# CS264: Beyond Worst-Case Analysis
# Lecture #6: Perturbation-Stable Clustering*

Tim Roughgarden†

January 26, 2017

## 1 Clustering Is Hard Only When It Doesn't Matter

In some optimization problems, the objective function can be taken pretty literally. If one wants to maximize profit or accomplish some goal at minimum cost, then the goal translates directly into a numerical objective function.

In other applications, an objective function is only a means to an end. Consider, for example, the problem of *clustering*. Given a set of data points, the goal is to cluster them into "coherent groups," with points in the same group being "similar" and those in different groups being "dissimilar." There is not an obvious, unique way to translate this goal into a numerical objective function, and as a result many different objective functions have been studied ($k$-means, $k$-median, $k$-center, etc.) with the intent of turning the fuzzy notion of a "good/meaningful clustering" into a concrete optimization problem. In this case, we do not care about the objective function value per se; rather, we want to discover interesting structure in the data. So we're perfectly happy to compute a "meaningful clustering" with suboptimal objective function value, and would be highly dissatisfied with an "optimal solution" that fails to indicate any patterns in the data (which suggests that we were asking the wrong question, or expecting structure where none exists).

The point is that *if we are trying to cluster a data set, then we are implicitly assuming that interesting structure exists in the data.*[1] This perspective suggests that an explicit model of data could sharpen the insights provided by a traditional worst-case analysis framework (cf., modeling locality of reference in online paging). This lecture develops theory to support the idea that *clustering is hard only when it doesn't matter.*[2] That is, clustering instances with "meaningful solutions" are computationally easier to solve than worst-case instances.



†Department of Computer Science, Stanford University, 474 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

[1]There are exceptions. For example, if one is partitioning a graph as part of a divide-and-conquer algorithm, then the partition is only a means to an end and is not interesting in its own right.

[2]This phrase is adapted from the title of [6], which was first suggested by Tali Tishby.

There has been an explosion of work on this idea over the past decade, and we'll only have time to cover some of the most recent developments.

## 2 The $k$-Median Problem

We study the *$k$-median* problem. The input is an $n$-point metric space $(X, d)$, meaning that $d$ is a nonnegative function on $X \times X$ satisfying $d(x, x) = 0$ for all $x \in X$, $d(x, y) = d(y, x)$ for all $x, y \in X$ (symmetry), and, most importantly, the *triangle inequality*:

$$d(x, y) \leq d(x, z) + d(z, y)$$

for all $x, y, z \in X$. That is, the shortest path between two points is a direct hop—intermediate stops can only lengthen the journey.

One generally interprets $d$ as a distance function or (dis)similarity measure. Metric spaces come up all the time in data analysis. For example, $X$ could be a set of points in $\mathbb{R}^m$ (representing images, say) and $d$ could be Euclidean distance (or distance with respect to some other norm). Or $X$ could be a set of strings (e.g., genomes), with $d$ being Hamming distance.[3]

A *$k$-clustering* of a finite metric space is a partition of its points into $k$ non-empty sets.[4] We next posit a specific objective function over $k$-clusterings; the main results in this lecture carry over to other popular choices, as well (see Homework #3).

To define the objective function in the $k$-median problem, consider a $k$-clustering $C_1, C_2, \ldots, C_k$ and a choice of a *center* $c_i \in C_i$ for each cluster. The goal is to minimize the sum of distances from points to their cluster centers:

$$\sum_{i=1}^{k} \left( \sum_{x \in C_i} d(c_i, x) \right). \tag{1}$$

It's redundant to describe both the clusters and the cluster centers. Given a $k$-clustering $C_1, \ldots, C_k$, it's clear how to choose the centers to minimize the objective function: for each cluster $C_i$ independently, choose the point $c_i \in C$ that minimizes the $i$th inner sum in (1) (trying all possibilities, if necessary). Conversely, given a choice of centers $c_1, \ldots, c_k$, the optimal thing to do is to assign each point to its closest center, so that $C_i$ is the set of points closer to $c_i$ than to any other center.

The $k$-median problem is $NP$-hard in the worst case (as are almost all clustering problems). Could the presence of a "meaningful solution" introduce structure that makes the problem easier? To answer this question, we need to commit to mathematical definition of a "meaningful clustering."

[3] For a non-example, think of airline fares!

[4] We'll think of $k$ as known a priori. In some applications one has domain knowledge about what $k$ should be; in others one runs a $k$-clustering algorithm for varying values of $k$ and goes with the solution that is "best" in some sense.

# 3 Perturbation Stability

Intuitively, we want a definition that captures the property of being "clearly optimal," like a uniqueness assumption on steroids. We first define a notion of "nearby" instances.

**Definition 3.1 ($\gamma$-Perturbation)** For $\gamma \geq 1$, a *$\gamma$-perturbation* of a metric space $(X, d)$ is another metric space $(X, d')$ on the same point set such that

$$d'(x, y) \in \left[\tfrac{1}{\gamma} d(x, y), d(x, y)\right].$$

That is, distances only go down in a $\gamma$-perturbation, and they can only go down by a $\gamma$ factor. The interesting case is where different edge lengths are scaled by different factors.[5] Feel free to think of $\gamma$ as 2.

The key definition is this: the optimal solution should be the same in all nearby instances.

**Definition 3.2 (Perturbation Stability)** A $k$-median instance $(X, d)$ is *$\gamma$-perturbation-stable* if there is a $k$-clustering $C_1^*, \ldots, C_k^*$ such that, for every $\gamma$-perturbation $(X, d')$, $C_1^*, \ldots, C_k^*$ is the unique optimal $k$-clustering.

Note that the condition in Definition 3.2 does not assert that the optimal objective function *value* is the same in all $\gamma$-perturbations, and in almost all cases a $\gamma$-perturbation will strictly decrease the optimal value. Definition 3.2 doesn't even assert that the optimal cluster centers $c_1, \ldots, c_k$ stay the same in all $\gamma$-perturbations. It is the partition of the point set $C_1^*, \ldots, C_k^*$—generally what we really care about—that must be invariant over $\gamma$-perturbations.

When $\gamma = 1$, Definition 3.2 is equivalent to asserting the existence of a unique optimal solution. The condition becomes more and more stringent as $\gamma$ increases, and so the $k$-median problem (restricted to $\gamma$-perturbation-stable instances) can only get easier for larger $\gamma$.

Our goal is to prove that, for all $\gamma$ at least a sufficiently large constant (hopefully not too large), there is a polynomial-time algorithm that recovers the optimal clustering in every $\gamma$-perturbation-stable instance. That is, we're hoping that the complexity of the problem switches from $NP$-hard to polynomial-time solvable at some reasonable value of $\gamma$. We'll get bonus points if we can prove our positive results using algorithms that might conceivably be used (or even better, are already being used) in practice.

Recalling the two escape routes from $NP$-hardness in the last lecture, this goal falls under the approach of relaxing correctness. While last time we considered approximation algorithms (which are approximately correct on all instances), here we're looking for algorithms that are always correct on a designated subset of instances (the $\gamma$-perturbation-stable ones). That is, we're looking for a well-motivated and tractable special case of the problem, rather than relaxed guarantees for the general version of the problem.

[5] In the literature, it is common to see the requirement that distances can only go up, but by at most a $\gamma$ factor. The two definitions differ only by a scaling factor of $\gamma$, and there is no difference between the two definitions for the purposes of Definition 3.2.

# 4 Discussion

Why Definition 3.2? In fact, many other formalizations of "clustering instances with a meaningful solution" have been studied. Perturbation stability was first proposed by Bilu and Linial [5] in the context of the Maximum Cut problem; see also Lecture #8. Around the same time, Balcan, Blum, and Gupta [3] proposed the related notion of "approximation stability;" see Lecture #6 from the 2014 version of this course. There are at least three other related definitions, as well; see [2, 4] for surveys. Perturbation stability seems to be emerging as the most well-studied of these definitions, so it is the one we will focus on. But what's the best way to interpret the definition, and positive results for instances that satisfy it?

The motivation behind the $\gamma$-perturbation-stability definition is that, in many clustering applications, the distance function $d$ is heuristic. In some cases, data points really do belong to some normed space and distances can be taken literally. But when the points represent images, proteins, documents, etc., there is no "true" distance function, and various standard distance/dissimilarity measures are used. Solving a clustering problem with such a heuristic distance function and expecting good results implicitly assumes that the optimal solution of the problem is not overly sensitive to small perturbations of the distance function. Put differently, sensitivity of the output to the details of the distance function suggests that the wrong question is being asked. The $\gamma$-perturbation-stability condition is a natural way to make this implicit assumption precise.

There are two obvious complaints about the definition of perturbation stability. First, taken literally, it is pretty strong, and it's not clear that it holds (with reasonable parameter values) in "real-world" instances. However, the set of instances where one can formally prove guaranteed good performance of an algorithm is typically a small subset of the instances where the algorithm performs well empirically. For example, one would hope that the same algorithms would "work well" even if the stability requirement only holds approximately. An important challenge for theory is to prove (perhaps approximate) recovery guarantees under relaxed, more plausible stability notions.

A second critique concerns the difficulty of empirical validation. Given a data set, it is highly non-trivial to estimate the smallest $\gamma$ for which it is $\gamma$-perturbation stable. (How would you approach it?) This is different from typical 20th-century research that identified polynomial-time solvable special cases of $NP$-hard problems, which typically considered instance restrictions that can be verified in polynomial time (e.g., planarity in graphs, 2-SAT and Horn-SAT, polynomially bounded numbers, etc.). But a condition like perturbation stability remains valuable even in the absence of empirical validation. There is a plausible story for why real-world data sets might satisfy some approximation version of perturbation stability (namely, that the optimal solution to a real-world clustering instance should be roughly the same for any reasonable distance measure). For example, for images of cats and dogs, it seems unlikely that minor changes to the (dis)similarity measure between images would yield a highly incorrect classification of the images. And ultimately, we should judge the value of the definition by the extent to which it leads to a better understanding of an existing algorithm or (in this case) the development of new algorithmic ideas that might

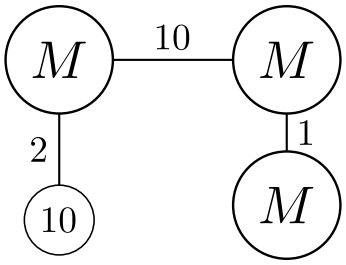


Figure 1: A bad example for single-link clustering when $k = 3$. The circles represent a number of co-located nodes, and the edges are distances. Single-link clustering will cut the "2" and the "10" edge, allocating a single center for the right-hand side.

prove useful for solving the problem.

# 5 Single-Link Clustering

*Single-link clustering* is a simple and widely known clustering algorithm. The idea is to think of the input metric space $(X, d)$ as a complete graph, with vertices $X$ and edge weights given by $d$. The algorithm runs Kruskal's minimum spanning tree (MST) algorithm,[6] except it stops when there are $k$ connected components, where $k$ is the desired number of clusters. That is, we skip the final $k - 1$ edge additions of Kruskal's algorithm.[7]

The simplest statement that might be true is: for $\gamma$ sufficiently large, single-link clustering recovers the optimal $k$-median solution in every $\gamma$-perturbation-stable instance. This would be the best-case scenario where a highly practical algorithm recovers an optimal solution under reasonable conditions. But given that single-link clustering doesn't even look at the objective function that we're optimizing (i.e., the $k$-median objective), this is perhaps too much to hope for. Indeed, the example shown in Figure 1 shows that this statement is false. In the example, there are three cities, each with $M$ co-located citizens, and a tiny village, with 10 co-located citizens. If $k = 4$, then it's clear where to locate the centers — one per city, and one in the village. With $k = 3$, the optimal solution is also clear: locate one center per city, with the 10 villagers each traveling 2 units to the nearest center, for an objective function value of 20. Single-link clustering, however, skips the final 2 iterations of Kruskal's algorithm (i.e., the "2" and "10" edges), thus retaining the "1" edge and forcing the location of a single center between the two cities on the right-hand side. The objective function value of this solution is $M$, arbitrarily larger than the optimal value. Moreover, such instances are $\gamma$-perturbation-stable for arbitrarily large $\gamma$ (as $M \to \infty$); see Homework #3. This example shows that if we want to recover optimal solutions of $\gamma$-perturbation-stable instances, we

[6]Recall this is the algorithm where you sort the input graph's edges from cheapest to most expensive, and then do a single pass through them, including an edge in the tree-so-far if and only if it does not create a cycle.

[7]Each edge addition fuses two connected components into one and hence decreases the number of components by 1; at some point, there are exactly $k$ connected components remaining.

need a better algorithm than single-link clustering. But a variant of single-link clustering works!

# 6 Single-Link++

We now describe `single-link++`, a more sophisticated version of single-link clustering.

`single-link++`

1. Run Kruskal's algorithm until completion to compute the minimum spanning tree $T$ of the complete graph induced by $(X, d)$.
2. Among all $\binom{n-1}{k-1}$ subsets of $k-1$ edges of $T$ and the induced $k$-clusterings (with one cluster per connected component), compute the one with the minimum $k$-median objective function value.[8]

You'd be right to wonder how to implement the second step—the number of possibilities is scaling exponentially with $k$. Have we just replaced one $NP$-hard problem another? Fortunately, many problems that are $NP$-hard in general are polynomial-time solvable on trees, usually via a dynamic programming algorithm. This is also the case here, although the dynamic program is relatively elaborate (too hard for CS161, but just right for CS264) — see Homework #3. We will not discuss further the running time of `single-link++`.

Observe that, in the example in Figure 1, `single-link++` recovers the optimal solution (leaving the 2 but deleting the 1 and 10 edges). Could the algorithm be correct in general? Presumably not, since the algorithm runs in polynomial time and, in general, the $k$-median problem is $NP$-hard. (See Homework #3 for an explicit example.)

It is easy to characterize the inputs for which the algorithm is correct.

**Lemma 6.1** *`single-link++` recovers the optimal solution of a $k$-median instance $(X, d)$ if and only if every optimal cluster $C_i^*$ induces a connected subgraph of the minimum spanning tree $T$.*[9]

See Figure 2 for a cartoon of an instance that does not satisfy the condition in Lemma 6.1.

*Proof of Lemma 6.1:* The `single-link++` algorithm can only output a $k$-clustering obtained by removing $k-1$ edges from the MST $T$. Such an output necessarily produces clusters that are connected subgraphs of $T$. Thus if some optimal cluster $C_i^*$ is not a connected subgraph of $T$, the `single-link++` algorithm has no chance of finding it.

Conversely, every partition of $X$ into $k$ (non-empty) connected subgraphs of $T$ can be obtained by deleting $k-1$ edges from $T$ (namely, every edge with an endpoint in two different

[8] The centers of the $C_i$'s are computed independently and optimally, as usual.

[9] If $(X, d)$ has multiple optimal solutions, it is enough that one them satisfies this condition.

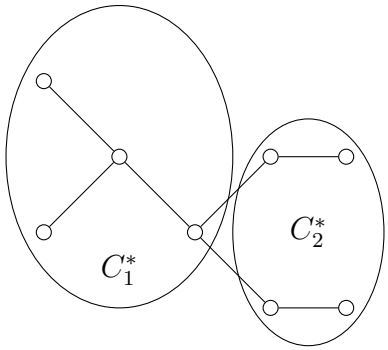


Figure 2: Optimal cluster $C_2^*$ is not connected in this MST.

optimal clusters). Since the `single-link++` algorithm explicitly optimizes over $k$-clusterings of this form, if the optimal algorithm has this form, then the algorithm will recover it. ■

The main result of this lecture is the following.

**Theorem 6.2 ([1])** *In every 2-perturbation-stable $k$-median instance, the `single-link++` algorithm recovers the optimal solution (in polynomial time).*

Theorem 6.2 identifies a novel tractable special case of the $k$-median problem. To the extent that we believe that "real-world" clustering instances with "meaningful solutions" are 2-perturbation-stable, Theorem 6.2 gives a formal sense in which clustering is hard only when it doesn't matter. It is a largely open research direction to prove robust versions of Theorem 6.2, where perturbations can cause a small number of points to switch clusters. (See [3] for such robust results under a stability notion that is more stringent than perturbation stability.)

# 7 Proof of Theorem 6.2

The heavy lifting in the proof of Theorem 6.2 is largely done by the following lemma, which identifies structure that is automatically possessed by any $\gamma$-perturbation-stable instance.

**Lemma 7.1** *For every $\gamma \geq 1$, if $(X, d)$ is a $\gamma$-perturbation-stable $k$-median instance with optimal $k$-clustering $C_1^*, \ldots, C_k^*$ and optimal centers $c_1, \ldots, c_k$, then for every cluster $i$ and point $x \in C_i$ of the corresponding cluster,*

$$d(x, c_j) > \gamma \cdot d(x, c_i) \tag{2}$$

*for every $j \neq i$.*

In every $k$-median instance, every point is assigned to the closest center in any optimal solution. Lemma 7.1 strengthens this property for $\gamma$-perturbation-stable instances: every point is significantly closer to its own center (in the optimal solution) than to any other center.

*Proof of Lemma 7.1:* We prove the contrapositive. Consider a $k$-median instance with optimal $k$-clustering $C_1^*, \ldots, C_k^*$ and corresponding centers $c_1, \ldots, c_k$. Suppose there is a point $x \in C_i^*$ with

$$d(x, c_j) \leq \gamma \cdot d(x, c_i). \tag{3}$$

The plan is to exhibit a $\gamma$-perturbation (Definition 3.1) that certifies that this $k$-median instance is not $\gamma$-perturbation-stable. The rough idea is to decrease the distance between $x$ and $c_j$ so that $x$ is equidistant between $c_i$ and $c_j$; then, reassigning $x$ from $c_i$ to $c_j$ will yield another optimal clustering, violating perturbation stability.

As a notational shorthand, write $r$ for $d(x, c_i)$. Since we start with an optimal solution for $(X, d)$, we have $d(x, c_j) \geq r$ for every $j \neq i$ (otherwise we would reassign $x$ from $c_i$ to $c_j$).

We define a $\gamma$-perturbation $(X, d')$ of $(X, d)$ using the following steps:

1. Let $G$ denote the complete graph on $X$, with the length of edge $(y, z)$ defined as $d(y, z)$.
2. Obtain $G'$ from $G$ by reducing the length of the edge $(x, c_j)$ from $d(x, c_j)$ to $r$.
3. Define $d'(y, z)$ as the length of a shortest path between $y$ and $z$ in $G'$.

We need to argue that $(X, d')$ is indeed a $\gamma$-perturbation. In the second step, by (3), we decreased the length of the edge $(x, c_j)$ by at most a $\gamma$ factor (and all other lengths stayed the same). This means that the length of any path in $G'$ is at most an $\gamma$ factor less than its length in $G$. In particular, the length of a shortest path between any two points only drops by an $\gamma$ factor, so $d'(y, z) \geq \frac{1}{\gamma} d(y, z)$ for every $y, z \in X$. Since all edge lengths in $G'$ are only less than those in $G$, we also have $d'(y, z) \leq d(y, z)$ for every $y, z \in X$. Finally, as shortest-path distances, $(X, d')$ is automatically a metric (for any $y, z, w$, one option for an $y$-$z$ path is to concatenate shortest paths from $y$ to $w$ and from $w$ to $z$, and the shortest $y$-$z$ path can only be shorter).[10]

We can complete the proof by showing that, in the $\gamma$-perturbation $(X, d')$, $C_1^*, \ldots, C_k^*$ is not the unique optimal clustering. The idea is that reassigning $x$ from $c_i$ to $c_j$ should produce an equally good clustering. But there is a missing step: we forced $x$ to be equidistant from $c_i$ and $c_j$ in the second step of construction, but does this property continue to hold for the shortest-path distances $d'$ constructed in the third step?

The final part of the proof shows that $d$ and $d'$ agree everywhere inside $C_i^*$ and $C_j^*$ (i.e., shrinking the distance between $x$ and $c_j$ does not shrink any distances inside either cluster). We denote by $d_G(P)$ and $d_{G'}(P)$ the length of a path $P$ in $G$ and $G'$.

Consider two points $y, z \in C_i^*$, and let $P$ be a shortest $y$-$z$ path in $G'$ (see Figure 3). If $P$ does not contain the newly shorter edge $(x, c_j)$, then $d_G(P) = d_{G'}(P)$ and the shortest $y$-$z$ path distance is the same in $G$ and $G'$. If $P$ does contain the edge $(x, c_j)$, then it either has the form

$$\underbrace{y \rightsquigarrow x}_{:=P'} \to \underbrace{c_j \rightsquigarrow z}_{:=P''}$$

[10] The requirement that $d'$ is a metric is the reason the construction goes through the graphs $G$ and $G'$. If we tried to just modify $d$ directly by shrinking the distance between a pair of points, we might violate the triangle inequality (why?).

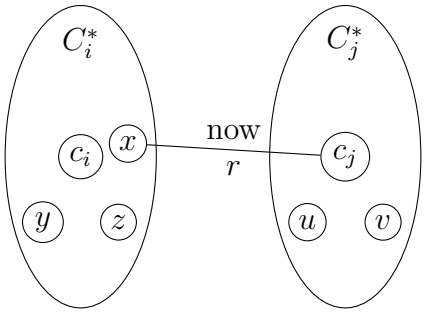


Figure 3: Clusters $C_i^*$ and $C_j^*$ as we reduce the length of $(x, c_j)$.

or

$$y \rightsquigarrow c_j \rightarrow x \rightsquigarrow z,$$

where the subpaths between $y, z$ and $x, c_j$ do not contain the edge $(x, c_j)$. Suppose $P$ is of the former type. Let $Q$ consist of the the same $y \rightsquigarrow x$ subpath $P'$, followed by the edge $(x, c_i)$, followed the edge $(c_i, z)$ (removing any cycles, if necessary). Since $P'$ does not include the edge $(x, c_j)$, we have $d_G(P') = d_{G'}(P')$. We also have $d(x, c_i) = r = d'(x, c_j)$. Finally, since $(X, d)$ is a metric, $d_G(x, c_i)$ is at most $d_G(P'')$; since $P'$ does not contain $(x, c_j)$, $d_G(P'') = d_{G'}(P'')$. We conclude that

$$d_G(Q) = d_G(P') + d_G(x, c_i) + d_G(x, c_i) \leq d_{G'}(P') + d_{G'}(x, c_j) + d_{G'}(P'') = d_{G'}(P),$$

and hence the length of a shortest path between $y$ and $z$ is the same in $G'$ as it is in $G$ (and hence $d(y, z) = d'(y, z)$). The arguments for paths of the second type, and for pairs $y, z \in C_j^*$, are similar. ■

Our last two lemmas are much shorter. The first lemma is where the "2" comes in.

**Lemma 7.2** *In every 2-perturbation-stable $k$-median instance $(X, d)$, for every optimal cluster $C_i^*$, point $x \in C_i^*$, and point $y \notin C_i^*$, $d(x, c_i) < d(x, y)$.*

That is, in a 2-perturbation-stable instance, every point is closer to its center (in the optimal clustering) than it is to any other point of any other optimal cluster.

*Proof of Lemma 7.2:* Suppose $y \in C_j^*$. Using the triangle inequality, Lemma 7.1, and the triangle inequality again, we obtain

$$\begin{aligned} d(x, y) &\geq d(x, c_j) - d(y, c_j) \\ &> 2d(x, c_i) - \frac{d(y, c_i)}{2} \\ &\geq 2d(x, c_i) - \frac{d(x, y) + d(x, c_i)}{2}; \end{aligned}$$

rearranging gives

$$\frac{3}{2}d(x,y) > \frac{3}{2}d(x,c_i)$$

and proves the lemma. ■

Finally, we can complete the proof of Theorem 6.2.

*Proof of Theorem 6.2:* It is enough to show that the correctness condition in Lemma 6.1 holds—that is, in every 2-perturbation-stable $k$-median instance, every optimal cluster $C_i^*$ induces a connected subgraph of $T$.

We proceed by contradiction. If not, there is a point $x \in C_i^*$ such that the (unique) $c_i$-$x$ path in $T$ concludes with the edge $(y, x)$ with $y \notin C_i^*$. At the time $(y, x)$ was added by Kruskal's algorithm, $x$ and $c_i$ were in different connected components (otherwise the addition of $(y, x)$ would have created a cycle). Thus, Kruskal's algorithm also had the option of including the edge $(x, c_i)$ instead. Since the algorithm chose $(y, x)$ over $(x, c_i)$, $d(x, y) \leq d(x, c_i)$. But then $x$ is as close to $y \notin C_i^*$ as its own center, contradicting Lemma 7.2. ■

# References


[1] H. Angelidakis, K. Makarychev, and Y. Makarychev. Algorithms for stable and perturbation-resilient problems. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC)*, 2017. To appear.

[2] P. Awasthi and M.-F. Balcan. Center based clustering: A foundational perspective. In C. M. Henni, M. Meila, F. Murtagh, and R. Rocci, editors, *Handbook of Cluster Analysis*. CRC Press, 2015.

[3] M.-F. Balcan, A. Blum, and A. Gupta. Approximate clustering without the approximation. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1068–1077, 2009.

[4] S. Ben-David. Computational feasibility of clustering under clusterability assumptions. Manuscript, 2014.

[5] Y. Bilu and N. Linial. Are stable instances easy? *Combinatorics, Probability & Computing*, 21(5):643–660, 2012.

[6] A. Daniely, N. Linial, and M. Saks. Clustering is difficult only when it does not matter. `http://arxiv.org/abs/1205.4891`, 2012.