

# Scalable and Globally Optimal Generalized $L_1$ $K$ -center Clustering via Constraint Generation in Mixed Integer Linear Programming

Aravinth Chembu<sup>1</sup>, Scott Sanner<sup>1\*</sup>, Hassan Khurram<sup>1</sup>, Akshat Kumar<sup>2</sup>

<sup>1</sup> University of Toronto, Toronto

<sup>2</sup> 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.

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

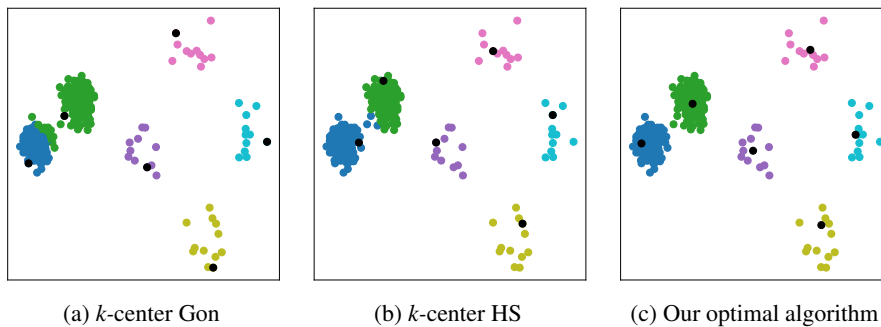


Figure 1: A motivating example comparing  $k$ -center clustering with Gonzalez (1985) (Gon) and Hochbaum and Shmoys (1985) (HS) algorithm, and our proposed optimal clustering for imbalanced clusters. The Gon and HS algorithms choose boundary points of clusters as centers while our optimal algorithm results 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 with $L_1$ $k$ -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  $\mathbf{x}_i$  with  $d$  dimensions in the dataset  $\mathbf{X} \in \mathbb{R}^{n \times d}$  where  $i \in N = \{1, \dots, n\}$ . The goal in hard-partitioning clustering is to assign each of the  $n$  observations to one of the  $k$  clusters  $C_j$  where  $j \in K = \{1, \dots, k\}$ . Let  $\mathbf{z}_j$  represent the centroid for  $C_j$  such that we minimize the following objective for the  $L_1$   $k$ -center problem:

$$\min_{\mathbf{z}} \max_{j \in K} \max_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \mathbf{z}_j\|_1 \quad (1)$$

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)

$$\begin{aligned} & \min_{\mathbf{c}, \mathbf{z}} \max_{j \in K} \max_{i \in N} \|\mathbf{x}_i - \mathbf{z}_j\|_1 c_{ij} \\ \text{s.t. } & \sum_{j \in K} c_{ij} = 1, \quad i \in N; \quad z_{j,1} < z_{j+1,1}, \quad j \in K \setminus \{k\}; \\ & c_{ij} \in \{0, 1\}, \quad i \in N, \quad j \in K; \quad \mathbf{z}_j \in \mathbb{R}^d, \quad j \in K \end{aligned} \quad (2)$$

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).

$$\begin{aligned} & \min_{\mathbf{c}, \mathbf{z}, \varepsilon} \varepsilon \\ \text{s.t. } & \varepsilon = \max_{j \in K} \max_{i \in N} \|\mathbf{x}_i - \mathbf{z}_j\|_1 c_{ij}; \\ & \sum_{j \in K} c_{ij} = 1, \quad i \in N; \quad z_{j,1} < z_{j+1,1}, \quad j \in K \setminus \{k\}; \\ & c_{ij} \in \{0, 1\}, \quad i \in N, \quad j \in K; \quad \mathbf{z}_j \in \mathbb{R}^d, \quad j \in K \end{aligned} \quad (3)$$

We achieve this transformation to a bi-level optimization problem by introducing an additional variable  $\varepsilon$ . We can further relax the equality in the first constraint of Problem (3) to  $\varepsilon \geq \max_j \max_i \|\mathbf{x}_i - \mathbf{z}_j\|_1 c_{ij}$ , which preserves optimality since the minimization of  $\varepsilon$  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  $\{\varepsilon \geq \|\mathbf{x}_i - \mathbf{z}_j\|_1 c_{ij}\}$  for all  $i$  and  $j$  and hence also the maximum over all  $i$  and  $j$  as required by the first constraint ( $\varepsilon = \max_{j \in K} \max_{i \in N} \|\mathbf{x}_i - \mathbf{z}_j\|_1 c_{ij}$ ) in Problem (3). This leads us to the final form of our *pure* MILP clustering problem (4):

$$\begin{aligned} & \min_{\mathbf{c}, \mathbf{z}, \varepsilon} \varepsilon \\ \text{s.t. } & c_{ij} = 1 \implies \varepsilon \geq \|\mathbf{x}_i - \mathbf{z}_j\|_1 \quad i \in N, \quad j \in K; \\ & \sum_{j \in K} c_{ij} = 1, \quad i \in N; \quad z_{j,1} < z_{j+1,1}, \quad j \in K \setminus \{k\}; \\ & c_{ij} \in \{0, 1\}, \quad i \in N, \quad j \in K; \quad \mathbf{z}_j \in \mathbb{R}^d, \quad j \in K \end{aligned} \quad (4)$$

#### 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  $\varepsilon$ , whose value is governed by the logical constraints in the MILP. As mentioned previously,  $\varepsilon$  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  $\mathbf{z}_j$ . We then solve the MILP

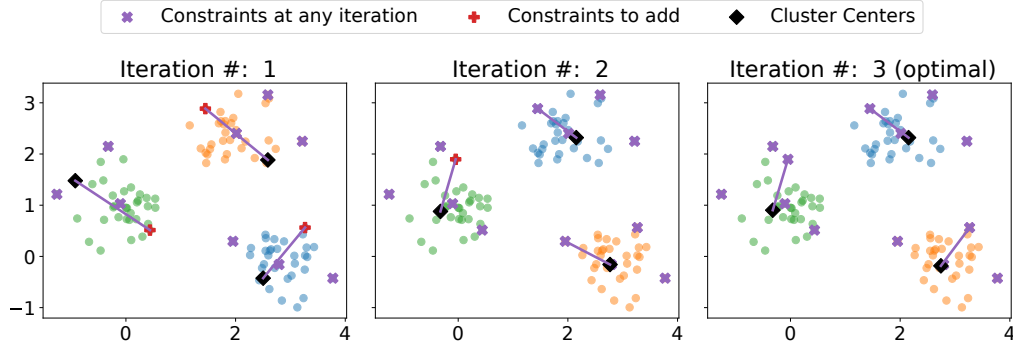


Figure 2: We show the constraint generation process using a synthetic example where we start (left) with 9 constraints (denoted with  $\times$ ), add 3 constraints (denoted with  $+$ ) in the first iteration corresponding to the extreme points from the centers (left), 1 constraint in second iteration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

---

#### Algorithm 1: kc-Opt algorithm

---

**Input** Data  $\mathbf{x}_i$ , and  $k$

- 1:  $\varepsilon^* \leftarrow 0$ ,  $\hat{\varepsilon} \leftarrow \infty$
- 2:  $I \neq \emptyset$ ,  $C \neq \emptyset$ ,  $\mathbf{z}_j^* \leftarrow$  Warm-start  
 $\triangleright$  Initial constraints  $C$  for points in  $I$
- 3:  $\varepsilon^*$ ,  $\mathbf{z}_j^*$ ,  $c_{ij}^* \leftarrow$  Solve MILP subject to  $C$
- 4:  $\hat{c}_{ij} \leftarrow \{\mathbb{1}_{j=\hat{j}} \mid \hat{j} = \arg \min_j \|\mathbf{x}_i - \mathbf{z}_j^*\|_1\}$ ,  
 $\triangleright$  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  
 $\triangleright$  Re-solve MILP with augmented constraints set  $C$  and points in  $I$
- 8: **end if**
- 9: **return**  $\varepsilon^*$ ,  $\mathbf{z}_j^*$ ,  $\hat{c}_{ij}$       $\triangleright$  Optimal solution

---

(with constraints in  $C$ ) to obtain the current best solution  $\varepsilon^*$  (line 3 in Algorithm 1). Using the current centers  $\mathbf{z}_j^*$ , 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  $\varepsilon^*$ . If found greater than  $\varepsilon^*$ , 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  $\varepsilon^*$  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  $\varepsilon^*$  (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* con-

---

#### Algorithm 2: Add-Constraints subroutine

---

**Input** Data  $\mathbf{x}_i$ ,  $\mathbf{z}_j^*$ ,  $\hat{c}_{ij}$ ,  $I$ ,  $C$

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

---

straints 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  $\{c_{5,1}=1 \implies \varepsilon \geq \left\| \begin{bmatrix} 0.44 \\ 0.51 \end{bmatrix} - \mathbf{z}_1 \right\|_1\}$ . In the example 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  $d$ . Fortunately, 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 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:

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

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

---

#### Algorithm 3: Add-Constraints-Outliers subroutine

---

**Input Data**  $\mathbf{x}_i, \mathbf{z}_j^*, \hat{c}_{ij}, I, C$

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

---

(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.

## 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.<sup>1</sup> 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 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

<sup>1</sup><https://github.com/Aravinthck/Optimal-KCenters>

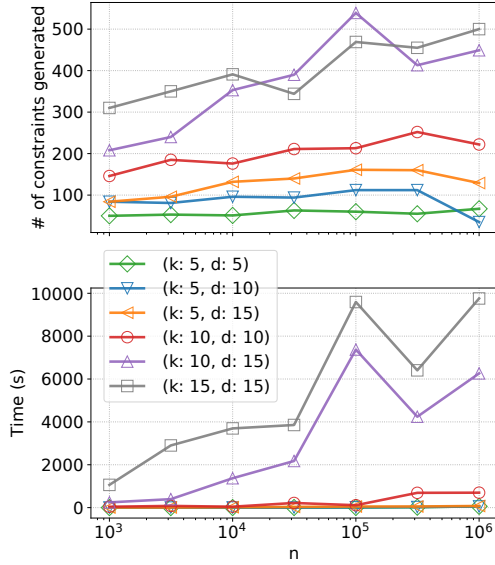
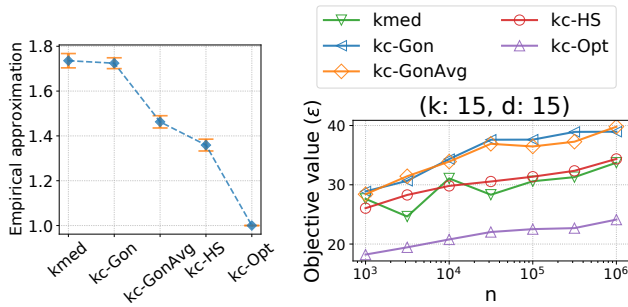


Figure 3: The performance of our kc-Opt algorithm for different values of  $n$ ,  $k$ , and  $d$  with the *Norm* dataset.



(a) Approximation factor with 95% confidence interval around mean (b) Objective values for baselines with the optimal value from our algorithm on *Norm* dataset.

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

performance against the competing popular algorithms.

**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  $\approx 450$  for the case with 1,000,000 data points distributed in 15 clusters (cf. Figure 3).

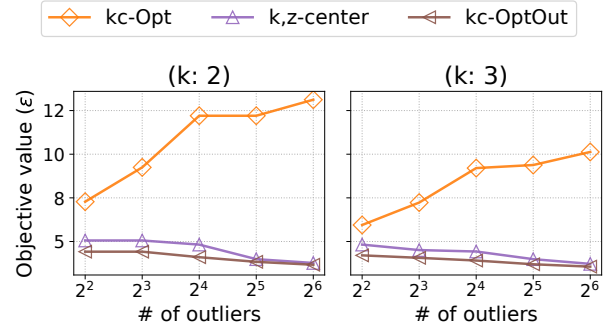


Figure 5: Comparison of outlier-aware  $k,z$ -center with our optimal kc-OptOut on *Norm-Out* dataset; both are parameterized by the number of outliers to remove ( $x$ -axis).

**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.

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

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 ( $\varepsilon$ )	<b>2.3</b>	<b>255.6</b>	<b>5.1</b>	<b>43.3</b>	<b>145.7</b>	<b>0.8</b>	<b>6.5</b>	<b>25064.3</b>	<b>3024.0</b>	<b>42.0</b>

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 ( $\varepsilon$ )	<b>18.3</b>	<b>74.9</b>	<b>3.4</b>	<b>3.1</b>	<b>517.5</b>	<b>391747.0</b>	<b>4318.5</b>	<b>482.9</b>	<b>27.5</b>	<b>318.5</b>

Table 1: Comparative evaluation of objective values for our kc-Opt (optimal  $\varepsilon$  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.

the previous two simulation studies, we constructed synthetic datasets (*Norm-Out*) with Gaussian clusters in two-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  $\varepsilon$  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  $\varepsilon$  for the baseline is higher 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 at 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, 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  $\varepsilon$  for the baselines and compare them with our optimal algorithm (kc-Opt) in Table 1. We present the optimal value  $\varepsilon$  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  $\varepsilon$  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.
- Batani, 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.
- 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.
- 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 approximation* *Generalized 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. In *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.