Unimodal-Cost k-Median on a Line

Yike Chen
University of Electronic Science and Technology of China, Chengdu, China

Chao Xu
University of Electronic Science and Technology of China, Chengdu, China

Abstract

Given n piecewise-linear unimodal functions $f_1, \ldots, f_n : \mathbb{R} \to \mathbb{R}$ and an integer $1 \le k \le n$, the Unimodal-Cost k-Median problem asks for k real numbers $y_1, \ldots, y_k$ minimizing $\sum_{i=1}^{n} \min_{1 \le r \le k} f_i(y_r)$. Let m be the number of breakpoints: the total number of affine-piece endpoint occurrences plus one occurrence at a chosen minimizer of each function. We give an exact algorithm running in

$$O\Big((m + n \log n) \log m \cdot \min\big\{k,\ \log m \sqrt{k \log m},\ \log m \cdot 2^{O(\sqrt{\log k \ \log\log m})}\big\}\Big).$$

The first term inside the minimum comes from a direct k-stage dynamic program. The other two use the minimum-weight k-link path algorithms of Aggarwal et al. [2] and Schieber [12] for Monge costs, replacing their $O(1)$ edge-weight access by batched access to the implicit transition costs.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases unimodal-cost k-median, Monge, k-link path, dynamic programming, total monotonicity

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.79

Funding Supported by the National Natural Science Foundation of China under grant 62372093 and by the CCF-Huawei Populus Grove Fund – Theoretical Computer Science Special Project CCF-HuaweiLK2025003.

Acknowledgements We thank Da Wei Zheng, who came up with the variation of the algorithm for the $k = 2$ case and contributed to initial discussions for higher k.

1 Introduction

Let $f_1, \ldots, f_n : \mathbb{R} \to \mathbb{R}$ be piecewise-linear unimodal functions with m breakpoints, and let $1 \le k \le n$. The *Unimodal-Cost k-Median* problem asks for k real numbers $y_1, \ldots, y_k$ minimizing

$$\Phi(y_1, \ldots, y_k) = \sum_{i=1}^{n} \min_{1 \le r \le k} f_i(y_r).$$

We use facility-location terminology as follows. A solution opens k *facilities*, where a facility is a chosen real number y_r. A *client* is one input function f_i. Assigning client i to facility y_r has service cost $f_i(y_r)$, and each client is assigned to a facility of minimum service cost among the chosen facilities. Thus Φ is the total service cost of all clients. The classical weighted k-median on a line corresponds to $f_i(x) = w_i|x - z_i|$ [7], while facility-location models also include variants with opening costs or an unfixed number of facilities [5].

For the symmetric line model, dynamic programming goes back to Love [10]. Hassin and Tamir [7] solve the general monotone-distance model in $O(n^2)$ time and the linear weighted k-median case in $O(kn)$ time, and Auletta et al. [3] independently obtain an $O(kn)$ algorithm for line k-median. Both algorithms assume $O(1)$ time to obtain the value of the function at a point.

For a positive integer s, define

$$T_{s,k} := \min\{\sqrt{k \log s},\ 2^{O(\sqrt{\log k \log\log s})}\}.$$

Chen and Wang give accelerated algorithms for linear costs $w_i|x - z_i|$: weighted k-median on a line in $O(nT_{n,k} \log n)$ time and the uniform case in $O(nT_{n,k})$ time. With an additional facility setup cost at each candidate location, their algorithm for this linear model runs in $O(nT_{n,k} \log^2 n)$ time [4]. Wang and Zhang [13] study the related but different line-constrained setting in the plane, where facilities lie on a prescribed line and clients are ordered by projection onto that line. These symmetric and line-constrained results rely on an ordered setting: after sorting the clients, optimal assignments are consecutive. The Unimodal-Cost 2-Median problem was introduced by Yu et al. [15], motivated by binary splits of categorical features under the mean-absolute-error criterion in CART; their algorithm runs in $O((m + n \log n) \log m)$ time, and the column-minimum primitive of Section 4 is built on their matrix search.

Several location models with asymmetric or nonstandard distances are related to ours. Drezner and Wesolowsky [6] study single-facility minisum and minimax models under asymmetric rectilinear and Euclidean distances; Plastria [11] studies Fermat-Weber problems under quasimetrics and semidirected networks; Jackson et al. [8] study one-sided directional k-median. Woeginger [14] exploits Monge structure in a linear-network proxy-placement problem, and Lin and Xue [9] study k-median under graded distance matrices.

Contributions

We give an exact algorithm for Unimodal-Cost k-Median running in time

$$O\big(\min\big\{k(m + n \log n) \log m,\ (m + n \log n) T_{m,k} \log^2 m\big\}\big),$$

where n is the number of input functions and m is the number of breakpoints. The bound is the minimum of the following three upper bounds:

- $O(k(m + n \log n) \log m)$ (Theorem 7): a direct k-stage dynamic program.
- $O((m + n \log n) \log^2 m \cdot \sqrt{k \log m})$ (Theorem 10).
- $O((m + n \log n) \log^2 m \cdot 2^{O(\sqrt{\log k \log\log m})})$ (Theorem 11).

For $k = 2$ the bound matches the previous best bound for Unimodal-Cost 2-Median [15].

The algorithm has two ingredients. The first is a reduction to a minimum-weight k-link path problem on the sorted breakpoints whose edge cost $C(u, v)$ is the marginal cost of making breakpoint v available after breakpoint u. The classical consecutive-interval formulation fails for asymmetric costs (Section 3), but this marginal cost is Monge, including the edges leaving the source (Theorem 2). The second is batched access: we do not know how to evaluate a single entry of C quickly, but the column minima of a row-shifted submatrix of C can be computed in near-linear time by extending the matrix search of Yu et al. [15] (Section 4), and the dynamic program as well as both k-link path frameworks access C only through such batches.

2 Preliminaries

A function $f : \mathbb{R} \to \mathbb{R}$ is *unimodal* if there exists $z \in \mathbb{R}$ such that f is nonincreasing on $(-\infty, z]$ and nondecreasing on $[z, \infty)$. Every such z is a minimizer of f. It is *piecewise-linear* if it is affine between consecutive points of a finite sorted set and on the two outer rays. Throughout the paper, $f_1, \ldots, f_n$ are piecewise-linear unimodal.

Let $x_1 \le \cdots \le x_m$ be the *breakpoints*: the sorted multiset obtained by taking all affine-piece endpoint occurrences of every f_i and inserting, for each i, one additional occurrence at a chosen finite minimizer of f_i. Let $\mu_i \in \{1, \ldots, m\}$ be the index of the occurrence added for i. The μ_i are pairwise distinct by construction (one inserted occurrence per function), even if the underlying coordinates x_{μ_i} coincide with other x_j. Thus m equals the total number of affine-piece endpoint occurrences plus n, and the sequence $f_i(x_1), \ldots, f_i(x_m)$ is nonincreasing up to x_{μ_i} and nondecreasing from x_{μ_i}.

The breakpoints and the μ_i are built in $O(m \log m)$ time from unsorted endpoint lists. Each f_i is stored by its endpoint list and affine pieces, so evaluating $f_i(x_v)$ takes logarithmic time by binary search and constant amortized time during the scans used below.

Write $Q := \{\mu_1, \ldots, \mu_n\} \subseteq \{1, \ldots, m\}$ for the set of minimizer indices; $|Q| = n$. For integers $L \le R$, the *index interval* $[L : R]$ is the set $\{L, L+1, \ldots, R\}$; for $K = [L : R]$ write $|K| = R - L + 1$. Then $|Q \cap K| = |\{i : \mu_i \in K\}| \le \min(|K|, n)$, and disjoint intervals partitioning $\{1, \ldots, m\}$ induce a partition of Q.

It's obvious that the optima are on breakpoints. We therefore seek $1 \le v_1 < \cdots < v_k \le m$ minimizing $\Phi(x_{v_1}, \ldots, x_{v_k})$. Since $k \le n \le m$, repeated facilities in an optimum with at most k distinct facilities can be supplemented by unused breakpoint indices, and adding facilities cannot increase $\min_r f_i(x_r)$.

3 Reduction to minimum-weight link paths

Classical algorithms for the simpler, symmetric version of the k-median problem reduce to minimum-weight link paths with Monge costs. The setting is a weighted complete directed acyclic graph on ordered vertices $1, \ldots, m$: for every $u < v$ there is an edge $u \to v$ of weight $W(u, v)$. The edge weights form a triangular matrix W, and we use the graph language and the matrix language interchangeably: a path visits an increasing sequence of vertices, its links are its edges, and each link weight is an entry of W. A k-link path is a path with exactly k edges. In the fixed-endpoint form, the minimum-weight k-link path problem asks for

$$1 = v_0 < v_1 < \cdots < v_k = m$$

minimizing $\sum_{r=1}^{k} W(v_{r-1}, v_r)$. A matrix W is Monge if $W(u_1, v_1) + W(u_2, v_2) \le W(u_1, v_2) + W(u_2, v_1)$ for all $u_1 \le u_2$ and $v_1 \le v_2$. This is the convention called (concave) Monge. For a triangular matrix defined only on pairs $u < v$, Monge means the same inequality for every $u_1 \le u_2 < v_1 \le v_2$. When W is Monge, Aggarwal et al. [2] give an $O(m\sqrt{k \log m} + m \log m)$-time algorithm and Schieber [12] gives an $m \cdot 2^{O(\sqrt{\log k \, \log\log m})}$-time algorithm for $k = \Omega(\log m)$, both assuming each entry of W is available in $O(1)$ time.

3.1 Symmetric case

Classical line k-median dynamic programs sort clients by their locations or minimizers and work on intervals of consecutive clients [10, 7, 3]. Let the sorted clients be $1, \ldots, n$. In the graph above, take the vertices to be $0, \ldots, n$; the edge $i \to j$ represents assigning the consecutive interval of clients $i+1, \ldots, j$ to one facility, and its weight $B(i, j)$ is the one-facility optimum for that interval. Thus a k-link path $0 = i_0 < i_1 < \cdots < i_k = n$ has cost $\sum_{r=1}^{k} B(i_{r-1}, i_r)$, where each summand is the one-facility cost for one consecutive interval of clients. We call this the consecutive-interval formulation. Chen and Wang [4] use it. For sorted client locations $z_1, \ldots, z_n$ and nondecreasing distance penalties ϕ_ℓ with $f_\ell(x) = \phi_\ell(|x - z_\ell|)$,

the edge weight can be written as $B(i,j) = \min_{i+1\le t\le j} \sum_{\ell=i+1}^{j} \phi_\ell(|z_\ell - z_t|)$. Symmetry is what makes this formulation work. If the cost of serving client i at x is $f_i(x) = \phi_i(|x - z_i|)$ with ϕ_i nondecreasing, then for two chosen facilities $a < b$, the clients that prefer a over b lie to the left of the clients that prefer b over a. Thus the assignment boundary between adjacent facilities is ordered on the line, and the objective can be written as a sum of one-facility costs for consecutive client intervals.

3.2 Asymmetric case

Unfortunately, with asymmetric unimodal costs, this ordered-assignment argument is no longer available. Sorting clients by minimizer does not determine how they compare two facilities: the client's preference between a and b is the sign of $f_i(a) - f_i(b)$, and this sign depends on the shape of f_i on both sides of its minimizer, not only on the location of that minimizer. A client whose minimizer lies between two others can prefer a farther facility if the relevant side of its function is flatter, or reject a nearer facility if that side is steeper. Hence clients assigned to one selected facility need not form a consecutive interval in minimizer order.

Here is a concrete failure of the interval weights B. For $c \in \mathbb{R}$ and $L, R > 0$, write

$$h_{c,L,R}(x) := \begin{cases} L(c-x), & x \le c, \\ R(x-c), & x \ge c. \end{cases}$$

Let $k = 2$ and let the four clients be

$$f_1 = h_{0,1,4}, \qquad f_2 = h_{1,3,1}, \qquad f_3 = h_{2,1,3}, \qquad f_4 = h_{3,4,1}.$$

The minimizers are ordered as $0, 1, 2, 3$. The two facilities 0 and 3 give value

$$\min\{f_1(0), f_1(3)\} + \cdots + \min\{f_4(0), f_4(3)\} = 0 + 2 + 2 + 0 = 4,$$

with clients 1 and 3 served by 0 and clients 2 and 4 served by 3. This assignment is not consecutive in minimizer order. If one instead uses consecutive-interval edge weights after sorting by minimizer, the three possible partitions into two intervals have values

$$\{1\} \mid \{2,3,4\} : 5, \qquad \{1,2\} \mid \{3,4\} : 6, \qquad \{1,2,3\} \mid \{4\} : 5.$$

Thus the consecutive-interval formulation gives value at least 5, although the true two-facility objective is at most 4 (Figure 1).

3.3 Our reduction

Our reduction uses a different order and a different edge weight. We sort the breakpoints and charge the objective to adjacent selected breakpoints. The edge $u \to v$ represents the marginal cost of making x_v available after x_u. This removes the need for the symmetry-based consecutive-interval property: the path formulation only uses the unimodality of each f_i along the selected breakpoint sequence.

Using 0 as a dummy index, define the aggregate two-point cost M by

$$M(0, v) := \sum_{i=1}^{n} f_i(x_v) \qquad (1 \le v \le m),$$

and

$$M(u, v) := \sum_{i=1}^{n} \min\{f_i(x_u), f_i(x_v)\} \qquad (1 \le u < v \le m).$$

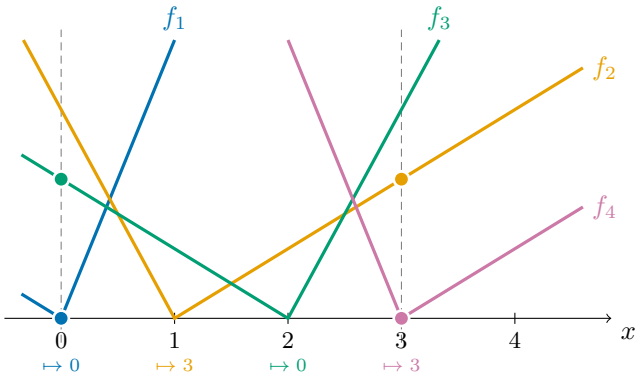

Figure 1 A two-facility instance ($k = 2$) on which consecutive intervals fail. The clients $f_1 = h_{0,1,4}$, $f_2 = h_{1,3,1}$, $f_3 = h_{2,1,3}$, $f_4 = h_{3,4,1}$ have minimizers $0, 1, 2, 3$. Facilities 0 and 3 (dashed) attain objective $0 + 2 + 2 + 0 = 4$; each dot marks the smaller of a client's two values, and the tag $\mapsto 0$ or $\mapsto 3$ beneath a minimizer gives that client's serving facility. In minimizer order the serving facilities read $0, 3, 0, 3$, so the assignment $\{f_1, f_3\} \mid \{f_2, f_4\}$ interleaves and matches no split into two consecutive intervals; the three consecutive splits cost $5, 6, 5$.

For $1 \le u < v \le m$, define the marginal edge cost $C(u, v) := M(u, v) - M(0, u)$; for the dummy index, set $C(0, v) := M(0, v)$. Define a directed acyclic graph on vertices $\{1, \dots, m\}$, with an edge $u \to v$ of cost $C(u, v)$ for every $u < v$. A feasible solution is an increasing sequence of vertices, and its cost can be charged to adjacent pairs in this graph. For fixed indices $v_1 < \cdots < v_k$, each sequence $f_i(x_{v_1}), \dots, f_i(x_{v_k})$ is unimodal: as the selected vertices pass the minimizer of f_i, the values first do not increase and then do not decrease. Hence the contribution of f_i is the value at the first selected vertex plus, for each adjacent pair, the decrease obtained when the next selected vertex is better:

$$\min_{1 \le r \le k} f_i(x_{v_r}) = f_i(x_{v_1}) + \sum_{r=2}^{k} \big(\min\{f_i(x_{v_{r-1}}), f_i(x_{v_r})\} - f_i(x_{v_{r-1}})\big).$$

After summing over i, the first term depends only on v_1, and every summand depends only on the adjacent pair (v_{r-1}, v_r). Thus minimizing over $v_1 < \cdots < v_k$ is a minimum-weight path problem with a free endpoint and exactly k selected vertices. Equivalently, add the dummy index 0 as a source vertex and an edge $0 \to v$ of cost $C(0, v) = M(0, v)$ for every vertex $v \in \{1, \dots, m\}$. The source is not counted; after the source edge, a path is an increasing sequence of vertices $v_1 \to \cdots \to v_h$ with initial cost $C(0, v_1)$ and edge costs $C(u, v)$ for $1 \le u < v \le m$. It selects the h vertices $v_1, \dots, v_h$ and has cost

$$C(0, v_1) + \sum_{r=2}^{h} C(v_{r-1}, v_r).$$

For $h = k$, this cost is $\Phi(x_{v_1}, \dots, x_{v_k})$. The endpoint is minimized over all vertices; forcing the path to end at vertex m would force x_m to be selected. (Section 5.2 converts this free endpoint into a fixed one with a sentinel construction.)

It remains to show that the transition cost C is Monge, including the edges leaving the source: the k-link path frameworks of Section 5.2 exchange path prefixes, so they use the Monge inequality also for pairs of edges whose earlier edge leaves the source. The first step is the following single-function inequality. Theorem 3 of Yu et al. [15] gives total monotonicity for $M_f(u, v) := \min\{f(x_u), f(x_v)\}$ on the triangular domain $u \le v$; here we need the stronger Monge inequality, extended to the dummy index 0.

▶ **Lemma 1.** *Let $f : \mathbb{R} \to \mathbb{R}$ be piecewise-linear unimodal and let $x_1 \le \cdots \le x_m$ be any sorted real sequence. Define $M_f(u, v) := \min\{f(x_u), f(x_v)\}$ for $1 \le u < v \le m$, and $M_f(0, v) := f(x_v)$ for $1 \le v \le m$. Then, for all $0 \le u_1 \le u_2 < v_1 \le v_2$,*

$$M_f(u_1, v_1) + M_f(u_2, v_2) \le M_f(u_1, v_2) + M_f(u_2, v_1).$$

Proof. Let $z \in \mathbb{R}$ be a minimizer of f and let $p \in \{0, \dots, m\}$ be maximal with $x_p \le z$; set $x_0 := -\infty$ and $f(x_0) := +\infty$. Then $M_f(0, v) = \min\{f(x_0), f(x_v)\}$, indices at most p lie on the nonincreasing side of f, and indices greater than p lie on the nondecreasing side. Fix $0 \le u_1 \le u_2 < v_1 \le v_2$. There are five possible positions of the cut p: $p < u_1$ (Case 1), $u_1 \le p < u_2$ (Case 2), $u_2 \le p < v_1$ (Case 3), $v_1 \le p < v_2$ (Case 4), and $p \ge v_2$ (Case 5). If $u_1 = 0$, Case 1 cannot occur because $p \ge 0$, and the other cases read with $f(x_{u_1}) = +\infty$. We settle the cases in order of difficulty.

Case 1 ($p < u_1$). All four indices are greater than p, so f is nondecreasing across them: $f(x_{u_j}) \le f(x_{v_k})$ for all j, k, whence each $M_f(u_j, v_k) = f(x_{u_j})$. Both sides of the Monge inequality equal $f(x_{u_1}) + f(x_{u_2})$.

Case 5 ($p \ge v_2$). All four indices are at most p, so f is nonincreasing across them and each $M_f(u_j, v_k) = f(x_{v_k})$. Both sides equal $f(x_{v_1}) + f(x_{v_2})$.

Case 2 ($u_1 \le p < u_2$). Then u_2, v_1, v_2 all lie on the nondecreasing side, so $f(x_{u_2}) \le f(x_{v_1}) \le f(x_{v_2})$ and $M_f(u_2, v_1) = M_f(u_2, v_2) = f(x_{u_2})$. These two cancel between LHS and RHS, reducing the Monge inequality to $M_f(u_1, v_1) \le M_f(u_1, v_2)$; min is nondecreasing in each argument and $f(x_{v_1}) \le f(x_{v_2})$.

Case 4 ($v_1 \le p < v_2$). Symmetrically, u_1, u_2, v_1 all lie on the nonincreasing side, so $f(x_{u_1}) \ge f(x_{u_2}) \ge f(x_{v_1})$ and $M_f(u_1, v_1) = M_f(u_2, v_1) = f(x_{v_1})$. These cancel, reducing Monge to $M_f(u_1, v_2) \ge M_f(u_2, v_2)$, which holds because $f(x_{u_1}) \ge f(x_{u_2})$.

Case 3 ($u_2 \le p < v_1$). Set $a_j := f(x_{u_j})$ and $b_j := f(x_{v_j})$; then $a_1 \ge a_2$ (both on the nonincreasing side) and $b_1 \le b_2$ (both on the nondecreasing side). The function $g(x) := \min\{a_1, x\} - \min\{a_2, x\}$ has value 0 on $(-\infty, a_2]$, $x - a_2$ on $[a_2, a_1]$, and $a_1 - a_2$ on $[a_1, \infty)$; since $a_1 \ge a_2$, g is nondecreasing, so $g(b_1) \le g(b_2)$. Expanding, $\min\{a_1, b_1\} + \min\{a_2, b_2\} \le \min\{a_2, b_1\} + \min\{a_1, b_2\}$, which is $M_f(u_1, v_1) + M_f(u_2, v_2) \le M_f(u_2, v_1) + M_f(u_1, v_2)$. ◀

We next prove that the transition cost is Monge.

▶ **Theorem 2.** *Let $f_1, \dots, f_n$ be piecewise-linear unimodal with breakpoints $x_1 \le \cdots \le x_m$, and let*

$$C(u, v) = \sum_{i=1}^{n} \min\{f_i(x_u), f_i(x_v)\} - \sum_{i=1}^{n} f_i(x_u) \quad (1 \le u < v \le m), \qquad C(0, v) = \sum_{i=1}^{n} f_i(x_v).$$

Then the edge cost C is Monge, including the source row: for every $0 \le u_1 \le u_2 < v_1 \le v_2$,

$$C(u_1, v_1) + C(u_2, v_2) \le C(u_1, v_2) + C(u_2, v_1).$$

Proof. Each summand $\min\{f_i(x_u), f_i(x_v)\}$, extended to $u = 0$ by $M_{f_i}(0, v) = f_i(x_v)$, is Monge by Lemma 1. Summing the four-point inequality over i gives, for every $0 \le u_1 \le u_2 < v_1 \le v_2$,

$$M(u_1, v_1) + M(u_2, v_2) \le M(u_1, v_2) + M(u_2, v_1).$$

Set $\sigma(0) := 0$ and $\sigma(u) := M(0, u)$ for $1 \le u \le m$, so that $C(u, v) = M(u, v) - \sigma(u)$ for every $0 \le u < v \le m$. Then

$$C(u_1, v_1) + C(u_2, v_2) = M(u_1, v_1) + M(u_2, v_2) - \sigma(u_1) - \sigma(u_2),$$

and the same row-shift term $\sigma(u_1) + \sigma(u_2)$ is subtracted from the right-hand side. Thus the Monge inequality for M gives the Monge inequality for C. ◀

4 Column-minimum primitives

All row/column minima of an $r \times c$ Monge matrix (one minimum per row) can be computed in $O(r + c)$ time, i.e. linear in the matrix dimensions rather than in its rc entries, assuming $O(1)$ access to each entry [1]. The fast minimum k-link path algorithms of Section 3 are built on this kind of batched access. In Chen and Wang's line-median setting [4], preprocessing gives a single-entry oracle for the interval edge weight $B(i, j)$ in $O(\log n)$ time, and in $O(1)$ time for the uniform case. However, for our transition cost,

$$C(u, v) = \sum_{i=1}^{n} \min\{f_i(x_u), f_i(x_v)\} - \sum_{i=1}^{n} f_i(x_u),$$

direct evaluation of one entry scans the input functions, and we do not know a preprocessing scheme that makes individual entries fast enough.

However, the three algorithms we care about – the direct dynamic program of Section 5.1 and the two k-link path frameworks of Section 5.2 – access the Monge matrix only through three higher-level primitives: ColumnMin, Path, and Decision. This section describes how to compute them quickly.

The primitives need the values $M(0, v)$ for all breakpoints v, both as the row shifts inside C and as the source-edge costs $C(0, v)$. They are not a bottleneck: they are the values of the pointwise sum of the input piecewise-linear functions on the breakpoints, and can be computed once within the $O(m \log m)$ preprocessing.

The selected-vertex count of a path is the number of vertices in $\{1, \ldots, m\}$ on it, excluding the source. We use one of two tie-breaking rules, fixed per computation: when a minimum is attained by several paths with the same value, keep the minimum selected-vertex count, or keep the maximum selected-vertex count.

For $\tau \in \mathbb{R}$, let $\mathcal{G}_\tau$ be the penalized graph on the vertices $\{0, 1, \ldots, m\}$ with an edge $u \to v$ of cost $C(u, v) + \tau$ for every $0 \le u < v \le m$; by Theorem 2, its costs are Monge for every τ. A path $0 \to v_1 \to \cdots \to v_h$ in $\mathcal{G}_\tau$ has selected-vertex count h.

For an interval $K = [L : R]$, an initial vector $A : K \to \mathbb{R} \cup \{+\infty\}$, and $\tau \in \mathbb{R}$, let $\mathcal{H}(K, A, \tau)$ be the graph with source s_K, vertices K, source edge $s_K \to v$ of cost $A(v)$ for every finite $A(v)$, and edge $u \to v$ of cost $C(u, v) + \tau$ for every $L \le u < v \le R$. The finite entries of A may carry a predecessor pointer and a selected-vertex count, and in $\mathcal{H}(K, A, \tau)$ the selected-vertex count of a path is counted cumulatively from this metadata: a source edge $s_K \to v$ carries the predecessor pointer and selected-vertex count stored with $A(v)$, and every internal edge $u \to v$ appends v and increases the selected-vertex count by one.

The *finite support* of a vector $A : K \to \mathbb{R} \cup \{+\infty\}$ is the set of indices $v \in K$ at which $A(v)$ is finite. Every offset and initial vector used below has finite support equal to an index interval.

▶ **Definition 3.** *We use the following primitives.*

- ColumnMin(K, a)*: for an interval* $K = [L : R] \subseteq \{1, \dots, m\}$ *and a row-offset vector* $a : [L : R-1] \to \mathbb{R} \cup \{+\infty\}$ *whose finite entries form an index interval, return, for every* $L < v \le R$*,*

$$U(v) := \min_{L \le u < v} \{a(u) + C(u, v)\},$$

together with an argmin predecessor u *when the minimum is finite. If the finite entries of* a *are stored with selected-vertex counts and a tie-breaking rule is specified, then the predecessor is chosen among the rows attaining* $U(v)$ *according to that rule.*

- Path(K, A, τ)*: for an interval* $K = [L : R]$*, an initial vector* $A : K \to \mathbb{R} \cup \{+\infty\}$ *whose finite entries form an index interval, and a scalar edge penalty* $\tau \in \mathbb{R}$*, return, for every* $v \in K$*, the shortest-path value from* s_K *to* v *in* $\mathcal{H}(K, A, \tau)$*, together with a predecessor and the selected-vertex count determined by the tie-breaking rule.*

- Decision(τ)*: return the selected-vertex count of a shortest path in* $\mathcal{G}_\tau$ *from* 0 *to the last vertex* m*. Under minimum-count tie-breaking, return the minimum such count among shortest* $0 \to m$ *paths; under maximum-count tie-breaking, return the maximum such count.*

In words: every edge carries the same penalty τ, and the three primitives ask for cheapest penalized paths in $\mathcal{G}_\tau$. ColumnMin extends known path values by one edge, for all destinations in one batch; it takes no penalty argument, since a one-edge extension adds the same τ to every candidate, which the caller adds to the output. Path extends by any number of edges, returning all penalized shortest-path values from a common source; it is computed by iterating ColumnMin (Theorem 5). Decision is one Path call on the whole penalized graph that reports only how many vertices a cheapest penalized $0 \to m$ path selects (Theorem 6).

▶ **Theorem 4.** *Let* $f_1, \dots, f_n$ *be piecewise-linear unimodal with* m *breakpoints, and let* $Q = \{\mu_1, \dots, \mu_n\}$ *be the set of minimizer indices. Then* ColumnMin(K, a) *can be computed in time* $O\big((|K| + |Q \cap K| \log |Q \cap K|) \log |K|\big)$.

Proof. Yu et al. search the matrix $M(u, v) = \sum_{i=1}^{n} \min\{f_i(x_u), f_i(x_v)\}$ over a rectangular index range by divide and conquer [15]: the minimum of the middle line is found by an amortized sweep, and by monotonicity of the argmins the search recurses on the two complementary quadrants, in time $O\big((|K| + |Q \cap K| \log |Q \cap K|) \log |K|\big)$ on ranges inside K. We modify this search in three ways. First, their algorithm reports the minimum entry of the range; we record the middle-line minimum computed at every node of the recursion instead. Every column of K is a middle line at exactly one node, so this returns the minimum and an argmin of every column within the same bound. Second, we add the offset: the searched entries are $a(u) + C(u, v) = M(u, v) + \big(a(u) - \sum_i f_i(x_u)\big)$, where the parenthesized row offset is available in $O(1)$ time per access from the vector a and the precomputed sums, and its finite entries form an index interval, so the search runs on that interval of rows. A row offset preserves the Monge inequality (it cancels from both sides, as in the proof of Theorem 2), hence also the argmin monotonicity that drives the recursion, and it adds $O(1)$ per accessed entry to the sweeps. Third, count tie-breaking compares $(a(u) + C(u, v), c(u))$ or $(a(u) + C(u, v), -c(u))$ lexicographically, where $c(u)$ is the selected-vertex count stored with $a(u)$. The second coordinate depends only on the row. By the Monge inequality the difference $\big(a(u_2) + C(u_2, v)\big) - \big(a(u_1) + C(u_1, v)\big)$ is nonincreasing in v for $u_1 < u_2$; hence a strict win of the later row persists at all later columns, a tie between two rows resolves by the count identically at every column where it occurs, and the lexicographic argmin is monotone in the column. The matrix search therefore applies unchanged. ◀

The bound is local: the $|K|$ term pays for the rows and columns of the submatrix, while the $|Q \cap K| \log |Q \cap K|$ term pays only for the functions whose minimizer indices lie in that submatrix.

The matrix search never materializes the submatrix; it reads entries of C only along monotone sweeps of the row and column indices, inside the divide and conquer. Along such a sweep the aggregate $\sum_i \min\{f_i(x_u), f_i(x_v)\}$ is maintained incrementally: each input function changes which of its two endpoints is the cheaper one at most once as an index advances, so over the sweep the running sum is updated in $O(1)$ amortized time per accessed cell, using the precomputed $M(0, v) = \sum_i f_i(x_v)$ and the function-sum data structure of Yu et al. [15]. A function whose minimizer index lies outside the swept interval is monotone there and contributes a single endpoint value already folded into the precomputed sums; only the $|Q \cap K|$ functions whose minimizer lies inside the interval can switch and need individual attention. This is why the cost in Theorem 4 is governed by $|K|$ and $|Q \cap K|$ rather than by the $\Theta(|K|^2)$ entries. It is also why a single entry $C(u, v)$ is not evaluated in constant time in isolation: a constant-size interval is cheap only because at most a constant number of functions can have a minimizer inside it, the rest being accounted for in bulk by the precomputed sums.

▶ **Theorem 5.** *Let $f_1, \ldots, f_n$ be piecewise-linear unimodal with m breakpoints, and let $Q = \{\mu_1, \ldots, \mu_n\}$ be the set of minimizer indices. Then* $\textsc{Path}(K, A, \tau)$ *can be computed in time $O\big((|K| + |Q \cap K| \log |Q \cap K|) \log^2 |K|\big)$. For $K = [1 : m]$ this is $O((m + n \log n) \log^2 m)$.*

Proof. The shortest-path values in $\mathcal{H}(K, A, \tau)$ satisfy

$$P(L) := A(L), \qquad P(v) := \min\Big\{A(v),\ \min_{L \le u < v} \big(P(u) + C(u, v) + \tau\big)\Big\} \quad (L < v \le R),$$

with the empty minimum equal to $+\infty$. The term $A(v)$ is the source edge $s_K \to v$; the second term chooses the last internal edge $u \to v$.

The computation is defined by divide and conquer. For $K = [L : L]$, it returns $P(L) := A(L)$. For $K = [L : R]$ with $L < R$, let $c := \lfloor (L + R)/2 \rfloor$. The left subproblem is $\textsc{Path}([L : c], A|_{[L:c]}, \tau)$, where $A|_{[L:c]}$ denotes the restriction of A to $[L : c]$. For its returned values $P(u)$, define $a : [L : R - 1] \to \mathbb{R} \cup \{+\infty\}$ by

$$a(u) := \begin{cases} P(u), & L \le u \le c, \\ +\infty, & c < u < R. \end{cases}$$

Let U be the output of $\textsc{ColumnMin}(K, a)$. For $c < v \le R$, define

$$A^+(v) := \min\{A(v), U(v) + \tau\},$$

with the predecessor and selected-vertex count coming from the term that attains the minimum, increasing the count by one when the term is $U(v) + \tau$. The right subproblem is $\textsc{Path}([c + 1 : R], A^+, \tau)$.

We prove by induction on $|K|$ that this computation returns the shortest-path values in $\mathcal{H}(K, A, \tau)$, stores the predecessor and count metadata, and returns finite entries forming either the empty set or a suffix of K. If $L = R$, the recurrence gives $P(L) = A(L)$, and the finite support is either empty or $\{L\}$. Otherwise, by induction, the finite entries returned by the left subproblem form an index interval; hence the row offset a satisfies the precondition of $\textsc{ColumnMin}$.

The right initial vector also satisfies the finite-support precondition. If the left subproblem returns no finite value, then the finite entries of A^+ are exactly the finite entries of A restricted to $[c + 1 : R]$. Otherwise $U(v)$ is finite for every $c < v \le R$, and the finite entries of A^+ are

all of $[c+1:R]$. Thus the finite entries of A^+ form an index interval. The returned finite entries are the empty set or a suffix of K: if the left support is empty this follows from the right subproblem, and otherwise the left suffix is followed by all of $[c+1:R]$.

For correctness at $v \in [c+1:R]$, the shortest path either uses its last internal edge from $[L:c]$ (captured by $U(v)+\tau$), uses its last internal edge from $[c+1:v-1]$ (captured by the right subproblem with initial vector A^+), or uses the source edge $s_K \to v$ (captured by $A(v)$). The case $v \in [L:c]$ is the left subproblem.

For the running time, write $\mathrm{work}(K)$ for the total time the computation spends on interval K: the ColumnMin(K,a) call together with its two recursive calls, so that

$$\mathrm{work}(K) = \mathrm{work}\big([L:c]\big) + \mathrm{work}\big([c+1:R]\big) + O\big((|K| + |Q\cap K|\log|Q\cap K|)\log|K|\big).$$

Unrolling the recurrence over $O(\log|K|)$ depths gives disjoint subintervals at each depth with $\sum_J |J| \le |K|$ and $\sum_J |Q\cap J| \le |Q\cap K|$. For every subinterval J, $|Q\cap J|\log|Q\cap J| \le |Q\cap J|\log|Q\cap K|$, so $\sum_J |Q\cap J|\log|Q\cap J| \le |Q\cap K|\log|Q\cap K|$. Thus each depth does $O\big((|K| + |Q\cap K|\log|Q\cap K|)\log|K|\big)$ work, and summing over the $O(\log|K|)$ depths gives the stated bound.

Each ColumnMin call returns both the optimal value and an argmin u^*, chosen under the requested count tie-breaking when values are equal. If a transition chooses u^*, the selected-vertex count is the count stored at u^* plus one; if it chooses $A(v)$, the stored count of $A(v)$ is retained. The comparison with $A(v)$ uses the same tie-breaking rule and costs $O(1)$. ◀

▶ **Theorem 6.** *Let $f_1,\dots,f_n$ be piecewise-linear unimodal with m breakpoints. Then* Decision(τ) *can be computed in time $O((m+n\log n)\log^2 m)$.*

Proof. Set

$$A_\tau(v) := C(0,v) + \tau,$$

store selected-vertex count 1 with every entry, and run Path$([1:m], A_\tau, \tau)$. The graph $\mathcal{H}([1:m], A_\tau, \tau)$ is exactly $\mathcal{G}_\tau$, identifying $s_{[1:m]}$ with the source 0, so the value returned at vertex m is the shortest-path value from 0 to m. The stored count is initialized to 1 and increases once per appended vertex, so the same tie-breaking rule returns the required selected-vertex count. The time follows from Theorem 5 on $K=[1:m]$. ◀

5 Applications of the primitives

5.1 Direct dynamic program

▶ **Theorem 7.** *Unimodal-Cost k-Median on n piecewise-linear unimodal functions with m breakpoints can be solved in time $O\big(k(m+n\log n)\log m\big)$.*

Proof. Define

$$D_1(v) := C(0,v) \qquad (1 \le v \le m),$$
$$D_t(v) := \min_{1\le u<v} \big\{D_{t-1}(u) + C(u,v)\big\} \qquad (2 \le t \le k,\ 1 \le v \le m),$$

with the empty minimum equal to $+\infty$. Here $D_t(v)$ is the best path value using t vertices and ending at v: the term $C(0,v)$ is the one-vertex cost, and $C(u,v)$ is the marginal cost of appending v after u. Therefore the reduction in Section 3 gives the optimum as $\min_v D_k(v)$.

By induction, the finite entries of D_t are exactly $[t : m]$: this is true for D_1, and the recurrence has a finite predecessor for v exactly when $v \geq t$. Therefore, for $t \geq 2$, the transition for columns $2, \ldots, m$ is exactly ColumnMin$([1 : m], a)$ with $a(u) := D_{t-1}(u)$ for $1 \leq u < m$, while $D_t(1) = +\infty$ by the empty minimum. The finite entries of a form the index interval $[t-1 : m-1]$. Since $Q \cap [1 : m] = Q$, one stage costs $O((m + n \log n) \log m)$. Over $k-1$ stages this is $O(k(m + n \log n) \log m)$. The argmin predecessors returned by ColumnMin recover an optimal index tuple $(v_1, \ldots, v_k)$ in the same time. ◀

The $k = 2$ case. Taking $k = 2$ recovers the running time of Yu et al.'s Unimodal-Cost 2-Median algorithm [15]: here $D_2(v) = \min_{1 \leq u < v}\{C(0, u) + C(u, v)\}$, so the optimum $\min_v D_2(v)$ is a single ColumnMin$([1 : m], a)$ call with $a(u) := C(0, u)$, followed by a linear scan. The cost is $O((m+n \log n) \log m)$, matching their bound. Thus ColumnMin is exactly the 2-median primitive, and Theorem 7 solves k-median by iterating it $k-1$ times.

5.2 Path algorithms

Finding an unconstrained shortest path is easy: it is a single Path computation. The hard part is forcing the path to have exactly k links. Aggarwal et al. [2] and Schieber [12] solve this count-constrained problem, and share one idea. Fix a penalty τ and pass to the penalized graph $\mathcal{G}_\tau$: a minimum-weight path there trades its true cost against τ times its link count, so increasing τ decreases the link count of the cheapest path. Searching for a penalty at which some shortest path uses exactly k links solves the count-constrained problem through unconstrained penalized computations, a Lagrangian relaxation of the count constraint. The two frameworks differ only in how they search. We reuse that scaffolding unchanged and supply every access to the graph through the primitives of Section 4.

Both frameworks fix both endpoints of the path, and their correctness rests on this: the existence of an optimal penalty is proved through the convexity of the minimum k-link path weight in k [2, 12], and the exchange argument behind that convexity compares paths with a common endpoint. Our path formulation instead has a free endpoint, and for free endpoints this convexity can fail for general Monge costs. We therefore make the endpoint fixed with a sentinel: we tilt every function upward after the last breakpoint, so that one additional rightmost breakpoint is a facility that no client ever uses.

▶ **Lemma 8.** *Let $f_1, \ldots, f_n$ be piecewise-linear unimodal with breakpoints $x_1 \leq \cdots \leq x_m$. Set $\lambda := 1 + \max_{1 \leq i \leq n} \max\{f_i(x_1) - f_i(x_m), 0\}$ and define the tilted functions*

$$f_i'(x) := f_i(x) + \lambda \cdot \min\big\{\max\{x - x_m, 0\}, 1\big\}.$$

Then each f_i' is piecewise-linear unimodal with the same chosen minimizer as f_i and two additional affine-piece endpoints, at x_m and at $x_m + 1$; moreover $f_i' = f_i$ on $(-\infty, x_m]$ and $f_i'(x_m + 1) > f_i(x)$ for every i and every $x \in [x_1, x_m]$. Consequently, for every nonempty finite $S \subseteq [x_1, x_m]$,

$$\sum_{i=1}^{n} \min_{y \in S \cup \{x_m+1\}} f_i'(y) = \sum_{i=1}^{n} \min_{y \in S} f_i(y).$$

Proof. The tilt $\lambda \min\{\max\{x - x_m, 0\}, 1\}$ is nondecreasing, piecewise-linear with affine-piece endpoints x_m and $x_m + 1$, and vanishes on $(-\infty, x_m]$. The chosen minimizer z_i of f_i is a breakpoint, so $z_i \leq x_m$; adding a nondecreasing function that vanishes on $(-\infty, x_m]$ keeps f_i' nonincreasing on $(-\infty, z_i]$ and nondecreasing on $[z_i, \infty)$, and $f_i' = f_i$ on $(-\infty, x_m]$. For $x \in [x_1, x_m]$, unimodality gives $f_i(x) \leq \max\{f_i(x_1), f_i(x_m)\}$, while

$$f_i'(x_m+1) = f_i(x_m+1) + \lambda \geq f_i(x_m) + \lambda > \max\{f_i(x_1), f_i(x_m)\},$$

using that f_i is nondecreasing on $[x_m, \infty)$ and the choice of λ. Hence for nonempty $S \subseteq [x_1, x_m]$, the minimum of f_i' over $S \cup \{x_m+1\}$ is never attained at the sentinel, and on S the tilted and original functions agree. ◀

Run the reduction of Section 3 on the tilted instance. Its breakpoint list is the old list plus, for each function, one endpoint occurrence at x_m and one at x_m+1, so it has $m'' := m + 2n \leq 3m$ breakpoints and the same minimizer-index count n; the last index m'' has coordinate x_m+1 and serves as the sentinel. The tilted instance is built in $O(m)$ time after the preprocessing of Section 2, and we write C for its transition cost, which is Monge including the source row by Theorem 2.

▶ **Lemma 9.** *The minimum weight of a $(k+1)$-link path from 0 to m'' in the graph of Section 3 on the tilted instance equals the Unimodal-Cost k-Median optimum, and an optimal path yields an optimal set of k facilities in $O(m)$ additional time.*

Proof. A $(k+1)$-link path from 0 to m'' selects k interior vertices and the sentinel. Let T be the multiset of interior coordinates at most x_m, and let OPT denote the k-median optimum. Every selected coordinate is either in T or equals x_m+1, so by the identity of Section 3 the path costs $\sum_i \min_{y \in T \cup \{x_m+1\}} f_i'(y)$. If $T \neq \emptyset$ this equals $\sum_i \min_{y \in T} f_i(y) \geq \mathrm{OPT}$ by Lemma 8, since $|T| \leq k$ and adding facilities cannot increase the objective; if $T = \emptyset$ the cost is $\sum_i f_i'(x_m+1) > \sum_i f_i(x_1) \geq \mathrm{OPT}$. Conversely, an optimal solution consists of k distinct original breakpoint indices (Section 2); appending the sentinel index m'' gives a $(k+1)$-link path from 0 to m'' of cost OPT by Lemma 8. From an optimal path, the coordinates in T, padded to k facilities as in Section 2 when $|T| < k$, form an optimal solution. ◀

Both frameworks access the graph in two ways, which our primitives supply on the tilted instance. A *relaxation* is a batched computation $\min_u\{a(u) + C(u,v)\}$ over a row interval and a later column interval, that is, a ColumnMin call after padding the offset vector with $+\infty$. A *penalized decision* at τ computes the link counts of minimum-weight $0 \to m''$ paths in $\mathcal{G}_\tau$ under both tie-breaking rules, that is, two Decision(τ) calls. Every weight the frameworks compare at a candidate penalty τ has the form of a true cost plus τ times a link count; the counts stored by our primitives supply these link counts, since a path from the source with h selected vertices has exactly h links. The frameworks' exchange arguments rewire a crossing pair of edges and use the Monge inequality to certify that the total weight does not increase [2, 12]; the earlier edge of a crossing pair can leave the source, which is why Lemma 1 and Theorem 2 include the source row.

▶ **Theorem 10.** *Unimodal-Cost k-Median on n piecewise-linear unimodal functions with m breakpoints can be solved in time $O\big((m + n\log n)\log^2 m\sqrt{k\log m}\big)$.*

Proof. Let $g := m + n\log n$. If $k < \log m$, Theorem 7 gives $O(kg\log m) \subseteq O(g\log^2 m\sqrt{k\log m})$, so assume $k \geq \log m$. By Lemma 9 it suffices to find a minimum-weight $(k+1)$-link path from 0 to m'' on the tilted instance; since $m'' = O(m)$ and the minimizer-index count is still n, we keep writing m and g for its parameters.

Use the guide algorithm of Aggarwal et al. [2] with stage size $b := \lceil m\sqrt{\log m/k}\rceil$ (their stage size is $\min\{m\sqrt{\log m/k}, m\}$, which equals $m\sqrt{\log m/k}$ since $k \geq \log m$). The algorithm partitions the m vertices into m/b intervals of b consecutive vertices and runs one stage per interval, keeping an interval of candidate penalties τ that contains an optimal one

and over which every comparison it makes is invariant. Within a stage it (i) computes, for each vertex of the stage's interval, a minimum-weight penalized path whose last edge leaves the prefix of vertices before the interval; such paths take one of two link counts [2], so this is $O(1)$ relaxations per stage whose row offsets are values of minimum-weight paths with a fixed link count on the prefix, spanning at most $[1 : m]$; (ii) computes, inside the stage's interval, candidate paths for each attained link count, one local relaxation per count, which over all stages telescopes to $O(k)$ relaxations; and (iii) localizes τ by $O(\log m)$ penalized decisions. The offset vectors in (i) and (ii) are finite exactly on the vertices reached by a minimum-weight path with the given link count, and those vertices form an index interval [2], so every relaxation satisfies the precondition of Definition 3. A relaxation inside one stage spans an interval J of $O(b)$ indices and costs $O\big((|J|+|Q\cap J|\log|Q\cap J|)\log|J|\big) = O(b\log^2 m)$ by Theorem 4 and $|Q\cap J| \le |J|$; a decision costs $O(g\log^2 m)$ by Theorem 6; a prefix-anchored relaxation spans at most $[1 : m]$ and costs $O(g\log m)$. The comparisons among already-computed candidate values that the parametric search resolves are lexicographic comparisons of value-count pairs at the endpoints of the penalty interval, $O(1)$ each and $O(kb)$ in total. Therefore the costs are

$$O(kb\log^2 m), \qquad O\big((m/b)g\log^3 m\big), \qquad O\big((m/b)g\log m\big).$$

Since $m/b = O(\sqrt{k/\log m})$, the second term is $O(g\log^2 m\sqrt{k\log m})$ and dominates the third. Also $O(kb\log^2 m) = O(m\log^2 m\sqrt{k\log m}) \subseteq O(g\log^2 m\sqrt{k\log m})$. At the optimal penalty the algorithm has minimum-weight $0 \to m''$ paths with the minimum and the maximum link count, and combines them into a minimum-weight $(k+1)$-link path by one path swap [2] in $O(m)$ time; the stored predecessors recover the selected vertices, and Lemma 9 converts the path into k facilities. ◀

▶ **Theorem 11.** *Unimodal-Cost k-Median on n piecewise-linear unimodal functions with m breakpoints can be solved in time $O\big((m+n\log n)\log^2 m\cdot 2^{O(\sqrt{\log k\ \log\log m})}\big)$.*

Proof. Let $g := m + n\log n$. If $k < \log m$, Theorem 7 gives $O(kg\log m) \subseteq O(g\log^2 m)$, so assume $k \ge \log m$. Then $\log\log m \le \log k$, so $\log m = 2^{\log\log m} \le 2^{\sqrt{\log k\ \log\log m}}$: any factor polylogarithmic in m is $2^{O(\sqrt{\log k\ \log\log m})}$ and will be absorbed into the final bound. By Lemma 9 it suffices to find a minimum-weight $(k+1)$-link path from 0 to m'' on the tilted instance; as before we keep writing m and g for its parameters.

Schieber's algorithm [12] runs k/ℓ stages for a parameter ℓ. Stage t computes the maximal vertex range $[L_t : R_t]$, with $L_t > R_{t-1}$, on which the minimum-weight penalized paths from the source have exactly $t\ell$ links, together with the values of those paths, by $O(\log m)$ binary-search iterations. Each iteration runs one penalized decision on the whole graph plus one recursive computation with target count $O(\ell)$ on an auxiliary graph: a source whose edge to each vertex carries the value of the cheapest penalized path anchored in $[L_{t-1} : R_{t-1}]$, followed by an interval J_t of consecutive vertices starting at $R_{t-1}+1$; the auxiliary costs are Monge including the source row [12]. Schieber bounds the size of the largest auxiliary interval of stage t by $2(R_t - R_{t-1})$ [12], so writing $m_t := |J_t|$ for that largest size, telescoping gives $\sum_t m_t \le 2m$.

In our implementation, a penalized decision is two Decision(τ) calls, at $O(g\log^2 m)$ by Theorem 6. The relaxation building the anchored values of stage t spans at most $[1 : m]$ and costs $O(g\log m)$; its offsets are the exactly-$((t-1)\ell)$-link path values, finite on all of $[L_{t-1} : R_{t-1}]$, so the precondition of Definition 3 holds. The recursive computation is the same count-constrained problem one level down: its internal edge costs are the restriction of C to J_t and its source-edge values form an initial vector with interval support, so all of its primitive calls are Path and ColumnMin calls on subintervals of J_t.

Inside the recursion we charge count-obliviously: on an interval J with $|J| = m'$ we bound $|Q \cap J| \le m'$, so each primitive call on J costs $O\big((m' + m' \log m') \log^2 m'\big) = O(m' \log^3 m')$. Let $S(m', k')$ be the resulting cost of the recursion on an interval of m' vertices with target count k'. The stage structure one level down is identical, so

$$S(m', k') \le c \log m' \left(\frac{k'}{\ell'} m' \log^3 m' + \sum_t S(m'_t, \ell') \right), \qquad \sum_t m'_t \le 2m',$$

which is Schieber's Section 3.1 recurrence with the linear unit m' replaced by $m' \log^3 m'$. His induction uses only his choice of ℓ', the bound $\sum_t m'_t \le 2m'$, and a base case $k' = O(1)$, which for us is $O(1)$ ColumnMin stages seeded with the initial vector, as in the proof of Theorem 7, at cost $O(m' \log^2 m')$. Since $\sum_t m'_t \log^3 m'_t \le 2m' \log^3 m'$, the unit replacement goes through, and the recurrence solves to $S(m', k') = m' \log^3 m' \cdot 2^{O(\sqrt{\log k' \ \log\log m'})}$.

At the top level, the $O((k/\ell) \log m)$ iterations cost $O(g \log^2 m)$ each for the decisions and the anchored relaxations, plus the recursions, for a total of

$$O\left(\frac{k}{\ell} g \log^3 m\right) + O(\log m) \cdot \sum_{t=1}^{k/\ell} S(m_t, \ell) = O\left(\frac{k}{\ell} g \log^3 m\right) + O\big(m \log^4 m \cdot 2^{O(\sqrt{\log \ell \ \log\log m})}\big).$$

Choose $\ell := \lceil k \log m \cdot 2^{-\sqrt{\log k \ \log\log m}} \rceil$. If the ceiling is 1 then $k \log m \le 2^{\sqrt{\log k \log\log m}}$ and the first term is at most $g \log^2 m \cdot 2^{\sqrt{\log k \log\log m}}$; otherwise $k/\ell \le 2^{\sqrt{\log k \log\log m}} / \log m$ gives the same bound. The second term is $g \log^2 m$ times $\log^2 m \cdot 2^{O(\sqrt{\log \ell \ \log\log m})}$, and since $\ell \le k$, the polylogarithmic factor is absorbed as noted above. Finally, at the optimal penalty a minimum-weight $(k+1)$-link path is assembled by one path swap [12] in $O(m)$ time, and Lemma 9 converts it into k facilities. ◀

▶ **Corollary 12.** *Unimodal-Cost k-Median on n piecewise-linear unimodal functions with m breakpoints can be solved in time*

$$O\Big((m + n \log n) \log m \cdot \min\big\{k,\ \log m \sqrt{k \log m},\ \log m \cdot 2^{O(\sqrt{\log k \ \log\log m})}\big\}\Big).$$

Proof. Take the best of Theorem 7, Theorem 10, and Theorem 11. ◀

References

1. Alok Aggarwal, Maria M. Klawe, Shlomo Moran, Peter Shor, and Robert Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(1–4):195–208, 1987. doi:10.1007/BF01840359.
2. Alok Aggarwal, Baruch Schieber, and Takeshi Tokuyama. Finding a minimum-weight k-link path in graphs with the concave Monge property and applications. *Discrete & Computational Geometry*, 12(3):263–280, 1994. doi:10.1007/BF02574380.
3. Vincenzo Auletta, Domenico Parente, and Giuseppe Persiano. Placing resources on a growing line. *Journal of Algorithms*, 26(1):87–100, 1998.
4. Danny Z. Chen and Haitao Wang. New algorithms for facility location problems on the real line. *Algorithmica*, 69(2):370–383, 2014. doi:10.1007/s00453-012-9737-0.
5. Mark S. Daskin. *Network and Discrete Location: Models, Algorithms, and Applications*. Wiley, 2 edition, 2013. doi:10.1002/9781118537015.
6. Zvi Drezner and George O. Wesolowsky. The asymmetric distance location problem. *Transportation Science*, 23(3):201–207, 1989. doi:10.1287/trsc.23.3.201.
7. Refael Hassin and Arie Tamir. Improved complexity bounds for location problems on the real line. *Operations Research Letters*, 10(7):395–402, 1991. doi:10.1016/0167-6377(91)90041-M.

8 Laura E. Jackson, George N. Rouskas, and Matthias F. M. Stallmann. The directional p-median problem: Definition, complexity, and algorithms. *European Journal of Operational Research*, 179(3):1097–1108, 2007. doi:10.1016/j.ejor.2005.06.080.

9 Guo-Hui Lin and Guoliang Xue. K-center and k-median problems in graded distances. *Theoretical Computer Science*, 207(1):181–192, 1998. doi:10.1016/S0304-3975(98)00063-2.

10 Robert F. Love. Note – One-dimensional facility location-allocation using dynamic programming. *Management Science*, 22(5):614–617, 1976. doi:10.1287/mnsc.22.5.614.

11 Frank Plastria. Asymmetric distances, semidirected networks and majority in Fermat–Weber problems. *Annals of Operations Research*, 167(1):121–155, 2009. doi:10.1007/s10479-008-0351-0.

12 Baruch Schieber. Computing a minimum weight k-link path in graphs with the concave monge property. *Journal of Algorithms*, 29(2):204–222, 1998. doi:10.1006/jagm.1998.0955.

13 Haitao Wang and Jingru Zhang. Line-constrained k-median, k-means, and k-center problems in the plane. *International Journal of Computational Geometry & Applications*, 26(03n04):185–210, 2016. doi:10.1142/S0218195916600049.

14 Gerhard J. Woeginger. Monge strikes again: Optimal placement of web proxies in the internet. *Operations Research Letters*, 27(3):93–96, 2000. doi:10.1016/S0167-6377(00)00041-9.

15 Peng Yu, Yike Chen, Chao Xu, Albert Bifet, and Jesse Read. Binary split categorical feature with mean absolute error criteria in CART. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(33):27961–27968, 2026. doi:10.1609/AAAI.V40I33.40020.