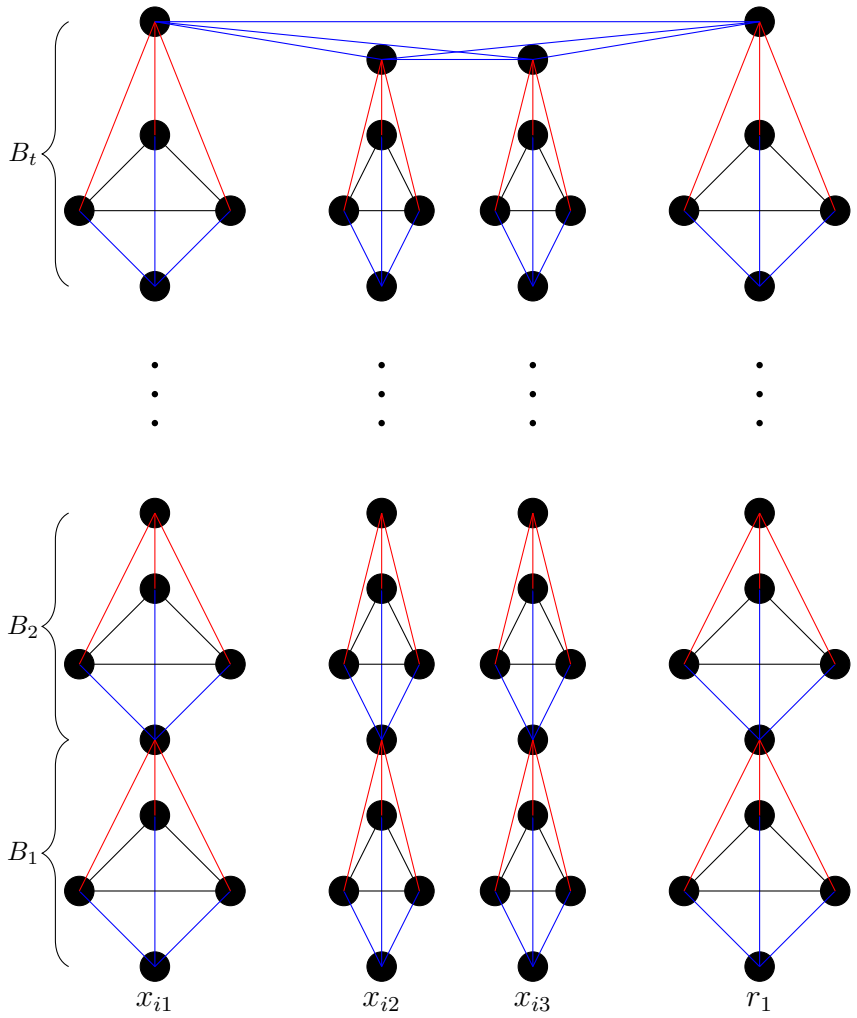

Figure 1: Part of graph G constructed for the subset $S_i = \{x_{i1}, x_{i2}, x_{i3}\}$. The graph is constructed by local replacement when for $p = 4$. If S_i is included in the exact cover then the edges colored black and the edges colored blue represent the corresponding clustering of this part of the graph G. If S_i is not included in the exact cover then the edges colored red and the edges colored black represent the clustering of this part of the graph.

This decision problem is known to be NP-Hard. We will now reduce an instance of X3C to the promise correlation clustering problem. For each three set $S_i = \{x_{i1}, x_{i2}, x_{i3}\}$, we construct a replacement gadget as described

in Fig. 1. The gadget is similar to the one used in the proof of partition into triangles problem. However, instead of triangles the graph is 'made of' cliques of size p.

Given an instance of X3C, we construct $G = (V, E)$ using local replacement described in Fig. 1. Let A be an algorithm which solves the promise problem described in Eqn. 6. Then, we can use this algorithm to decide exact cover by three sets as follows.

If A outputs a clustering C such that all the clusters have size exactly p and E_C makes no negative errors w.r.t E (that is $\mu(E_C) = 0$) then output YES. Otherwise, output NO. Next, we will prove that this procedure decides X3C.

Let there exists an exact cover for the X3C instance. Let C be the clustering corresponding to the exact cover. That is, the edges colored blue and black correspond to this clustering and the corresponding vertices are in the same cluster (Fig. 1). Note that this clustering makes no negative errors. Furthermore, each point is in a cluster of size exactly p. Thus, the positive error corresponding to any vertex is the degree of that vertex minus $p - 1$. Since, the size of a cluster is atmost p, this is the minimum possible positive error for any vertex. Hence, any other clustering strictly makes more positive errors than C.

It is easy to see from the construction that if A finds a clustering which has no negative errors and all the clusters have size p, then this corresponds to exact cover of the X3C instance and hence we output YES. If this does not happen then there does not exist any exact cover for (U, S). This is because if there was an exact cover then the corresponding clustering would satisfy our condition. Thus, A decides X3C. Since, X3C is NP-Hard, no polynomial time algorithm A exists unless $P = NP$.

In the construction, for each clause, we have $p^2t + (p - 3)$ vertices and a vertex for each of the variables. Therefore, $|V| = m(p^2t + (p - 3)) + 3q$ and $|E| = pt(\binom{p}{2} + p - 1) + \binom{p}{2}$. Consider a clustering C which places all the x_i's and r_i's in singleton clusters and places rest of the points in clusters of size p. For $t \geq 2$,

$$\beta = \frac{pt\binom{p}{2}}{pt(\binom{p}{2} + p - 1) + \binom{p}{2}} = \frac{1}{1 + \frac{2}{p} + \frac{1}{pt}} > \frac{1}{2} \text{ and } \alpha = 0$$

3.2 Hardness of PCC in the presence of an oracle

In the previous sections, we have shown that the PCC problem is NP-Hard without queries. It is trivial to see that by making $\beta|X|$ queries to the same-cluster oracle allows us to solve (in polynomial time) the Promise Correlation Clustering problem for all p and $\alpha = 0$. In this section, we prove that the linear dependence on $n = |X|$ is tight. We prove that if the exponential time hypothesis (ETH) holds then any algorithm that runs in polynomial time makes atleast $\Omega(n)$ same-cluster queries.

Theorem 8. *Given that the Exponential Time Hypothesis (ETH) holds then any algorithm for the Promise Correlation Clustering problem that runs in polynomial time makes $\Omega(|X|)$ same-cluster queries for all $p \geq 3$ and for $\alpha = 0$ and $\beta = \frac{1}{2}$.*

Below, we give a proof sketch but a detailed proof is in the supplementary material. The exponential time hypothesis says that any solver for 3-SAT runs in $2^{o(m)}$ time (where m is the number of clauses in the 3-SAT formula). We use a reduction from 3-SAT to 3DM to X3C to show that the exact cover by 3-sets (X3C) problem also can't be solved in $2^{o(m)}$ time (if ETH holds). Then, using the reduction from the previous section implies that PCC also can't be solved in $2^{o(n)}$ time. Thus, any query based algorithm for PCC needs to make atleast $\Omega(n)$ queries where $n = |X|$ is the number of vertices in the graph.

Definition 9 (3-SAT)**.** .
Input: A boolean formulae ϕ in 3CNF with n literals and m clauses. Each clause has exactly three literals.
Output: YES if ϕ is satisfiable, NO otherwise.

Exponential Time Hypothesis
There does not exist an algorithm which decides 3-SAT and runs in $2^{o(m)}$ time.
To prove that (X3C) is NP-Hard, the standard We will reduce 3-SAT to 3-dimensional matching problem. 3DM is already known to be NP-Hard. However, the standard reduction of 3-SAT to 3DM constructs a set with number of matchings in $\Theta(m^2n^2)$. Hence, using the standard reduction, the exponential time hypothesis would imply there does not exist an algorithm for 3DM which runs in $\Omega(m^{\frac{1}{4}})$. Our reduction is based on the standard

reduction. However, we make some clever optimizations especially in the way we encode the clauses. This improves the lower bound to $\Omega(m)$.

Using the above result, we immediately get an $\Omega(2^m)$ lower bound on the run-time of X3C. Now, using the same reduction of X3C to PCC as in Section **??**, gives the same lower bound of $\Omega(n)$ on the running time of PCC.

For the sake of contradiction, let us assume that there exists an algorithm which solves PCC in polynomial time by making $o(n)$ same-cluster queries (n is the number of vertices). Then by simulating all possible answers for the oracle, we get a non-query algorithm which solves PCC in $2^{o(n)}$. Hence, no such query algorithm exists.