# Clustering With or Without the Approximation*

Frans Schalekamp
ITCS, Tsinghua University
frans@tsinghua.edu.cn

Michael Yu
MIT
mikeyu@mit.edu

Anke van Zuylen
ITCS, Tsinghua University
anke@tsinghua.edu.cn

**Abstract**

We study algorithms for clustering data that were recently proposed by Balcan, Blum and Gupta in SODA'09 [6] and that have already given rise to two follow-up papers. The input for the clustering problem consists of points in a metric space and a number $k$, specifying the desired number of clusters. The algorithms find a clustering that is provably close to a target clustering, provided that the instance has the "$(1+\alpha,\varepsilon)$-property", which means that the instance is such that all solutions to the $k$-median problem for which the objective value is at most $(1+\alpha)$ times the optimal objective value correspond to clusterings that misclassify at most an $\varepsilon$ fraction of the points with respect to the target clustering. We investigate the theoretical and practical implications of their results.

Our main contributions are as follows. First, we show that instances that have the $(1+\alpha,\varepsilon)$-property and for which, additionally, the clusters in the target clustering are large, are easier than general instances: the algorithm proposed in [6] is a constant factor approximation algorithm with an approximation guarantee that is better than the known hardness of approximation for general instances. Further, we show that it is $NP$-hard to check if an instance satisfies the $(1+\alpha,\varepsilon)$-property for a given $(\alpha,\varepsilon)$; the algorithms in [6] need such $\alpha$ and $\varepsilon$ as input parameters, however. We propose ways to use their algorithms even if we do not know values of $\alpha$ and $\varepsilon$ for which the assumption holds. Finally, we implement these methods and other popular methods, and test them on real world data sets. We find that on these data sets there are no $\alpha$ and $\varepsilon$ so that the dataset has both $(1+\alpha,\varepsilon)$-property and sufficiently large clusters in the target solution. For the general case, we show that on our data sets the performance guarantee proved by [6] is meaningless for the values of $\alpha,\varepsilon$ such that the data set has the $(1+\alpha,\varepsilon)$-property. The algorithm nonetheless gives reasonable results, although it is outperformed by other methods.

*This work was supported in part by the National Natural Science Foundation of China Grant 60553001, and the National Basic Research Program of China Grant 2007CB807900, 2007CB807901. Part of this work was done while the second author was visiting the Institute for Theoretical Computer Science at Tsinghua University in the summer of 2009.

# 1 Introduction

Clustering is an important problem which has applications in many situations where we try to make sense of large amounts of data, such as in biology, marketing, information retrieval, et cetera. A common approach is to infer a distance function on the data points based on the observations, and then to try to find the correct clustering by solving an optimization problem such as the $k$-median problem, $k$-means problem or min-sum clustering problem. Unfortunately, these three optimization problems are all $NP$-hard, hence we do not expect to find algorithms that find the optimal solution in polynomial time. Research has therefore focused on finding good heuristics (such as for example the popular $k$-means++ algorithm [3]), exact methods (see for example [10]), and approximation algorithms: polynomial time algorithms that come with a guarantee $\beta$ that the returned solution has objective value at most $\beta$ times the optimum value. Research into approximation algorithms for these three clustering problems has produced a large number of papers that demonstrate approximation algorithms as well as lower bounds on the best possible guarantee. However, in many cases there is still a gap between the best known approximation algorithm and the best known lower bound.

In a recent paper, Balcan, Blum and Vempala [7] (see also Balcan, Blum and Gupta [6]) observe the following: The optimization problems that we try to solve are just proxies for the real problem, namely, finding the "right" clustering of the data. Hence, if researchers try so hard to find better approximation algorithms, that must mean that we believe that this will help us find clusterings that are closer to the target clustering. More precisely, Balcan, Blum and Vempala [7] turn this implicit belief into the following explicit assumption: there exist $\alpha > 0, \varepsilon > 0$ such that any solution with objective value at most $(1 + \alpha)$ times the optimum value misclassifies at most an $\varepsilon$ fraction of the points (with respect to the unknown target clustering). We will call this the $(1 + \alpha, \varepsilon)$-property. By making this implicit assumption explicit, Balcan et al. [6] are able to show that, given $(\alpha, \varepsilon)$ such that the $(1+\alpha, \varepsilon)$-property holds, quite simple algorithms will give a clustering that misclassifies at most an $O(\varepsilon)$-fraction of the points. In the case when the clusters in the target clustering are "large" (where the required size is a function of $\varepsilon/\alpha$), they give an algorithm that misclassifies at most an $\varepsilon$ fraction of the points. In the general case, they give an algorithm that misclassifies at most an $O(\varepsilon/\alpha)$-fraction. They do not need better approximation algorithms for the $k$-median problem to achieve these results: in fact, [6] shows that finding a $(1+\alpha)$-approximation algorithm does not become easier if the instance satisfies the $(1 + \alpha, \varepsilon)$-property.

These results seem quite exciting, because they allow us to approximate the target clustering without approximating the objective value of the corresponding optimization problem. As Balcan et al. [6] point out, especially if approximating the objective to within the desired accuracy is hard, we have no choice but to "bypass" the objective value if we want to approximate the target clustering.

However, it is not immediately clear how useful these results are in practice. A first concern is that the algorithms need parameters $\alpha$ and $\varepsilon$ such that the instance satisfies the $(1 + \alpha, \varepsilon)$-property. The paper by Balcan et al. [6] gives no suggestions on how a practitioner can find such $\alpha$ and $\varepsilon$. And of course an interesting question is whether these new algorithms outperform previously known methods in approximating the target clustering, if we do know $\alpha$ and $\varepsilon$, especially in the case when $\alpha$ is smaller than the guarantee of the best known approximation algorithm.

In this paper, we set out to investigate practical and theoretical implications of the algorithms in Balcan et al. [6] We now briefly describe our contributions.

## 1.1 Our contributions

We focus on the case when the optimization problem we need to solve is the $k$-median problem. Our main theoretical contribution is a proof that the algorithm for "large clusters" given by Balcan et al. [6] is in fact an approximation algorithm with a guarantee $1 + \frac{1}{1/2+5/\alpha}$. One could argue that the algorithm of Balcan et al. [6] is most interesting when $\alpha \leq 2$ (since for $\alpha > 2$ one can use the algorithm of Arya et al. [4] to obtain the claimed result), and hence in those cases their algorithm has an approximation guarantee of at most $\frac{4}{3}$. However, for the general case of the $k$-median problem, there is a hardness of approximation of $1+\frac{1}{e}$ [16]. As $1+\frac{1}{e} \approx 1.37$ is larger than $\frac{4}{3}$, this means that these instances are provably easier than the general class of instances. We note that Balcan et al. [6] show that approximating the $k$-median objective does *not* become easier if we are guaranteed that the instance has the $(1 + \alpha, \varepsilon)$-property. We show that it *does* become easier for those instances that have the $(1 + \alpha, \varepsilon)$-property and

for which the clusters in the target clustering are "large".

For the general case, we show that it is $NP$-hard to check whether a data sets satisfies the $(1+\alpha, \varepsilon)$-property for a given $\alpha, \varepsilon$; however, knowledge of such parameters is necessary to run the algorithm of Balcan et al. [6].

We implement the algorithms of Balcan et al. [6] and compare the results to the outcome of previously known methods for various real world data sets. We show how to efficiently run the algorithms for all possible values of the parameters $(\alpha, \varepsilon)$ (regardless of whether the assumption holds for the pair of values), and suggest a heuristic for choosing a good solution among the generated clusterings. The algorithm for "large clusters" fails to find a solution on any of our instances and for any value of the parameters $\alpha, \varepsilon$. The algorithm for the general case, however, does return reasonably good solutions. We compare these results to other methods, and find that they are reasonable, but that there are other methods, both heuristics and approximation algorithms, which are significantly better both in terms of approximating the target clustering and approximating the $k$-median objective.

We also show how to enumerate all values of $\alpha, \varepsilon$ for which the $(1+\alpha, \varepsilon)$-property holds, which we note are not practical as they need or calculate the optimal $k$-median solution. We find that indeed our data sets never satisfy the $(1+\alpha, \varepsilon)$-property *and* the large clusters assumption. For the general case, we find that the proven guarantee on the misclassification of $O(\varepsilon/\alpha)$ is greater than one.

## 1.2 Related Work

We focus our discussion on research that is most relevant to our work; we stress that this is by no means an exhaustive description of the literature on clustering and the $k$-median problem.

The $k$-median problem and the related facility location problem, in which there is a cost associated with the decision to have a cluster center at each given point, have been extensively studied by the approximation algorithms community. Charikar, Guha, Tardos and Shmoys [14] give the first constant-factor approximation algorithm for the $k$-median problem using LP rounding. Jain and Vazirani [18] give a primal-dual 6-approximation algorithm for the $k$-median problem. This was improved by Charikar and Guha [13] to a 4-approximation algorithm. Jain, Mahdian, Markakis, Saberi and Vazirani [17] use dual-fitting to obtain a 4-approximation algorithm. The paper by [17] compares the approximation algorithms in [18] and [17] for the facility location problem on several data sets, and finds that the algorithm in [17] is significantly better than the method in [18] and finds solutions that are on average less than 3% from optimal. Arya, Garg, Khandekar, Meyerson, Munagala and Pandit [4] show that one can obtain a $(3+2/p)$-approximation algorithm by using a local search approach, where a move consists of exchanging at most $p$ centers. Since we use these algorithms as comparisons to the results obtained by the algorithms of Balcan et al. [6], we describe these algorithms in more detail in Appendix E. If we restrict attention to Euclidean metrics in constant dimension the problem admits a PTAS [1].

Balcan, Blum and Vempala [7] study the problem of approximating an unknown target clustering given a distance metric on the points. They identify properties of the distance metric that allow us to approximate an unknown target clustering. One of the properties they define is the $(1+\alpha, \varepsilon)$-property which states that any solution with objective value at most $(1+\alpha)$ times the optimal value of a given optimization problem is $\epsilon$-close to the target clustering. One of their results in an algorithm that finds a hierarchical clustering such that some pruning of the hierarchy is close to the target clustering if the data satisfies the $(2, \varepsilon)$-property with respect to the $k$-median objective.

Balcan, Blum and Gupta [6] show how to use the $(1+\alpha, \varepsilon)$-property to find clusterings that are provably close to the target clustering. We describe their work in more detail in the next section. Two follow-up papers have extended their results in two directions: Balcan, Röglin and Teng [9] consider the setting where all but a $\gamma$-fraction of the points have the $(1+\alpha, \varepsilon)$-property and this $\gamma$-fraction of points is adversarially chosen. Balcan and Braverman [8] improve the results in [6] if the goal is to approximate the target clustering and the input satisfies the $(1+\alpha, \varepsilon)$-property with respect tot the min-sum objective.

Finally, we mention a paper by Ostrovsky, Rabani, Schulman and Swamy [20], which identifies natural properties under which variants of Lloyd's algorithm are guaranteed to quickly find near-optimal solutions to the $k$-means problem, and the recent paper of Bilu and Linial [12], which gives polynomial time algorithms for a certain class of inputs to the max-cut problem, which they call "stable" instances. There are strong similarities between [6] and [12]: both approaches define classes of inputs, for which they can give algorithms that perform better than what is possible for general instances. In fact, it is possible to show that the $(1+\alpha, \varepsilon)$-property implies a stability property in similar vein to the stability

property defined in [12]. Theorem 1 in this paper shows that the class of inputs defined by $(1+\alpha,\varepsilon)$-property combined with the assumption that the clusters of the target clustering are large, is easier to approximate than general instances of the $k$-median problem.

# 2 Problem Definition

## 2.1 $k$-median Problem

In the $k$-median problem, we are given a set of elements $X$ and a distance function $d : X \times X \to \mathbb{R}^{\geq 0}$ which forms a metric (i.e., $d$ satisfies the triangle inequality), a subset of elements $V \subset X$ that need to be covered and a parameter $k \in \mathbb{N}$. We denote $|V| = n$. The goal is to choose $k$ cluster centers $v_1, \ldots, v_k \in X$ so as to minimize $\sum_{u \in V} \min_{i=1,\ldots,k} d(u, v_i)$.

We denote by $OPT$ the optimum ojective value of a given instance, and we say an algorithm is an $\beta$-approximation algorithm for the $k$-median problem if for any instance it is guaranteed to output cluster centers $v_1, \ldots, v_k \in X$ so that $\sum_{u \in V} \min_{i=1,\ldots,k} d(u, v_i) \leq \beta OPT$.

## 2.2 Setting of Balcan, Blum and Gupta

In the setting considered by Balcan et al. [6], an instance also includes an unknown target clustering, i.e. a partition $C_1^*, \ldots, C_k^*$ of $V$. We say a clustering $C_1, \ldots, C_k$ is $\varepsilon$-close to the target clustering, if there exists a permutation $\pi$ such that $\frac{1}{n}\sum_{i=1}^{k} |C_i^* \backslash C_{\pi(i)}| \leq \varepsilon$. The misclassification is defined as the smallest $\varepsilon$ such that the clustering is $\varepsilon$-close to the target clustering.

When clustering data into $k$ clusters, an often used approach is to define a distance function on the data based on observations, and to solve an optimization problem (for example, the $k$-median problem) to obtain a clustering. Balcan et al. [6] argue that the quest for better approximation algorithms thus implies a belief that better approximations will result in solutions that are closer to the unknown target clustering. They formalize this implicit belief into the following property.

**Definition 1** (**$(1+\alpha,\varepsilon)$-property).** *An instance satisfies the $(1+\alpha,\varepsilon)$-property, if any $k$-median solution with objective value at most $(1+\alpha)OPT$ is $\varepsilon$-close to the target clustering.*

We now summarize the algorithms and contributions made by Balcan et al. in [6]. In the remainder of this paper, we will use the abbreviation BBG to refer to the algorithms proposed by Balcan, Blum and Gupta in [6].

## 2.3 BBG Algorithm for Large Clusters

Note that if the size of each cluster in the target clustering $C_1^*, \ldots, C_k^*$ is greater than $2\varepsilon n$, and the instance has the $(1+\alpha,\varepsilon)$-property, then there is a unique way to label the clusters $C_1, \ldots, C_k$ of the optimal $k$-median clustering so that $|C_i^* \backslash C_i| \leq \varepsilon n$ for each $i$. We will say the optimal $k$-median solution and the target clustering agree on $v$ if $v \in C_i^* \cap C_i$. In addition, we define $\varepsilon^*$ as the misclassification of the optimal $k$-median solution.

If the clusters in the target clustering are large, Balcan et al. [6] show in Lemma 3.1 that the following property follows from the $(1+\alpha,\varepsilon)$-property in Definition 1.

**Definition 2** (**weak $(1+\alpha,\varepsilon)$-property if all clusters in the target clustering have more than $2\varepsilon n$ points).** *In the optimal $k$-median solution, there are at most $(\varepsilon - \varepsilon^*)n$ points on which the target clustering and the optimal $k$-median solution agree for which the second closest center is strictly less than $\alpha OPT/(\varepsilon n)$ farther than the closest center.*

The algorithm proposed in [6] is given in Figure 1. Note that for this algorithm we need to know $\alpha, \varepsilon$ and $OPT$. If these three values are known, then Balcan et al. [6] show that the clustering found by BBGlarge is $\varepsilon$-close to the target clustering, provided that each cluster in the target clustering has size at least $(3 + 10/\alpha)\varepsilon n + 2$.

Balcan et al. [6] also show that, if $OPT$ is not known, then one can iterate through all possible threshold graphs $G_{\alpha,\varepsilon}$ and identify one on which their algorithm gives a solution which is $\varepsilon$-close to the target clustering. The $G_{\alpha,\varepsilon}$ chosen is the graph with the lowest threshold value $\tau$ for which the largest $k$

**BBGlarge [6], input parameters:** $(V, d, k, \alpha, \varepsilon, OPT)$

Let $\tau = 2\alpha OPT/(5\varepsilon n)$. Form the threshold graph $G_{\alpha,\varepsilon} = (V, E)$ where $E = \{\{u, v\} : d_{uv} \leq \tau\}$.
Let $b = (1 + 5/\alpha)\,\varepsilon n$. Form the neighborhood graph $H_{\alpha,\varepsilon} = (V, E_H)$ where
    $E_H = \{\{u, v\} : u, v$ have at least $b$ common neighbors in $G_{\alpha,\varepsilon}\}$.
Let $C_1, \ldots, C_k$ be the vertices in the $k$ largest connected components of $H_{\alpha,\varepsilon}$.
Add remaining nodes to an arbitrary cluster.
*//Reassignment Step:*
Let $C'_1, \ldots, C'_k \leftarrow \emptyset$.
For each $v \in V$
    Let $d_{med}(v, i)$ be the median distance from $v$ to $C_i \backslash \{v\}$.
    Add $v$ into cluster $C'_j$ where $j = \arg\min_i \{d_{med}(v, i)\}$.
Return $C'_1, \ldots, C'_k$.

Figure 1: Clustering algorithm proposed by Balcan, Blum and Gupta [6] for the case when the target clustering has large clusters.

**BBGgeneral [6], input parameters:** $(V, d, k, \alpha, \varepsilon, OPT)$

Let $\tau = \alpha OPT/(5\varepsilon n)$. Form the threshold graph $G_{\alpha,\varepsilon} = (V, E)$ where $E = \{\{u, v\} \in V : d_{uv} \leq \tau\}$.
For $i = 1, \ldots, k$
    choose the highest degree vertex $v$ in $G_{\alpha,\varepsilon}$ and let $C_i = \{v\} \cup \{u \in V : \{u, v\} \in E\}$.
    remove $C_i$ and the edges adjacent to them from $G_{\alpha,\varepsilon}$.
Add the remaining nodes in $G_{\alpha,\varepsilon}$ to $C_k$.
Return $C_1, \ldots, C_k$.

Figure 2: Clustering algorithm proposed by Balcan, Blum and Gupta [6] for the general case.

connected components have size at least $(1+5/\alpha)\varepsilon n$ and there are at most $(1+5/\alpha)\varepsilon n$ remaining nodes. To prove that this solution is $\varepsilon$-close, they need a slightly stronger assumption on the cluster sizes: the size of the clusters in the target clustering must be at least $(4 + 15/\alpha)\varepsilon n + 2$.

## 2.4 BBG Algorithm for the General Case

Balcan et al. [6] show in Lemma 3.1 that in the general case, the $(1+\alpha, \varepsilon)$-property implies the following property, which we will call the weak $(1 + \alpha, \varepsilon)$-property:

**Definition 3 (weak $(1 + \alpha, \varepsilon)$-property).** *In the optimal $k$-median solution, there are at most $6\varepsilon n$ points for which the second closest center is strictly less than $\alpha OPT/(2\varepsilon n)$ farther than the closest center.*

The algorithm proposed by Balcan et al. [6] for the general case is remarkably simple, and is given in Figure 2. Balcan et al. [6] show that if the *weak* $(1 + \alpha, \varepsilon)$-property holds, then this algorithm finds a solution which is $O(\varepsilon/\alpha)$-close to the target clustering. We note that the points that are added to $C_k$ in the final step of the algorithm are all counted as misclassified points in the analysis in [6]. We can thus replace this step with any reasonable assignment; in our implementation, we assign each point to the cluster containing the closest among the $k$ high degree vertices chosen by the algorithm. This algorithm again assumes knowledge of the optimal value of the $k$-median problem, $OPT$, as well as a pair of $(\alpha, \varepsilon)$ such that the weak $(1 + \alpha, \varepsilon)$-property holds for a particular instance. Of course, finding $OPT$ requires solving an $NP$-hard problem, however, [6] notes that a $\beta$-approximation for the $k$-median problem suffices, if one accepts a deterioration of the misclassification guarantee by a factor $\beta$.

# 3 Theoretical Aspects of the BBG Algorithms

We now show that in fact the algorithm BBGlarge succeeds in finding a solution which not only has a classification error of at most $\varepsilon$, but is also an approximately optimal $k$-median solution.

**Theorem 1.** *If the $k$-median instance satisfies the $(1+\alpha,\varepsilon)$-property and each cluster in the target clustering has size at least $(3+10/\alpha)\varepsilon n+2$, then the algorithm for large clusters proposed by Balcan et al. [6] with the Reassigment Step repeated twice is a $\left(1+\frac{1}{1/2+5/\alpha}\right)$-approximation algorithm for $k$-median clustering.*

*Proof.* We begin by repeating some properties that were shown to hold by Balcan et al. [6] in the proof of their Theorem 6. They distinguish two types of points: red points and non-red points. Call a point $x$ red if $w_2(x)-w(x)<\alpha OPT/(\varepsilon n)$, where $w(x)$ is the distance from $x$ to its cluster center in the optimal $k$-median solution, and $w_2(x)$ is the distance from $x$ to the second closest cluster center in the optimal $k$-median solution. It is shown by [6] that the algorithm for large clusters finds a clustering that satisfies the following properties:

(i) Each non-red point is in the same cluster as in the optimal $k$-median solution.

(ii) If the Reassignment Step is repeated then all non-red points are again in the same cluster as in the optimal $k$-median solution.

Let $c_1^*,\ldots,c_k^*$ be the cluster centers in the optimal $k$-median solution, and let $C_1^*,\ldots,C_k^*$ be the corresponding clusters in the optimal solution. Let $C_1,\ldots,C_k$ be the outcome of the algorithm (after two Reassignment Steps). By (ii) we can assume that the labeling of the clusters is such that any non-red point is in $C_i\cap C_i^*$. We want to show that $\sum_{i=1}^k\sum_{x\in C_i}d(x,c_i^*)\le\left(1+\frac{1}{1/2+5/\alpha}\right)OPT$. Recall that $OPT=\sum_{x\in V}w(x)$ and that for the non-red points $x$, we have $d(x,c_i^*)=w(x)$ by property (i) above. Hence it is enough to show that

$$\sum_{i=1}^{k}\sum_{x\in C_i:x\text{ is red}}d(x,c_i^*)\le\frac{1}{1/2+5/\alpha}\sum_{x\in V}w(x).$$

We consider a red point $x$. Let $o$ be the index of the cluster that $x$ is assigned to in the optimal $k$-median solution, and let $a$ be the index of the cluster that $x$ is assigned to in the algorithm's clustering. Suppose we have a non-red point $y\in C_a\cap C_a^*$ and $z\in C_o\cap C_o^*$ such $d(x,y)\le d(x,z)$. By the triangle inequality, $d(x,c_a^*)\le d(x,y)+d(y,c_a^*)$ and by the property of $y$ and $z$, the latter value is at most $d(x,z)+d(y,c_a^*)$. We use the triangle inequality again to see that $d(x,z)+d(y,c_a^*)\le d(x,c_o^*)+d(z,c_o^*)+d(y,c_a^*)=w(x)+w(z)+w(y)$. Hence we can charge the distance $d(x,c_a^*)$ against $w(x)+w(z)+w(y)$. To prove the approximation guarantee, we need to show that we don't charge $w(z)$ and $w(y)$ too often. In the following claim, we show that there is a large number of $y,z$ so that each $w(y),w(z)$ only need to get charged a small fraction of $d(x,c_a^*)$.

**Claim 2.** *Let $x$ be a red point, such that $x\in C_o^*\cap C_a$. If $o\neq a$, then there exist sets of non-red points $Y\subseteq C_a\cap C_a^*, Z\subseteq C_o\cap C_o^*$ such that $|Y|,|Z|\ge(1/2+5/\alpha)\varepsilon n$ and $d(x,y)\le d(x,z)$ for every $y\in Y,z\in Z$.*

Note that the non-red points do not change their cluster in the second Reassignment Step. Also note that since $x$ is in $C_a$ instead of in $C_o$, its median distance to the points in the $a$-th cluster in the second Reassignment Step was closer than the median distance to the points in the $o$-th cluster. We let $Y$ be the non-red points in the $a$-th cluster for which the distance to $x$ is less than the median distance from $x$ to the $a$-th cluster, and we let $Z$ be the non-red points in the $o$-th cluster for which the distance to $x$ is more than the median distance from $x$ to the $o$-th cluster. Then $d(x,y)\le d(x,z)$ for every $y\in Y,z\in Z$.

Let $N_r$ be the number of red points, and let $N_b$ be the number of non-red points in a given cluster before the second Reassignment Step. Note that $N_r\le\varepsilon n$, and that $N_b\ge(2+10/\alpha)\varepsilon n+2$. The first fact follows because there are at most $\varepsilon n$ red points overall, and the second because each cluster in the target clustering has size at least $(3+10/\alpha)\varepsilon n+2$, and hence at least $(2+10/\alpha)\varepsilon n+2$ non-red points, all of which are correctly clustered before the second Reassignment Step. Since there are $\lfloor\frac{1}{2}(N_b+N_r)\rfloor$ points that $x$ is closer to (farther from) than the median distance to the cluster, there are at least $\lfloor\frac{1}{2}(N_b-N_r)\rfloor$

non-red points for which this is the case. ◇

By the claim, and the fact that $d(x, c_a^*) \leq w(x) + w(z) + w(y)$ for all $y \in Y, z \in Z$, we get that

$$\begin{aligned} d(x, c_a^*) &\leq w(x) + \frac{1}{|Z|}\sum_{z\in Z} w(z) + \frac{1}{|Y|}\sum_{y\in Y} w(y) \\ &\leq w(x) + \frac{1}{(1/2+5/\alpha)\varepsilon n}\sum_{z\in Z} w(z) + \frac{1}{(1/2+5/\alpha)\varepsilon n}\sum_{y\in Y} w(y). \end{aligned}$$

Hence we can "charge" the difference $d(x, c_a^*) - w(x)$ by charging at most $\frac{1}{(1/2+5/\alpha)\varepsilon n} w(v)$ to each non-red point $v$. Since there are at most $\varepsilon n$ red points $x$, the total amount charged to each non-red point $v$ is at most $\frac{1}{(1/2+5/\alpha)} w(v)$. Hence we get that

$$\sum_{j=1}^{k}\sum_{x\in C_j} d(x, c_j^*) \leq \sum_{x\in V} w(x) + \frac{1}{(1/2+5/\alpha)} \sum_{x\in V: x \text{ not red}} w(x). \quad \square$$

**Remark 1.** *If we have stronger lower bounds on the size of the clusters in the target clustering, we also get improved approximation guarantees. In particular, if each cluster in the target clustering has size at least* $(3 + 10/\alpha)\varepsilon n + 2 + \kappa\varepsilon n$*, then our algorithm is a* $(1 + \frac{1}{1/2+5/\alpha+\kappa/2})$*-approximation algorithm.*

Note that for the approximation algorithm in Theorem 1 we do not need to know the values of $\alpha$ and $\varepsilon$ or $OPT$: there are only $n$ relevant values for $b = (1 + 5/\alpha)\varepsilon n$ and $n^2$ different values for $\tau = 2\alpha OPT/(5\varepsilon n)$ that give different graphs $H_{\alpha,\varepsilon}$ in the algorithm in Figure 1. Hence we could run the algorithm for each of the possible values of $\tau, b$, and return the solution with smallest objective value.

We finally remark that the algorithms proposed by Balcan et al. [6] are most interesting for instances with a $(1+\alpha, \varepsilon)$-property with $\alpha \leq 2$: if $\alpha > 2$ then we could then find an $\varepsilon$-close clustering by running a $(3 + 2/p)$-approximation algorithm [4] for sufficiently large $p$. Hence for those $\alpha$ for which the Balcan et al. algorithms are interesting, we have shown that the large clusters algorithm gives a $\frac{4}{3}$-approximation algorithm.

Our second result in this section is to show that verifying if an instance has the $(1 + \alpha, \varepsilon)$-property for a given $\alpha, \varepsilon$ is $NP$-hard. The proof is a reduction from max $k$-cover, and is deferred to Appendix A.

**Lemma 3.** *It is NP-hard to verify whether an instance has the (weak)* $(1 + \alpha, \varepsilon)$*-property for a given* $\alpha, \varepsilon$*.*

We should remark two things about Lemma 3. First of all, in our proof we need to choose $\alpha \approx \varepsilon/n^3$. In that case, the guarantee given by Balcan et al. [6] is $O(\varepsilon/\alpha) = O(n^3)$, hence this does not constitute an interesting case for their algorithm. Second, our lemma does not say that it is $NP$-hard to find *some* $\alpha, \varepsilon$ for which the $(1 + \alpha, \varepsilon)$-property holds. However, we do not know how to find such $\alpha, \varepsilon$ efficiently.

# 4 Practical Aspects of the BBG Algorithms

## 4.1 Data Sets

We use two popular sets of data to test the algorithms, and compare their outcomes to other methods.

We use the pmed data sets from the OR-Library [11] to investigate whether the methods proposed by Balcan et al. [6] give improved performance on commonly used $k$-median data sets compared to known algorithms in either misclassification, objective value or performance relative to the running time. These instances are distance based but do not have a ground truth clustering. Note that the $(1+\alpha, \varepsilon)$-property implies that the optimal $k$-median clustering is $\varepsilon$-close to the target clustering (whatever the target clustering is), hence we can assume that the optimal $k$-median clustering is the target clustering while changing the misclassification of any solution with respect to the target by at most $\varepsilon$.

The second data sets we use come from the University of California, Irivine (UCI) Machine Learning Repository [5]. For these data sets a ground truth clustering is known and given. The data sets we use

have only numeric attributes and no missing values. To get a distance functions, we first apply a "$z$-transform" on each of the dimensions, i.e., for each attribute we normalize the values to have mean 0 and standard deviation 1. Next, we calculate the Euclidean distance between each pair of points. We note that it may be possible to define distance functions that give better results in terms of approximating the target clustering. This is not within the scope of this paper, as we are only interested in comparing the performance of different algorithms for a given distance function.

The data sets from UCI that we use are the wine data set with 178 elements, 13 attributes and 3 clusters, the iris data set (150 elements, 4 attributes, 3 clusters), the yeast data set (1484 elements, 8 attributes, 10 clusters), the letter data set (20000 elements, 16 attributes, 26 clusters). We sample this latter data set down to 1000 and 2000 elements. The sizes of the different data sets in the pmed collection can be found in Table 3.

## 4.2 Algorithm for Large Clusters

Our first experimental results are those obtained by running the algorithm in Figure 1. Balcan et al. [6] point out that we do not need to know the value of $OPT$ to run this algorithm; instead we can iterate through all possible threshold graphs $G_{\alpha,\varepsilon}$ and choose the graph with the lowest threshold value $\tau = 2\alpha OPT/(5\varepsilon n)$ for which the largest $k$ connected components have size at least $(1 + 5/\alpha)\varepsilon n$ and there are at most $(1 + 5/\alpha)\varepsilon n$ remaining nodes.

Balcan et al. [6] do not discuss how to find values of $\alpha$ and $\varepsilon$ to use. Indeed, Lemma 3 gives an indication that this may be far from trivial. We therefore try all possible values of $\alpha$ and $\varepsilon$: note that the only dependence on these parameters in the modified algorithm (that does not need to know the value of $OPT$) is the value $b = (1 + 5/\alpha)\varepsilon n$. Hence, we interate over all possible values of $b = 0, \ldots, n$. Given $b$, we iterate over all possible threshold graphs to find the smallest threshold value such that the largest $k$ connected components have size at least $(1 + 5/\alpha)\varepsilon n$ and there are at most $(1 + 5/\alpha)\varepsilon n$ remaining nodes.

Since we thus need to run the algorithm in Figure 1 $O(n^3)$ times, we reuse as much of the information from the previous iteration as we can. In particular, for a fixed value $b$, we note that the algorithm basically looks at the square of the adjacency matrix of different threshold graphs (call this $A^2$). Instead of updating the adjacency matrix when we increase the threshold, we instead update $A^2$ directly, which can be done by updating two rows and two columns of $A^2$ per edge that is added to the threshold graph.

We ran this algorithm on all instances, and found a somewhat surprising outcome: it did not return any clustering on any instance! This means that there exists no $\alpha, \varepsilon$ such that the data satisfied the $(1 + \alpha, \varepsilon)$-property *and* the clusters in the target clustering had size at least $(3 + 10/\alpha)\varepsilon n + 2$.

## 4.3 Algorithm for General Case

Similar to the algorithm for large clusters, we propose bypassing the fact that we do not know which values of $\alpha$ and $\varepsilon$ to use by iterating over all possible outcomes of the algorithm in Figure 2. Since there are only $O(n^2)$ possibilities for the threshold graph $G_{\alpha,\varepsilon}$, hence it is indeed possible to do so in polynomial time.

One could, of course, naïvely generate all possible threshold graphs (by sorting the edges and adding them in turn) and form clusters as in Figure 2, but here is a faster way of doing it. We can keep track of how many more edges need to be added adjacent to $v$, before $v$ becomes the center of cluster $i$, for every node $v$ and every cluster $i$. This way, we know which clusters of the clustering of the previous threshold graph remain unchanged, and reduce the amount of work done by the algorithm. We give more details on our implementation in Appendix B.

Table 4 in Appendix C shows how many solutions where generated using this method, which is at most 5% of the $n(n-1)/2$ solutions that would be generated with the naïve method. Note that these solutions are not necessarily distinct; the number of distinct solution is given in the table as well.

### 4.3.1 Choosing a Solution

Even though we managed to reduce the number of solutions to consider from $O(n^2)$ to a much smaller number, the number of solutions is still very large. Our next challenge is how to choose a good solution, that is close to the target clustering.

A natural choice is to choose the outcome $C_1, \ldots, C_k$ with the lowest $k$-median objective value defined as $\sum_{i=1}^{k} \min_{c \in X} \sum_{v \in C_i} d(c, v)$. The following lemma shows that unfortunately the $k$-median objective is not always a reliable indicator of the best solution. The proof is deferred to Appendix A.

**Lemma 4.** *For a given instance, let $\delta$ be the misclassification of the solution with lowest $k$-median objective value among all outcomes obtained by the algorithm from Section 4.3. Then $\delta \neq O(\varepsilon/\alpha)$, even if the instance has the weak $(1+\alpha, \varepsilon)$-property.*

We have tried several other criteria which are inspired by the analysis of Balcan et al. [6]; they show that most of the points of each cluster in the target clustering form cliques in the threshold graph, and the cliques corresponding to different clusters are not connected. There are only $O(\varepsilon/\alpha)$ points that do not conform to this. Our criteria measure how far the threshold graph is from this ideal form by counting the number of edges between clusters plus the number of non-edges within clusters, or by computing the size of the vertex cover on these edges found by the greedy algorithm.

In Appendix D we compare these three criteria. None of these criteria is guaranteed to choose a solution with small misclassification, but some of them, including the $k$-median objective value, seem to work quite well in practice. In Table 6 in Appendix D we show the minimum misclassification and the additional misclassification in percentage points for each of the criteria.

Since the $k$-median objective value gives good results and is quick to evaluate, we chose this criterion for our experimental comparison: on average, the misclassification of the best solution among all solutions generated is 6 percentage points lower than the misclassification of the solution with the best $k$-median objective value.

## 4.4 Verifying the $(1+\alpha, \varepsilon)$-property for these data sets

To investigate the extent to which our data sets exhibit the properties needed for the algorithm, we use two methods, which need or calculate an optimal $k$-median solution, and which find settings of $\alpha$ and $\varepsilon$ for which the $(1+\alpha, \varepsilon)$-property holds. The goal of these experiments is two-fold. First of all, we want to verify whether the algorithm by Balcan et al. [6] is meaningful on our data sets (which seems to be true for the general case algorithm based on the results in Section 4.3) or if perhaps the necessary assumption is too strong (which may be the case for the "large" clusters case). In addition, we may be able to find some way of choosing values of $\alpha$ and $\varepsilon$ that work "often" (not provably, but empirically), which would get us around the problem of having to come up with a solution to an $NP$-hard problem, or having to evaluate a large number of solutions.

Note that we can test for different properties: the (original) $(1+\alpha, \varepsilon)$-property given in Definition 1, and the derived weak $(1+\alpha, \varepsilon)$-property in Definition 2 and Definition 3. The weak properties are sufficient for the theoretical analysis algorithms and as they are easier to verify because we know an optimal solution for most instances, we focused most of our efforts on verifying the weak properties.

### 4.4.1 Weak $(1+\alpha, \varepsilon)$-property for Large Clusters

To test the weak property for the large clusters case in Definition 2, let $w(i)$ be the distance from point $i \in V$ to its closest center in the optimal $k$-median solution, and let $w_2(i)$ be the distance from point $i$ to its second closest center. We assume that the points are labeled so that $w_2(i) - w(i) \leq w_2(i+1) - w(i+1)$ for $i =, \ldots, n-1$.

We set $\varepsilon_i = i/n$ for $i = 1, \ldots, n-1$ and we compute $\alpha_i$ such that for at most $\varepsilon_i n$ points the second closest center is less than $\alpha_i OPT/(\varepsilon_i n)$ farther than the closest center, i.e. $\alpha_i = i/OPT(w_2(i+1) - w(i+1))$. If $w_2(i+1) - w(i+1) = w_2(i+2) - w(i+2)$, then we ignore the values of $\varepsilon_i$ and $\alpha_i$.

For any $(\alpha, \varepsilon)$ with $\varepsilon < 1$ such that the property in Definition 2 holds, let $i$ be such that $\varepsilon_i n = \lfloor \varepsilon n \rfloor$. Then it is not hard to see that $\alpha \leq \alpha_i$. Hence for any such $(\alpha, \varepsilon)$, we have some $i$ such that $\alpha_i \geq \alpha$ and $\varepsilon_i \leq \varepsilon$.

We also compute the required minimum cluster size $b_i = (3 + 10/\alpha_i)\varepsilon_i n + 2$ (which we note is the minimum cluster size if the value $OPT$ is known — if $OPT$ is not known the minimum cluster size is even larger). Note that $\varepsilon_i n/\alpha_i = OPT/(w_2(i+1) - w(i+1))$, so we find that $b_i = 3i + 10OPT/(w_2(i) - w(i)) + 2$ for $i = 1, \ldots, n-1$. Note that $OPT/(w_2(i) - w(i))$ is decreasing in $i$, and $3i$ is increasing in $i$. We found that on our data sets, it was always the case that $b_{n-1} = \min_i b_i$. Since $b_{n-1} > 3(n-1) + 2 = n - 1$, the

assumptions for the algorithm for large clusters are thus never satisfied by our data sets, since, clearly, the clusters of the target clustering do not have size at least $n$.

#### 4.4.2 Weak $(1+\alpha, \varepsilon)$-property for the General Case

The weak property for the general case given in Definition 3 can be computed in a similar way as for the large clusters case, but now we set $6\varepsilon_i = i/n$ for $i = 1, \ldots, n$ and compute $\alpha_i$ such that $\alpha_i OPT/(2\varepsilon_i n) = w_2(i+1) - w(i+1)$.

By similar reasoning as in Section 4.4.1, it is the case that for any $(\alpha, \varepsilon)$ with $\varepsilon < 1$ such that the weak $(1+\alpha, \varepsilon)$ property holds, there exists some $i$ such that $\varepsilon_i \leq \varepsilon$ and $\alpha_i \geq \alpha$.

The guarantee on the misclassification proved in Theorem 8 of Balcan et al. [6] is $O(\varepsilon/\alpha)$, or, to be precise, $25\varepsilon + 40\varepsilon/\alpha$. Hence, the best possible guarantee on a data set is given by $\min_i(25\varepsilon_i + 40\varepsilon_i/\alpha_i)$. In this case, $\varepsilon_i/\alpha_i = OPT/(2n(w_2(i+1) - w(i+1)))$ which is decreasing in $i$, whereas $\varepsilon_i = i/(6n)$ is increasing in $i$.

Similar to the result in Section 4.4.1, we find that the $\varepsilon_i/\alpha_i$ term dominates and that the best guarantee is obtained for $i = n-1$. Recall that the threshold graph is created by using a threshold $\tau = \alpha OPT/(5\varepsilon n)$. If we use $\varepsilon = \varepsilon_{n-1} = (n-1)/(6n)$ and $\alpha = \alpha_{n-1}$, we find $\tau = 2(w_2(n) - w(n))/5$.

In Tabel 1, we show the values $\varepsilon_{n-1}, \alpha_{n-1}$, the corresponding $\tau = 2(w_2(n) - w(n))/5$ and worst case guarantee $25\varepsilon_{n-1} + 40\alpha_{n-1}/\varepsilon_{n-1}$. The worst case guarantee is pretty terrible for our data sets, however, one would hope that these parameter settings result in better solutions in practice.

Note that we can use these observations to give a recommendation for how to use the BBGgeneral algorithm in Figure 2 if we do not know $\alpha$ and $\varepsilon$: Define the *center separation* of an instance as the maximum, over all centers in the optimal $k$-median solution, of the distance to the closest other center. It is not hard to show that if $V = X$ (i.e. the only possible cluster centers are points that need to be clustered) then $w_2(n) - w(n)$ is equal to the center separation. It may be more intuitive in practice to "guess" the center separation of an instance. Then, using 2/5 times this value as the threshold in the BBGgeneral algorithm is equivalent to using the $(\varepsilon, \alpha)$ that minimize the value $\varepsilon/\alpha$.

#### 4.4.3 Strong $(1+\alpha, \varepsilon)$-property

For our smallest data sets we also attempted to find the values of $\alpha, \varepsilon$ for which the original $(1+\alpha, \varepsilon)$-property in Definition 1 holds. To do this, we need to enumerate all possible clusterings, order them by non-decreasing $k$-median objective value, and for each solution of value $(1+\alpha)OPT$, find the maximum misclassification of all clusterings with objective value at most $(1+\alpha)OPT$. Since there are $k^n$ possible clusterings, we cannot implement this even for smaller sized instances. However, if $k$ is not too big, there are only $O(n^k)$ different $k$-median solutions. We can enumerate all of these, and compute $(\alpha, \varepsilon)$- pairs based on this list of solutions. Note that for a given $\alpha$, we *underestimate* the value of $\varepsilon$ for which the $(1+\alpha, \varepsilon)$-property holds.

The resulting pairs of $(1+\alpha, \varepsilon)$ for which the wine and iris data sets have the "strong" $(1+\alpha, \varepsilon)$-property are shown in Figure 3. In Table 2 we give the values of $\alpha, \varepsilon$ for which the ratio $\varepsilon/\alpha$ is minimal and the corresponding value $\tau = \alpha OPT/(5\varepsilon n)$. Note that, this value of $\tau$ is in fact an overestimation of the maximum value of $\tau$ which corresponds to some $(\alpha, \varepsilon)$ for which the original $(1+\alpha, \varepsilon)$-property holds. We also show the minimum distance between distinct points; in the wine data set we note that this value is less than the value of $\tau$, indicating that there are no settings of $\alpha, \varepsilon$, for which the original $(1+\alpha, \varepsilon)$-property holds, that give a meaningful BBGgeneral algorithm.

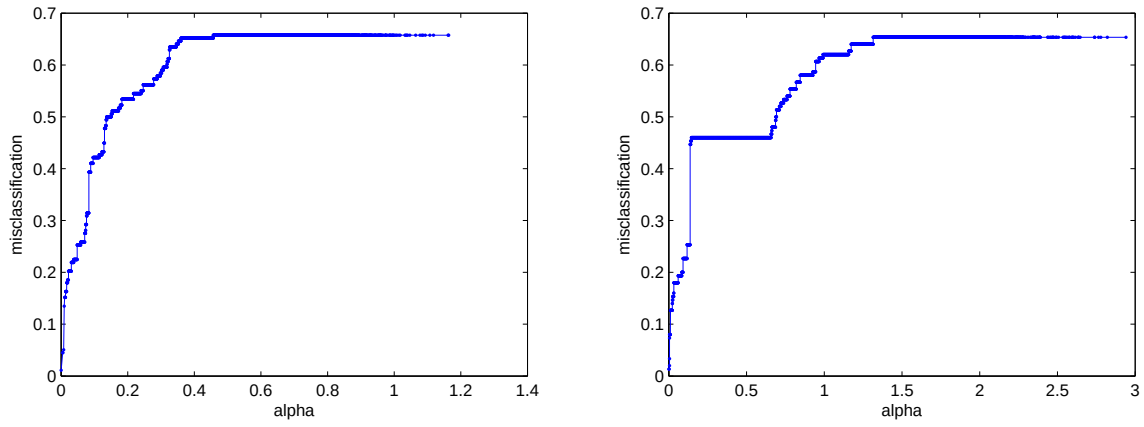
## 5 Comparison to other methods

In light of the discussion above, we will evaluate BBGgeneral by generating the outcome for each different threshold graph using the method in Section 4.3 and choosing the solution with the smallest $k$-median objective value. We compare the quality and running time needed for this solution to various algorithms that are often used in practice to solve $k$-median problems, as well as approximation algorithms that were proposed in the Operations Research and Theoretical Computer Science community. More specifically, we implemented the following algorithms:

- the primal-dual algorithm proposed by Jain and Vazirani [18] (denoted by JV)

| | $\varepsilon$ | $\alpha$ | guar. | $\tau$ | $\tau_{k-\text{med}}$ |
|---|---|---|---|---|---|
| 1 | 0.165 | 0.664 | 1407% | 46.8 | 73 |
| 2 | 0.165 | 0.903 | 1143% | 44.8 | 65 |
| 3 | 0.165 | 0.808 | 1230% | 41.6 | 79 |
| 4 | 0.165 | 1.109 | 1007% | 40.8 | 74 |
| 5 | 0.165 | 2.655 | 661% | 43.6 | 45 |
| 6 | 0.166 | 0.449 | 1891% | 21.2 | 41 |
| 7 | 0.166 | 0.601 | 1519% | 20.4 | 39 |
| 8 | 0.166 | 1.283 | 931% | 34.4 | 32 |
| 9 | 0.166 | 2.205 | 715% | 36.4 | 38 |
| 10 | 0.166 | 4.070 | 578% | 30.8 | 22 |
| 11 | 0.166 | 0.466 | 1840% | 14.4 | 29 |
| 12 | 0.166 | 0.616 | 1494% | 16.4 | 28 |
| 13 | 0.166 | 1.413 | 886% | 24.8 | 27 |
| 14 | 0.166 | 2.485 | 683% | 29.6 | 24 |
| 15 | 0.166 | 2.937 | 642% | 20.4 | 18 |
| 16 | 0.166 | 0.342 | 2359% | 8.4 | 23 |
| 17 | 0.166 | 0.399 | 2082% | 8.4 | 21 |
| 18 | 0.166 | 0.940 | 1123% | 13.6 | 18 |
| 19 | 0.166 | 1.771 | 791% | 15.2 | 16 |
| 20 | 0.166 | 3.414 | 610% | 18.4 | 15 |
| 21 | 0.166 | 0.328 | 2447% | 7.2 | 22 |
| 22 | 0.166 | 0.427 | 1976% | 8.8 | 23 |
| 23 | 0.166 | 1.259 | 944% | 14.0 | 17 |
| 24 | 0.166 | 2.075 | 736% | 14.8 | 16 |
| 25 | 0.166 | 4.353 | 569% | 19.2 | 13 |
| 26 | 0.166 | 0.403 | 2072% | 8.0 | 19 |
| 27 | 0.166 | 0.409 | 2048% | 6.8 | 17 |
| 28 | 0.166 | 2.574 | 676% | 23.2 | 13 |
| 29 | 0.166 | 2.502 | 683% | 15.2 | 14 |
| 30 | 0.166 | 3.999 | 583% | 16.0 | 10 |
| 31 | 0.166 | 0.323 | 2478% | 5.6 | 14 |
| 32 | 0.166 | 0.426 | 1981% | 6.8 | 15 |
| 33 | 0.166 | 1.140 | 1001% | 9.2 | 13 |
| 34 | 0.166 | 3.171 | 627% | 16.4 | 11 |
| 35 | 0.166 | 0.333 | 2416% | 5.2 | 15 |
| 36 | 0.166 | 0.509 | 1723% | 7.6 | 15 |
| 37 | ? | ? | ? | ? | 11 |
| 38 | 0.166 | 0.379 | 2169% | 5.6 | 13 |
| 39 | 0.166 | 0.413 | 2025% | 5.2 | 12 |
| 40 | ? | ? | ? | ? | 11 |
| wine | 0.166 | 0.553 | 1613% | 1.87 | 3.45 |
| iris | 0.166 | 1.043 | 1049% | 1.10 | 1.39 |
| yeast | 0.167 | 2.357 | 695% | 4.39 | 3.40 |
| letter1000 | 0.167 | 0.812 | 1237% | 2.42 | 2.45 |
| letter2000 | 0.167 | 0.780 | 1271% | 2.34 | 3.26 |

Table 1: The "best" $(\alpha, \varepsilon)$ for which the weak $(1+\alpha, \varepsilon)$-property holds: The table shows the values of $(\alpha, \varepsilon)$ for which the guarantee on the misclassification $25\varepsilon + 40\varepsilon/\alpha$ is minimal, the corresponding guarantee, the threshold implied by these settings, and threshold which gives the minimum $k$-median value among all thresholds. For the pmedian data set 37 and 40, we do not have the optimal $k$-median solution, and hence could not compute these values.




(a) wine (b) iris

Figure 3: Pairs of $(1+\alpha, \varepsilon)$ values for which the instance has the strong $(1+\alpha, \varepsilon)$-property

| | $\varepsilon$ | $\alpha$ | $\tau$ | $\min\{d_{uv}\}$ |
|---|---|---|---|---|
| wine | 0.6573 | 1.164 | 0.994 | 1.161 |
| iris | 0.6533 | 2.941 | 0.788 | 0.121 |

Table 2: The "best" $(\alpha, \varepsilon)$ for which the strong $(1+\alpha, \varepsilon)$-property holds: The table shows the values of $(\alpha, \varepsilon)$ for which $\varepsilon/\alpha$ is minimal, the threshold implied by these settings, and the minimum distance in the graph.

- the primal-dual algorithm proposed by Jain, Mahdian, Markakis, Saberi and Vazirani [17] (denoted by JMMSV)
- Lloyd's algorithm [19]
- k-means++ proposed by Arthur and Vassilvitskii [3]
- two variants of Local Search [4]: LSbest which chooses the best improving move, and LSrandom which chooses an improving move at random.

All implementations were done in MATLAB. In Appendix E we briefly discuss some of the choices we made when implementing these algorithms.

Before we discuss the results, we make one remark about our setup which may have a big impact on the comparison. For the pmed data sets, we chose an optimal $k$-median solution as the target clustering. The optimal values for these data sets are known, but the optimal solution is not given. Since the Local Search algorithm found an optimal solution in all but a handful of these data sets, we chose the Local Search solution as the target clustering for these data sets. Note that there may be multiple optimal solutions and hence this choice may have a big impact on the results. This explains in part the far superior performance of Local Search in terms of misclassification. For the remaining pmed data sets we used an Integer Linear Program to find the optimal solution. This approach failed for data sets 37 and 40.

The primal-dual algorithms have a randomized rounding step which is performed 10 times, after which we choose the best solution. Since the other algorithms also make random choices, we run each of them 10 times and choose the best solution. Figure 4 plots the $k$-median objective value divided by $OPT$ against the running time. Because the best misclassification among all the algorithms varies significantly over the different instances, the average or median misclassification is not very informative. However, we cannot compute the misclassification relative to the best possible misclassification (as this means we would divide by zero). We therefore report the misclassification relative to the best algorithm for each instance: we compute this as the difference in the number of points misclassified by each algorithm compared to the best algorithm, divided by the number of points correctly classified by the best algorithm. We note that the LSrandom was the best algorithm for each instance, except for the yeast data set, for which BBGgeneral algorithm gives a slightly better solution. Hence the results look basically the same if we take the misclassification relative to the LSrandom algorithm rather than the best algorithm for each instance. The results are shown in Figure 5. In each figure there is a dot for each instance and algorithm, where the color of the dot indicates to which algorithm the dot corresponds. For the median time and performance, there is a larger marker of the same color, and the name of the algorithm is given next to the larger marker. The results for Lloyd's Algorithm are suppressed, as they are always worse than kmeans++.

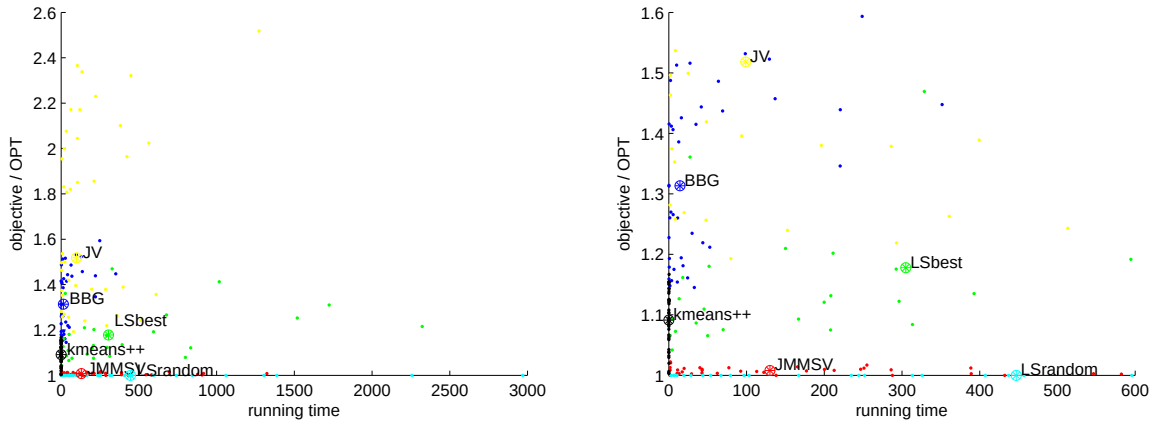
The BBGgeneral algorithm generally outperforms the JV algorithm in terms of both $k$-median objective and misclassification. Also, it is quite a bit faster. The JMMSV algorithm performs very well, but is also slower than the BBGgeneral algorithm. The Local Search algorithm which chooses the best improving move is mostly better than BBGgeneral, but it can be much slower. Perhaps surprisingly, Local Search which chooses an improving move at random gives much better results, although it is also much slower. We should note that we did not try to optimize the speed of the local search algorithms; it should be possible to improve these running times. Finally, kmeans++ has very good performance, although worse than JMMSV and Local Search, but it is extremely fast.

Overall, in terms of performance, Local Search, which chooses random improving moves, is superior, both in terms of $k$-median objective and closeness to the target clustering. The JMMSV algorithm

| | | | BBG general | | | Jain-Vazirani | | | Jain et al. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $n$ | $k$ | obj. | misclass. | time | obj. | misclass. | time | obj. | misclass. | time |
| 1 | 100 | 5 | 116% | 25% | 0.3 | 128% | 36% | 1.6 | 101% | 20% | 1.8 |
| 2 | 100 | 10 | 123% | 36% | 0.2 | 146% | 39% | 2.1 | 100% | 1% | 1.5 |
| 3 | 100 | 10 | 131% | 51% | 0.2 | 150% | 41% | 2.1 | 102% | 30% | 2.4 |
| 4 | 100 | 20 | 131% | 40% | 0.3 | 137% | 24% | 4.0 | 101% | 8% | 1.8 |
| 5 | 100 | 33 | 142% | 36% | 0.4 | 195% | 23% | 5.8 | 102% | 9% | 2.0 |
| 6 | 200 | 5 | 118% | 60% | 0.6 | 126% | 44% | 8.6 | 101% | 28% | 11.3 |
| 7 | 200 | 10 | 119% | 49% | 0.8 | 135% | 47% | 7.6 | 100% | 4% | 10.7 |
| 8 | 200 | 20 | 126% | 33% | 1.3 | 154% | 31% | 8.6 | 100% | 9% | 13.4 |
| 9 | 200 | 40 | 149% | 49% | 2.3 | 183% | 36% | 18.6 | 101% | 9% | 19.8 |
| 10 | 200 | 67 | 141% | 35% | 3.3 | 200% | 26% | 20.5 | 101% | 5% | 17.4 |
| 11 | 300 | 5 | 115% | 54% | 1.6 | 127% | 59% | 19.8 | 100% | 6% | 29.4 |
| 12 | 300 | 10 | 127% | 50% | 2.7 | 150% | 60% | 25.0 | 100% | 9% | 42.8 |
| 13 | 300 | 30 | 141% | 55% | 5.4 | 181% | 51% | 37.2 | 101% | 13% | 40.0 |
| 14 | 300 | 60 | 151% | 49% | 10.1 | 208% | 38% | 33.2 | 101% | 8% | 42.9 |
| 15 | 300 | 100 | 139% | 36% | 12.7 | 205% | 30% | 104.7 | 101% | 12% | 43.0 |
| 16 | 400 | 5 | 116% | 47% | 3.7 | 126% | 60% | 48.0 | 101% | 33% | 77.3 |
| 17 | 400 | 10 | 127% | 60% | 5.7 | 142% | 55% | 48.5 | 100% | 14% | 62.4 |
| 18 | 400 | 40 | 143% | 51% | 16.2 | 182% | 53% | 59.7 | 101% | 13% | 92.6 |
| 19 | 400 | 80 | 152% | 49% | 27.4 | 218% | 41% | 63.6 | 101% | 9% | 113.1 |
| 20 | 400 | 133 | 145% | 41% | 41.9 | 237% | 27% | 104.2 | 101% | 5% | 122.4 |
| 21 | 500 | 5 | 118% | 61% | 6.7 | 119% | 41% | 79.8 | 100% | 2% | 138.1 |
| 22 | 500 | 10 | 126% | 67% | 11.3 | 140% | 60% | 93.7 | 101% | 28% | 165.6 |
| 23 | 500 | 50 | 142% | 55% | 35.0 | 185% | 44% | 106.0 | 101% | 15% | 177.2 |
| 24 | 500 | 100 | 149% | 47% | 63.9 | 217% | 42% | 120.3 | 101% | 12% | 200.2 |
| 25 | 500 | 167 | 154% | 40% | 98.5 | 235% | 30% | 134.8 | 101% | 5% | 212.8 |
| 26 | 600 | 5 | 115% | 63% | 11.0 | 124% | 65% | 152.8 | 102% | 47% | 254.9 |
| 27 | 600 | 10 | 119% | 48% | 16.0 | 138% | 59% | 196.3 | 101% | 26% | 251.1 |
| 28 | 600 | 60 | 144% | 52% | 69.2 | 186% | 48% | 212.2 | 101% | 19% | 286.1 |
| 29 | 600 | 120 | 153% | 53% | 129.6 | 224% | 42% | 222.7 | 102% | 13% | 388.6 |
| 30 | 600 | 200 | 135% | 35% | 220.6 | 253% | 32% | 1273.3 | 101% | 10% | 246.9 |
| 31 | 700 | 5 | 118% | 46% | 18.3 | 122% | 53% | 293.2 | 100% | 4% | 287.7 |
| 32 | 700 | 10 | 123% | 55% | 30.0 | 138% | 57% | 286.1 | 100% | 6% | 389.2 |
| 33 | 700 | 70 | 146% | 60% | 136.8 | 210% | 55% | 381.3 | 102% | 20% | 695.9 |
| 34 | 700 | 140 | 160% | 49% | 248.8 | 233% | 40% | 449.6 | 101% | 9% | 582.3 |
| 35 | 800 | 5 | 116% | 59% | 24.3 | 126% | 63% | 360.8 | 100% | 2% | 432.1 |
| 36 | 800 | 10 | 122% | 61% | 43.8 | 139% | 64% | 399.3 | 100% | 12% | 546.6 |
| 37 | 800 | 80 | 144% | 50% | 220.7 | 197% | 46% | 423.7 | 101% | 15% | 916.0 |
| 38 | 900 | 5 | 115% | 51% | 32.9 | 124% | 59% | 513.3 | 100% | 11% | 878.7 |
| 39 | 900 | 10 | 121% | 59% | 52.7 | 136% | 57% | 611.7 | 100% | 11% | 907.7 |
| 40 | 900 | 90 | 145% | 57% | 351.8 | 203% | 50% | 564.2 | 101% | 14% | 1324.1 |
| wine | 178 | 3 | 522.2 | 24% | 0.6 | 526.5 | 20% | 5.7 | 499.5 | 9% | 5.6 |
| iris | 150 | 3 | 148.3 | 25% | 0.3 | 169.6 | 36% | 3.6 | 132.7 | 14% | 3.2 |
| yeast | 1484 | 10 | 2840.8 | 59% | 210.6 | 3312.8 | 72% | 2514.8 | 2316.7 | 60% | 3482.2 |
| letter1000 | 1000 | 26 | 2826.2 | 75% | 155.2 | 3480.5 | 76% | 651.7 | 2486.4 | 69% | 1173.3 |
| letter2000 | 2000 | 26 | 5764.2 | 76% | 1034.8 | 7039.8 | 78% | 5002.2 | 5015.3 | 69% | 9635.8 |

Table 3: (continued on next page) Comparison of the performance of the different algorithms. "obj." gives the objective value divided by the optimal $k$-median objective value, "misclass." gives the misclassification compared to the target clustering.

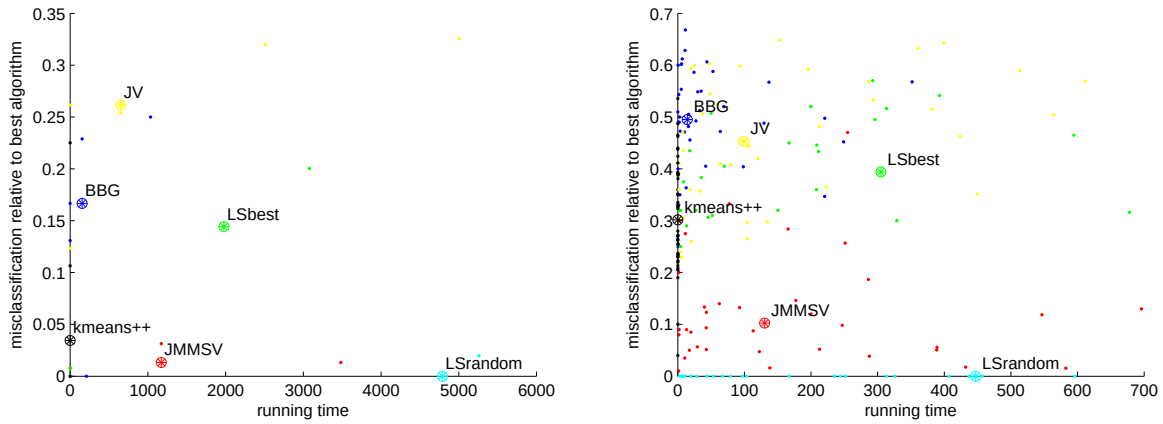
| Local Search (best move) | | | Local Search (random) | | | Lloyd's Algorithm | | | kmeans++ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| obj. | misclass. | time | obj. | misclass. | time | obj. | misclass. | time | obj. | misclass. | time |
| 107% | 32% | 2.0 | 100% | 0% | 2.6 | 137% | 57% | 0.0 | 100% | 4% | 0.0 |
| 109% | 32% | 4.4 | 100% | 0% | 6.2 | 121% | 29% | 0.0 | 109% | 27% | 0.0 |
| 104% | 25% | 4.4 | 100% | 0% | 6.8 | 119% | 40% | 0.0 | 106% | 28% | 0.0 |
| 113% | 29% | 13.3 | 100% | 0% | 19.5 | 128% | 40% | 0.0 | 108% | 23% | 0.0 |
| 136% | 32% | 27.3 | 100% | 0% | 53.8 | 148% | 41% | 0.0 | 112% | 19% | 0.0 |
| 107% | 38% | 8.7 | 100% | 0% | 10.7 | 111% | 34% | 0.0 | 104% | 33% | 0.0 |
| 116% | 44% | 18.1 | 100% | 0% | 28.6 | 107% | 32% | 0.0 | 107% | 22% | 0.0 |
| 118% | 31% | 51.9 | 100% | 0% | 79.2 | 124% | 42% | 0.0 | 109% | 23% | 0.0 |
| 121% | 32% | 150.3 | 100% | 0% | 252.0 | 127% | 34% | 0.1 | 109% | 21% | 0.0 |
| 147% | 30% | 328.8 | 100% | 0% | 843.9 | 146% | 33% | 0.1 | 115% | 24% | 0.1 |
| 109% | 38% | 35.3 | 100% | 0% | 43.9 | 112% | 51% | 0.1 | 101% | 10% | 0.0 |
| 111% | 31% | 45.4 | 100% | 0% | 96.9 | 107% | 33% | 0.0 | 110% | 32% | 0.0 |
| 120% | 43% | 211.3 | 100% | 0% | 407.2 | 123% | 47% | 0.0 | 110% | 39% | 0.1 |
| 127% | 34% | 678.0 | 100% | 3% | 1060.3 | 123% | 37% | 0.1 | 115% | 29% | 0.1 |
| 141% | 33% | 1017.5 | 100% | 0% | 2970.3 | 161% | 40% | 0.1 | 114% | 21% | 0.1 |
| 107% | 51% | 50.0 | 100% | 0% | 67.1 | 103% | 26% | 0.0 | 100% | 44% | 0.0 |
| 108% | 41% | 70.0 | 100% | 0% | 103.3 | 107% | 47% | 0.0 | 106% | 47% | 0.0 |
| 119% | 47% | 594.5 | 100% | 0% | 1387.2 | 112% | 37% | 0.1 | 110% | 34% | 0.1 |
| 131% | 33% | 1724.4 | 100% | 0% | 3593.6 | 132% | 41% | 0.2 | 113% | 26% | 0.2 |
| 143% | 33% | 3239.5 | 100% | 0% | 8860.0 | 148% | 34% | 0.2 | 115% | 22% | 0.3 |
| 109% | 45% | 167.0 | 100% | 0% | 167.1 | 103% | 34% | 0.1 | 103% | 21% | 0.1 |
| 112% | 52% | 199.8 | 100% | 0% | 325.9 | 108% | 43% | 0.1 | 109% | 44% | 0.1 |
| 125% | 45% | 1519.2 | 100% | 0% | 2257.1 | 114% | 34% | 0.2 | 111% | 35% | 0.1 |
| 132% | 41% | 3286.7 | 100% | 0% | 9371.2 | 128% | 38% | 0.2 | 113% | 27% | 0.3 |
| 146% | 33% | 8122.7 | 100% | 0% | 20717.0 | 146% | 37% | 0.4 | 117% | 26% | 0.4 |
| 108% | 36% | 208.1 | 100% | 0% | 234.9 | 106% | 46% | 0.2 | 103% | 33% | 0.1 |
| 112% | 50% | 296.3 | 100% | 0% | 457.0 | 113% | 52% | 0.1 | 108% | 35% | 0.1 |
| 122% | 44% | 2323.2 | 100% | 0% | 5120.3 | 116% | 41% | 0.2 | 114% | 39% | 0.2 |
| 129% | 42% | 6846.2 | 100% | 8% | 16208.0 | 124% | 36% | 0.2 | 112% | 30% | 0.4 |
| 144% | 35% | 12963.0 | 100% | 0% | 35372.0 | 144% | 35% | 0.4 | 116% | 23% | 0.5 |
| 113% | 45% | 208.5 | 100% | 0% | 244.5 | 106% | 49% | 0.1 | 104% | 42% | 0.1 |
| 118% | 57% | 292.6 | 100% | 0% | 437.2 | 113% | 50% | 0.1 | 108% | 49% | 0.1 |
| 124% | 46% | 3590.3 | 100% | 8% | 7193.4 | 116% | 43% | 0.3 | 115% | 38% | 0.3 |
| 133% | 39% | 10435.0 | 100% | 8% | 26795.0 | 131% | 36% | 0.3 | 115% | 31% | 0.5 |
| 108% | 52% | 313.5 | 100% | 0% | 313.4 | 108% | 51% | 0.1 | 104% | 41% | 0.1 |
| 114% | 54% | 392.7 | 100% | 0% | 595.8 | 108% | 59% | 0.2 | 109% | 46% | 0.1 |
| 128% | 46% | 5607.8 | 100% | 0% | 15501.0 | 119% | 42% | 0.4 | 113% | 39% | 0.6 |
| 108% | 47% | 800.0 | 100% | 0% | 739.5 | 107% | 54% | 0.3 | 102% | 38% | 0.2 |
| 112% | 48% | 833.8 | 100% | 0% | 1304.8 | 111% | 52% | 0.2 | 106% | 54% | 0.2 |
| 128% | 50% | 10607.0 | 100% | 0% | 24185.0 | 116% | 44% | 0.4 | 113% | 39% | 0.4 |
| 500.2 | 9% | 3.3 | 499.5 | 9% | 3.4 | 567.8 | 52% | 0.0 | 499.5 | 9% | 0.0 |
| 130.9 | 14% | 2.3 | 130.8 | 13% | 2.0 | 149.4 | 49% | 0.0 | 130.8 | 13% | 0.0 |
| 2552.4 | 67% | 3077.8 | 2304.1 | 60% | 5260.4 | 2449.9 | 66% | 0.4 | 2410.9 | 68% | 0.7 |
| 2726.4 | 73% | 1979.4 | 2477.2 | 68% | 4788.8 | 2659.6 | 72% | 0.3 | 2597.5 | 69% | 0.2 |
| 5506.7 | 74% | 12188.0 | 4990.3 | 68% | 27683.0 | 5469.0 | 73% | 0.8 | 5193.8 | 71% | 0.6 |




(a) Outcomes obtained in less than 3000 seconds, median is over all instances

(b) Zoomed in

Figure 4: Running time versus $k$-median objective divided by OPT for the pmed data sets.




(a) UCI data sets (outcomes obtained in less than 6000 seconds, median is over all instances)

(b) pmed data sets (outcomes obtained in less than 700 seconds, median is over all instances)

Figure 5: Running time versus misclassification relative to best algorithm.

is second in terms of performance, followed by kmeans++. The first two algorithms are much slower however than kmeans++.

Although we found in the previous section that the theoretical guarantees of the BBGgeneral algorithm are meaningless for our data sets, it is clear from our experiments that the algorithm does give reasonable clusterings, and it is fast, even when checking all threshold graphs. However, for our data sets, other algorithms clearly outperform the BBGgeneral algorithm.

# 6 Conclusion and Open Problems

In this paper, we investigate theoretical and practical aspects of a new approach to clustering proposed by Balcan et al. [6]. We show that the assumption needed for their strongest result (the "large" clusters case) defines a set of "easy" instances: instances for which we can approximate the $k$-median objective to within a smaller ratio than for general instances. Our practical evaluations show that our instances do not fall into this category. For the algorithm for the general case, we give some theoretical justification that it may be hard to find the values of parameters $\alpha, \varepsilon$ that are needed as input. We show how to adapt the algorithm so we do not need to know these parameters, but this approach does not come with any guarantees on the misclassification of the resulting clustering. In our experimental comparison, the

performance is reasonable but some existing methods are significantly better.

An interesting direction to evaluate the pratical performance of the algorithms by Balcan et al. [6] would be to test them on "easy" instances, i.e. instances for which the $(1 + \alpha, \varepsilon)$-property holds for values $\varepsilon, \alpha$ for which $\varepsilon/\alpha$ is small, perhaps by identifying a small set of points whose removal ensures that this is the case, which was studied by Balcan, Röglin and Teng [9].

Theoretically, our results also raise the question whether it is possible to show an approximation guarantee for the algorithms for instances that satisfy the $(1 + \alpha, \varepsilon)$-property and for which the target clustering has "large" clusters that were proposed for other objective functions, namely $k$-means and min-sum $k$-clustering, by Balcan et al. [6] and Balcan and Braverman [8]. For the general case, more research into exploiting this property may lead to algorithms which outperform existing methods. On the other hand, it would be interesting to have a lower bound on the misclassification of any (reasonable) algorithm when given an $\alpha, \varepsilon$, such that the $(1+\alpha, \varepsilon)$-property holds. In particular, it would be interesting to know if the dependence on $\varepsilon/\alpha$ in either the guarantee on the misclassification or in the minimum cluster size is unavoidable.

Finally, an interesting direction is to find (other) classes of inputs defined by natural properties for which one can give algorithms that perform better than what is possible for the general class of inputs, both for the $k$-median problem and other optimization problems.

# References


[1] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for Euclidean $k$-medians and related problems. In *STOC '98: Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 106–113. 1999.

[2] D. Arthur and S. Vassilvitskii. How slow is the k-means method? In *SCG '06: 22d Annual Symposium on Computational Geometry*, pages 144–153, 2006.

[3] D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA '07: 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, 2007.

[4] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for $k$-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004.

[5] A. Asuncion and D. Newman. UCI machine learning repository, 2007. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[6] M.-F. Balcan, A. Blum, and A. Gupta. Approximate clustering without the approximation. In *SODA '09: 19th Annual ACM -SIAM Symposium on Discrete Algorithms*, pages 1068–1077, 2009.

[7] M.-F. Balcan, A. Blum, and S. Vempala. A discriminative framework for clustering via similarity functions. In *STOC 2008: 40th Annual ACM Symposium on Theory of Computing*, pages 671–680, 2008.

[8] M.-F. Balcan and M. Braverman. Finding low error clusterings. In *COLT 2009: 22nd Annual Conference on Learning Theory*, 2009.

[9] M.-F. Balcan, H. Röglin, and S.-H. Teng. Agnostic clustering. In *ALT 2009: 20th International Conference on Algorithmic Learning Theory*, volume 5809 of *Lecture Notes in Computer Science*, pages 384–398, 2009.

[10] J. E. Beasley. A note on solving large p-median problems. *European Journal of Operational Research*, 21(2):270–273, August 1985.

[11] J. E. Beasley. OR-Library $p$-median - uncapacitated, 1985. `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/pmedinfo.html`.

[12] Y. Bilu and N. Linial. Are stable instances easy? In *ICS 2010: The First Symposium on Innovations in Computer Science*, pages 332–341, 2010.

[13] M. Charikar and S. Guha. Improved combinatorial algorithms for facility location problems. *SIAM J. Comput.*, 34(4):803–824 (electronic), 2005.

[14] M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the $k$-median problem. *J. Comput. System Sci.*, 65(1):129–149, 2002.

[15] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[16] A. Gupta. personal communication, 2009.

[17] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *J. ACM*, 50(6):795–824, 2003.

[18] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and $k$-median problems using the primal-dual schema and Lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001.

[19] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[20] R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy. The effectiveness of Lloyd-type methods for the k-means problem. In *FOCS'06:47th Annual IEEE Symposium on Foundations of Computer Science*, pages 165–176, 2006.

# A Proofs

**Lemma 3.** *It is NP-hard to verify whether an instance has the (weak) $(\alpha, \varepsilon)$-property for a given $\alpha, \varepsilon$.*

*Proof.* Our reduction is similar to the proof that it is $NP$-hard to approximate the $k$-median objective to within $(1 + 1/e)$. (We note that the $(1 + 2/e)$-hardness of approximation in [17] assumes a slightly different setting from ours, the $(1 + 1/e)$-hardness result for our setting was communicated to us by Gupta [16].) We use a reduction from max $k$-cover, in which we have a groundset of elements $U$ with $|U| = n$, a collection $\mathcal{C}$ of subsets of $U$ and an integer $k$. It is $NP$-hard to decide whether there exist $k$ subsets from $\mathcal{C}$ whose union is $U$ (a "yes" instance), or if the union of any $k$ subsets has at most $(1 - 1/e + \eta)|U|$ elements (a "no" instance) [15].

Give an instance of max $k$-cover, we form a bipartite graph with the elements on the left and the sets on the right. We set $d(i, j) = 1$ if element $i$ is contained in set $j$. All other distances are 2. We now slightly perturb the edge distances that are 1: we number the edges of length 1, say $1, 2, \ldots, a \leq n^2$ and change the length of the $i$-th edge to $1 + i\rho$ for a small value $\rho$. We take $\rho = 1/n^3$ and note that the resulting distance function obeys the triangle inequality.

Our $k$-median instance now has $X$ equal to all the nodes in the bipartite graph, and $V$ is the nodes corresponding to the elements. In a $k$-median solution to this instance, if a point is at distance less than 2 from its center, then it is also the case that its second closest center is at least $\rho$ farther than its closest center. Conversely, if a point is at distance 2 from its center than its second center is as close as its closest center. We call the first type of points "good" and the second "bad".

If we have a "yes" instance, then an optimal $k$-median solution has at most $k$ bad points: there is a k-median solution of value strictly less than $n(1+n^2\rho) \leq n+1$ (locate centers at the points corresponding to the sets in the maximum $k$-cover solution). If a solution has $b$ bad points, then these $b$ points are at distance 2 from their center, at most $k$ points are at distance 0, and the remaining $n - k - b$ points are at distance at least 1. Hence the cost of such a solution is at least $n - k - b + 2b = n + b - k$. Hence a solution can be optimal only if $b < k + 1$.

If we have a "no" instance, then an optimal $k$-median solution has at least $(1/e - \eta)n$ bad points: suppose there is a $k$-median solution with fewer bad points, then its centers cover more than $(1-1/e+\eta)n$ of the elements at distance less than 2. Note that a center on the left can cover only itself at distance less than 2, and we can move such a center to one of the sets containing the element and not decrease the number of elements that are covered at a distance less than 2. Hence we obtain a $k$-cover solution which covers more than $(1 - 1/e + \eta)n$ of the elements, contradicting the fact that we have a "no" instance.

We can now choose $\varepsilon$ and $\alpha$ so that the $(1 + \alpha, \varepsilon)$-property tells us whether or not there are fewer than $(1/e - \eta)n$ bad points, and hence whether we are in the "yes" or "no" case. □

**Lemma 4.** *For a given instance, let $\delta$ be the misclassification of the solution with lowest $k$-median objective value among all outcomes obtained by the BBGgeneral algorithm for all threshold graphs. Then $\delta \neq O(\varepsilon/\alpha)$, even if the instance has the weak $(1 + \alpha, \varepsilon)$-property.*

*Proof.* Consider the following instance: The vertices in the instance are separated into two groups, $V_1$ and $V_2$, and $k = 3$. The two groups are "far apart": for $u \in V_1, v \in V_2$ the distance is $M$. We think of the input as having "light", "medium" and "heavy" vertices. A light vertex is just a regular point, a medium vertex means that there are 2 points arbitrarily close together, and a heavy vertex means that there are $N$ points arbitrarily close together. In $V_1$, we just have two light vertices, $\ell_1(1)$ and $\ell_1(2)$ at distance 2 from each other. In $V_2$, we have 6 vertices; two heavy vertices $h_2(1), h_2(2)$, two medium vertices $m_2(1)$ and $m_2(2)$ and two light vertices $\ell_2(1)$ and $\ell_2(2)$. The distances $d(h_2(i), m_2(i))$ for $i = 1, 2$ and $d(\ell_2(1), \ell_2(2))$ are 1 and the other distances between vertices in $V_2$ is 2.

We'll let the target clustering be the following clustering: $C_1^* = \{\ell_1(1), \ell_1(2)\}$, $C_2^* = \{h_2(1), m_2(1)\}$, $C_3^* = \{h_2(2), m_2(2), \ell_2(1), \ell_2(2)\}$. Note that this clustering has objective value 10 if we choose cluster centers inside the heavy vertices $h_2(1), h_2(2)$ and at either $\ell_1(1)$ or $\ell_1(2)$.

Note that this instance has the weak $(1 + \alpha, \varepsilon)$-property for $\varepsilon > 2/(2N + 8)$ and $\alpha < \frac{2}{5}$, since then $\alpha OPT/2\varepsilon n < 1$, and indeed the only points for which the second closest center is strictly less than 1 farther than the closest center are $\ell_2(1)$ and $\ell_2(2)$.

Note that if we do not split $V_1$ into two clusters, then we misclassify all points in one of the heavy vertices, i.e. at least $N$ out of the $2N + 8$ points. On the other hand, if we do not make a separate

cluster for the points in $V_2$, then the $k$-median objective value is at least $2M$.

We now consider what the algorithm will do for different values of $\tau = \alpha OPT/(5\varepsilon n)$. If $\tau$ is less than the tiny distance inside the heavy and light vertices, then the threshold graph is empty and the algorithm could output any clustering of two singleton clusters and one cluster containing the remaining points. If $\tau$ is greater than that value, but less than 1, the algorithm will make $h_2(1), h_2(2)$ into two clusters and cluster the remaining points together. This solution is close to the target clustering (only 6 out of $2N+8$ points are misclassified) but has $k$-median objective at least $2M$. If $1 \leq \tau < 2$, the algorithm will find clusters $\{h_2(1), m_2(1)\}, \{h_2(2), m_2(2)\}$ and cluster the remaining points together, which misclassifies only 2 points, but again, the $k$-median objective is at least $2M$. Finally, if $\tau \geq 2$, the algorithm just finds two clusters, $V_1$ and $V_2$. This solution misclassifies approximately half the points, but its $k$-median objective is only approximately $2N$. If we choose $N < M$ then this last solution will have the smallest objective value. □

# B Implementation Details of BBGgeneral

We give our implementation which returns the outcome of the BBGalgorithm from Figure 2 for all possible values of $(\alpha, \epsilon)$. Rather than just run the algorithm for all different threshold graphs, we keep track of how many more edges need to be added adjacent to $v$, before $v$ becomes the center of cluster $i$, for every node $v$ and every cluster $i$ (these values are denoted by $\lambda(v, i)$). When adding an edge $\{u, v\}$, we can look at the clusters from the largest to the smallest, and check whether $u$ or $v$ is the cluster center for cluster $i$, or whether $\lambda(u, i)$ or $\lambda(v, i)$ equals 0. In the latter case, we need to regenerate all clusters smaller than cluster $i$. In the first case, say $u$ is the cluster center for cluster $i$. We add $v$ to this cluster, and increase $\lambda(w, j)$ by one for $j > i$ if $v$ was not in clusters $1, \ldots, j$ and $w$ is adjacent to $v$. We regenerate all clusters starting from $j$ if $v$ is the cluster center for center $j$ or the cluster center $c(j)$ is no longer the vertex with largest degree. Note that if there are multiple vertices with the highest degree, we thus force the algorithm to choose the vertex it chose before.

**BBGgeneral practical version for unknown $\alpha, \varepsilon, OPT$, input parameters $(V, d, k)$**

1. let $\lambda(v,i) \leftarrow 0$ for all $v \in V$, $i \in 1, 2, \ldots, k$
2. let $c(i) \leftarrow \emptyset$ for all $i \in 1, 2, \ldots, k$ (the cluster centers)
3. let $C_1 \leftarrow V$, $C_2 \leftarrow C_3 \leftarrow \ldots \leftarrow C_k \leftarrow \emptyset$ (the clusters)
4. let $E \leftarrow \emptyset$ (the edges in the threshold graph)
5. sort distances in increasing order
6. take next pair $\{u, v\}$ for which distance is smallest among all remaining pairs, and add $\{u, v\}$ to $E$
7. let $i \leftarrow 1$
8. if $\lambda(u,i) = 0$ or $\lambda(v,i) = 0$
9. if $u$ or $v$ equals $c(i)$
   (assume $u$ equals $c(i)$, otherwise swap $u$ and $v$)
10. add $v$ to $C_i$
11. increase $\lambda(w,i)$ by 1 for all $w \neq u, v$
12. set $\lambda(v,j) \leftarrow \infty$ for all $j = i+1, i+2, \ldots, k$
13. let $j \leftarrow i+1$
14. while $j \leq k$ and $C_j \not\ni v$
15. adjust $\lambda(w,j)$ for all nodes $w$ that are adjacent to $v$ in $G = (V, E)$, making sure $\min_x \lambda(x,j) = 0$
16. if $v$ equals $c(j)$, or $\lambda(c(j), j)$ is not minimal amongst $\lambda(\cdot, j)$
17. use Subroutine on the graph induced by $V \setminus (C_1 \cup C_2 \cup \cdots \cup C_{j-1})$ with $\ell = k - j + 1$ to determine $C_j, C_{j+1}, \ldots, C_k$ and the accompanying $\lambda$s and $c$s and goto 28
18. end if
19. increase $j$ by 1
20. end while
21. goto 28
22. end if
23. use Subroutine on the graph induced by $V \setminus (C_1 \cup C_2 \cup \cdots \cup C_{i-1})$ with $\ell = k - i + 1$ to determine $C_i, C_{i+1}, \ldots, C_k$ and the accompanying $\lambda$s and $c$s and goto 28.
24. end if
25. decrease $\lambda(u,i)$ and $\lambda(v,i)$ by 1
26. if $C(i) \ni u$ or $C(i) \ni v$ then goto 6.
27. increase $i$ by 1 and goto 8.
28. record the clustering (where all points not in $\bigcup_{i=1}^{k} C_i$ are assigned to the cluster containing the closest among the $k$ high degree vertices chosen by the Subroutine) and goto 6.

**Subroutine, input parameters $(G = (V, E), \ell)$**

1. for $i = 1, \ldots, \ell$
2. choose the highest degree vertex $v$ in $G$ and let $C_i = \{v\} \cup \{u \in V : \{u, v\} \in E\}$.
3. remove $C_i$ and the edges adjacent to them from $G$.
4. end for
5. return $C_1, \ldots, C_\ell$.

# C Number of Solutions of BBGgeneral

| instance | # of solutions | # of distinct sol. | instance | # of solutions | # of distinct sol. |
|---|---|---|---|---|---|
| 1 | 179 | 157 | 24 | 1653 | 1645 |
| 2 | 198 | 187 | 25 | 1404 | 1399 |
| 3 | 229 | 222 | 26 | 1054 | 822 |
| 4 | 248 | 242 | 27 | 1257 | 1113 |
| 5 | 241 | 241 | 28 | 1749 | 1722 |
| 6 | 375 | 321 | 29 | 1929 | 1919 |
| 7 | 411 | 380 | 30 | 2038 | 2030 |
| 8 | 477 | 461 | 31 | 1337 | 1099 |
| 9 | 539 | 537 | 32 | 1704 | 1571 |
| 10 | 509 | 507 | 33 | 2249 | 2237 |
| 11 | 458 | 342 | 34 | 2416 | 2406 |
| 12 | 722 | 685 | 35 | 1487 | 1211 |
| 13 | 783 | 765 | 36 | 2006 | 1878 |
| 14 | 953 | 952 | 37 | 2785 | 2760 |
| 15 | 841 | 838 | 38 | 1669 | 1310 |
| 16 | 700 | 558 | 39 | 2053 | 1861 |
| 17 | 871 | 799 | 40 | 3131 | 3110 |
| 18 | 1199 | 1189 | wine | 285 | 223 |
| 19 | 1274 | 1268 | iris | 236 | 191 |
| 20 | 1278 | 1276 | yeast | 3154 | 2938 |
| 21 | 913 | 742 | letter1000 | 2655 | 2583 |
| 22 | 1146 | 1063 | letter2000 | 5345 | 5195 |
| 23 | 1440 | 1425 | | | |

Table 4: The number of solutions returned by the algorithm from Section 4.3, and the number of distinct solutions among these.

# D Criteria for Choosing a Solution

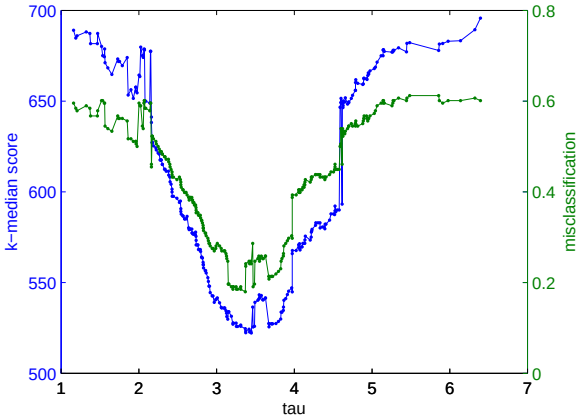
We compare three criteria for choosing a solution with small misclassification among the solutions generated by the algorithm in Section 4.3. Our first criterion is the $k$-median objective value of the solution, which is defined as $\sum_{i=1}^{k} \min_{c \in X} \sum_{v \in C_i} d(c, v)$. Our other two criteria measure how far the threshold graph is from this ideal form by counting the number of edges between clusters plus the number of non-edges within clusters (called the "edit distance to cliques" criterion), or by computing the size of the vertex cover on these edges found by the greedy algorithm (the "greedy vertex cover" criterion).

In Figures 6 to 11, we show the performance of the different criteria for 6 data sets. The pmed data sets were chosen to show an example where all criteria perform well (pmed 5), an example where the misclassification is very unstable and the correlation with all criteria is low (pmed 26), and an example where the $k$-median criterion is outperformed by the other criteria (pmed34). For the UCI data sets, the target clustering is not equal to the optimal $k$-median solution, however, the plots show that the $k$-median criterion nonetheless outperforms the other criteria.

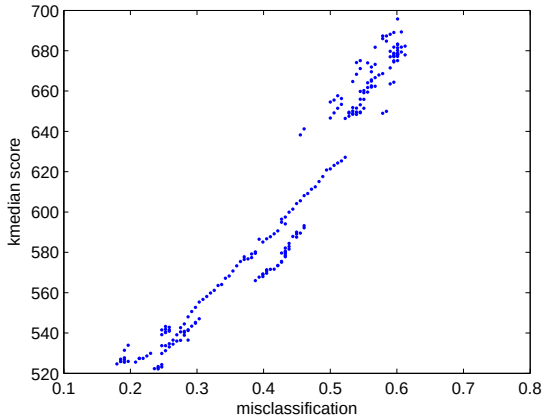
For each data set and each criterion, we show two plots. The first plot shows the value of the criterion of all solutions generated by the BBGgeneral algorithm and their misclassification compared to the target clustering, against the value of the threshold that created this solution. The second plot is a scatter plot of the criterion value against the misclassification. The closer this plot is to a (strictly) monotone function, the better the performance of the criterion in tracking the misclassification.

Note that the "edit distance to cliques" and "greedy vertex cover" criteria are not informative everywhere: if the threshold graph is the complete graph, then the algorithm finds a single cluster and these two criteria are 0. However, the criteria do seem to be informative for roughly the first half of the threshold values. In Table 5, we show the (Pearson's) correlation between the misclassification and the three criteria for each data set, as well as the correlation if we only consider the first half of the threshold values. In terms of correlation on the full set of thresholds, the $k$-median criterion is clearly much better than the other criteria, but if we only look at the first half of the threshold values, the "greedy vertex cover" criterion is better. We remark that these conclusions do not depend on the fact that we used Pearson's correlation. The same conclusion holds for the rank correlation functions Spearman's $\rho$ and Kendall's $\tau$.

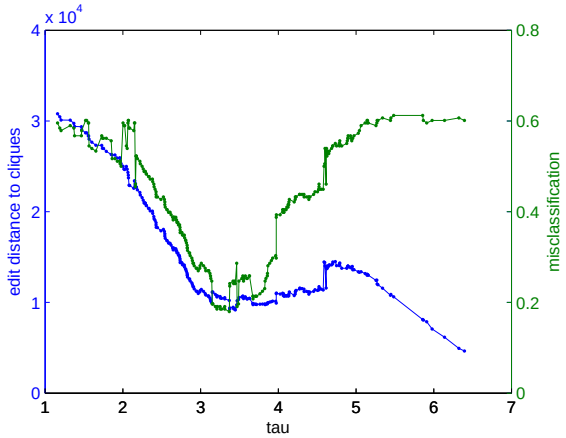
Note however, that we don't necessarily want a criterion that is highly correlated with the misclassification, but that we really want the solution with minimum value for the criterion to have small misclassification. In Table 6 we show the minimum misclassification and the additional misclassification in percentage points for each of the criteria.



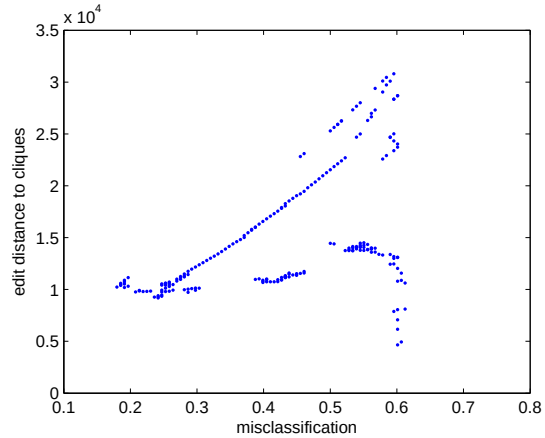

(a) $k$-median value versus $\tau$



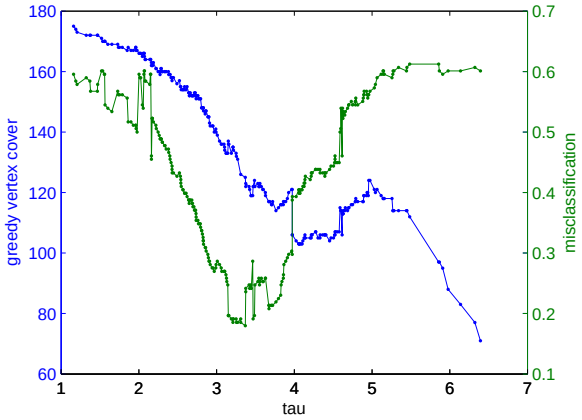

(b) $k$-median value scatter plot



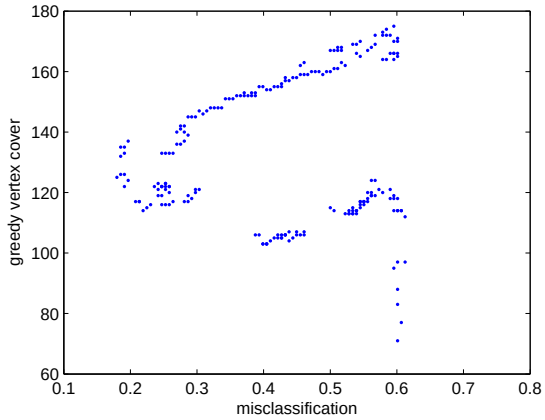

(c) edit distance to cliques versus $\tau$




(d) edit distance to cliques scatter plot

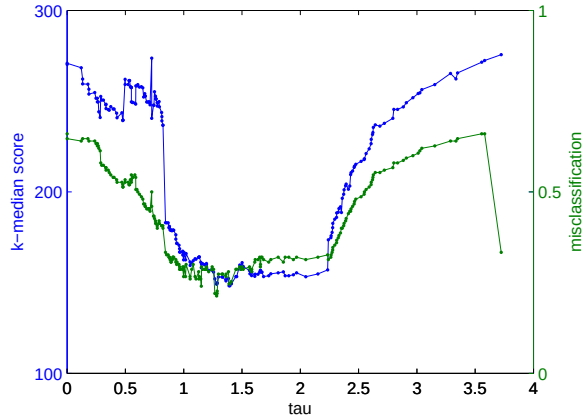



(e) greedy vertex cover versus $\tau$

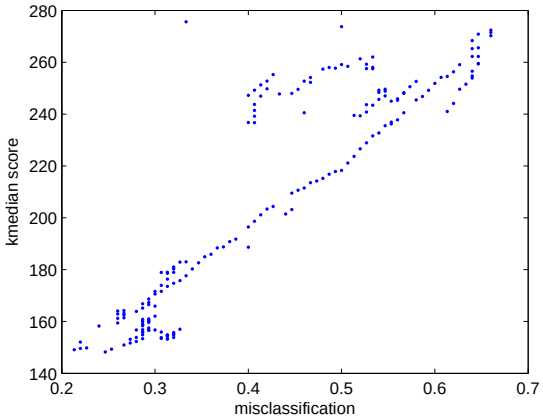



(f) greedy vertex cover scatter plot

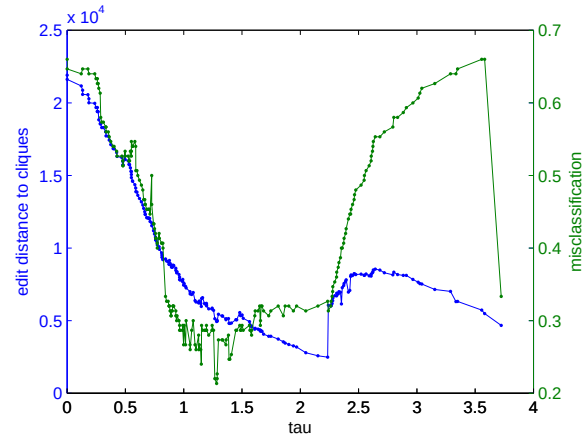
Figure 6: Wine data set



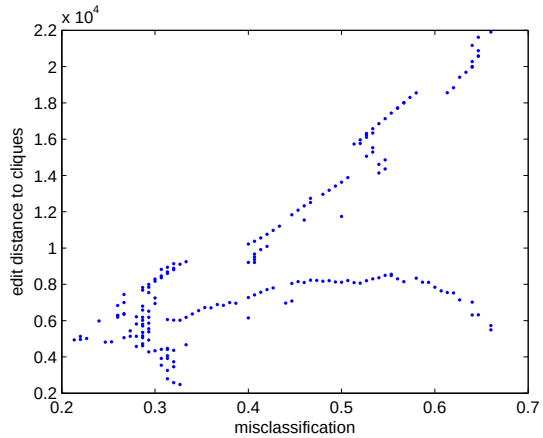

(a) $k$-median value versus $\tau$



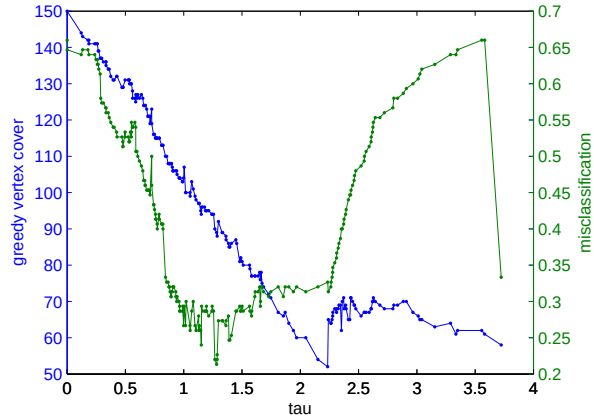

(b) $k$-median value scatter plot



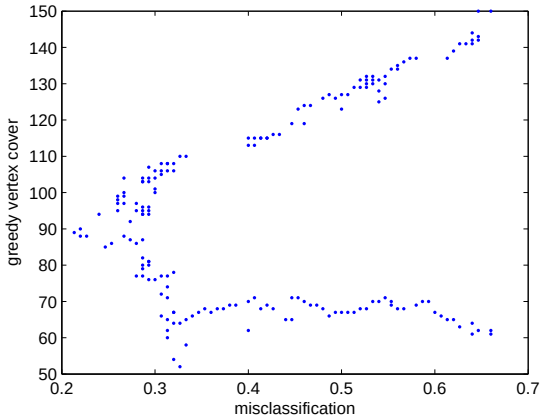

(c) edit distance to cliques versus $\tau$




(d) edit distance to cliques scatter plot

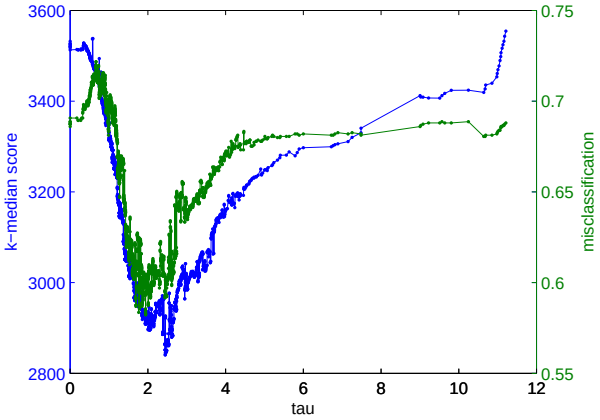



(e) greedy vertex cover versus $\tau$

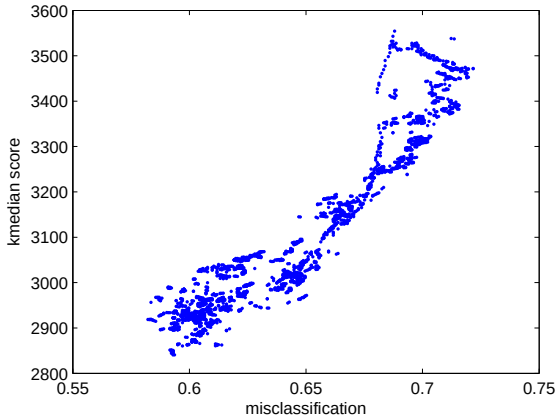



(f) greedy vertex cover scatter plot

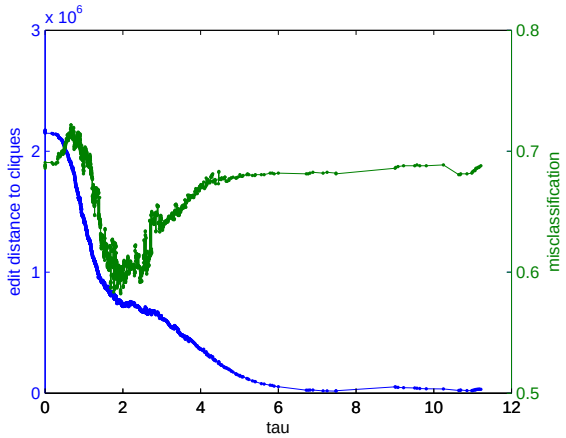
Figure 7: Iris data set



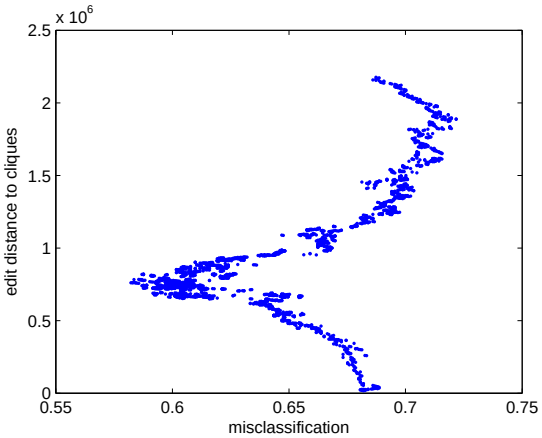

(a) $k$-median value versus $\tau$



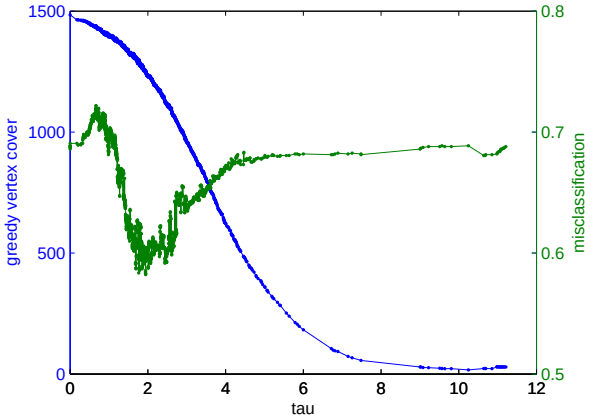

(b) $k$-median value scatter plot



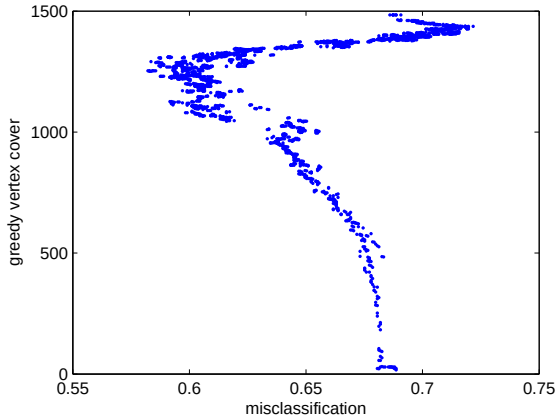

(c) edit distance to cliques versus $\tau$




(d) edit distance to cliques scatter plot

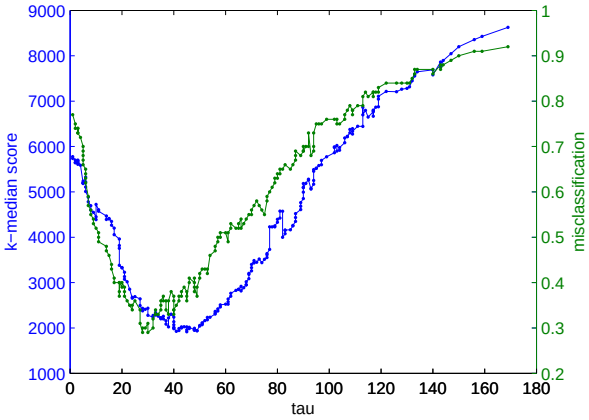



(e) greedy vertex cover versus $\tau$

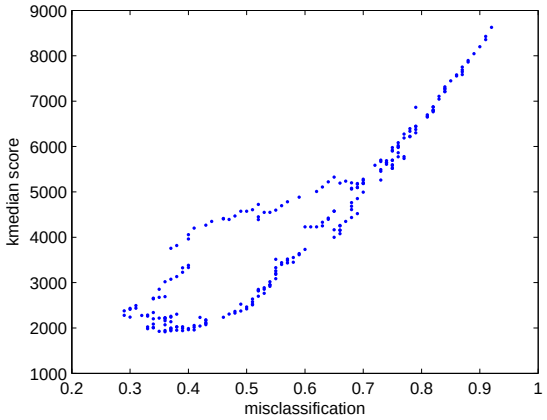



(f) greedy vertex cover scatter plot

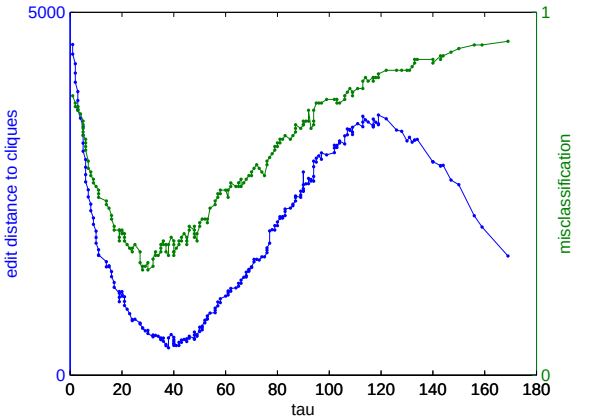
Figure 8: Yeast data set



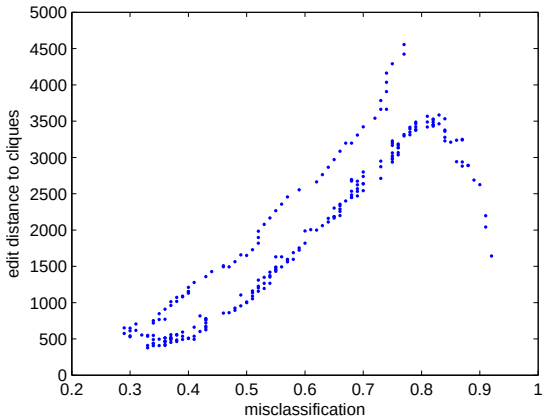

(a) $k$-median value versus $\tau$



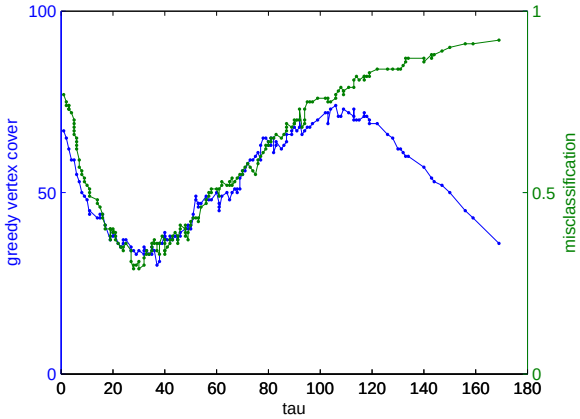

(b) $k$-median value scatter plot



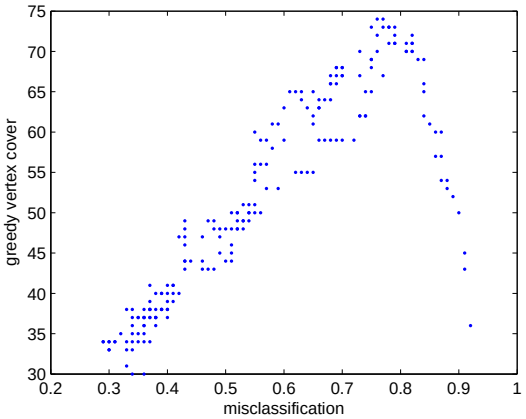

(c) edit distance to cliques versus $\tau$




(d) edit distance to cliques scatter plot

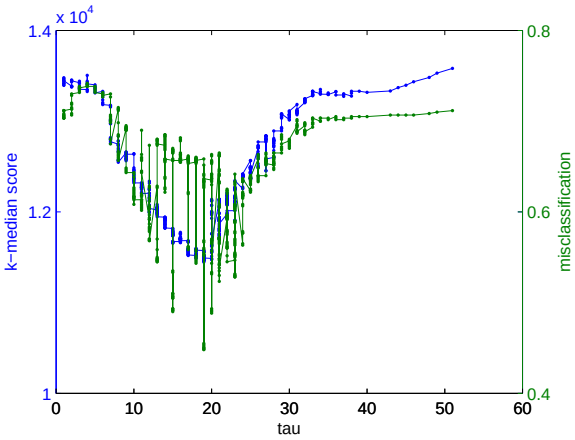



(e) greedy vertex cover versus $\tau$

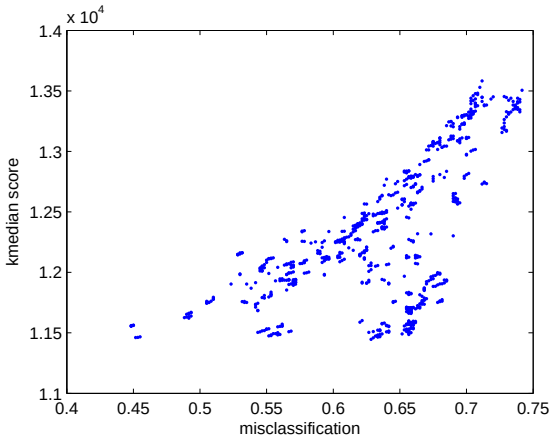



(f) greedy vertex cover scatter plot

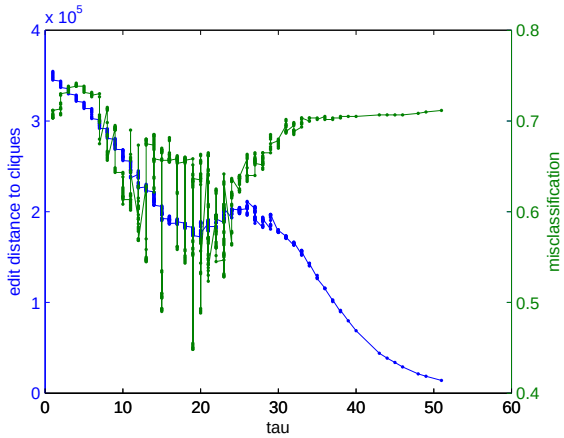
Figure 9: pmed 5 data set



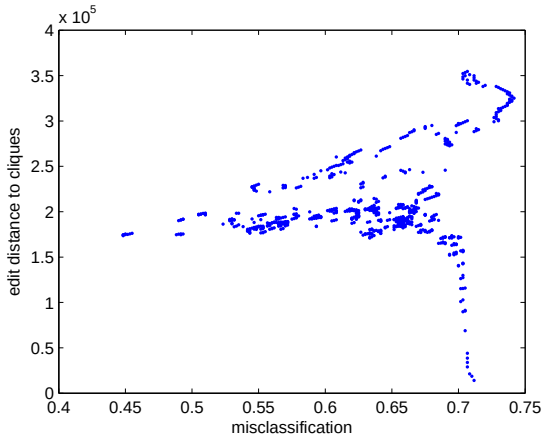

(a) $k$-median value versus $\tau$



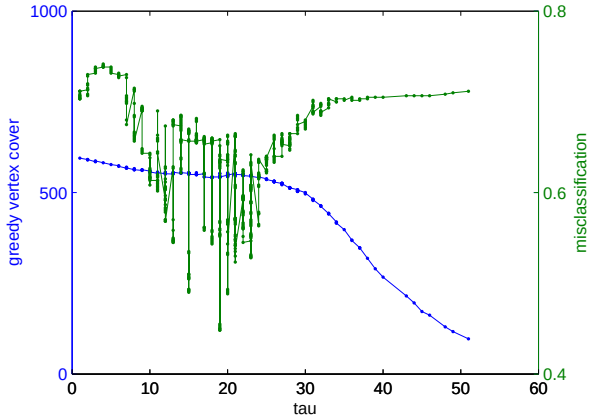

(b) $k$-median value scatter plot



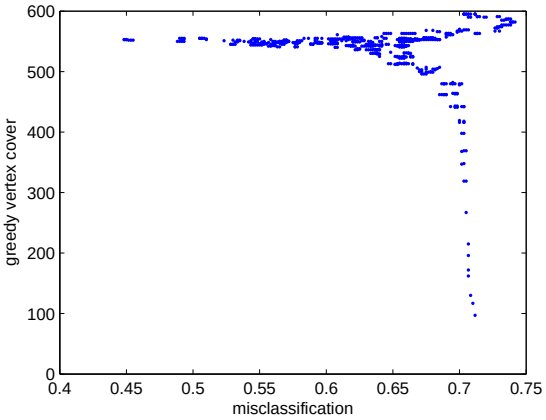

(c) edit distance to cliques versus $\tau$




(d) edit distance to cliques scatter plot

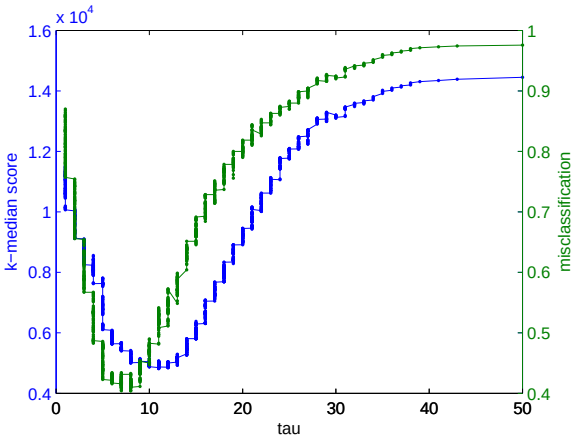



(e) greedy vertex cover versus $\tau$

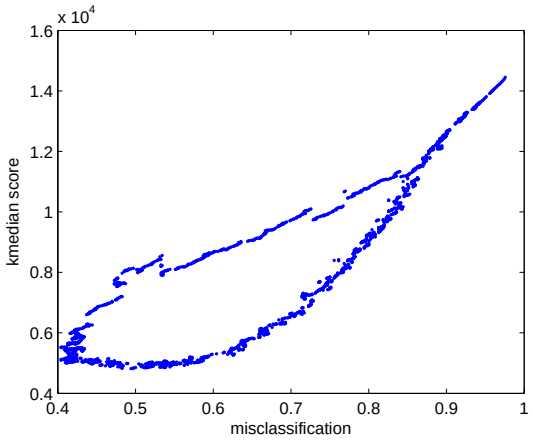



(f) greedy vertex cover scatter plot

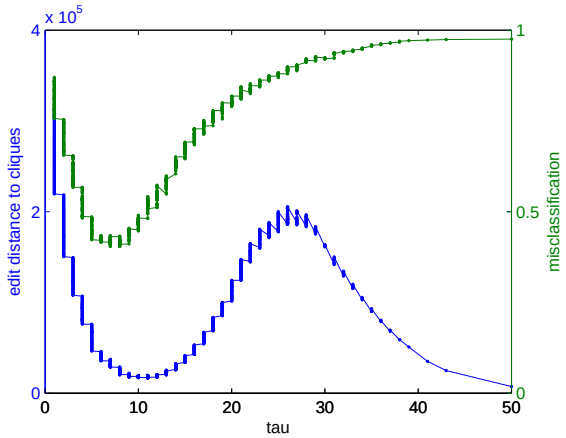
Figure 10: pmed 26 data set



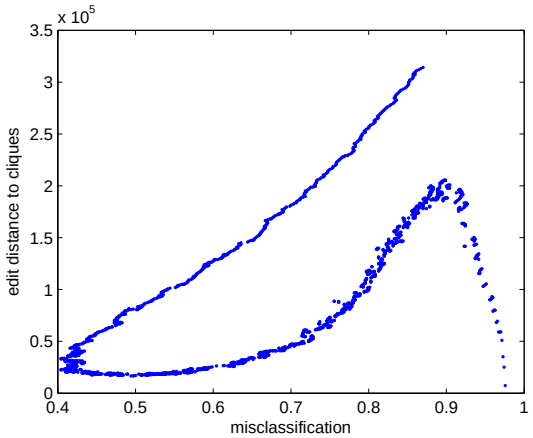

(a) $k$-median value versus $\tau$



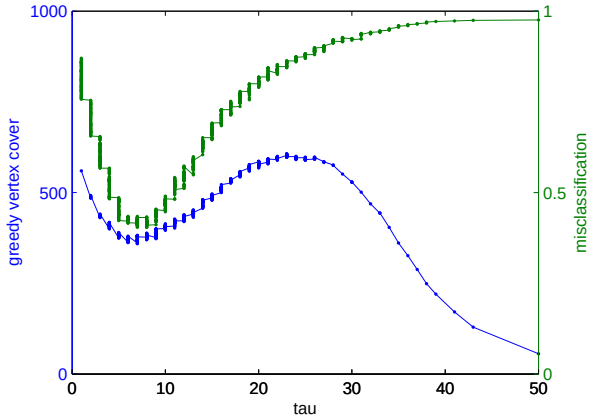

(b) $k$-median value scatter plot



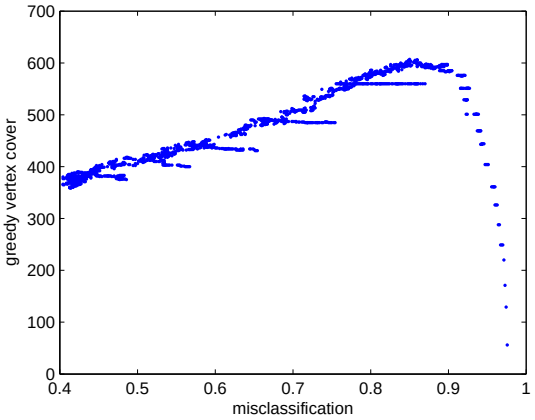

(c) edit distance to cliques versus $\tau$




(d) edit distance to cliques scatter plot




(e) greedy vertex cover versus $\tau$




(f) greedy vertex cover scatter plot

Figure 11: pmed 34 data set

| instance | full | | | half | | |
|---|---|---|---|---|---|---|
| | $k$-median | edit distance | vertex cover | $k$-median | edit distance | vertex cover |
| 1 | 0.92 | 0.53 | 0.05 | 0.93 | 0.93 | 0.92 |
| 2 | 0.88 | 0.56 | 0.50 | 0.95 | 0.90 | 0.93 |
| 3 | 0.83 | 0.47 | 0.34 | 0.92 | 0.95 | 0.96 |
| 4 | 0.92 | 0.82 | 0.83 | 0.93 | 0.94 | 0.93 |
| 5 | 0.93 | 0.92 | 0.86 | 0.85 | 0.95 | 0.97 |
| 6 | 0.91 | 0.24 | -0.31 | 0.88 | 0.89 | 0.92 |
| 7 | 0.91 | 0.50 | 0.00 | 0.97 | 0.96 | 0.94 |
| 8 | 0.91 | 0.67 | 0.39 | 0.94 | 0.94 | 0.95 |
| 9 | 0.92 | 0.74 | 0.64 | 0.91 | 0.95 | 0.96 |
| 10 | 0.97 | 0.89 | 0.81 | 0.94 | 0.97 | 0.96 |
| 11 | 0.92 | 0.06 | -0.40 | 0.95 | 0.94 | 0.91 |
| 12 | 0.90 | 0.48 | -0.12 | 0.95 | 0.96 | 0.95 |
| 13 | 0.92 | 0.67 | 0.29 | 0.94 | 0.97 | 0.96 |
| 14 | 0.87 | 0.74 | 0.67 | 0.84 | 0.93 | 0.91 |
| 15 | 0.96 | 0.89 | 0.89 | 0.93 | 0.96 | 0.89 |
| 16 | 0.79 | 0.25 | -0.28 | 0.72 | 0.64 | 0.71 |
| 17 | 0.87 | 0.33 | -0.27 | 0.92 | 0.94 | 0.93 |
| 18 | 0.87 | 0.64 | 0.53 | 0.92 | 0.94 | 0.94 |
| 19 | 0.89 | 0.77 | 0.83 | 0.82 | 0.89 | 0.97 |
| 20 | 0.92 | 0.81 | 0.90 | 0.74 | 0.74 | 0.97 |
| 21 | 0.84 | 0.07 | -0.45 | 0.87 | 0.88 | 0.87 |
| 22 | 0.90 | 0.35 | -0.39 | 0.89 | 0.90 | 0.86 |
| 23 | 0.84 | 0.63 | 0.50 | 0.88 | 0.93 | 0.97 |
| 24 | 0.91 | 0.78 | 0.85 | 0.90 | 0.93 | 0.94 |
| 25 | 0.95 | 0.87 | 0.90 | 0.89 | 0.91 | 0.97 |
| 26 | 0.61 | 0.23 | -0.26 | 0.56 | 0.56 | 0.51 |
| 27 | 0.92 | 0.40 | -0.30 | 0.96 | 0.96 | 0.90 |
| 28 | 0.89 | 0.71 | 0.59 | 0.93 | 0.94 | 0.95 |
| 29 | 0.91 | 0.77 | 0.83 | 0.88 | 0.94 | 0.96 |
| 30 | 0.93 | 0.80 | 0.93 | 0.81 | 0.73 | 0.94 |
| 31 | 0.95 | 0.32 | -0.37 | 0.96 | 0.95 | 0.88 |
| 32 | 0.88 | 0.40 | -0.26 | 0.92 | 0.87 | 0.87 |
| 33 | 0.85 | 0.69 | 0.79 | 0.80 | 0.86 | 0.96 |
| 34 | 0.86 | 0.75 | 0.88 | 0.78 | 0.86 | 0.97 |
| 35 | 0.79 | 0.00 | -0.42 | 0.74 | 0.78 | 0.79 |
| 36 | 0.86 | 0.34 | -0.29 | 0.79 | 0.74 | 0.87 |
| 37 | 0.85 | 0.64 | 0.82 | 0.81 | 0.80 | 0.97 |
| 38 | 0.90 | 0.17 | -0.39 | 0.90 | 0.87 | 0.87 |
| 39 | 0.92 | 0.39 | -0.30 | 0.91 | 0.87 | 0.96 |
| 40 | 0.85 | 0.63 | 0.78 | 0.81 | 0.83 | 0.96 |
| wine | 0.97 | 0.55 | 0.13 | 0.97 | 0.95 | 0.96 |
| iris | 0.90 | 0.74 | 0.37 | 0.86 | 0.97 | 0.99 |
| yeast | 0.95 | 0.67 | 0.13 | 0.94 | 0.85 | 0.93 |
| letter1000 | 0.93 | 0.56 | -0.09 | 0.94 | 0.96 | 0.96 |
| letter2000 | 0.93 | 0.56 | -0.17 | 0.96 | 0.97 | 0.98 |

Table 5: The correlation between the misclassification and each criterion, computed over all threshold values (full), and over the first half of the threshold values (half).

| instance | minimum misclassification | add. misclass. $k$-median criterion | add. misclass. edit distance criterion (only half of all $\tau$s) | add. misclass. greedy vertex cover criterion (only half of all $\tau$s) |
|---|---|---|---|---|
| 1 | 25% | 0% | 0% | 2% |
| 2 | 29% | 7% | 6% | 6% |
| 3 | 46% | 5% | 2% | 3% |
| 4 | 29% | 11% | 7% | 5% |
| 5 | 29% | 7% | 4% | 7% |
| 6 | 46% | 14% | 3% | 7% |
| 7 | 48% | 2% | 2% | 5% |
| 8 | 33% | 0% | 1% | 1% |
| 9 | 38% | 11% | 6% | 3% |
| 10 | 29% | 7% | 6% | 1% |
| 11 | 50% | 4% | 0% | 2% |
| 12 | 48% | 2% | 2% | 3% |
| 13 | 49% | 6% | 2% | 3% |
| 14 | 38% | 10% | 7% | 5% |
| 15 | 34% | 2% | 2% | 1% |
| 16 | 44% | 3% | 5% | 11% |
| 17 | 59% | 1% | 6% | 7% |
| 18 | 46% | 5% | 5% | 3% |
| 19 | 40% | 10% | 9% | 4% |
| 20 | 33% | 8% | 5% | 2% |
| 21 | 52% | 9% | 8% | 5% |
| 22 | 59% | 8% | 4% | 2% |
| 23 | 50% | 5% | 5% | 6% |
| 24 | 39% | 9% | 10% | 0% |
| 25 | 34% | 7% | 2% | 2% |
| 26 | 45% | 18% | 19% | 21% |
| 27 | 45% | 3% | 4% | 5% |
| 28 | 49% | 4% | 4% | 3% |
| 29 | 43% | 10% | 7% | 3% |
| 30 | 32% | 2% | 1% | 1% |
| 31 | 43% | 3% | 9% | 1% |
| 32 | 51% | 3% | 0% | 1% |
| 33 | 47% | 13% | 12% | 2% |
| 34 | 40% | 9% | 9% | 1% |
| 35 | 52% | 6% | 6% | 8% |
| 36 | 54% | 6% | 6% | 9% |
| 37 | 44% | 6% | 6% | 2% |
| 38 | 44% | 7% | 10% | 8% |
| 39 | 58% | 0% | 1% | 1% |
| 40 | 47% | 10% | 10% | 1% |
| wine | 18% | 6% | 7% | 8% |
| iris | 21% | 3% | 7% | 5% |
| yeast | 58% | 1% | 1% | 4% |
| letter1000 | 74% | 2% | 1% | 2% |
| letter1000 | 75% | 1% | 1% | 1% |

Table 6: The minimum misclassification of the solution among all solutions that are found by the algorithm in Section 4.3 and the additional misclassification in percentage points for each of the criteria.

# E Implementation Details of Other Algorithms

## E.1 Primal-Dual Algorithm proposed by Jain and Vazirani [18]

Jain and Vazirani propose a primal-dual algorithm for the facility location problem with an approximation guarantee of 3. They then show that by using bisection search on the (uniform) facility cost, one can use the primal-dual algorithm to get two solutions, such that the difference in facility costs for the two solutions is very small, and in one solution there are at least $k$ facilities, while in the other solution at most $k$ facilities are open. Since the maximum facility cost that needs to be considered (such that only 1 facility is opened), is bounded by a polynomial in the size of the input, this bisection search runs in polynomial time. Next they give a randomized algorithm (and a derandomized version) to combine these two solutions into one solution with exactly $k$ open facilities. The resulting approximation guarantee is 6 for this algorithm.

At the end of the first phase of the primal dual algorithm, a subset of the tentatively opened facilities has to be chosen. They show that any maximal independent set in an appropriately defined graph (see the paper for details), will result in a solution that has objective value at most a factor 3 worse than the objective value of the optimal solution.

Because of this, we also implemented a slightly modified version the algorithm, that tries 100 random maximal independent sets at the end of the first phase and keeps track of the solutions where the number of open facilities is as close to $k$ on either side as possible. This is really fast (relative to the other operations of the algorithm), and has the potential to stop the bisection search in an earlier stage, maybe even with a solution with $k$ open facilities, which would imply that we have found a solution that is guaranteed to be within a factor 3 of optimal, rather than the guarantee of a factor 6 that we get if we need to combine two solutions. (We note that it is not clear whether the algorithm as originally proposed by Jain and Vazirani, or our variant has a better probability of getting a solution with exactly $k$ open facilities at the end of a run of the primal-dual algorithm.)

Note that it is not necessarily so that there exists a maximal independent set with exactly $k$ nodes, even if there exist maximal independent sets with strictly less, and strictly more than $k$ nodes, as examplified by the star graph. If no solution with exactly $k$ open facilities was found, we ran the randomized procedure 10 times to get multiple solutions of which we chose the best.

## E.2 Primal-Dual Algorithm of Jain, Mahdian, Markakis, Saberi and Vazirani [17]

The second primal-dual algorithm for the facility location problem that we implemented is based on a different linear program, which has a variable for every possible set of clients with exactly one facility. The primal-dual algorithm also differs from the algorithm by Jain and Vazirani [18] in how clients pay toward facilities.

This primal-dual algorithm is used in a similar way as described in the subsection above to obtain an algorithm for the $k$-median problem. Jain et al. [17] show that this gives a 4-approximation for the $k$-median problem.

As for the algorithm in the previous section, we ran the randomized procedure 10 times if it was needed to get a solution with exactly $k$ open facilities.

## E.3 Lloyd's Algorithm

Lloyd's algorithm is often used in practice. This algorithm is so popular that its name is often erroneously conflated with the problem, as for example in Matlab where the function "`kmeans`" is associated with Lloyd's Algorithm for Euclidean instances.

The algorithm works by successively finding an optimal clustering, given cluster centers, and next finding optimal cluster centers, given the clustering. If the clustering does not change after an iteration, then this clustering is returned.

Theoretically Lloyd's algorithm as described in this subsection is not very satisfactory: it is known that it can give arbitrarily bad solutions, and its worst case running time is proved to be superpolynomial [2, 3].

In our implementation of Lloyd's algorithm, we restrict the centers to be points in our metric space. We run the algorithm 10 times, with the initial $k$ centers chosen uniformly at random each time, and return the best among the 10 solutions.

### E.4 k-means++ proposed by Arthur and Vassilvitskii

To remedy the drawback of Lloyd's algorithm, Arthur and Vassilvitskii [3] propose a specific way of choosing centers randomly (called a "seeding" method), that does give provable guarantees for the $k$-means problem. Like for Llloyd's algorithm, we run the algorithm 10 times, with the initial $k$ centers chosen according to the seeding method proposed by Arthur and Vassilvitskii, and return the best among the 10 solutions.

### E.5 Local Search

We implemented the local search algorithm by Arya et al. [4], where a local move consists of "closing" one center and "opening" a different center in its place, and then assigning each point to its closest center. Arya et al. [4] show that a locally optimal solution is a 5-approximation to the optimal $k$-median solution.

We have two versions: one where we always choose the best improving move, and one where we choose a random improving move. The initial set of centers is chosen uniformly at random. The results reported are the best result found after running the algorithm 10 times.

We note that our implementation does not reuse any information from the previous iteration, so running time improvements can be made by better implementations.