Scalable and Globally Optimal Generalized L_1 *k*-center Clustering via Constraint Generation in Mixed Integer Linear Programming

Aravinth Chembu, [1] **Scott Sanner,** [1*] **Hassan Khurram,** [1] **Akshat Kumar** [2]

[1] Department of Mechanical and Industrial Engineering, University of Toronto, Toronto
[2] School of Computing and Information Systems, Singapore Management University, Singapore
(aravinth.chembu, hassan.khurram)@mail.utoronto.ca, ssanner@mie.utoronto.ca, akshatkumar@smu.edu.sg

Abstract

The *k*-center clustering algorithm, introduced over 35 years ago, is known to be robust to class imbalance prevalent in many clustering problems and has various applications such as data summarization, document clustering, and facility location determination. Unfortunately, existing *k*-center algorithms provide highly suboptimal solutions that can limit their practical application, reproducibility, and clustering quality. In this paper, we provide a novel scalable and globally optimal solution to a popular variant of the *k*-center problem known as generalized L_1 *k*-center clustering that uses L_1 distance and allows the selection of arbitrary vectors as cluster centers. We show that this clustering objective can be reduced to a mixed-integer linear program (MILP) that facilitates *globally optimal* clustering solutions. However, solving such a MILP may be intractable for large datasets; to remedy this, we present a *scalable* algorithm that leverages constraint generation to efficiently and provably converge to its global optimum. We further enhance outlier handling through a simple but elegant extension to our MILP objective. We first evaluate our algorithm on a variety of synthetic datasets to better understand its properties and then validate on 20 real benchmark datasets where we compare its performance to both traditional L_1 distance *k*-center and *k*-medians baselines. Our results demonstrate significant suboptimality of existing algorithms in comparison to our approach and further demonstrate that we can find *optimal* generalized L_1 *k*-center clustering solutions up to an unprecedented 1,000,000 data points.

1 Introduction

Clustering is a classical data analysis tool extensively used in problems in unsupervised learning. It is found ubiquitously in a varied range of machine learning and data mining tasks, including pattern recognition, document clustering, image segmentation, medical and social sciences (Jain, Murty, and Flynn 1999; Xu and Wunsch 2005; Xu and Tian 2015; Jain 2010). Clustering is often formulated as an optimization problem to aggregate *similar* data observations into clusters; formalizing this notion of similarity with different objective functions leads to different clustering algorithms.

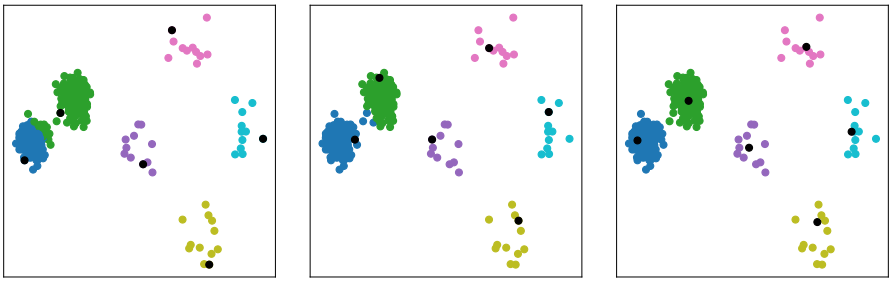
***k*-center clustering** One of the most traditionally used center-based clustering problems is the *k*-center problem. It is an essential and fundamental task in exemplar-based clustering (Kaufman and Rousseeuw 2009) and has been a well-studied formulation for metric clustering (Gonzalez 1985) for over three decades. The key idea in the vertex *k*-center problem is to choose a subset of k points from the data observations as the respective centers of k clusters such that each point is assigned to its closest center. The objective of the problem is to minimize the maximum distance from any point to its nearest cluster center. The generalized variant of the *k*-centers problem relaxes the centers to be arbitrary vectors (Calik and Tansel 2013; Xu, Peng, and Xu 2018).

The k-center clustering problem has many real-world applications. Recently, *k*-center clustering was used for the unsupervised learning task of detecting suspected spammers in a dataset of Amazon reviews (Zhong, Tan, and Qu 2020). Bateni et al. (2021) discuss the application of k-center clustering in spam and fraud detection. Further use cases of *k*-centers for online advertisement and document search are also discussed in the literature (Wang et al. 2009).

We note that the *k*-centers problem is inherently immune to cluster imbalance, e.g., when we have both dense and sparse clusters in the data (cf. Figure 1). This is a result of the objective minimizing the within-cluster worst-case distance (or radius) (Bateni et al. 2021), making it *insensitive* to the number of points that lie within this radius. This contrasts with the minimum sum-of-squared error clustering (MSSC) objective where the within-cluster sum of distances is considered (Fränti and Sieranoja 2018). In fact, imbalance in cluster sizes is a common occurrence in many real-world use cases for clustering, including medical diagnosis (Lin et al. 2017), financial fraud, and bioinformatics (Liang et al. 2012) and hence *k*-centers is appropriate for such settings.

The vertex k-center problem is known to be NP-Hard in the metric space and cannot be solved in polynomial time within an approximation factor of less than 2 unless P = NP (Gonzalez 1985). Some exact algorithms can be found in the literature as discussed in Section 2. However, they are mostly restricted to the vertex *k*-center problem and do not extend to its generalized variant. Moreover, most algorithms do not scale beyond 1,000's of data points (cf. Section 2). On the other hand, there exist several polynomial time heuristic greedy algorithms that match the best possible 2-approximation bound (Gonzalez 1985; Hochbaum and Shmoys 1985) that we discuss further in Section 2 and

[*] Affiliate to Vector Institute, Toronto, Canada.



(a) k-center Gon (b) k-center HS (c) Our optimal algorithm

Figure 1: A motivating example comparing k-center clustering with Gonzalez (1985) (Gon) and Hochbaum and Shmoys (1985) (HS) algorithm, and our proposed optimal clustering for imbalanced clusters. The Gon and HS algorithms choose boundary points of clusters as centers while our optimal algorithm results in perfect clustering with the L_1 $k-$center objective.

compare to experimentally in this work. However, these approaches are sensitive to initialization and do not provably converge to a globally optimal solution, hence risking poor clustering and lack of consistency in results.

Proposed contributions In this work, we address these shortcomings by designing a scalable and optimal algorithm for the generalized k-center clustering problem. As mentioned, the minimax objective of the k-center problem is immune to the class imbalance in the data and consequently finds many applications in the real world. Hence, we choose the generalized k-center objective for our clustering model. However, unlike most k-center problems, we use the L_1 distance metric in our objective instead of the commonly used Euclidean metric, similar to Bajpai et al. (2021) and Angelidakis et al. (2022). We use the L_1 distance metric to exploit its well-known robustness property over the Euclidean distance (Celebi, Kingravi, and Vela 2013); this partially alleviates the issue of the objective's sensitivity to outliers that is a consequence of minimizing the within-cluster worst-case distance (Charikar et al. 2001; Malkomes et al. 2015).

Our major contributions are as following:

- We show the generalized L_1 k-center objective can be elegantly reduced to a MILP, facilitating its optimal solution. As solving the MILP may be intractable for large datasets, we present a *scalable* algorithm that leverages a constraint generation technique to efficiently and optimally solve the MILP within a practically feasible time.
- Furthermore, we enhance outlier handling and account for the objective's sensitivity to outliers through a clever constraint modification step to our MILP and constraint generation algorithm. We demonstrate our model's ability to identify ground-truth clusters in the presence of outliers through experiments with synthetic data.
- We then demonstrate the efficacy of our algorithm through rigorous simulation studies and experiments with 20 real benchmark datasets. Through our experiments, we demonstrate significant suboptimality of existing algorithms in comparison to our approach and further demonstrate that we can find optimal generalized L_1 k-center clustering solutions up to 1,000,000 data points.

2 Related Work

Greedy Algorithms It has been shown that the best possible polynomial time algorithm for the vertex k-center problem is a 2-approximation algorithm unless P = NP (Gonzalez 1985). Gonzalez (1985) and Hochbaum and Shmoys (1985) are the most popular greedy algorithms with this best possible approximation bound. The Gonzalez (1985) algorithm (kc-Gon) is $O(kn)$ time complexity algorithm that selects the first center at random and then chooses the farthest point from the previously selected centers as the next center. In contrast, the other popular algorithm (kc-HS) (Hochbaum and Shmoys 1985) has a time complexity of $O(kn \log n)$ and takes its formal characterization from the dominating set problem in square graphs.

Researchers over the years have come up with other polynomial time heuristic algorithms for the vertex k-center problem that try to provide better empirical results (Robič and Mihelič 2005; Garcia-Diaz et al. 2017; Rana and Garg 2008). However, none of these approaches provably converge to a globally optimal solution. Nonetheless, among the greedy algorithms, kc-Gon and kc-HS algorithms remain popular as standard baselines due to their efficiency, simplicity, and the best possible 2-approximation guarantee.

Optimal algorithms Exact algorithms for the metric k-center problem can also be found in the literature. In one of the earliest works, a set-covering based approach was used to obtain the optimal solution for the problem (Minieka 1970). In a seminal work, Daskin (2000) iteratively solves several maximal covering sub-problems with a binary search procedure over possible solution values. For each radius value from this search procedure, the total number of points covered within the radius is maximized while the number of centers (or facilities open) is restricted to k. Further, Elloumi, Labbé, and Pochet (2004) proposed a new integer programming formulation for the set-covering problem. They perform a binary search on the ordered list of distinct distances between their proposed tighter lower and upper bounds and solve a set covering problem for each selected distance. It was the first optimal algorithm to solve a problem with 1817 nodes. Calik and Tansel (2013) improve on the above formulation to provide tighter bounds and solve

problems up to 3038 nodes in the data. More recently, a general constrained-clustering based clustering algorithm (Dao, Duong, and Vrain 2017) was proposed to solve for up to 5000 samples in the dataset. Other recent works scale for larger datasets (Shi et al. 2022; Aloise and Contardo 2018). However, most of these exact solutions were proposed for the vertex *k*-center problem and are not applicable to its generalized variant. Moreover, they are not practically feasible for very large datasets.

3 Optimal Clustering via L_1 *k*-center

3.1 L_1 *k*-center problem as a MILP

As introduced previously, the *k*-center problem objective minimizes the maximum radius around all the cluster centers. Consider that we have n observations $\boldsymbol{x_i}$ with d dimensions in the dataset $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ where $i \in N = \{1, ..., n\}$. The goal in hard-partitioning clustering is to assign each of the n observations to one of the k clusters C_j where $j \in K = \{1, ..., k\}$. Let $\boldsymbol{z_j}$ represent the centroid for C_j such that we minimize the following objective for the L_1 *k*-center problem:

$$\min_{\boldsymbol{z}} \max_{j \in K} \max_{x_i \in C_j} \|\boldsymbol{x_i} - \boldsymbol{z_j}\|_1 \tag{1}$$

We can rewrite the above objective as a minimax mixed integer linear program (MILP) using binary indicator variables c_{ij} that identify cluster assignments for all the observations. If a point i is associated with cluster C_j, then we have $c_{ij} = 1$; $c_{ij} = 0$, otherwise. With this definition, we can have a minimax MILP (but not pure MILP) formulation in problem (2)

$$\begin{aligned} &\min_{\boldsymbol{c}, \boldsymbol{z}} \max_{j \in K} \max_{i \in N} \|\boldsymbol{x_i} - \boldsymbol{z_j}\|_1 c_{ij} \\ &\text{s.t.} \sum_{j \in K} c_{ij} = 1,\ i \in N;\ \ z_{j,1} < z_{j+1,1},\ j \in K \backslash \{k\}; \\ &\qquad c_{ij} \in \{0, 1\},\ \ i \in N,\ j \in K;\ \ \boldsymbol{z_j} \in \mathbb{R}^d,\ \ j \in K \end{aligned} \tag{2}$$

The product of binary variables c_{ij} with the distance term in the above objective ensures that we only account for distances from any observation to its corresponding center. Moreover, the first set of constraints guarantee that every observation is associated with exactly one cluster. The second set of constraints are symmetry-breaking constraints that enforce that the k centers be found in increasing order of its value along the first dimension. This makes sure that the other permutations of point-cluster assignments are removed from the feasible search space. The above minmax saddle point optimization problem (2) can then be rewritten as a bi-level MILP problem (3).

$$\begin{aligned} &\min_{\boldsymbol{c}, \boldsymbol{z}, \varepsilon} \ \varepsilon \\ &\text{s.t.}\ \ \varepsilon = \max_{j \in K} \max_{i \in N} \|\boldsymbol{x_i} - \boldsymbol{z_j}\|_1 c_{ij}; \\ &\qquad \sum_{j \in K} c_{ij} = 1,\ i \in N;\ \ z_{j,1} < z_{j+1,1},\ j \in K \backslash \{k\}; \\ &\qquad c_{ij} \in \{0, 1\},\ \ i \in N,\ j \in K;\ \ \boldsymbol{z_j} \in \mathbb{R}^d,\ \ j \in K \end{aligned} \tag{3}$$

We achieve this transformation to a bi-level optimization problem by introducing an additional variable ε. We can further relax the equality in the first constraint of Problem (3) to $\varepsilon \geq \max_j \ \max_i \|\boldsymbol{x_i} - \boldsymbol{z_j}\|_1 c_{ij}$, which preserves optimality since the minimization of ε guarantees that the constraint will be satisfied at equality. This realization permits one further serendipitous transformation that allows us to *remove* the second level of the bi-level problem. Specifically, we can further reduce this bi-level optimization problem to a pure MILP formulation by introducing $n \times k$ logical constraints that guarantee $\{\varepsilon \geq \|\boldsymbol{x_i} - \boldsymbol{z_j}\|_1 c_{ij}\}$ for all i and j and hence also the maximum over all i and j as required by the first constraint ($\varepsilon = \max_{j \in K} \ \max_{i \in N} \|\boldsymbol{x_i} - \boldsymbol{z_j}\|_1 c_{ij}$) in Problem (3). This leads us to the final form of our *pure* MILP clustering problem (4):

$$\begin{aligned} &\min_{\boldsymbol{c}, \boldsymbol{z}, \varepsilon} \ \varepsilon \\ &\text{s.t.}\ \ c_{ij} = 1 \implies \varepsilon \geq \|\boldsymbol{x_i} - \boldsymbol{z_j}\|_1\ \ i \in N,\ j \in K; \\ &\qquad \sum_{j \in K} c_{ij} = 1,\ i \in N;\ \ z_{j,1} < z_{j+1,1},\ j \in K \backslash \{k\}; \\ &\qquad c_{ij} \in \{0, 1\},\ \ i \in N,\ j \in K;\ \ \boldsymbol{z_j} \in \mathbb{R}^d,\ \ j \in K \end{aligned} \tag{4}$$

3.2 Constraint generation for scalable clustering

The above transformation gave us an elegant MILP in problem (4) that can be solved optimally with standard solvers; however, such a procedure would be computationally prohibitive, largely due to the size of this MILP. We have $n \times k$ integer variables c_{ij} and logical constraints in this formulation. Unfortunately, it would be practically infeasible to solve the problem optimally for large datasets, and it would seem like we have made little progress with this reformulation. However, the key observation here is that we can potentially scale if we reduce the number of variables and constraints while keeping the structure of the problem intact.

A crucial insight is that the objective function depends only on a single variable ε, whose value is governed by the logical constraints in the MILP. As mentioned previously, ε essentially holds the value of the maximum distance from any point to its corresponding cluster center. This means that all points close to an optimal center do not impact the objective value and hence may be ignored during optimization. This is especially useful since clusters often tend to have a denser collection of points around the center, which is reflected in the implicit Gaussian data distribution assumption underlying *k*-means clustering (Fränti and Sieranoja 2018). This insight can be leveraged to design an efficient constraint generation methodology that omits logical constraints corresponding to points interior to the cluster boundaries.

We formally describe our constraint generation optimization methodology we call kc-Opt through Algorithm 1 and illustrate this approach with a synthetic example in Figure 2. We broadly perform two operations iteratively, typical of the main and sub-problem setting. We begin with a small set of variables c_{ij} and constraints C corresponding to points in I. We warm start the MILP with a strong upper bound by initializing the cluster centers $\boldsymbol{z_j}$. We then solve the MILP (with constraints in C) to obtain the current best solution ε^*

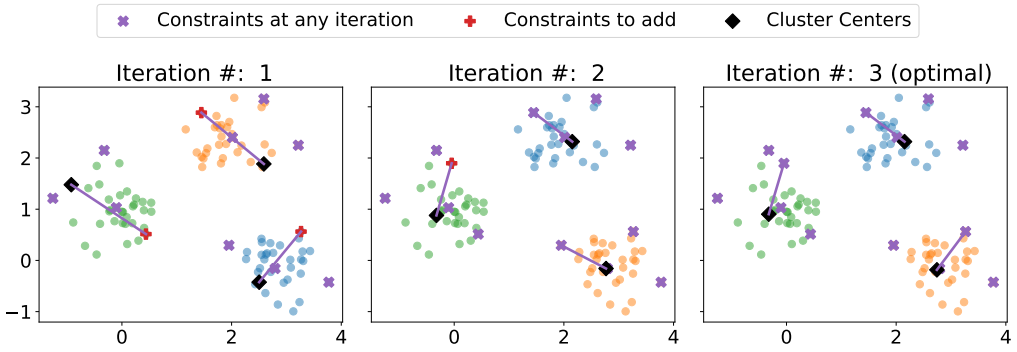


Figure 2: We show the constraint generation process using a synthetic example where we start (left) with 9 constraints (denoted with $\times$), add 3 constraints (denoted with $+$) in the first iteration corresponding to the extreme points from the centers (left), 1 constraint in second iteration (middle), and none in third iteration where we reach the optimal solution (right).

Algorithm 1: kc-Opt algorithm

Input Data $\boldsymbol{x_i}$, and k

1: $\varepsilon^* \leftarrow 0, \quad \hat{\varepsilon} \leftarrow \infty$
2: $I \neq \emptyset, \quad C \neq \emptyset, \quad \boldsymbol{z}_j^* \leftarrow$ Warm-start
▷ *Initial constraints C for points in I*
3: $\varepsilon^*, \boldsymbol{z}_j^*, c_{ij}^* \leftarrow$ Solve MILP subject to C
4: $\hat{c_{ij}} \leftarrow \{\mathbb{1}_{j=\hat{j}} | \hat{j} = \arg\min_j \|\boldsymbol{x_i} - \boldsymbol{z_j^*}\|_1\}$,
▷ *Assign points $i \in N$ to their closest center*
5: $\hat{\varepsilon}, I, C \leftarrow$ Add-Constraints()
6: **if** $\hat{\varepsilon} > \varepsilon^*$ **then**
7: **go to** line 3
▷ *Re-solve MILP with augmented constraints set C and points in I*
8: **end if**
9: **return** $\varepsilon^*, \boldsymbol{z}_j^*, \hat{c_{ij}}$ ▷ *Optimal solution*

Algorithm 2: Add-Constraints subroutine

Input Data $\boldsymbol{x_i}, \boldsymbol{z_j^*}, \hat{c_{ij}}, I, C$

1: $\hat{\varepsilon} = \max_{i \in N, \ j \in K} \{\|\boldsymbol{x_i} - \boldsymbol{z_j^*}\|_1 \hat{c_{ij}}\}$
▷ *Check whether current ε^* represents the max distance from points to their center*
2: **if** $\hat{\varepsilon} > \varepsilon^*$ **then**
3: **for** $j \in K$ **do**
4: $I_{add} \leftarrow \{\arg\max_i \|\boldsymbol{x_i} - \boldsymbol{z_j^*}\|_1 \hat{c_{ij}}\}$
▷ *Add farthest point in clusters to I*
5: $I \leftarrow I \ \cup \ I_{add}$
6: **end for**
7: $C \leftarrow C \cup \{c_{ij}{=}1 \implies \varepsilon \geq \|\boldsymbol{x_i} - \boldsymbol{z_j}\|_1, i \in I_{add}, j \in K\}$
8: **end if**
9: **return** $\hat{\varepsilon}, I, C$

(line 3 in Algorithm 1). Using the current centers z_j^*, we explicitly assign all points to their closest centers and capture these assignments in binary variables $\hat{c_{ij}}$ (line 4 in Algorithm 1). This is an important step because the point-cluster binary variables c_{ij}^* are only available for points $i \in I \subset N$. We then call subroutine Add-Constraints shown in Alg. 2.

In Algorithm 2, we evaluate the maximum distance from the points to their centers $\hat{\varepsilon}$ and compare it with ε^*. If found greater than ε^*, it is clear that the solution is not optimal and that we need to add lower bounding constraints corresponding to points that are farther than ε^* distance from their centers. We identify the farthest points per cluster, which are equivalent to identifying the most violated constraint, and add them to the constraint set C (line 7 in Algorithm 2). We then re-solve the MILP with the additional constraints.

We stop this iterative process when $\hat{\varepsilon} = \varepsilon^*$, i.e., when none of the points in the dataset incur a larger error than ε^* (right plot in Figure 2). At this point of termination, we are guaranteed to have the optimal solution since adding further constraints could only maintain or increase the objective value; we know that the current solution satisfies *all* constraints at the *current* objective value, hence maintaining it and implying it is optimal w.r.t. all constraints. This optimality guarantee is formalized in the following proposition, where we establish that it is sufficient to use Algorithm 1 to obtain the optimal solution for the MILP problem (4):

Proposition 1. *The MILP formulation in problem* (4) *and kc-Opt algorithm achieve the same objective value and optimal solution variable assignment.*

We provide a full proof for Proposition 1 in the Appendix. The equivalence of objective values follows from this proof and the established provably convergent nature of constraint generation methods in the case of finite constraints. We also remark that the symmetry breaking constraints of the MILP (4) ensure solution uniqueness.

As a concrete example of constraint generation, in Figure 2 (left), three points that were furthest from the centers (marked as $+$) gave a $\hat{\varepsilon} = \max\{2.32, 1.75, 2.14\}$ which was greater than $\varepsilon^* = 1.27$ at the first iteration; hence we added the nine constraints corresponding to these three points for each of the three clusters. For example, for cluster 1 where the 5th data point $[0.44, 0.51]^T$ is most violating, we add the constraint $\{c_{5,1}{=}1 \implies \varepsilon \geq \left\| \begin{bmatrix} 0.44 \\ 0.51 \end{bmatrix} - \boldsymbol{z_1} \right\|_1\}$. In the exam-

ple of Figure 2, we continued the constraint generation for two more iterations after which we had no more violated constraints and hence we reach the optimal solution shown in Figure 2 (right).

We conclude our discussion by observing that the algorithm will provably always terminate and converge to the optimal solution in finite time since we are bounded by the finite number of points for which we can generate constraints. The choice of L_1 distance in the objective facilitates solving it as a MILP, which is usually computationally less expensive than solving MIQCPs with an L2 objective (Leyffer 2001). We empirically show in Section 4.1 that the number of constraints we need to generate is *significantly* smaller than the number of observations and generally increases linearly with the number of clusters k and dimensions d. Fortuitously, we also empirically demonstrate in Section 4.1 that the number of constraints generated in some standard clustering settings appears to increase very minimally as dataset size n increases by orders of magnitude.

3.3 Extensions to the algorithm to handle outliers

The L_1 objective of k-center problem provides some robustness to outliers in comparison to Euclidean distance metric. However, L_1 k-center problem can still be sensitive to outliers since it accounts for the worst-case distance between points and centers (Malkomes et al. 2015; Charikar et al. 2001). A common strategy to deal with outliers in k-clustering is to discard l number of observations during clustering (Malkomes et al. 2015; Charikar et al. 2001; Chawla and Gionis 2013), where l is a hyperparameter. Specifically, in the k-center case, it has been shown that the best possible algorithm is a 3-approximation algorithm (k, z-center) (Charikar et al. 2001).

We use the above strategy to handle outliers in our model. We believe that having a tunable hyperparameter l is a clear and systematic approach to clustering. Fortunately, this approach can be integrated into our MILP to obtain a *guaranteed optimal solution* for excluding the l worst outliers. We have the following modified MILP:

$$
\begin{aligned}
\min_{\boldsymbol{c},\boldsymbol{z},\varepsilon} \quad & \varepsilon \\
\text{s.t.} \quad & c_{ij} = 1 \implies \varepsilon \geq \|\boldsymbol{x_i} - \boldsymbol{z_j}\|_1,\ i \in N,\ j \in K\ ; \\
& \sum_{i \in N} \sum_{j \in K} c_{ij} = n - l;\ \sum_{j \in K} c_{ij} \leq 1,\ i \in N \\
& z_{j,1} < z_{j+1,1},\ \ j \in K \setminus \{k\} \\
& c_{ij} \in \{0,1\},\ i \in N,\ j \in K;\ \ \boldsymbol{z_j} \in \mathbb{R}^d,\ j \in K
\end{aligned}
\tag{5}
$$

In problem (5), the constraints ensure that exactly l points are not assigned to any of the k clusters. We accommodate the above MILP in our constraint generation scheme by calling the Add-Constraints-Outliers subroutine presented in Algorithm 3 at line 5 in Algorithm 1. We refer to this algorithm as kc-OptOut throughout the paper. The subroutine provides an updated set of points in I for which we generate constraints C such that it also includes the l farthest points (outliers). Additionally, here the value of $\hat{\varepsilon}$ is computed as the maximum distance from cluster centers to all non-outlier points (line 3). Furthermore, we provide a strategy to estimate the value of hyperparameter l in the Appendix.

Algorithm 3: Add-Constraints-Outliers subroutine

Input Data $\boldsymbol{x_i}, \boldsymbol{z_j^*}, \hat{c_{ij}}, I, C$

1: $I_{out} \leftarrow \{\arg\max_{A \subset N} \sum_{i \in A} (\|\boldsymbol{x_i} - \boldsymbol{z_j^*}\|_1 \hat{c_{ij}})\}$
▷ *Where $|A| = l$ to get l farthest (outlier) points*
2: $I \leftarrow I \ \cup \ I_{out}$
3: $\hat{\varepsilon} = \max_{i \in N \setminus I_{out},\ j \in K} \{\|\boldsymbol{x_i} - \boldsymbol{z_j^*}\|_1 \hat{c_{ij}}\}$
4: **if** $\hat{\varepsilon} > \varepsilon^*$ **then**
5: **for** $j \in K$ **do**
6: $I_{add} \leftarrow \{\arg\max_{i \in N \setminus I_{out}} \|\boldsymbol{x_i} - \boldsymbol{z_j^*}\|_1 \hat{c_{ij}}\}$
7: $I \leftarrow I \ \cup \ I_{add}$
8: **end for**
9: $C \leftarrow \{c_{ij} = 1 \implies \varepsilon \geq \|\boldsymbol{x_i} - \boldsymbol{z_j}\|_1, i \in I, j \in K\}$
10: **end if**
11: **return** $\hat{\varepsilon}, I_{out}, I, C$

4 Empirical performance

We analyze the performance of our approach on an array of synthetic datasets and real data benchmarks used to evaluate competing peer algorithms. We empirically compare our algorithm with two popular *k*-center baselines: Gonzalez (1985) (kc-Gon) and Hochbaum and Shmoys (1985) (kc-HS) algorithms. Both provide the best possible 2 approximation solution guarantees. Since the Gonzalez (1985) algorithm is sensitive to initialization, we also report the mean results from 10 independent runs (kc-GonAvg). We note that we use the L_1 objective variants for the above *k*-center algorithms to provide a relevant comparison. We also compare to the highly popular state-of-the-art initialization strategy (Arthur and Vassilvitskii 2007) with *k*-medians (kmed) from the pyclustering package (Novikov 2019) as an additional popular baseline that uses L_1 distance. Although *k*-medians does *not* optimize the *k*-centers objective, it provides insight into how much an alternate L_1 clustering objective will comparatively impact the clustering results.

We present mean results from 10 independent runs. Since our algorithm (kc-Opt) is optimal and deterministic, we report this single value. As a scalable and optimal *k*-centers algorithm is our *key contribution* in this work, we primarily focus on comparing the optimal value of the *k*-centers objective obtained from our kc-Opt algorithm relative to that of the other baselines as datasets and their sizes are varied.

All code is available to reproduce results in this paper.[1] All experiments were conducted on a Macbook-Air laptop (8-core CPU at 3.2 GHz and 8 GB memory). Furthermore, we employed the commercially available Gurobi solver (Gurobi Optimization 2021) to execute our MILP.

4.1 Synthetic Experiments

We consider a series of experiments with simulated data to examine the complexity of our approach and benchmark its performance against the competing popular algorithms.

[1] https://github.com/Aravinthck/Optimal-KCenters

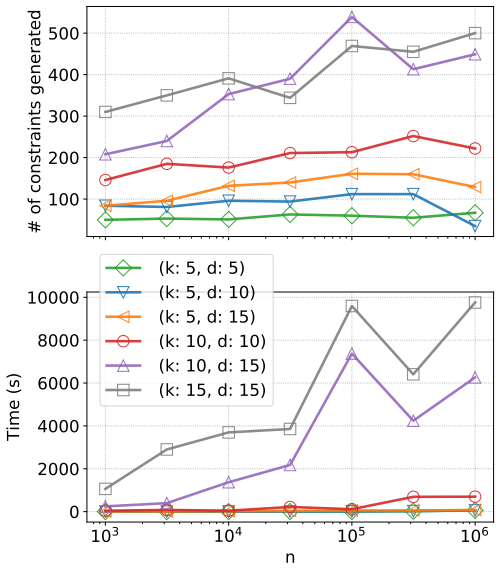
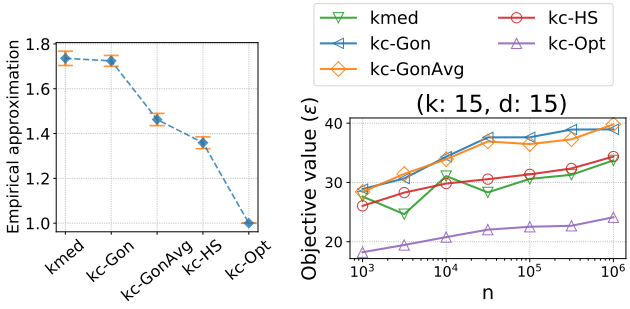


Figure 3: The performance of our kc-Opt algorithm for different values of n, k, and d with the *Norm* dataset.



(a) Approximation factor with 95% confidence interval around mean

(b) Objective values for baselines

Figure 4: Comparison of the objective values from baselines with the optimal value from our algorithm on *Norm* dataset.

Algorithm complexity The first set of experiments was aimed at understanding the computational efficiency of the constraint generation procedure of kc-Opt. The theoretical worst-case complexity of our kc-Opt algorithm is trivially bounded by the worst-case exponential running times for solving the MILP at every iteration of the constraint generation procedure. However, modern MILP solvers are typically efficient when the number of constraints is limited and we show empirically that the algorithm converges to the optimal solution in relatively few iterations after adding constraints for a small fraction of points (and in practically feasible time). Specifically, we show that the maximum number of constraints we had was ≈ 450 for the case with 1,000,000 data points distributed in 15 clusters (cf. Figure 3).

Well-separated clusters We constructed synthetic datasets (called *Norm*) similar to (Arthur and Vassilvitskii 2007; Chakraborty et al. 2020; Xu and Lange 2019), with $k \in \{5, 10, 15\}$ centers chosen uniformly at random from a $d \in \{5, 10, 15\}$ dimensional hypercube of side 500. The number of observations n was varied from 1,000 to 1,000,000. These points were drawn from normal distributions with a standard deviation of 1 around the k centers to generate well-separated Gaussian clusters.

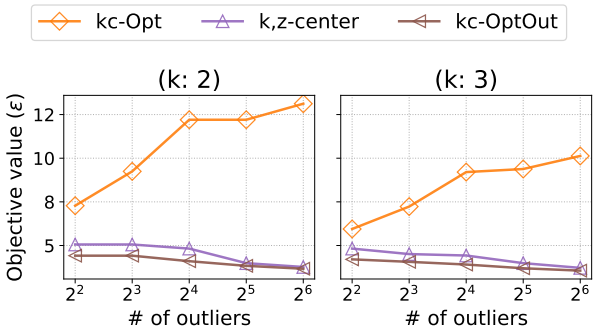


Figure 5: Comparison of outlier-aware k,z-center with our optimal kc-OptOut on *Norm-Out* dataset; both are parameterized by the number of outliers to remove (x-axis).

We first studied the number of constraints (data points) generated as well as the optimization time to understand our algorithm's ability to scale. We report the values we found for 6 different k, d combinations for the *Norm* dataset in Figure 3 to examine the effect of varying the n, k and d values. We ran the experiments to a maximum optimality gap of 5% to provide a bound on the obtained solution.

It is evident from Figure 3 that the number of constraints we had to add increased very minimally as we increased the number of observations n for a specific combination of k, d. Furthermore, we notice that the number of constraints we need at the final solution scales linearly with k. We also report the time taken to run these experiments in Figure 3. Overall, this study illustrates our model's ability to scale to large datasets and obtain fast convergence irrespective of the number of points n, especially when k is small.

All models recovered the ground truth clusters on this controlled synthetic data. The farthest first traversal nature of the k-center baselines and careful seeding of k-medians contribute to this success for the baselines. However, we saw that the objective value for baselines was not close to the optimal value obtained with our algorithm but was within an empirical approximation factor of 2. In Figure 4a, we report the average ratio of the objective values obtained from the baselines with our optimal solution across all n, k, and d values for the *Norm* dataset. We also report the actual objective values obtained for the case of $d = 15, k = 15$ in Figure 4b for more clarity. We provide plots for the remaining k, d combinations in the Appendix, along with a similar analysis with non-isotropic Gaussian clusters.

Clustering with outliers We now test the robustness of our outlier-aware algorithm (kc-OptOut). Similar to the previous two simulation studies, we constructed synthetic datasets (*Norm-Out*) with Gaussian clusters in two-

Table 1: Comparative evaluation of objective values for our kc-Opt (optimal ε by definition, shown in bold) with baseline methods on 20 datasets shown in the columns and ordered by number of data points (n) and split over two tables for readability.

Model	Iris	Wine	Seeds	Newthyroid	Vertebral	Ecoli	Balance	Australian	Blood	Mammographic
# data (n)	150	178	210	215	310	336	625	690	748	830
# dim (d)	4	13	7	5	6	7	4	14	4	5
kmed	3.7	473.1	7.3	78.1	464.5	1.1	7.4	31249.2	8240.4	57.2
kc-Gon	3.4	462.7	7.6	63.1	223.2	1.1	8.0	49925.7	4269.0	49.0
kc-GonAvg	3.6	397.0	8.4	69.4	226.2	1.1	8.0	50117.1	4919.3	52.9
kc-HS	3.1	312.6	7.1	66.4	196.3	1.0	7.0	49048.4	4279.0	54.0
kc-Opt (ε)	**2.3**	**255.6**	**5.1**	**43.3**	**145.7**	**0.8**	**6.5**	**25064.3**	**3024.0**	**42.0**

Model	Banknote	Winequality	Banana	Wallrobot	Pendigits	Census	Shuttle	Codrna	Sepsis	Skinsegment
# data (n)	1372	4898	5300	5456	10992	48842	58000	59535	129392	245057
# dim (d)	4	11	2	4	16	14	9	8	3	3
kmed	30.4	426.7	4.4	5.3	604.3	1237579.8	9480.8	1358.4	44.2	394.5
kc-Gon	24.9	131.5	4.6	4.6	531.0	732152.0	5085.0	680.0	47.0	395.0
kc-GonAvg	29.3	120.9	4.6	4.9	527.8	658588.3	4846.5	791.7	41.3	389.8
kc-HS	23.3	103.5	4.4	3.8	522.0	606000.0	4928.0	653.5	38.0	343.0
kc-Opt (ε)	**18.3**	**74.9**	**3.4**	**3.1**	**517.5**	**391747.0**	**4318.5**	**482.9**	**27.5**	**318.5**

dimensions (square of size 20) with $n = 1000$ and $k \in \{2, 3\}$. Following the design of experiments in (Chawla and Gionis 2013; Gupta et al. 2017), we introduced outliers in the data by sampling l points uniformly at random in the same two-dimensional space. We inspect the optimal values ε obtained from kc-OptOut and the *k,z*-center baseline (Charikar et al. 2001) as we increase the number of outliers added to the data. From Figure 5, we see that ε for the baseline is higher that than the optimal value from kc-OptOut as expected. However, it has been noted in the literature that the *k,z*-center algorithm's ability to perform well comes as the cost of scalability. Malkomes et al. (2015) claim that it can take *k,z*-center algorithm $\approx$ 100 hours to run for a dataset with 45,000 points.

In the absence of outlier handling, i.e., when we ran our experiments just with the kc-Opt algorithm for *Norm-Out* dataset, the optimal value was found to be much higher (as seen in Figure 5). These high values with kc-Opt can be attributed to the random nature in which we introduce outliers throughout the two-dimensional box of size 20.

4.2 Real dataset experiments

We now move on from our controlled synthetic evaluations to benchmark our kc-Opt model on 20 real datasets from the UCI (Dua and Graff 2017), Keel (Alcalá-Fdez et al. 2011), and LibSVM (Chang and Lin 2011) machine learning repositories that are commonly used clustering benchmarks (Malkomes et al. 2015; Chakraborty et al. 2020). We follow standard procedure (Malkomes et al. 2015; Kleindessner, Awasthi, and Morgenstern 2019; Bateni et al. 2021) and report objective values ε for the baselines and compare them with our optimal algorithm (kc-Opt) in Table 1. We present the optimal value ε obtained from kc-Opt. This provides an understanding of how well kc-Opt performs compared to existing *k*-center methods as well as *k*-medians. We do not compare the results from the outlier-aware models since leaving out l outliers will always result in a lower *k*-center objective value and thus would not provide a fair objective comparison to the algorithms presented here. Full descriptive information regarding the datasets and comparative running time of all methods is provided in the Appendix.

From Table 1, it is evident that the optimal objective value ε obtained from the kc-Opt algorithm is much lower than all baselines. We note that the difference in the objective values of the baselines and kc-Opt gets larger as we increase the number of data points n (exceptions are Pendigits and Shuttle datasets). This performance gap improvement as we increase the size of the dataset again highlights kc-Opt's ability to optimally scale to large datasets.

5 Conclusion and discussion

We presented the generalized L_1 *k*-center clustering objective and an outlier-aware variant that both reduce to a mixed integer linear program (MILP). We showed how to leverage a constraint generation methodology to efficiently achieve *globally optimal* results for large datasets up to an unprecedented 1,000,000 data points. We further demonstrated the ability of the kc-Opt algorithm to practically scale by running experiments with 20 real-world benchmark datasets and also note that the relative performance improvement over baselines increased with the size of the datasets. As with all clustering algorithms, inherent bias in the data and its attributes along with potential misinterpretation of results may have detrimental effects and thus all clustering results must be handled cautiously. Nonetheless, our contribution represents a novel optimization methodology for L_1 *k*-center clustering that performs well with imbalanced clusters that are prevalent in a range of modern data exploration tasks critical to both society and industry.

References

Alcalá-Fdez, J.; Fernández, A.; Luengo, J.; Derrac, J.; and García, S. 2011. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *J. Multiple Valued Log. Soft Comput.*, 17: 255–287.

Aloise, D.; and Contardo, C. 2018. A sampling-based exact algorithm for the solution of the minimax diameter clustering problem. *Journal of Global Optimization*, 71(3): 613–630.

Angelidakis, H.; Kurpisz, A.; Sering, L.; and Zenklusen, R. 2022. Fair and Fast k-Center Clustering for Data Summarization. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; and Sabato, S., eds., *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 669–702. PMLR.

Arthur, D.; and Vassilvitskii, S. 2007. k-Means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, 1027–1035. USA: Society for Industrial and Applied Mathematics. ISBN 9780898716245.

Bajpai, T.; Chakrabarty, D.; Chekuri, C.; and Negahbani, M. 2021. Revisiting Priority k-Center: Fairness and Outliers. In Bansal, N.; Merelli, E.; and Worrell, J., eds., *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 21:1–21:20. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-195-5.

Bateni, M.; Esfandiari, H.; Fischer, M.; and Mirrokni, V. 2021. Extreme K-Center Clustering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5): 3941–3949.

Calik, H.; and Tansel, B. C. 2013. Double Bound Method for Solving the P-Center Location Problem. *Comput. Oper. Res.*, 40(12): 2991–2999.

Celebi, M. E.; Kingravi, H. A.; and Vela, P. A. 2013. A Comparative Study of Efficient Initialization Methods for the k-means Clustering Algorithm. *Expert Systems with Applications*, 40(1): 200–210.

Chakraborty, S.; Paul, D.; Das, S.; and Xu, J. 2020. Entropy Weighted Power k-Means Clustering. In Chiappa, S.; and Calandra, R., eds., *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, 691–701. PMLR.

Chang, C.-C.; and Lin, C.-J. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.*, 2(3).

Charikar, M.; Khuller, S.; Mount, D. M.; and Narasimhan, G. 2001. Algorithms for Facility Location Problems with Outliers. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, 642–651. USA: Society for Industrial and Applied Mathematics. ISBN 0898714907.

Chawla, S.; and Gionis, A. 2013. k-means–: A Unified Approach to Clustering and Outlier Detection. In *Proceedings of the 2013 SIAM International Conference on Data Mining (SDM)*, 189–197.

Dao, T.-B.-H.; Duong, K.-C.; and Vrain, C. 2017. Constrained Clustering by Constraint Programming. *Artificial Intelligence*, 244: 70–94. Combining Constraint Solving with Mining and Learning.

Daskin, M. S. 2000. A New Approach To Solving the Vertex P-center Problem to Optimality: Algorithm and Computational Results. *Communications of the Operations Research Society of Japan*, 45(9): 428–436.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository.

Elloumi, S.; Labbé, M.; and Pochet, Y. 2004. A New Formulation and Resolution Method for the P-Center Problem. *INFORMS Journal on Computing*, 16(1): 84–94.

Fränti, P.; and Sieranoja, S. 2018. K-Means Properties on Six Clustering Benchmark Datasets. *Applied Intelligence*, 48(12): 4743–4759.

Garcia-Diaz, J.; Sanchez-Hernandez, J.; Menchaca-Mendez, R.; and Menchaca-Mendez, R. 2017. When a Worse Approximation Factor Gives Better Performance: A 3-Approximation Algorithm for the Vertex K-Center Problem. *Journal of Heuristics*, 23(5): 349–366.

Gonzalez, T. F. 1985. Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science*, 38: 293–306.

Gupta, S.; Kumar, R.; Lu, K.; Moseley, B.; and Vassilvitskii, S. 2017. Local Search Methods for K-Means with Outliers. *Proc. VLDB Endow.*, 10(7): 757–768.

Gurobi Optimization, L. 2021. Gurobi Optimizer Reference Manual.

Hochbaum, D. S.; and Shmoys, D. B. 1985. A Best Possible Heuristic for the k-Center Problem. *Mathematics of Operations Research*, 10(2): 180–184.

Jain, A. K. 2010. Data Clustering: 50 Years Beyond K-means. *Pattern Recognition Letters*, 31(8): 651–666. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR).

Jain, A. K.; Murty, M. N.; and Flynn, P. J. 1999. Data Clustering: A Review. *ACM Comput. Surv.*, 31(3): 264–323.

Kaufman, L.; and Rousseeuw, P. J. 2009. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons.

Kleindessner, M.; Awasthi, P.; and Morgenstern, J. 2019. Fair k-Center Clustering for Data Summarization. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 3448–3457. PMLR.

Leyffer, S. 2001. *Generalized outer approximationGeneralized Outer Approximation*, 787–794. Boston, MA: Springer US. ISBN 978-0-306-48332-5.

Liang, J.; Bai, L.; Dang, C.; and Cao, F. 2012. The K-Means-Type Algorithms Versus Imbalanced Data Distributions. *IEEE Transactions on Fuzzy Systems*, 20(4): 728–745.

Lin, W.-C.; Tsai, C.-F.; Hu, Y.-H.; and Jhang, J.-S. 2017. Clustering-Based Undersampling in Class-Imbalanced Data. *Information Sciences*, 409-410: 17–26.

Malkomes, G.; Kusner, M. J.; Chen, W.; Weinberger, K. Q.; and Moseley, B. 2015. Fast Distributed k-Center Clustering with Outliers on Massive Data. In Cortes, C.; Lawrence, N.; Lee, D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 28.

Minieka, E. 1970. The m-Center Problem. *SIAM Review*, 12(1): 138–139.

Novikov, A. 2019. PyClustering: Data Mining Library. *Journal of Open Source Software*, 4(36): 1230.

Rana, R.; and Garg, D. 2008. The Analytical Study of K-Center Problem Solving Techniques. *Int. J. Inf. Technol. Knowl. Manag*, 1(2): 527–535.

Robič, B.; and Mihelič, J. 2005. Solving the K-Center Problem Efficiently With a Dominating Set Algorithm. *Journal of computing and information technology*, 13(3): 225–234.

Shi, M.; Hua, K.; Ren, J.; and Cao, Y. 2022. Global Optimization of K-Center Clustering. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; and Sabato, S., eds., *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 19956–19966. PMLR.

Wang, H.; Liang, Y.; Fu, L.; Xue, G.-R.; and Yu, Y. 2009. Efficient Query Expansion for Advertisement Search. SIGIR '09, 51–58. New York, NY, USA: Association for Computing Machinery. ISBN 9781605584836.

Xu, D.; and Tian, Y. 2015. A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*, 2: 165–193.

Xu, J.; and Lange, K. 2019. Power k-Means Clustering. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 6921–6931. PMLR.

Xu, R.; and Wunsch, D. 2005. Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks*, 16(3): 645–678.

Xu, Y.; Peng, J.; and Xu, Y. 2018. The Mixed Center Location Problem. *J. Comb. Optim.*, 36(4): 1128–1144.

Zhong, M.; Tan, L.; and Qu, X. 2020. Identification of Opinion Spammers Using Reviewer Reputation and Clustering Analysis. *International Journal of Computers Communications & Control*, 14(6): 759–772.

A Proof of the theoretical result in Section 3.2

In this section, we restate the theoretical result in Section 3.2 and provide the proof.

Proposition 1. *The MILP formulation in problem* (4) *and kc-Opt algorithm achieve the same objective value and optimal solution variable assignment.*

Proof. Let us assume that we have generated constraints in C^* for the candidate points whose indexes were in I^* with the constraint generation methodology described in Algorithm 1. Also, assume that the current solution ε^* corresponds to the optimal objective value for the overall problem defined in problem (4). Here, ε^* is the optimal solution obtained with constraints C^* of the form $\{c_{ij} = 1 \implies \varepsilon \geq \|\boldsymbol{x_i} - \boldsymbol{z_j}\|_1,\ i \in I^*,\ j \in K\}$, where $I^* \subseteq N$. Note that we implicitly assume that constraints of the form $\{\sum_{j=1}^{k} c_{ij} = 1,\ \ i \in I\}$ and $\{z_{j,1} < z_{j+1,1},\ \ j \in K \setminus k\}$ as seen in problem (4) are also being added at each iteration of constraint generation to complete the formulation.

To prove that ε^* is the optimal solution when we have constraints for all points in N, it is sufficient to show that ε^* satisfies the indicator constraints $\{c_{ij} = 1 \implies \varepsilon^* \geq \|\boldsymbol{x_i} - \boldsymbol{z_j^*}\|_1,\ j \in K\}$ for the remaining points $i \in N \setminus I^*$. In other words, we simply need to show that $\{\varepsilon^* \geq \|\boldsymbol{x_i} - \boldsymbol{z_j^*}\|_1\ c_{ij}\}$ for $i \in N \setminus I^*$.

In Algorithm 1, we have two additional variables $\hat{\varepsilon}$ and $\hat{c_{ij}}$. Here, variables $\hat{c_{ij}}$ defined as $\{\hat{c_{ij}} \leftarrow \{\mathbb{1}_{j=\hat{j}} | \hat{j} = \arg\min_j \|\boldsymbol{x_i} - \boldsymbol{z_j^*}\|_1\},\ i \in N\}$ ensure that all points are assigned to their closest cluster. Also, $\hat{\varepsilon}$ is defined as $\hat{\varepsilon} = \max_{i \in N,\ j \in K}\{\|\boldsymbol{x_i} - \boldsymbol{z_j^*}\|_1 \hat{c_{ij}}\}$ and is guaranteed to take the value of the maximum distance from any point to its cluster center. This directly implies that $\{\hat{\varepsilon} \geq \|\boldsymbol{x_i} - \boldsymbol{z_j^*}\|_1\ \hat{c_{ij}}\}$ for all values of $i \in N$. With this definition, we also have $\hat{c_{ij}} = c_{ij}^*$ for $i \in I^*$ because the optimality (with constraints C^* and points I^*) is achieved only when points in I^* are assigned to their closest centers.

Since we exit the constraint generation iterations only when $\hat{\varepsilon} = \varepsilon^*$, from the above definition of $\hat{\varepsilon}$ we directly have $\{\varepsilon^* \geq \|\boldsymbol{x_i} - \boldsymbol{z_j^*}\|_1\ \hat{c_{ij}}\}$ for all values of $i \in N$, and not just $i \in N \setminus I^*$. Hence, we show that we achieve the above sufficient condition for ε^* obtained from problem 1 to be the optimal value for the original formulation in problem (4). □

B Hyperparameter search methodology for Section 3.3

In the formulation in problem (5), determining a value for the hyperparameter l is a complex problem and remains an open research question (Chawla and Gionis 2013). To this end, we prescribe a search-based hyperparameter tuning method based on an internal cluster evaluation criterion.

In the presence of outliers, the centers identified with the standard L_1 k-center objective without outlier handling would be skewed towards the outlier points and away from the true cluster centers since it optimizes for the worst-case distance from centers. This suggests that the sum-of-squared error (SSE) computed for all points with these centers would be much higher than the ground truth SSE. As we treat for outliers by increasing the value of l in problem (5), we expect these centers to move closer to the true centers and the SSE to drop. Further, if values of l larger than the actual number of outliers were to be used, we expect the centers not to shift considerably. This is because we would exclude the furthest points of the ground truth clusters and expect SSE to remain flat. This indicates the possibility of an elbow curve for SSE as we increase the number of outliers. This is illustrated with a synthetic dataset in Figure 6 where the optimal value ε (as defined in problem (5)) and SSE both drop as we treat for outliers.

We leverage this critical insight to prescribe a fast search methodology that efficiently identifies the elbow of the SSE curve if it exists. We initially start with $l = 0$ and increase its value with a pre-decided step size (α). We then compute SSE at every step and the local slope of the curve between each pair of points. We continue this process until we notice a sudden drop in the slope value (up to a tolerance) compared with the slope from the previous step. This could be indicative of the elbow in the SSE curve. This is illustrated in Figure 6 where a step size of 2 was used (points marked as $\times$), and the actual elbow point was identified using this prescribed methodology.

We formalize the above prescribed slope-based methodology to search for this elbow in the SSE curve and find the best value for the hyperparameter l here with the following pseudocode presented in Algorithm 4.

In the Outliers-Search subroutine shown in Algorithm 4, we primarily search for the value of l at which we notice a sharp change in the value of the slope when compared to

Algorithm 4: Outliers-Search subroutine

Input $\alpha, max\_outliers, tol$

$u \leftarrow 0,\ \ v \leftarrow u + \alpha$
$l \leftarrow 0,\ \ pre\_slope \leftarrow 0$
▷ *Initialize the variables for search*
$SSE_u \leftarrow$ get_SSE$(l = u)$
▷ *Find SSE after running the kc-OptOut algorithm*
while $v \leq max\_outliers$ **do**
  $SSE_v \leftarrow$ get_SSE$(l = v)$
  $slope = (SSE_u - SSE_v)/\alpha$
  if $|slope| < tol * prev\_slope$ **then**
    $l = u$
▷ *Found the elbow in the SSE curve*
    break
  else
    $u = v$
    $SSE_u = SSE_v$
    $prev\_slope = slope$
    $v = u + \alpha$
▷ *Continue the search with updated value for v*
  end if
end while
return l

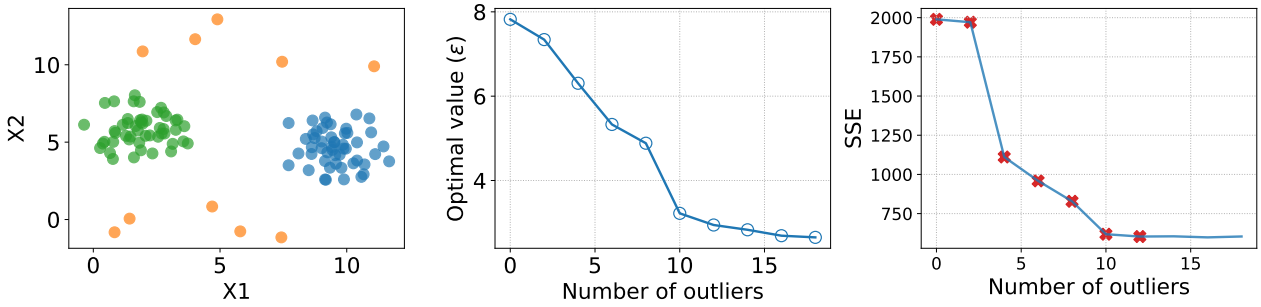


Figure 6: With the illustrative example (left) for clustering with outliers, we note the drop in optimal value ε (middle) and elbow curve for SSE (right) as we increase hyperparameter l. We also show a typical search procedure (right) involved in finding this l that determines the elbow is at $l = 12$.

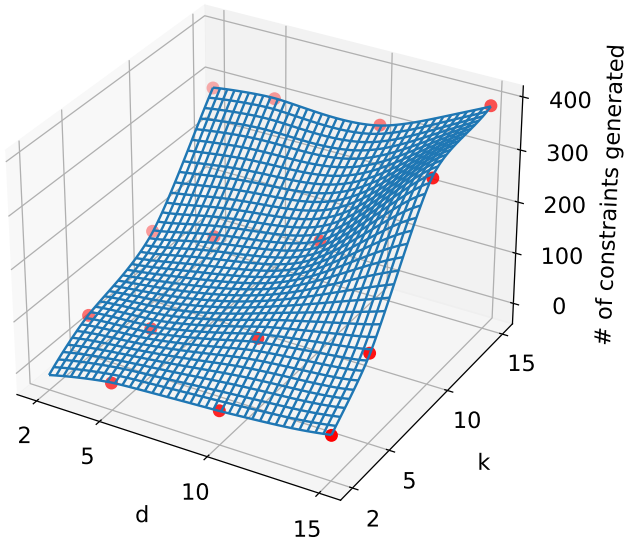


Figure 7: The number of constraints we had to add in the kc-Opt algorithm as we increase both k and d with the *Norm* dataset for a fixed $n = 10,000$.

the previous slope value as we increase the number of outliers. We initially start from $l = 0$ and use variables u and v to indicate two consecutive outlier values separated by a predecided step size α. We use get_SSE(l) function to return the value of SSE after running the kc-OptOut algorithm with l outliers. We compute SSE at l equals u and v and find the slope of the SSE curve between these points u and v. We then compare the current value of the slope with that from the previous step (with a predefined tolerance value tol) to observe any sudden change in slope.

When the ratio of the current slope to the previous slope is lower than the threshold defined by tol, i.e., when a sudden change in the slope between the two pairs of u and v is observed, we stop the search and assign the final value of l as u (from the recent step); otherwise, we continue the search by updating the value of both u and v by adding the value of α to them. We continue this search until the value of v reaches the maximum search value for the number of outliers $max\_outliers$. We decide this value heuristically by factoring in the number of points in the actual data and the number of points that are farther than six standard deviations from the population mean of the point to cluster center distances, where centers are taken from a greedy clustering solution.

C Additional analysis with synthetic datasets

C.1 Additional complexity analysis with well-separated clusters using *Norm* dataset

In Section 4.1, we provided a thorough analysis of the effectiveness of our constraint generation optimization methodology as we increased the number of observations n with different combinations of k, d. It was observed that the number of constraints we had to add increased very minimally as we increased the number of observations n for a specific combi-

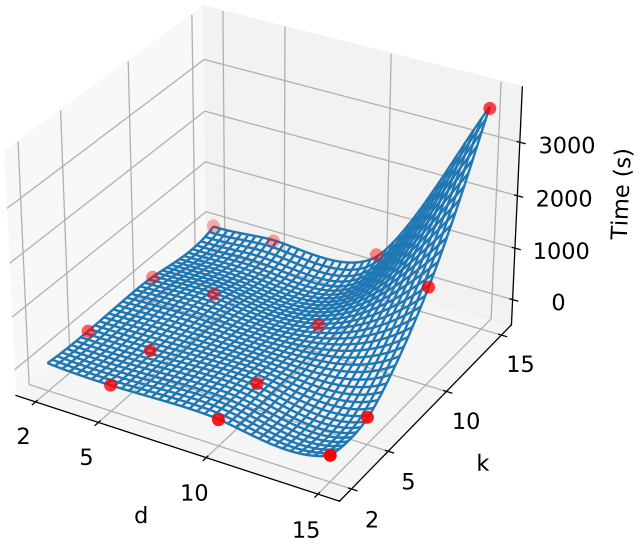


Figure 8: Total time taken to run the kc-Opt algorithm as we increase both k and d with the *Norm* dataset for a fixed $n = 10,000$.

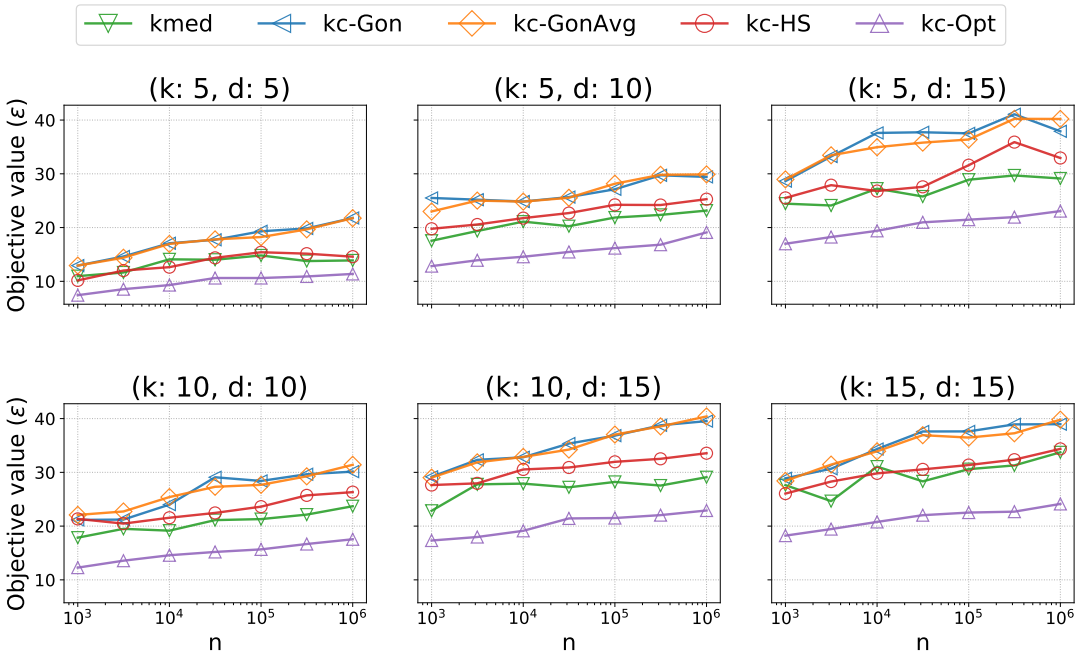


Figure 9: Comparison of the objective values from baselines with the optimal value from our algorithm on *Norm* dataset.

nation of k, d. Moreover, we discussed that increasing k and d increases the complexity of the problem, and extremely high values of k can potentially limit the model's ability to scale.

Here, we show additional plots with the *Norm* synthetic dataset to more clearly examine the effect of increasing k and d as we fix the number of observations $n = 10000$. In Figure 7, we show the number of points for which we had to generate constraints when we varied the values of k and d. It is evident from the plot that the number of constraints increases linearly with both dimensions d and the number of clusters k. Furthermore, in Figure 8, we also show the time it took to run the experiments. We can see from this plot that the time taken increases exponentially when k increases, which is expected behavior for an optimal MILP solution although still within reasonable time bounds for practical use

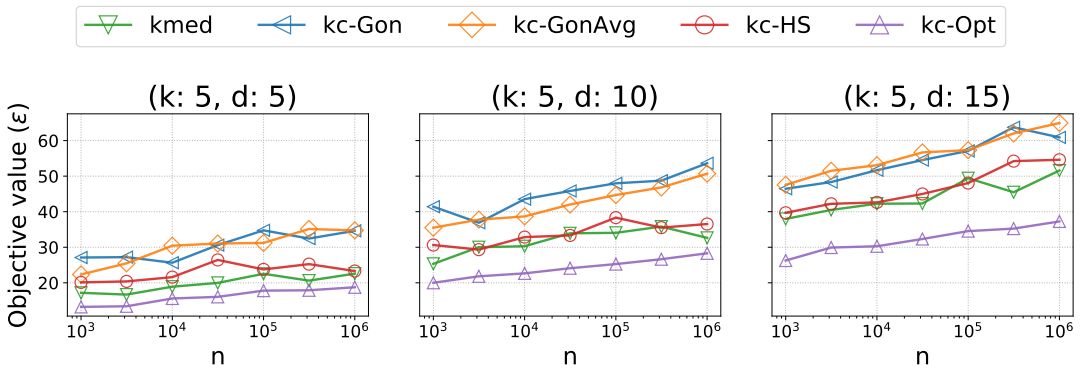


Figure 10: Comparison of the objective values from baselines with the optimal value from our algorithm on the *Norm-Noniso* dataset.

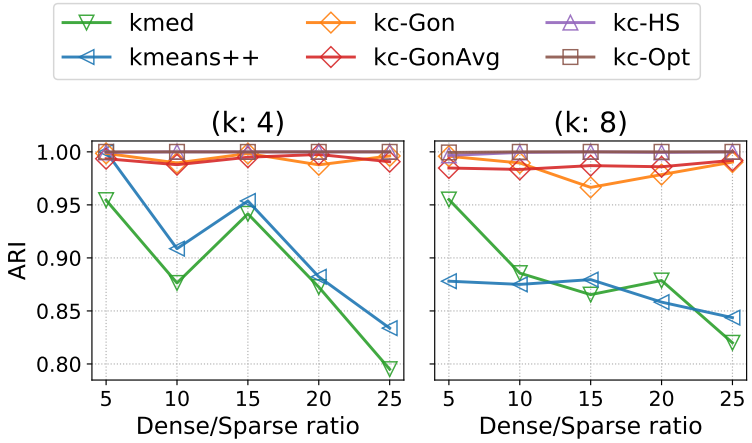


Figure 11: Comparison of the kc-Opt algorithm with baselines based on ARI for imbalanced (*Norm-Imb* dataset) clusters.

for the sizes evaluated here.

C.2 Comparison of baselines and kc-Opt optimal algorithm for *Norm* dataset

In Section 4.1, we compared the objective values obtained from the baseline algorithms and our optimal algorithm kc-Opt for a specific value of k, d with the *Norm* dataset. Here, we supplement the analysis with plots for all combinations of k, d. We report the results in Figure 9 where it is evident that the kc-Opt algorithm, which is optimal by definition, has the lowest objective value for all the different combinations of n, k, and d. This again confirms our claim that we are scalable and perform optimally as expected for datasets with up to 1,000,000 data observations.

C.3 Experiments with non-isotropic Gaussian clusters

We also conducted experiments with non-isotropic Gaussian clusters, which were created to observe whether having non-spherical clusters would affect the performance of the kc-Opt algorithm. The dataset we constructed (called *Norm-Noniso*) was similar to the synthetic examples containing ellipsoid-shaped Gaussian clusters in Liu et al. (2016); Lu and Wan (2012). The points were drawn from normal distributions with the standard deviations for the second half of dimensions set to twice that of the first half of dimensions. For example, with $d = 5$, the standard deviations were $\sigma = 1$ along the first two axes, and $\sigma = 2$ along the next 3 axes. These datasets were constructed using $k = 5$ centers chosen uniformly at random from a d dimensional hypercube with side 500 and $d \in \{5, 10, 15\}$, while n was varied from $1,000$ to $1,000,000$ observations. The plots in Figure 10 reaffirm that our optimal algorithm performed best with the consistently lowest objective value as n varied across all three configurations of k and d considered. Moreover, we observe that the objective value gap between the kc-Opt algorithm and the baselines increases as we increase the number of dimensions in the data.

C.4 Experiments with cluster imbalance

To study imbalanced clusters, we generated Gaussian clusters (called *Norm-Imb*) with $n = 5,000$, $d = 2$, and $k \in \{4, 8\}$ where one-half of the clusters were designed to be dense and one-half sparse with the dense/sparse ratio

Table 2: Description and source of real datasets along with the run-times (s) for the baselines and the kc-Opt algorithm.

data	n	d	k	Data source	kmed++	kc-Gon	kc-GonAvg	kc-HS	Opt-Cg
Iris	150	4	3	UCI	0.00	0.02	0.01	0.01	0.62
Wine	178	13	3	UCI	0.01	0.01	0.01	0.01	4.40
Seeds	210	7	3	UCI	0.00	0.01	0.01	0.02	3.77
Newthyroid	215	5	3	Keel	0.00	0.01	0.01	0.02	0.58
Vertebral	310	6	3	UCI	0.01	0.02	0.02	0.03	1.65
Ecoli	336	7	8	UCI	0.01	0.02	0.02	0.03	2636.84
balance	625	4	3	UCI	0.01	0.04	0.03	0.05	7812.59
Australian	690	14	2	LibSVM	0.01	0.04	0.04	0.09	0.11
blood	748	4	2	UCI	0.01	0.04	0.04	0.08	0.27
Mammographic	830	5	2	UCI	0.02	0.06	0.04	0.09	0.18
Banknote	1372	4	2	UCI	0.03	0.07	0.07	0.28	0.80
Winequality	4898	11	7	UCI	0.22	0.26	0.25	3.08	7742.57
banana	5300	2	2	Keel	0.10	0.28	0.28	2.86	0.56
Wallrobot	5456	4	4	UCI	0.12	0.29	0.28	2.87	96.33
Pendigits	10992	16	10	UCI	0.67	0.60	0.59	9.76	1208.26
census	48842	14	2	UCI	2.05	2.62	2.62	14.30	3.89
shuttle	58000	9	7	UCI	2.29	3.20	3.17	293.36	1845.94
Codrna	59535	8	2	LibSVM	1.16	3.15	3.17	503.72	3.83
sepsis	129392	3	2	UCI	1.87	6.82	6.90	14.53	7.35
Skinsegment	245057	3	2	UCI	4.33	12.92	13.17	20.64	15.12

specified as a ratio of the total number of points in the dense and sparse clusters. We compare the performance of the baselines with our algorithm for values of dense/sparse ratio ranging in $[5, 25]$. It is evident from the ARI values (Hubert and Arabie 1985) in Figure 11 that the kc-Opt optimal algorithm identified the true clusters (ARI equals 1) while other baselines fail in some cases. Moreover, it is very evident from the plots that k-means++ (Arthur and Vassilvitskii 2007) and k-medians algorithm algorithms struggle to find the true ground truth clusters. This is because when we have dense and sparse clusters in the data, *k*-means type clustering, which relies on within-cluster sum-of-distances (and not within cluster worst case distance), often produces clusters with a similar number of points in them. This balancing effect (Liang et al. 2012) is a natural consequence of the summation term in the objectives, which makes it sensitive to the number of points in the cluster. It is interesting to note that the performance of *k*-means and *k*-medians dropped substantially as we increased the dense/sparse ratio, even though ARI weakly penalizes when a small number of points (in sparse clusters) are incorrectly assigned.

D Details of experiments with real benchmark datasets

D.1 Description of the real datasets

We now provide descriptions for the benchmark datasets we used to evaluate the real-world performance of the kc-Opt algorithm with the baselines. In Table 2, we list the number of observations n, dimensions d, and the true number of clusters (or classes) k along with the information of the machine learning data repository from where we obtained the dataset. Moreover, we mention that these datasets are standard benchmarks used to validate the performance of competing clustering algorithms. We believe these datasets do not contain any personally identifiable information or offensive content.

D.2 Running time comparison

In Section 4.2, we reported the objective values we obtained from the kc-Opt and the other baselines for the chosen real datasets. In Table 2, we present the time it took for these experiments for the different algorithms. We observe that the time taken for the kc-Gon algorithm is the lowest among the k-center algorithms, while kc-HS performs at comparable speeds to our kc-Opt algorithm barring some exceptions. This supports our claim that we are both scalable and optimal.

References

Hubert, L. J.; and Arabie, P. 1985. Comparing Partitions. *Journal of Classification*, 2: 193–218.

Liu, L.; Sun, L.; Chen, S.; Liu, M.; and Zhong, J. 2016. K-PRSCAN: A Clustering Method Based on Pagerank. *Neurocomputing*, 175: 65–80.

Lu, Y.; and Wan, Y. 2012. Clustering by Sorting Potential Values (Cspv): A Novel Potential-Based Clustering Method. *Pattern Recognition*, 45(9): 3512–3522. Best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011).