Online k-Median with Consistent Clusters

Benjamin Moseley
Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA, USA

Heather Newman
Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA, USA

Kirk Pruhs
Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA

Abstract

We consider the problem in which n points arrive online over time, and upon arrival must be irrevocably assigned to one of k clusters where the objective is the standard k-median objective. Lower-bound instances show that for this problem no online algorithm can achieve a competitive ratio bounded by *any* function of n. Thus we turn to a beyond worst-case analysis approach, namely we assume that the online algorithm is a priori provided with a predicted budget B that is an upper bound to the optimal objective value (e.g., obtained from past instances). Our main result is an online algorithm whose competitive ratio (measured against B) is solely a function of k. We also give a lower bound showing that the competitive ratio of every algorithm must depend on k.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases k-median, online algorithms, learning-augmented algorithms, beyond worst-case analysis

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2024.20

Category APPROX

Related Version *Full Version*: `https://arxiv.org/abs/2303.15379` [20]

Funding *Benjamin Moseley*: Supported in part by a Google Research Award, an Inform Research Award, a Carnegie Bosch Junior Faculty Chair, and NSF grants CCF-2121744 and CCF-1845146.
Heather Newman: Supported in part by a Google Research Award, an Inform Research Award, a Carnegie Bosch Junior Faculty Chair, and NSF grants CCF-2121744 and CCF-1845146.
Kirk Pruhs: Supported by NSF grants CCF-1907673, CCF-2036077, CCF-2209654 and an IBM Faculty Award.

1 Introduction

Clustering problems, such as k-means clustering and k-median clustering, are a classic genre of learning / data mining problems [5]. Typically the input consists of a collection $X = \{x_1, \ldots, x_n\}$ of points in some metric space $\mathcal{M}$ (typically $\Re^d$ with the 1-norm or 2-norm) and a positive integer k. Typically k is a small constant [2, 5]. The output for a **center-based clustering problem** is a collection $c_1, \ldots, c_k$ of k points from X, called centers, that succinctly summarize the data points. The implicit cluster C_i corresponding to the center c_i is the collection of points in X whose closest center is c_i, that is $C_i = \{x_j \mid \arg\min_{h \in [k]} d(x_j, c_h) = i\}$, where $d(\cdot, \cdot)$ is the distance function for the metric space. The output for a **cluster-based clustering problem** is a partition $C_1, \ldots C_k$ of X into k parts, called clusters. The implicit center of each cluster C_i is then $c_i = \arg\min_{x_h \in C_i} \sum_{x_j \in C_i} d(x_h, x_j)$. For both center-based clustering and cluster-based clustering, the objective is to minimize the cost of the clustering. This paper considers the k-median objective which is the aggregate distance from each point to the center of its cluster, that is $\sum_{i=1}^{k} \sum_{x_j \in C_i} d(x_j, c_i)$.

Here we consider applications where the data points in X arrive online over time. In an **online center-based clustering problem**, the online algorithm maintains a collection of centers. In a **online cluster-based clustering problem**, the online algorithm needs to assign the data points to a cluster when they arrive, that is each point x_j needs to be assigned a label $\ell_j \in [k]$ when x_j arrives. In either case, these choices should (ideally) be irrevocable.

An application of online clustering given by [16] is the task of clustering news articles that arrive online, e.g., at Yahoo news or Google news. We refer to these outlets as the news providers. The news provider selects some (approximately) fixed number k of articles to feature on the news homepage, and has a "view complete coverage" link next to each article to see all the news stories on this topic. The problem of selecting the best k articles that summarize all current news articles is better modeled as a center-based clustering. The problem of partitioning all news articles into clusters of similar articles is better modeled as a cluster-based clustering. Other applications can be found in [17, 14], but we will the news story clustering application as our running canonical example.

A line of research [12, 11, 14, 17] within online clustering goes by moniker of consistent clustering. Research on consistent clustering studies the tradeoffs between the following objectives:

- **Maximizing the Quality of the Clustering:** One seeks a clustering of small cost. The most common metric to measure the quality of a solution is the ratio of the cost of this solution to cost of the optimal solution. The most common metric to measure the quality of an online algorithm is the competitive ratio, which is the maximum (over all inputs) of the ratio of the cost of the online algorithm's solution to the optimal cost.
- **Maximizing Consistency:** Ideally one would like the centers in a center-based problem, or the clusters (labels of points) in a cluster-based problem, to be consistent over time. That is, they should change as little as possible. E.g., the news provider does not want the clusters to completely change every time a new news article is written.

1.1 Prior Work on Consistent Clustering

k-median clustering is NP-hard, but constant-factor approximation algorithms are known [9, 13, 1, 15, 6].

All prior algorithmic research on consistent clustering that we are aware of [12, 11, 14, 17, 4] is center-based. That is, the online algorithm explicitly maintains a collection of centers, and the clustering is implicit; each point is associated with the closest center, but there are no restrictions on how often points' associated centers can change.

In the first paper in this line of research, Liberty et al. [17] gave a lower bound that showed that one cannot simultaneously have both high quality and maximum consistency. That is, they showed that if a center cannot be changed once it is established, then there is no algorithm whose competitive ratio can be bounded by any function of n and k. Thus various "beyond worst-case analysis" (see [21]) approaches have been used in the literature to attempt to circumvent the obstacle presented by this lower bound. One approach is to use bi-criteria analysis or *resource augmentation* analysis. This analysis allows the online algorithm to use more than k centers, and then compares the cost of the algorithm's clustering to the optimal one using k centers [17]. A second approach is to allow the algorithm *recourse*, which in this setting means allowing the algorithm to change the centers (or clusters) a small number of times [14, 11, 12].

Resource Augmentation. Liberty et al. [17] give a randomized algorithm for k-means clustering and analyzes this algorithm using *resource augmentation* analysis. They show that the expected number of clusters/centers used by their algorithm is $O(k \log n \log(n\Delta))$ and at all times the expected cost of the clustering using these centers is at most $O(\log n)$ times the optimal cost using k clusters. Here Δ is the aspect ratio of the data points, which is the ratio between the distance between the furthest pair of points and the distance between the closest pair of points. The algorithm leverages a randomized online algorithm for facility location of Meyerson [18] to decide whether to create a new center at a newly arriving data point. Once a center is established, it is maintained throughout the course of the algorithm. Finally, they give a randomized algorithm that requires a priori knowledge of n and a lower bound on the optimal with k centers, and that maintains a collection of $O(k \log n \log \alpha)$ centers in expectation that has expected cost $O(1)$ times the optimal cost with k centers. Here α is the ratio between the actual optimal cost with k centers and the lower bound provided a priori to the algorithm.

Recourse. Lattanzi and Vassilvitskii [14] give a randomized algorithm for k-median clustering that uses *recourse*. It maintains the invariant that the cost of the current centers is always $O(1)$-competitive with the optimal clustering of the data points seen *to date*. To maintain this invariant, the expected number of cluster center changes used is $O(k^2 \log^4(n\Delta))$. They show a similar lower bound, that is they show that every algorithm requires $\Omega(k \log_c \frac{\Delta}{k})$ center changes to maintain $O(c)$-competitiveness. Further, they show that it possible to maintain $O(1)$-competitiveness with $O(k \log^2(n\Delta))$ center changes, but this is given as an existential result. In a follow-up paper, Fichtenberger et al. [11] gave a randomized algorithm that is $O(1)$-competitive with $O(k \operatorname{polylog}(n\Delta))$ cluster center changes. In both papers, the result of Meyerson [18] is again a key subroutine. The results of Lattanzi and Vassilvitskii [14] were extended to k-median clustering with outliers (so one could opt to not cluster a pre-specified number of points) by Guo et al. [12]. Lattanzi and Vassilvitskii [14] also observe that for k-center clustering an algorithm of Charikar et al. [8] yields an $O(1)$-competitive clustering with $O(k \log(n\Delta))$ center changes.

While not directly germane to the work in this paper, there is also research on online clustering in the streaming setting, where the emphasis is more on the algorithm using a small amount of memory, or quickly responding to the arrival of a new data point (e.g. [7, 10, 3]).

1.2 Our Contribution

Our research investigates consistent clustering for cluster-based problems (recall that all the past algorithmic consistent clustering publications that we are aware of focus on center-based clustering). We are interested in applications where the focus is on explicitly maintaining consistent clusters (and not necessarily on maintaining consistent centers). The application where Google or Yahoo news is trying to maintain collections of similar news articles is an example of such an application. Note that even the algorithms from [17] that are perfectly consistent from a center perspective, in that once a center is established it persists until the end of the algorithm, are not necessarily consistent from a cluster perspective in that a data point could change clusters every time a new center is established. All one can say (at least naively) about the cluster consistency of the algorithms from [17] is that no data point changes clusters more than $O(k \log n \log(n\Delta))$ times.

▶ **Problem 1** ((Online) cluster-based clustering). *The points $X = \{x_1, \ldots, x_n\}$ from a metric space arrive online in this (adversarial) order. Each point must be given an irrevocable label from $\{1, \ldots, k\}$ (i.e., irrevocably assigned a cluster) upon arrival. (The number of points n is not known.) The goal is to minimize the k-median objective.*

Beyond Worst-Case Model. We first observe that the lower bound from [17] extends to the cluster-based setting, so no online algorithm can achieve a competitive ratio bounded by *any* function of n. Thus we turn to a learning-augmented approach, namely we assume that the algorithm is provided a priori with an estimated upper bound B on the cost OPT of the final optimal clustering; recourse and resource augmentation are **not** allowed. This approach is both natural and appealing, as the a priori information provided to the algorithm is *minimal*. Moreover, it finds motivation in the Google/Yahoo news application, where presumably the final objective values for prior instances could be used as a basis to obtain a reasonable estimate for B. Thus we then seek algorithms that will maintain a clustering of low cost relative to B (not the current optimal cost for the points that have arrived to date). We say an algorithm is c-competitive with respect to B if the algorithm's cost is at most $c \cdot B$ on instances where the optimal cost is at most B (after all points have arrived).

We first show that any deterministic algorithm must have dependence on k in the competitive ratio. The proof is deferred to the full version.

▶ **Proposition 1.** *Any deterministic algorithm for cluster-based clustering is $\Omega(k)$-competitive with respect to B.*

In almost all applications, k is a small constant [2, 5]. Thus, we ask if an algorithm can have performance only depending on k and not on the large parameters Δ and n.

Does there exist an online algorithm for Problem 1 that, given a priori knowledge of an upper bound B on OPT, achieves competitiveness independent of n and Δ?

1.3 Results

Our main question is whether there exists an algorithm with competitiveness depending only on k (and not n or Δ). We answer this constructively:

▶ **Theorem 2.** *There is a poly-time algorithm for cluster-based clustering that is $O(k^5 3^k)$-competitive with respect to B.*

Intuitively, our algorithm uses the value of B to determine a scale for which costs are cheap (namely that are small relative to B) and which are expensive (namely that are large relative to B). Thus, one upshot of our results is that this minimal scaling information is all that the online algorithm needs to overcome the strong lower bound. Moreover, existing algorithms/subroutines in the recourse and resource augmentation settings do not seem to translate to guarantees in our setting. Thus, our setting requires novel techniques and structural insights, which we turn to next.

2 Technical Overview

As our algorithm is fairly detailed, we begin with a technical overview to build the case for our design decisions.

To understand the motivation for the learning-augmented approach, let us consider the lower bound instance from [17]. It is sufficient to assume $k = 2$. The first point x_1 arrives and is assigned some irrevocable label. Then assume the second data point x_2 arrives a unit

distance from x_1. If the online algorithm assigns x_2 the same label as x_1, then the cost of the algorithm's clustering is 1, and the optimal cost is 0 (which is the cost if each of these data points were given a different label). This results in the algorithm having unbounded competitiveness. In contrast, if the algorithm gave x_2 a different label from x_1 then the third data point x_3 could arrive very far away. In which case, the algorithm's clustering would necessarily have very high cost (as x_3's label would have to be either the same as x_1's or the same as x_2's). However, the optimal clustering would have cost 1 (by giving x_1 and x_2 the same label and giving x_3 the remaining label). Again, this results in competitiveness that can only be bounded by Δ (which may be much larger than n or k).

Intuitively, the dilemma faced by the algorithm when x_2 arrives is that it does not know whether the distance between x_1 and x_2 is small or large. Equipped with an estimate of the optimal cost B, the algorithm could resolve the dilemma in this case by giving x_2 a different label than x_1 if their distance is larger than B and the same label otherwise.

2.1 Properties of competitive algorithms

To better understand our algorithm design it is useful to understand some instances that illustrate some properties that a competitive algorithm must have.

A simple first observation is that any reasonably competitive algorithm can never use $t+1$ labels if it is the case that the points to date could be clustered with cost at most B using at most t labels. If the algorithm ever allowed this to happen, it could be that the next $k-t$ data points could arrive very far from the previous data points, and very far from each other. Thus after these data points arrive, the algorithm's cost would increase by an amount depending on the diameter of the metric space, while there would still be a clustering of cost at most B, since the clustering that used t labels could be extended with no additional cost by giving each of the new $k-t$ data points a new label.

Natural greedy algorithm fails. In light of this last observation, a natural greedy algorithm would maintain the invariant that the number of labels it uses is always equal to the minimum number of labels necessary for a clustering of cost at most B, and then give each new data point the label that minimizes the increase in cost. To see why such an algorithm (and other similar algorithms) can have unbounded cost even when $k=2$, and the metric space is the real line, consider the following instance (see Figure 1). Let α be an arbitrarily large positive integer. We construct an example in which the budget $B=2$. The first point arrives at location -2. Say the algorithm gives this point the label blue. The next point arrives at location 1. Now, we know that any offline clustering with cost at most 2 must use at least 2 clusters. So the greedy algorithm would give this new point a second label, say green, as that would minimize the increase in the objective. Then a total of α additional points arrive at location 1. The algorithm labels these points green. Then α points arrive at the origin 0. It is still the case that only 2 clusters are required in order to have cost at most 2, since we may place the points at location -2 and the origin in one cluster, and the points at location 1 in the other cluster. However the algorithm would assign each point arriving at the origin the label green, since this increases the objective by at most 1 while assigning such a point the label blue increases the objective by 2. Yet, this results in a solution for the algorithm in which the contribution of green points towards the objective is α.

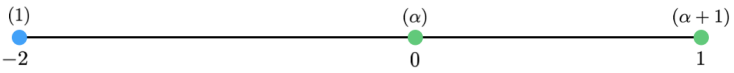

Figure 1 An example in which the natural greedy algorithm fails.

Upon reflection of this lower bound instance for the natural greedy algorithm, there appear to us to be two natural hypotheses as to the "mistake" that this algorithm is making, and correspondingly two natural paths towards a remedy:

- One hypothesis is that greedy assignment is a mistake, and then the natural remedy is some label assignment rule more sophisticated than greedy.
- Another is that the algorithm was too hasty in using a new label. Thus the natural remedy would be to delay using a new label until it is more clear as to a region where arriving data points should be given this new label. Note in the example in Figure 1 that if the algorithm had waited until some reasonable number of data points had arrived at the origin before using the second label, then the algorithm might have been able to see that the right choice was to give the remaining points arriving at the origin the second label of green.

2.2 Techniques

Here we primarily adopt the second remedy/approach (while also considering an alternate greedy assignment policy). To apply this remedy we must address the following:

- Under what conditions can the algorithm justify the use of an additional label, increasing from $t-1$ labels to t labels?
- When this can be justified, how should we modify our prior partition of space into $t-1$ parts to a partition into t parts? We would like to greedily assign each point to its "closest" part.

Well-separated points. At a high level our answer to the first question is that we do not use t labels until there exist t well-separated points $x_{\alpha(1)}, \ldots, x_{\alpha(t)}$. We will say that a collection of points $x_{\alpha(1)}, \ldots, x_{\alpha(t)}$ from a collection S of points is β-**well-separated with respect to w_S** (for some $\beta > 0$) if for all $i, j \in [t]$, $i \neq j$

$$\min\{w_S(x_{\alpha(i)}), w_S(x_{\alpha(j)})\} \cdot d(x_{\alpha(i)}, x_{\alpha(j)}) \geq \beta \cdot B \qquad (\star)$$

Here $w_S(x_h)$ is what we call the **natural weight** of point x_h in S, which is the maximum number of points in S whose distances to x_h sum to at most $2B$:

$$w_S(x_h) := \max\{|S'| : S' \subseteq S, \sum_{s \in S'} d(s, x_h) \leq 2B\}.$$

The condition $(\star)$ states that every pair of these t points is far apart – according to a weighted notion of distance. In turn, the weights used in this notion of distance are the so-called natural weight of each point, which captures the density of its nearby points. Intuitively, if we have t well-separated points, then not only must any near-optimal solution use t labels, but such a solution cannot combine the points near $x_{\alpha(i)}$ and the points near $x_{\alpha(j)}$ into a single cluster.

Pivots. The algorithm is divided into at most k **phases**. For each phase t, the algorithm maintains a collection of points $p_1, \ldots, p_t$ from the online stream X which we call *pivots*. The pivots $p_1, \ldots, p_t$ stay fixed during phase t. The key property they should satisfy is that they

are well-separated with respect to the points seen so far. Between phases, we increase the number of pivots, thus allowing the algorithm to use more labels. The pivot p_i is associated with the label i. Thus, during phase t, there are t clusters (i.e., t labels in use). When a new point arrives, it is assigned the label i of the pivot p_i nearest to it (so we maintain a greedy labelling rule). Importantly, though, the *location* of the pivot p_i for label i may change over time. Roughly speaking, this occurs when there is a better representative for cluster i.

Pivots vs. centers. While one might reasonably think that the pivots are intuitively (low-cost) centers for the clusters, this intuition is only partially correct. Part of the subtlety of the algorithm design is that there are in fact scenarios where some pivots are poor centers for the corresponding clusters, but still good representatives for making cluster assignment decisions. What is critical is that the pivots are located so as to guarantee that using greedy assignment in the future results in a relatively low cost assignment; so pivots serve to recruit points to the right clusters. Our algorithm evinces a distinction between a good representative for a cluster in the long-term (a pivot) and a good center at a single moment.

Invariants. In order for our cost analysis to be tractable, the algorithm should maintain the following invariants:

- Each pivot p_i is located in a region where it would not be too costly to assign points arriving there the label i.
- The pivots $p_1, \ldots, p_t$ for phase t are well-separated (for some appropriate choice of β) during phase t.[1]
- There is no other point that is well-separated from the pivots during a phase. (Otherwise, this indicates that another label can and should be in use.)
- The locations of the pivots should not move very often.

Note that some of these invariants can intuitively be in opposition to each other, which requires that the algorithm design be a bit detailed, as there are several different cases where maintaining this invariant requires different updates to the pivots. We now give an overview of how the algorithm maintains these invariants, highlighting representative cases.

2.3 Preliminaries

Assumptions. We state our results assuming $B = \mathsf{OPT}$, but all still hold by replacing OPT with B.

Terminology. Recall from above the *natural weights* $w_S(\cdot)$. We will always take S to be some prefix of the online stream X. Note $w_S(p)$ can only increase over time as S enlarges.

Other terms related to well-separation are: A pair of points x_i, x_j are **β-attached** with respect to w_S if $\min\{w_S(x_i), w_S(x_j)\} \cdot d(x_i, x_j) < \beta \cdot B$, i.e., the well-separated condition does *not* hold for this pair. A useful way of viewing attachment is that we may move a certain number of points lying near x_{α_j} to x_{α_i} at bounded cost (but perhaps not in the reverse direction). We say p is **β-well-separated from** a set of points $\{x_{\alpha(1)}, \ldots, x_{\alpha(m)}\}$ with respect to w_S if $\min\{w_S(p), w_S(x_{\alpha(i)})\} \cdot d(p, x_{\alpha(i)}) \geq \beta \cdot B$, $\forall i \in [m]$.

[1] β must be initialized sufficiently large and also decrease as the number of pivots increases.

Figure notation. In all figures below, dashed lines indicate well-separation. Solid lines indicate attachment; where included, arrows on solid lines specify the direction of attachment, i.e., point from smaller to larger natural weights (see previous paragraph). Colors except black correspond to cluster labels. Small circles drawn near a larger circle indicate the points attaining the natural weight of the larger point (the maximizing set S' in the definition of w_S).

2.4 Subroutines

There are two subroutines used to enlarge the set of pivots from phase t to phase $t+1$, the **Add Operation** and the **Exchange Operation**. There are subtleties to the execution of these operations that require we also keep track of good centers for the clusters built so far, called **estimated centers**.

Estimated Centers. As the pivots are not necessarily good centers (for example, pivot p_1 at location -2 as the points arrive at location 1 in Figure 1), the algorithm also maintains a collection $c_1, \ldots, c_T$ of estimated centers for the T labels[2] that have been used to date. The estimated centers are updated at the end of some phases, and satisfy the invariant that c_j is a center for label j's current cluster with bounded cost. Consider Figure 2. At the start of phase T, c_j^{old} is the estimated center for points in cluster j. Points arriving in phase T that are closer to p_j than to other pivots are given label j (by our greedy assignment rule). However, these new points may be concentrated around, for instance, y_2, so that p_j is not actually a good center for cluster j at the end of phase T (even though it is fine for labelling purposes).

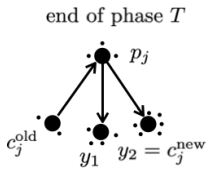

Figure 2 Updating the estimated centers at the end of a phase.

Thus, when it comes time to reset the pivots at the end of phase T, we might need to move p_j to the new estimated center $y_2 = c_j^{\text{new}}$ or to a nearby point. So, estimated centers are not only used in the cost analysis, but critically are used algorithmically to update the locations of pivots. The computation of the estimated centers will involve running an offline approximation algorithm on the points seen to date.

Add Operation. An Add Operation is applicable when there is a point x_α that is well-separated from the current pivots. Intuitively, this means a new label (cluster) can be justified, but the implementation requires the consideration of several possible scenarios. In the simplest scenario x_α is near a cluster of new points that are all far from previous points, and the pivot p_{t+1} for the new label $(t+1)$ is set to x_α. In some scenarios an old pivot p_i $(i \le t)$ is set to x_α and p_{t+1} is set to p_i (so the new pivot location inherits the old label i and an old pivot location gets the new label $t+1$). Intuitively, this occurs when the estimated

[2] We use T instead of t here to distinguish that this subroutine is only executed at the end of certain phases; see Section 3.

center c_i for cluster i is at or near the location of x_α. See Figure 3 (left); take $i = 2$, $t = 4$, and $x_\alpha = c_2$. Finally, there are scenarios where x_α is close to two different clusters; in this case x_α is never made a pivot and instead *two* pivots are added at the estimated centers of these clusters (so we skip straight to phase $t + 2$). See Figure 3 (right). One must show that this move maintains the well-separation invariant.

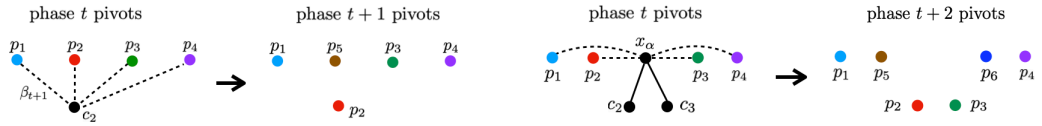

Figure 3 Two cases of the Add Operation.

Exchange Operation. An Exchange Operation is applicable when there are two points x_α and x_γ near a pivot p_j that are well-separated from each other and the other pivots (besides p_j). See Figure 4 (left); take $j = 3$. So intuitively the cluster of points labeled j appear to be splitting into two clusters. In the simplest scenario the location of pivot p_j is set to the location of one of x_α or x_γ, and the location of the new pivot p_{t+1} is set to the other. See Figure 4 (right); set $j = 3$, $t = 4$.

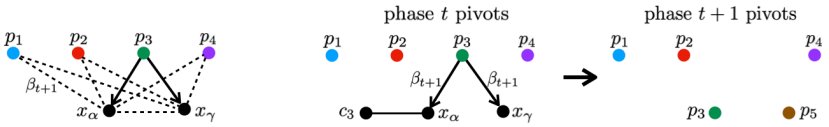

Figure 4 A case of the Exchange Operation.

This scenario occurs in the instance depicted in Figure 1. The first pivot p_1 is initially set to location -2. The points arriving at location 1 would all be assigned the label 1 (blue) as there is no point well-separated from p_1 (the points located at 1 are not separated from p_1 because the points at p_1 can be cheaply moved to location 1). When enough points have arrived at the origin, then the points $x_\alpha = 0$ and at $x_\gamma = 1$ are near p_1 (because the point at p_1 can be cheaply moved to either x_α or x_γ), but are well-separated from each other and the pivots other than p_1. Thus our algorithm would locate p_1 at 1 and p_2 at the origin. While this gives intuition, there are other more subtle scenarios.

3 Algorithm Description

The algorithm sees an online sequence $X = \{x_1, x_2, \ldots x_n\}$ of points. Let $X_i = \{x_1, x_2, \ldots x_i\}$. **Let w_i be shorthand for w_{X_i}.** During any phase t, the algorithm maintains:

- a collection of previously arriving points $p_1, \ldots, p_t$ that have been designated as *pivots*, where t is the number of labels used by the algorithm to date and pivot p_j is associated with label j,
- a separation parameter $\beta_t = 8 \cdot 3^{k-t+2}$, and
- a collection of previously arriving points $c_1, \ldots, c_s$ ($s \leq t$) that have been designated as estimated centers.

Phase t is the set of time steps when there are t pivots. Phase 1 is initialized as follows: the first point x_1 is given the label 1, the first pivot p_1 is set to x_1, and the collection of estimated centers is empty. Let T be the current phase. The algorithm handles the arrival

of each subsequent point x_i as follows. It checks whether there is an applicable Add or Exchange Operation, both of which will increase the number of pivots. If so, phase T ends. First estimated centers $c_1, \ldots, c_T$ are computed, and then, using these, the algorithm carries out consecutive Add and Exchange Operations (giving preference to Add Operations for technical reasons) until there are none left. With each operation, the phase increases and the pivots are reset. Call the last phase in this sequence of consecutive operations T^+. (Note that $T^+ \geq T+1$.) The point x_i is the first point labelled during phase T^+ (it is *not* labelled during phase T). In summary:

1. **If** there is an applicable Add or Exchange Operation upon the arrival of x_i **then** compute new Estimated Centers $c_1, \ldots c_T$.
   a. **Repeat** while there is an applicable Add Operation or Exchange Operation.
      i. **If** there is an applicable Add Operation **then** apply an arbitrary applicable one.
      ii. **Else** apply an arbitrary applicable Exchange Operation.
2. Give x_i label j, where p_j is the nearest pivot (among $p_1, \ldots, p_{T^+}$) to x_i

We then repeat the above steps upon the arrival of x_{i+1}. Note that if there is t such that $T < t < T^+$, then no points are labelled during phase t. We call such phases t during which no points are labelled **intermediate**. During each other phase, at least one point is labelled, and we call such phases **non-intermediate**. So for a non-intermediate phase T, T^+ is the first non-intermediate phase after T, and we will also use T^- to refer to the last intermediate phase before T. We now describe the three subroutines.

3.1 The Estimated Center Subroutine

This subroutine computes T new estimated centers $c_1, \ldots, c_T$ from pivots $p_1, \ldots p_T$, the points X_{i-1} that have arrived before x_i, and estimated centers $c_1, \ldots c_{T^-}$.

Choose $y_1, \ldots, y_k \in X_{i-1}$ to be an (offline) optimal collection of k centers[3] for the points in X_{i-1}. For each **offline optimal center** y_h, $h \in [k]$, define $p(y_h)$ to be the pivot with the minimum weighted distance to y_h, that is,

$$p(y_h) = \arg\min_{p_j} \left(\min\{w_{i-1}(p_j), w_{i-1}(y_h)\} \cdot d(p_j, y_h)\right) \tag{$\dagger$}$$

Say that y_h is *assigned to* p_j if $p(y_h) = p_j$. For each pivot p_j, we define the set $\delta(p_j)$ to contain a subset of the offline optimal centers that are assigned to p_j, and possibly c_j as well; the points in $\delta(p_j)$ are "close" to p_j in some sense. In particular, $y_h \in \delta(p_j)$ if $p(y_h) = p_j$ and $w_{i-1}(y_h) > w_{i-1}(p(y_h))$. Also, c_j is in $\delta(p_j)$ if $w_{i-1}(c_j) > w_{i-1}(p_j)$ and c_j is β_{t+1}-attached to p_j w.r.t. w_{i-1}.

As an example, **see Figure 2.** Here, $\delta(p_j) = \{y_1, y_2\}$, so c_j (denoted c_j^{old}) is not in $\delta(p_j)$, because the arrow from c_j^{old} to p_j (representing attachment) points in the wrong direction.

For each $j \in [T]$, we now define the new **estimated center** c_j: If $w_{i-1}(p_j) \geq \max_{p \in \delta(p_j)} w_{i-1}(p)$ then $c_j = p_j$, else

$$c_j = \arg\max_{p \in \delta(p_j)} w_{i-1}(p) \tag{$\ddagger$}$$

So in Figure 2, c_j is updated to y_2 (denoted c_j^{new}), because y_2 has the largest weight in $\delta(p_j)$. Intuitively, this means that c_j^{new} is now a better center for cluster j than, say, c_j^{old}.

[3] To run in poly-time, replace with any constant approximation algorithm. This algorithm's cost will change by a constant factor.

3.2 The Add Operation Subroutine

Let $t \geq T$ be the number of pivots when an Add Operation is called (during an execution of (i) above). The Add Operation applies if there is a point $x_\alpha \in X_i$ such that x_α is β_{t+1}-well-separated from the current pivots $p_1, \ldots, p_t$ with respect to the weights w_i. **(E.g., Figure 3, left, with $t = 4$.)** The Add Operation depends on x_α, X_i, the current pivots $p_1, \ldots, p_t$, and estimated centers $c_1, \ldots, c_T$. In most cases, the Add Operation adds x_α to the set of pivots, and changes the location of up to two previous pivots **(Figure 3)**.

Define $w_t := w_{i-1}$ if $t = T$ and $w_t := w_i$ if $t > T$.[4]

1. **If** there is an estimated center c_j that is β_{t+1}-well-separated from $p_1, \ldots, p_t$ w.r.t. w_i then set $p_{t+1} = p_j$ and set $p_j = c_j$. **(Figure 3, left)**
2. **Else if** it is the case that for every estimated center c_j that is β_{t+2}-attached to x_α w.r.t. w_i it is also the case that $w_t(c_j) < w_t(p_j)$, then set $p_{t+1} = x_\alpha$.
3. **Else if** there exists a unique estimated center c_j is β_{t+2}-attached to x_α w.r.t. w_i and $w_t(c_j) \geq w_t(p_j)$ then set $p_{t+1} = p_j$ and $p_j = x_\alpha$.
4. **Else** Let c_f and c_g be estimated centers such that each is β_{t+2}-attached to x_α w.r.t. w_i, $w_t(c_f) \geq w_t(p_f)$, and $w_t(c_g) \geq w_t(p_g)$. Set $p_{t+1} = p_f$, $p_{t+2} = p_g$, $p_f = c_f$, and $p_g = c_g$. **(Figure 3, right)**

Note in the last case that we skip to phase $t + 2$.

3.3 The Exchange Operation Subroutine

The Exchange Operation subroutine is applicable if there exists two points x_α and x_γ in X_i, and a pivot p_j such that:

- x_α and x_γ are each β_{t+1}-attached to p_j w.r.t. w_i,
- $w_i(p_j) \leq w_i(x_\alpha)$,
- $w_i(p_j) \leq w_i(x_\gamma)$, and
- The collection of the $t + 1$ points, consisting of x_α, x_γ, and the pivots other than p_j, are β_{t+1}-well-separated w.r.t. w_i. **(E.g., Figure 4, left, with $t = 4$.)**

The Exchange Operation depends on x_α, x_γ, X_i, the current pivots $p_1, \ldots, p_t$, and estimated centers $c_1, \ldots, c_T$. In most cases, the Exchange Operation adds x_α and x_γ to and deletes p_j from the set of pivots, and possibly changes the location of one previous pivot.

1. **If** $j > T$ then set $p_j = x_\alpha$ and $p_{t+1} = x_\gamma$.
2. **Else if** $w_i(c_j) < w_i(p_j)$ then set $p_j = x_\alpha$ and $p_{t+1} = x_\gamma$.
3. **Else if** c_j is β_{t+2}-attached to x_α w.r.t. w_i then set $p_j = x_\alpha$ and $p_{t+1} = x_\gamma$. **(Figure 4, right)**
4. **Else if** c_j is β_{t+2}-attached to x_γ w.r.t. w_i then set $p_j = x_\gamma$ and $p_{t+1} = x_\alpha$.
5. **Else** set $p_{t+1} = x_\alpha$, $p_{t+2} = x_\gamma$, and $p_j = c_j$.

4 Algorithm Invariants and Analysis

In this section, we state the key technical lemmas. We defer full proofs to the Appendix.

▶ **Theorem 3.** *The algorithm uses at most k labels.*

▶ **Theorem 4.** *The algorithm's cost is $O(k^5 \cdot 3^k \cdot OPT)$.*

[4] We are overloading subscripts here for ease. We could instead write v_t, but we retain w to recall weights.

4.1 Notation

- $p_1^t, \ldots, p_t^t$ denote the pivots for labels 1 through t, respectively, during phase t.
- w^t are the natural weights at the end of phase t.
- $X(t)$ is the set of points assigned a label before or during phase t.
- For $j \in [t]$, C_j^t is the set of points labelled j in phases 1 through t.
- For $j \in [T]$, c_j^T is the estimated center (‡) computed at end of non-intermediate phase T.
- $y_1^T, \ldots, y_k^T$ are the offline optimal centers computed at the end of phase T in The Estimated Center Subroutine.
- $P_T = \{p_1^T, \ldots, p_T^T, y_1^T, \ldots, y_k^T\}$
- $cost(S; c) = \sum_{p \in S} d(p, c)$ for $S \subseteq X$ and $c \in X$.

4.2 Invariants

In the next two lemmas, we show that our algorithm maintains two key invariants. The full proofs are deferred to the full version, although we give a proof sketch of Lemma 5 in the appendix.

▶ **Lemma 5.** *Let $t \in [k]$. The algorithm maintains the invariant that $p_1^t, \ldots, p_t^t$ are β_t-well-separated w.r.t. the natural weights at the start of phase t (and after).*

Lemma 5 directly implies Theorem 3 once we show that there can be no more than k well-separated points in X and note that we have set β_1 sufficiently large.

Next is a key technical lemma. It states that the estimated center $c_j^{T^-}$ for the points given label j *before* phase T is close, in a weighted sense, to the pivot for label j in phase T. This is key to showing that points in cluster j that are labelled *before* phase T can be combined with those that are labelled *during* phase T at bounded cost. This lemma is in tension with the prior one because a pivot must be placed in a location where it is both well-separated from other pivots and is close to previously arriving points in its cluster.

▶ **Lemma 6.** *Let T be a non-intermediate phase and let $j \in [T]$. If $T > 1$, at least one of the following holds:*

(a) $w^{T^-}(c_j^{T^-}) \leq w^T(p_j^T)$ *and* $w^{T^-}(c_j^{T^-}) \cdot d(c_j^{T^-}, p_j^T) \leq \beta_{T^-}(T - T^-) \cdot OPT$.

(b) $c_j^{T^-}$ *is β_{T+1}-attached to p_j^T w.r.t. w^T.*

5 Bounding Cost

We show by induction that the estimated center c_j^T is good for *all* points given label j by the end of phase T. Taking T to be the last phase gives our main result. We follow Figure 5. By definition of attached, a certain number of points sitting at the head (equal to the head's natural weight) of an arc can be moved to the tail at bounded cost. First we address the cost of $C_1^{T^-}$, the points given label 1 before phase T. We inductively assume these can be moved to $c_1^{T^-}$ at bounded cost. From there, we need to move them to c_1^T at bounded cost. This can be done by showing that (1)$|C_1^{T^-}|$ is a bounded factor away from the natural weight of $c_1^{T^-}$ (Lemma 8), and (2) c_1^T is "close" to p_1^T (Lemma 8). Together these imply that we can move the points in $C_1^{T^-}$ along the arc from $c_1^{T^-}$ to p_1^T at bounded cost.

Next we show that the points given label 1 *during* phase T, call them C_1, can also be moved to c_1^T at bounded cost. This is where we use the set of offline optimal centers $y_1^T, \ldots, y_k^T$ computed during the Estimated Centers Subroutine. Importantly, since during a phase every point is attached to at least one pivot (otherwise we execute an Add Operation

and leave the phase), each offline center y_i^T is attached to a pivot. We partition the points in C_1 based on which center y_i^T they are assigned to in the *offline* optimal solution. The set of points in C_1 that are assigned to centers attached to the pivot for label 1, p_1^T, is called $S_{near,1}$. In Figure 5, these are points assigned to y_1^T and y_2^T. One can show, using that during a phase no Exchange Operation occurs, that these can be moved to c_1^T at bounded cost. The set of points that are assigned to centers that are attached to a pivot for a different label, say label 2, is called $S_{far,1}$. These points are misclassified in the sense that the online and offline algorithms classify them differently. However, we show their cost is still controlled. Specifically, the well-separated invariant implies that (1) these points can be moved to p_1^T at bounded cost, and (2) the number of them is a bounded factor away from the natural weight of p_1^T (Lemma 7). These two properties imply we can move the points in $S_{far,1}$ to p_1^T, and then to c_1^T, at bounded cost.

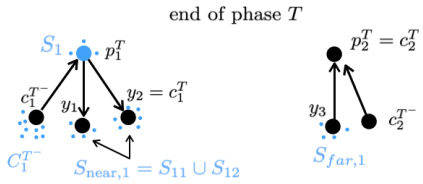

Figure 5 The points given label 1 (blue) before or during phase T are partitioned as in the text.

▶ **Lemma 7.** *Let T be a non-intermediate phase. For any $j \in [T]$, let C_j be the points given label j during phase T, i.e., $C_j = C_j^T \setminus C_j^{T^-}$. Define S_{ji} to be be the set of elements in C_j assigned to y_i in the clustering of $X(T) \setminus X(T^-)$ induced by P_T. Define $S_{far,j} = \bigcup_{i:p(y_i)\neq p_j^T} S_{ji}$. Then*

1. *$cost(S_{far,j}; p_j^T) \leq k \cdot (\beta_{T+1} + 2) \cdot OPT$, and*
2. *$|S_{far,j}| \leq k \cdot w^T(p_j^T)$, where w^T denotes the natural weights at the end of phase T.*

▶ **Lemma 8.** *Let T be a non-intermediate phase and $j \in [T]$. Let $w^T(c_j^T)$ be the natural weight of c_j^T at the end of phase T and C_j^T be the set of points in cluster j by the end of phase T. Then $|C_j^T| \leq (2k+1) \cdot T \cdot w^T(c_j^T)$.*

The final lemma below shows that the cost of our algorithm's solution at the end of phase T is bounded against OPT. Taking T to be the last phase gives Theorem 4.

▶ **Lemma 9.** *Let T be a non-intermediate phase and $j \in [T]$. Then $cost(C_j^T)$ is bounded against center c_j^T, i.e., $\sum_{x \in C_j^T} d(x, c_j^T) \leq g(T,k) \cdot OPT$, where*

$$g(T,k) = T \cdot g(k) \qquad \text{and} \qquad g(k) = \beta_1(2k^3 + 3k^2 + 5k + 1) + 2k + 4.$$

6 Conclusion

This paper gives the first online algorithm for **cluster-based** k-median clustering, with competitive ratio independent of n and Δ, that does not recluster or use additional centers. We take a learning-augmented approach, assuming minimal a priori information in the form of an upper bound B on the optimal cost. Prior to this work, it was not known that any algorithm could have bounded worst-case guarantees. Interestingly, we remark that if the algorithm does not know B and reclustering is allowed, our results imply an algorithm that maintains a solution competitive against the optimal solution on the points that have arrived *so far*. Reclustering an $O(\log(n\Delta))$ number of times, the algorithm is always $O(1)$-competitive when k is a constant at each point in time. This matches the number of reclusterings used in prior work for the consistent center case.

References

1 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal of Computing*, 33(3):544–562, 2004.

2 Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *Proc. VLDB Endow.*, 5(7):622–633, 2012.

3 Robi Bhattacharjee, Jacob Imola, Michal Moshkovitz, and Sanjoy Dasgupta. Online k-means clustering on arbitrary data streams. In *International Conference on Algorithmic Learning Theory*, pages 204–236. PMLR, 2023.

4 Robi Bhattacharjee and Michal Moshkovitz. No-substitution k-means clustering with adversarial order. In *Algorithmic Learning Theory*, pages 345–366. PMLR, 2021.

5 Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, Berlin, Heidelberg, 2006.

6 Jaroslaw Byrka, Thomas W. Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median and positive correlation in budgeted optimization. *ACM Transactions on Algorithms*, 13(2):23:1–23:31, 2017.

7 T-H. Hubert Chan, Arnaud Guerqin, and Mauro Sozio. Fully dynamic k-center clustering. In *World Wide Web Conference*, pages 579–587, 2018.

8 Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. *SIAM Journal of Computing*, 33(6):1417–1440, 2004.

9 Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and Systems Sciences*, 65(1):129–149, 2002.

10 Vincent Cohen-Addad, Niklas Hjuler, Nikos Parotsidis, David Saulpic, and Chris Schwiegelshohn. Fully dynamic consistent facility location. In *Conference on Neural Information Processing Systems*, pages 3250–3260, 2019.

11 Hendrik Fichtenberger, Silvio Lattanzi, Ashkan Norouzi-Fard, and Ola Svensson. Consistent k-clustering for general metrics. In *ACM-SIAM Symposium on Discrete Algorithms*, 2021.

12 Xiangyu Guo, Janardhan Kulkarni, Shi Li, and Jiayi Xian. Consistent k-median: Simpler, better and robust. In *International Conference on Artificial Intelligence and Statistics*, volume 130, pages 1135–1143, 2021.

13 K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of the ACM*, 48(2):274–296, 2001.

14 Silvio Lattanzi and Sergei Vassilvitskii. Consistent k-clustering. In Doina Precup and Yee Whye Teh, editors, *International Conference on Machine Learning*, pages 1975–1984, 2017.

15 Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. *SIAM Journal of Computing*, 45(2):530–547, 2016.

16 Edo Liberty, Ram Sriharsha, and Maxim Sviridenko. k-means clustering. talk slides.

17 Edo Liberty, Ram Sriharsha, and Maxim Sviridenko. An algorithm for online k-means clustering. In *Workshop on Algorithm Engineering and Experiments*, pages 81–89, 2016.

18 A. Meyerson. Online facility location. In *IEEE Symposium on Foundations of Computer Science*, pages 426–431, 2001.

19 Adam Meyerson, Liadan O'Callaghan, and Serge Plotkin. A k-median algorithm with running time independent of data size. *Machine Learning*, 56(1):61–87, 2004.

20 Benjamin Moseley, Heather Newman, and Kirk Pruhs. Online k-median with consistent clusters. *arXiv preprint*, 2023. `arXiv:2303.15379`.

21 Tim Roughgarden. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2021.

The Appendix is organized as follows. In Appendix A, we introduce some additional terminology and notation. In Appendix B, we introduce a few propositions that will be useful for proving the main lemmas. In Appendix C, we prove Lemma 5, the well-separation invariant; this is a rather involved proof, so we include both a proof sketch with the high-level ideas, and defer the full proof to the full version. Then, in Appendix D, we show how Lemma 5 swiftly implies Theorem 3. In the remaining appendices, we prove the lemmas in Section 5. The last lemma, Lemma 9, directly implies Theorem 4.

A Terminology

Throughout this appendix, we take $B = \mathsf{OPT}$ for simplicity. Our results still hold as long as $B \geq \mathsf{OPT}$. Below is some additional terminology used in the proofs.

- Let $y_1^T, \ldots, y_k^T$ be the optimal collection of k centers computed at the end of phase T in The Estimated Center Subroutine. Let $P_T = \{p_1^T, \ldots, p_T^T, y_1^T, \ldots, y_k^T\}$ and call this set the **offline centers** for phase T. When the context is clear, we may omit the superscript T in y_i^T.
- The **attachment digraph** $D(T)$ is a bipartite digraph with vertex set P_T, plus $\{c_j^{T^-} \mid j \leq T^-\}$ if $T > 1$, partitioned as $(\{p_1^T, \ldots, p_T^T\}, \{y_1, \ldots, y_k, c_1^{T^-}, \ldots, c_{T^-}^{T^-}\})$. There is a directed arc $(y_i, p(y_i))$ if $w^T(y_i) \leq w^T(p(y_i))$ and a directed arc $(y_i, p(y_i))$ otherwise. If $c_j^{T^-}$ and p_j^T are β_{T+1}-attached w.r.t. w^T, add the arc $(c_j^{T^-}, p_j^T)$ if $w^T(c_j^{T^-}) \leq w(p_j^T)$ and the arc $(p_j^T, c_j^{T^-})$ otherwise. $\delta^+(p_j^T)$ and $\delta^-(p_j^T)$ denote the out- and in-degree of p_j^T. See Figure 6.

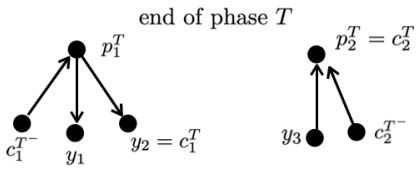

Figure 6 The attachment digraph $D(T)$. Arrows represent attached pairs, and arrows point from smaller to larger natural weights. So we may move a certain number of points near the head, to the tail at bounded cost.

B Helper Propositions

In this section, we present a few short propositions that will be useful in the remaining proofs. All excluded proofs are deferred to the full version.

The following fact justifies that the offline optimal centers $y_1^T, \ldots, y_k^T$ have cost at most $2\mathsf{OPT}$ on the points that arrive during phase T. This is used at various points in the analysis.

▶ **Fact 10** (Fact 2.1 in [19]). *Let N be a set of points, with $S \subseteq N$. Let k be an integer with $0 \leq k \leq n$. Let $K \subseteq S$ be the k-element subset of S minimizing $\sum_{x \in S} d(x, K)$ where $d(\cdot, \cdot)$ is the distance function on N, and $d(x, K)$ denotes $\min_{m \in K} d(x, m)$. Then if K' is a k-element subset of N, $\sum_{x \in S} d(x, K) \leq 2 \sum_{x \in S} d(x, K')$.*

The following proposition is a weighted version of the triangle inequality.

▶ **Proposition 11.** *Let x, y, p be three points in some set S, and let $w : S \to \mathbb{Z}_+$ be a weight function on S. Assume that $\beta, \beta_x, \beta_y, B > 0$. Suppose that $w(x) \leq w(p)$ and that x and y are β-well-separated w.r.t. w. If x and p are β_x-attached w.r.t. w, and y and p are β_y-attached w.r.t. w, then $\beta < \beta_x + \beta_y$.*

Next, we show that a point set cannot contain more than k pairwise β-well-separated points for β a sufficiently large constant. This allows us to bound the number of labels used.

▶ **Proposition 12.** *Let X be a set of points whose optimal k-median cost using k centers is OPT. Let $\{x_1, \ldots, x_l\}$ be a set of points in X, and let w_X denote their natural weights in X. Let $\beta > 8$. If $\{x_1, \ldots, x_l\}$ is β-well-separated w.r.t. w_X, then $l \leq k$.*

The next two propositions will be used to aid the proofs of Lemmas 5 and 6. Recall that for each non-intermediate phase T, we defined a set of offline centers P_T that has cost at most 2OPT on $X(T)$ (Appendix A and Fact 10). In order to compare the (low-cost) offline clustering induced by P_T to our online algorithm's clustering, we relate the offline set of centers P_T (which we *know* have bounded cost on $X(T)$) to the pivots in phase T (which are used to make the greedy online choices) in the next proposition.

▶ **Proposition 13.** *Let $P_T = \{p_1^T, \ldots, p_T^T, y_1, \ldots, y_k\}$ be as in Appendix A. Then y_i and $p(y_i)$ are β_{T+1}-attached w.r.t. the natural weights w^T at the end of phase T.*

Each attached pair in Proposition 13 is encoded in the digraph $D(T)$ by a directed arc. So, we can now think of this directed arc as representing the direction in which we could move a certain number of points sitting near one endpoint to the other at bounded cost.

Next we show that the estimated center for a cluster at the end of a phase is attached to the pivot for that cluster in that phase. Thus, while the pivot itself may not be a good center for the cluster, the pivot is close to the estimated center (in at least one direction, in a weighted sense).

▶ **Proposition 14.** *The estimated center c_j^T is β_{T+1}-attached to p_j^T w.r.t. the natural weights w^T at the end of phase T. Further, $w^T(c_j^T) \geq w^T(p_j^T)$, with equality if and only if $c_j^T = p_j^T$.*

C Proof of Lemma 5

As the proof of Lemma 5 is a rather involved double induction, we provide a proof sketch which pulls out the hard cases. In the full version, we give the full proof.

Proof sketch of Lemma 5. The proof is by induction. However, we need to couple the induction with a statement about the relative position of the estimated center for a cluster (which stays fixed between intermediate phases) to that cluster's pivot, which may change often as we consecutively reset the pivots between intermediate phases. Roughly, we prove below that if the estimated center for cluster j has not separated entirely from the present set of pivots, then it must be close (in a weighted sense) to the present pivot for label j.

▷ Claim 1. Let w_{i-1}, w_i, and w_t be as Section 3.2. For each $j \in [T]$ and $t \in [T, T^+]$ such that $p_1^t, \cdots, p_t^t$ are defined,[5]

$$p_1^t, \ldots, p_t^t \text{ are } \beta_t\text{-well-separated w.r.t. } w_t. \qquad (\Diamond)$$

Moreover, at least one of the following properties holds:
(a) c_j^T is β_{t+1}-well-separated from $p_1^t, \ldots, p_t^t$ w.r.t. w_i.
(b) c_j^T is β_{t+1}-attached to p_j^t w.r.t. w_t.
(c) c_j^T is $f(t,T)$-attached to p_j^t w.r.t. w_t and $w_t(c_j^T) < w_t(p_j^t)$, where $f(t,T) = \beta_T \cdot (t - T)$.

[5] Recall in Case 4 of the Add Operation and Case 5 of the Exchange Operation, we go directly from t to $t+2$ pivots, skipping phase $t+1$.

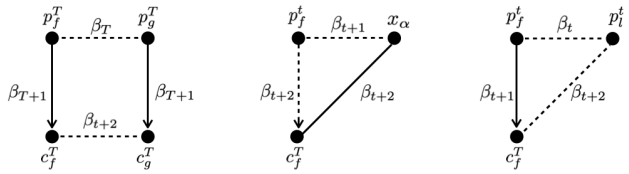

Figure 7 Cases (i) –(iii) in the proof sketch of Lemma 5. Dashed lines indicate well-separation and solid lines indicate attachment, labelled with the appropriate parameters. Arrows go from smaller to larger natural weights.

For the proof sketch we focus on Case 4 of the Add Operation, which will give a flavor of the arguments. This is a concerning case a priori; for, if we were to add x_α to the set of pivots as in Cases 2 and 3, it is ambiguous as to whether x_α should be associated with label f or g, as both c_f^T and c_g^T are close to x_α. We maneuver around the issue by making c_f^T and c_g^T new pivots and excluding x_α. However, it is not immediately clear that such a step will preserve the desired invariants. To give intuition, we suppress the separation parameters and the precise weights used, though emphasize both are brittle (e.g., the arguments rely heavily on β_t decreasing with t. The directions of attachment between points (arrows in Figure 7) are also crucial. We will also see why we need to couple the induction with (a) –(c).

To prove the inductive step for ($\Diamond$) when Case 4 of the Add Operation is performed, we need to show (i) c_f^T and c_g^T are well-separated, (ii), WLOG, c_f^T is well-separated from p_f^t, and (iii), WLOG, c_f is well-separated from p_l^t, $l \neq f$. See Figure 7. When we say "close" or "far" below, we always mean in a weighted sense. For (i), because p_f^T is close to c_f^T (Proposition 14) and likewise for p_g^T, c_g^T, then c_f^T and c_g^T cannot be close, since this would violate that p_f^T and p_g^T are (inductively) far. To prove (ii), note x_α is far from p_f^t by assumption of the Add Operation, and c_f^T is close to x_α by assumption of Case 4, so p_f^t and c_f^T must be far. Finally for (iii), one can (inductively) deduce that (b) must hold when $j = f$, so c_f^T and p_f^t are close; but, since p_f^t and p_l^t are (inductively) far, c_f^t and p_l^t must be far.

Proving the inductive step for (a) –(c) involves detailed casework. The Add and Exchange Operations are engineered so that, loosely speaking, an estimated center is either attached to the corresponding present pivot, or else breaks off to form its own pivot. A main subtlety is the direction and strength of attachment, e.g., property (c). Another is the sequence of operations, specifically, the Add Operation taking precedence over the Exchange Operation. ◀

D Proof of Theorem 3

Proof of Theorem 3. The number of labels used by the algorithm is the number of pivots in the last phase. By Lemma 5, we maintain the invariant that pivots $p_1^t, \ldots, p_t^t$ are β_t-well-separated w.r.t. the natural weights at every time step in phase t. Suppose to the contrary that the final number of pivots is strictly more than k. Then at some point there are $t = k+1$ or $t = k+2$ pivots[6] that are β_t-well-separated w.r.t. the natural weights throughout phase t. But $\beta_{k+2} = 8$, and it is impossible for $k+2$ points to be 8-well-separated, by Proposition 12. We conclude the final number of pivots is at most k. ◀

[6] The algorithm may skip a phase, hence we consider both cases.

E Proof of Lemma 7

Proof of Lemma 7. WLOG, let $j = T$. For $c \in P_T$, let $m(c)$ be the number of points assigned to c in the clustering of $X(T) \setminus X(T^-)$ induced by the centers P_T, i.e., in this clustering every point is assigned to the *nearest* point in P_T. For shorthand, let w denote the natural weights w^T of points at the end of phase T.

▶ **Observation 2.** *For $c \in P_T$, $w(c) \geq m(c)$.*

This follows from the definition of $w(c)$ and the fact that there are $m(c)$ points whose movement cost to c is at most $2\mathsf{OPT}$, by construction of P_T.

▶ **Observation 3.** *If $(p(y_i), y_i)$ is a directed edge in $D(T)$, then $w(p(y_i)) \cdot d(p(y_i), y_i) < \beta_{T+1} \cdot \mathsf{OPT}$. Likewise, if $(y_i, p(y_i))$ is a directed edge in $D(T)$, then $w(y_i) \cdot d(p(y_i), y_i) < \beta_{T+1} \cdot \mathsf{OPT}$.*

This follows from the definition of $D(T)$ and Proposition 13.

Call the points in $S_{far,T}$ *far* points. In the claims below, we show that the far points can be moved to p_T^T at bounded cost (Claims 1 and 2), and that there are not too many far points relative to the weight of p_T^T (Claim 3). In turn, we will be able to *charge* the cost of the far points to p_T^T.

▷ Claim 1. Let $p(y_i) \neq p_T^T$. Suppose $w(y_i) > w(p(y_i))$. Then $cost(S_{Ti}; p_T^T) \leq (\beta_{T+1} + 2)\mathsf{OPT}$.

Proof. WLOG, let $p(y_i) = p_1^T$. We consider two cases.

▶ **Case 1.** *$|S_{Ti}| \geq w(p_1^T)$. We will show this case cannot happen.*

We know that $w(y_i) \geq m(y_i) \geq |S_{Ti}| \geq w(p_1^T)$, and by Observation 3, that $w(p_1^T) \cdot d(p_1^T, y_i) < \beta_{T+1} \cdot \mathsf{OPT}$. By Proposition 11, this implies $w(p_1^T) \cdot d(y_i, p_T^T) \geq 2\beta_{T+1} \cdot \mathsf{OPT}$.

Since $|S_{Ti}| \geq w(p_1^T)$, there exists $S'_{Ti} \subseteq S_{Ti}$ such that $|S'_{Ti}| = w(p_1^T)$. In turn, $cost(S'_{Ti}; p_1^T) \leq cost(S'_{Ti}; y_i) + w(p_1^T) \cdot d(y_i, p_1^T) < (\beta_{T+1} + 2) \cdot \mathsf{OPT}$, since P_T is a clustering with cost at most $2\mathsf{OPT}$. On the other hand,

$$cost(S'_{Ti}; p_T^T) \geq \sum_{p \in S'_{Ti}} d(y_i, p_T^T) - \sum_{p \in S'_{Ti}} d(p, y_i) = w(p_1^T) \cdot d(y_i, p_T^T) - \sum_{p \in S'_{Ti}} d(p, y_i) \geq (2\beta_{T+1} - 2)\mathsf{OPT}.$$

Since $\beta_{T+1} \geq 4$, $\beta_{T+1} + 2 \leq 2\beta_{T+1} - 2$, so $cost(S'_{Ti}; p_1^T) < cost(S'_{Ti}; p_T^T)$, which violates that $T = \arg\min_{j \in [T]} d(p, p_j^T)$ for all $p \in S'_{Ti} \subseteq C_T$.

▶ **Case 2.** *$|S_{Ti}| \leq w_t(p_1^T)$.*

In this case, we know that since $w(p_1^T) \cdot d(y_i, p_1^T) < \beta_{T+1} \cdot \mathsf{OPT}$, we also have $|S_{Ti}| \cdot d(y_i, p_1^T) < \beta_{T+1} \cdot \mathsf{OPT}$. By the triangle inequality,

$$cost(S_{Ti}; p_1^T) \leq cost(S_{Ti}; y_i) + |S_{Ti}| \cdot d(y_i, p_1^T) \leq 2\mathsf{OPT} + \beta_{T+1} \cdot \mathsf{OPT}.$$

Since $cost(S_{Ti}; p_T^T) \leq cost(S_{Ti}; p_1^T)$ by the greedy procedure, this proves Claim 1. ◁

▷ Claim 2. Let $p(y_i) \neq p_T^T$. Suppose that $w(y_i) \leq w(p(y_i))$. Then $cost(S_{Ti}; p_T^T) \leq (\beta_{T+1} + 1)\mathsf{OPT}$.

Proof. WLOG, let $p(y_i) = p_1^T$. By Observation 3, $w(y_i) \cdot d(y_i, p_1^T) < \beta_{T+1} \cdot \mathsf{OPT}$. Further, $|S_{Ti}| \leq m(y_i) \leq w(y_i)$, so $|S_{Ti}| \cdot d(y_i, p_1^T) < \beta_{T+1} \cdot \mathsf{OPT}$. So:

$$cost(S_{Ti}; p_T^T) \leq cost(S_{Ti}; p_1^T) \leq cost(S_{Ti}; y_i) + |S_{Ti}| \cdot d(y_i, p_1^T) \leq 2\mathsf{OPT} + \beta_{T+1} \cdot \mathsf{OPT}.$$ ◁

▷ Claim 3. Let $p(y_i) \neq p_T^T$. Then $|S_{Ti}| \leq w(p_T^T)$.

Proof. As before, assume WLOG that $p(y_i) = p_1^T$.

▶ **Case 1.** $w(y_i) > w(p_1^T)$.

We know from the proof of Claim 1, Case 1 that this implies $|S_{Ti}| < w(p_1^T)$. We have

$$\begin{aligned} |S_{Ti}| \cdot d(p_T^T, y_i) = \sum_{p \in S_{Ti}} d(y_i, p_T^T) &\leq \sum_{p \in S_{Ti}} d(p, p_T^T) + \sum_{p \in S_{Ti}} d(p, y_i) \\ &\leq (\beta_{T+1} + 2)\mathsf{OPT} + 2\mathsf{OPT} \qquad \text{(Claim 1)} \\ &\leq 2\beta_{T+1} \cdot \mathsf{OPT} \leq w(p_T^T) \cdot d(p_T^T, y_i) \end{aligned}$$

where in the last line we have applied Proposition 11, using that $w(y_i) > w(p_1^T)$, Observation 3, and p_1^T and p_T^T are β_T-well-separated w.r.t. w. Finally, dividing both ends of the chain of inequalities by $d(p_T^T, y_j)$ gives $|S_{Tj}| \leq w(p_T^T)$, as desired.

▶ **Case 2.** $w(y_i) \leq w(p_1^T)$.

Consider when $w(p_T^T) \geq w(y_i)$. Then $w(p_T^T) \geq w(y_i) \geq m(y_i) \geq |S_{Ti}|$, so the claim follows.

So the last case to consider is when $w(p_T^T) < w(y_i)$. It suffices to show that $w(p_T^T) \cdot d(p_T^T, y_i) \geq 2\beta_{T+1} \cdot \mathsf{OPT}$; then, we can just apply the argument in Case 1. Suppose to the contrary that $w(p_T^T) \cdot d(p_T^T, y_i) < 2\beta_{T+1} \cdot \mathsf{OPT}$. Then

$$\begin{aligned} \beta_T \cdot \mathsf{OPT} \leq w(p_T^T) \cdot d(p_T^T, p_1^T) &\leq w(p_T^T) \cdot d(p_T^T, y_i) + w(p_T^T) \cdot d(y_i, p_1^T) \\ &\leq 2\beta_{T+1} \cdot \mathsf{OPT} + w(p_T^T) \cdot d(y_i, p_1^T) \\ &< 2\beta_{t+1} \cdot \mathsf{OPT} + w(y_i) \cdot d(y_i, p_1^T) \\ &< 2\beta_{T+1} \cdot \mathsf{OPT} + \beta_{T+1} \cdot \mathsf{OPT} = \beta_T \cdot \mathsf{OPT} \end{aligned}$$

where the second-to-last line follows from Observation 3. The left-hand and right-hand sides give a contradiction, concluding the proof of the case and the claim. ◁

▷ Claim 4. $cost(S_{far,T}; p_T^T) \leq k \cdot (\beta_{T+1} + 2)\mathsf{OPT}$ and $|S_{far,T}| \leq k \cdot w(p_T^T)$.

Proof. By Claims 1 and 2,

$$cost(S_{far,T}; p_T^T) = \sum_{i: p(y_i) \neq p_T^T} cost(S_{Ti}; p_T^T) \leq k \cdot (\beta_{T+1} + 2)\mathsf{OPT}$$

By Claim 3,

$$|S_{far,T}| = \sum_{i: p(y_i) \neq p_T^T} |S_{Ti}| \leq k \cdot w(p_T^T).$$ ◁

This concludes the proof of the claim, thus also of the lemma. ◀

F Proof of Lemma 8

Proof of Lemma 8. As in Lemma 7, let $C_j = C_j^T \setminus C_j^{T^-}$ and let S_{ji} be the set of elements in C_j assigned to y_i in the clustering of $X(T) \setminus X(T^-)$ induced by P_T. Let $S_{far,j} = \bigcup_{i: p(y_i) \neq p_j^T} S_{ji}$, $S_{near,j} = \bigcup_{i: p(y_i) = p_j^T} S_{ji}$, and S_j be the elements in C_j that are assigned to p_j^T in the clustering of $X(T) \setminus X(T^-)$ induced by P_T.

The proof is by induction. We have that

$$|C_j^T| = |C_j^{T^-}| + |C_j| = |C_j^{T^-}| + |S_{far,j}| + |S_{near,j}| + |S_j| \tag{1}$$

(Note we use that there are no points in C_j that are assigned to $p_{j'}^T$, $j' \neq j$, in the offline clustering induced by P_T, due to the greedy labelling rule. This is true as long as in the offline clustering induced by P_T we break ties consistent with how the online algorithm breaks ties.)

First, we bound the last three terms. Let w^t denote the natural weights at the end of phase t.

$$|S_{far,j}| \leq k \cdot w^T(p_j^T) \leq k \cdot w^T(c_j^T) \tag{2}$$

where the first inequality follows from Lemma 7 and the second inequality follows from the definition (‡) of estimated center. Next,

$$|S_{near,j}| = \sum_{i:p(y_i)=p_j^T} |S_{ji}| \leq \sum_{i:p(y_i)=p_j^T} w^T(y_i) \leq k \cdot w^T(c_j^T) \tag{3}$$

where the second inequality follows from the definition of w^T. The third inequality follows from the definitions of attachment digraph and estimated center: If $y_i \in \delta^-(p_j^T)$, then $w^T(y_i) \leq w^T(p_j^T)$ by construction of the attachment digraph $D(T)$. Otherwise, $y_i \in \delta^+(p_j^T)$, so by (‡), $w^T(c_j^T) \geq w^T(y_i)$. Finally,

$$|S_j| \leq w^T(p_j^T) \leq w^T(c_j^T) \tag{4}$$

where the first inequality is by the definition of w^T and the second inequality from (‡).

For simplicity, let $h(t,k) = (2k+1)t$. Now we need to bound $|C_j^{T^-}|$ in terms of $w^T(c_j^T)$. If $j \notin [T^-]$, then $|C_j^{T^-}| = 0$. So assume $j \in [T^-]$. Inductively, we have that

$$|C_j^{T^-}| \leq h(T^-,k) \cdot w^{T^-}(c_j^{T^-}).$$

We will prove that

$$|C_j^{T^-}| \leq h(T^-,k) \cdot w^T(c_j^T). \tag{5}$$

There are two cases to consider.

▶ **Case 1.** *(a) holds in Lemma 6.*

$$|C_j^{T^-}| \leq h(T^-,k) \cdot w^{T^-}(c_j^{T^-}) \leq h(T^-,k) \cdot w^T(p_j^T) \leq h(T^-,k) \cdot w^T(c_j^T).$$

▶ **Case 2.** *(b) holds in Lemma 6.*

This means that $c_j^{T^-}$ is β_{T+1}-attached to p_j^T w.r.t. w^T. If $w^T(c_j^{T^-}) \leq w^T(p_j^T)$, then $w^T(c_j^{T^-}) \leq w^T(c_j^T)$. Otherwise, $w^T(c_j^{T^-}) > w^T(p_j^T)$, so $c_j^{T^-} \in \delta^+(p_j^T)$. By (‡), $w^T(c_j^{T^-}) \leq w^T(c_j^T)$. In both cases we have $w^T(c_j^{T^-}) \leq w^T(c_j^T)$, so building from the inductive assumption,

$$|C_j^{T^-}| \leq h(T^-,k) \cdot w^{T^-}(c_j^{T^-}) \leq h(T^-,k) \cdot w^T(c_j^{T^-}) \leq h(T^-,k) \cdot w^T(c_j^T)$$

which concludes the case. Putting equations (1), (2), (3), (4), (5) together gives

$$|C_j^T| \leq \big(h(T^-,k) + 2k + 1\big) \cdot w^T(c_j^T) \leq h(T,k) \cdot w^T(c_j^T) = (2k+1) \cdot T \cdot w^T(c_j^T)$$

as desired. ◀

G Proof of Lemma 9

Proof of Lemma 9. The proof is by induction. Let C_j, S_{ji}, and $S_{far,j}$ be as in Lemma 7. Define $S_{near,j} = \bigcup_{i:p(y_i)=p_j^T} S_{ji}$ and S_j to be the elements in C_j that are assigned to p_j^T in the clustering of $X(T) \setminus X(T^-)$ induced by P_T. Let w^t denote the natural weights at the end of phase t. First we need the following key claim.

▷ Claim 1. For any $x, y \in \delta^+(p_j^T) \cup \delta^-(p_j^T) \cup \{p_j^T\}$, x and y are $2\beta_{T+1}$-attached w.r.t. w^T.

Proof of Claim 1. If x or y is p_j^T, then the claim automatically holds by Proposition 13. There are two other cases. The first case is, WLOG, $x \in \delta^-(p_j^T)$. Regardless of whether y is in $\delta^-(p_j^T)$ or $\delta^+(p_j^T)$, the claim holds by Propositions 13 and 11. The second case is that $x, y \in \delta^+(p_j^T)$. We prove the stronger statement that x and y are β_{T+1}-attached w.r.t. w^T. Suppose to the contrary that x and y are β_{T+1}-well-separated. We claim that this implies

$$\{p_1^T, \ldots, p_T^T\} \cup \{x, y\} \setminus \{p_j^T\} \tag{6}$$

is β_{T+1}-well-separated w.r.t. w^T; this would give a contradiction, since if an Exchange Operation were available, it would have been executed. Now suppose that (6) does not hold. Then WLOG $p_{j'}^T$ and x are β_{T+1}-attached w.r.t. w^T, for some $j' \neq j$. Since $x \in \delta^+(p_j^T)$ and since x and p_j^T are β_{T+1}-attached w.r.t. w^T, by Proposition 11, p_j^T and $p_{j'}^T$ are $2\beta_{T+1}$-attached w.r.t. w^T. This contradicts that p_j^T and $p_{j'}^T$ are β_T-well-separated w.r.t. w^T, since $2\beta_{T+1} < \beta_T$. This concludes the proof of the case and the claim. ◁

To bound the cost contribution of $C_j^{T^-}$, we case on which statement holds in Lemma 6.

▶ **Case 1.** *$c_j^{T^-}$ is β_{T+1}-attached to p_j^T w.r.t. w^T (i.e., (b) holds in Lemma 6).*

Since in Case 1, $c_j^{T^-}$ is β_{T+1}-attached to p_j^T w.r.t. w^T, $c_j^{T^-} \in \delta^+(p_j^T) \cup \delta^-(p_j^T)$. Also, c_j^T by definition is in $\delta^+(p_j^T) \cup \{p_j^T\}$. So by Claim 1, $c_j^{T^-}$ is $2\beta_{T+1}$-attached to c_j^T w.r.t. w^T. Using this, we bound $cost(C_j^{T^-}; c_j^T)$:

$$\begin{aligned}
cost(C_j^{T^-}; c_j^T) &\leq cost(C_j^{T^-}; c_j^{T^-}) + |C_j^{T^-}| \cdot d(c_j^{T^-}, c_j^T) \\
&\leq g(T^-, k) \cdot \mathsf{OPT} + |C_j^{T^-}| \cdot d(c_j^{T^-}, c_j^T) \\
&\leq g(T^-, k) \cdot \mathsf{OPT} + (2k+1) \cdot T^- \cdot w^{T^-}(c_j^{T^-}) \cdot d(c_j^{T^-}, c_j^T) \\
&\leq g(T^-, k) \cdot \mathsf{OPT} + (2k+1) \cdot T^- \cdot w^T(c_j^{T^-}) \cdot d(c_j^{T^-}, c_j^T) \\
&\leq g(T^-, k) \cdot \mathsf{OPT} + (2k+1) \cdot T^- \cdot 2\beta_{T+1} \cdot \mathsf{OPT}
\end{aligned} \tag{7}$$

where the third inequality is due to Lemma 8.

▶ **Case 2.** *(b) does not hold in Lemma 6, so (a) holds, i.e., $w^{T^-}(c_j^{T^-}) \leq w^T(p_j^T)$ and $w^{T^-}(c_j^{T^-}) \cdot d(c_j^{T^-}, p_j^T) \leq \beta_{T^-}(T - T^-) \cdot OPT$.*

We bound $cost(C_j^{T^-}; c_j^T)$:

$$\begin{aligned}
cost(C_j^{T^-}; c_j^T) &\leq cost(C_j^{T^-}; c_j^{T^-}) + |C_j^{T^-}| \cdot d(c_j^{T^-}, c_j^T) \\
&\leq g(T^-, k) \cdot \mathsf{OPT} + |C_j^{T^-}| \cdot d(c_j^{T^-}, p_j^T) + |C_j^{T^-}| \cdot d(p_j^T, c_j^T)
\end{aligned} \tag{8}$$

and now we use the assumptions of the case to continue bounding from (8):

$$\begin{aligned}
|C_j^{T^-}| \cdot d(c_j^{T^-}, p_j^T) &\leq (2k+1) \cdot T^- \cdot w^{T^-}(c_j^{T^-}) \cdot d(c_j^{T^-}, p_j^T) \\
&\leq (2k+1) \cdot T^- \cdot \beta_{T^-}(T - T^-) \cdot \mathsf{OPT}
\end{aligned} \tag{9}$$

where the first inequality is due to Lemma 8. Next,

$$|C_j^{T^-}| \cdot d(p_j^T, c_j^T) \leq (2k+1)T^- \cdot w^{T^-}(c_j^{T^-}) \cdot d(p_j^T, c_j^T) \leq (2k+1)T^- \cdot w^T(p_j^T) \cdot d(p_j^T, c_j^T)$$
$$\leq (2k+1)T^- \cdot \beta_{T+1} \cdot \mathsf{OPT} \quad (10)$$

where the first inequality is due to Lemma 8 and the last inequality is due to Proposition 14. So combining (8), (9), (10) gives

$$cost(C_j^{T^-}; c_j^T) \leq g(T^-, k) \cdot \mathsf{OPT} + (2k+1) \cdot T^- \cdot (\beta_{T^-}(T - T^-) + \beta_{T+1}) \cdot \mathsf{OPT}. \quad (11)$$

Now we have bounds (7) and (11) for $cost(C_j^{T^-}; c_j^T)$. Recall that $C_j^T = C_j^{T^-} \cup S_{far,j} \cup S_{near,j} \cup S_j$. The following bounds will hold regardless of whether we are in Case 1 or 2. We have

$$cost(S_j; c_j^T) \leq cost(S_j; p_j^T) + |S_j| \cdot d(p_j^T, c_j^T) \leq 2\mathsf{OPT} + w^T(p_j^T) \cdot d(p_j^T, c_j^T) \leq (2 + \beta_{T+1})\mathsf{OPT} \quad (12)$$

$$cost(S_{near,j}; c_j^T) = \sum_{i:p(y_i)=p_j^T} cost(S_{ji}; c_j^T) \leq \sum_{i:p(y_i)=p_j^T} \sum_{p \in S_{ji}} d(p, c_j^T)$$
$$\leq 2\mathsf{OPT} + \sum_{i:p(y_i)=p_j^T} w^T(y_i) \cdot d(y_i, c_j^T) \leq (2k\beta_{T+1} + 2)\mathsf{OPT} \quad (13)$$

where we have used Claim 1 and that $|S_{ji}| \leq w^T(y_i)$. Finally, by Lemma 7,

$$cost(S_{far,j}; c_j^T) \leq cost(S_{far,j}; p_j^T) + |S_{far,j}| \cdot d(p_j^T, c_j^T) \leq k(2\beta_{T+1} + 2)\mathsf{OPT} \quad (14)$$

Combining (12), (13), (14) with (7) or (11) gives the sought bound:

$$cost(C_j^T; c_j^T) \leq [g(T^-, k) + g(k)]\mathsf{OPT} \leq g(T, k) \cdot \mathsf{OPT}. \quad \blacktriangleleft$$