Speeding up Dynamic Programming in the Line-Constrained k-median

Paweł Gawrychowski[1] and Łukasz Zatorski[2(✉)]

[1] University of Haifa, Haifa, Israel
[2] Institute of Computer Science, University of Wrocław, Wroclaw, Poland
lzatorski@gmail.com

Abstract. In the planar k-median problem we are given a set of demand points and want to open up to k facilities as to minimize the sum of the transportation costs from each demand point to its nearest facility. In the line-constrained version the medians are required to lie on a given line. We present a new dynamic programming formulation for this problem, based on constructing a weighted DAG over a set of median candidates. We prove that, for any convex distance metric and any line, this DAG satisfies the concave Monge property. This allows us to construct efficient algorithms in L_∞ and L_1 and any line, while the previously known solution (Wang and Zhang, ISAAC 2014) works only for vertical lines. We also provide an asymptotically optimal $\mathcal{O}(n)$ solution for the case of $k = 1$.

Keywords: k-median · Dynamic programming · Monge property

1 Introduction

The planar k-median problem is a variation of the well-known facility location problem. For a given set P of *demand points*, we want to find a set Q of k *facilities*, such that the sum of all transportation costs from a demand point to its closest facility is minimized. Each $p \in P$ is associated with its own (positive) cost per unit of distance to assigned facility, denoted $w(p)$. Formally, we want to minimize:

$$S(P) = \sum_{p \in P} \min_{q \in Q} w(p) \cdot d(p, q)$$

Because the problem is NP-hard for many metrics [7], we further restrict it by introducing a *line-constraint* on the set Q. We require that all facilities should belong to a specified *facility line* χ defined by an equation $ax + by = c$, where $a, b, c \in \mathbb{R}$ and $a \cdot b \neq 0$. Such a constraint is natural when all facilities are by design placed along a path that can be locally treated as linear, e.g., pipeline, railroad, highway, country border, river, longitude or latitude.

For $k = 1$ we obtain the line-constrained 1-median problem. Despite the additional restriction, the complexity of this simplest variant strongly depends

V. Mäkinen et al. (Eds.): IWOCA 2016, LNCS 9843, pp. 293–305, 2016.
DOI: 10.1007/978-3-319-44543-4_23

on the metric. For a point $p \in \mathbb{R}^2$, let $x(p)$ and $y(p)$ denote its x- and y-coordinate. The most natural metric is the Euclidean distance, where $L_2(p,q) = \sqrt{(x(p)-x(q))^2 + (y(p)-y(q))^2}$. It is known that even for 5 points, it is not possible to construct the 1-median with a ruler and compass. It can also be proven that the general, line-constrained and 3-dimension versions of the k-median problem are not solvable over the field of rationals [2]. Hence it is natural to consider also other distance functions, for example:

Chebyshev distance $L_\infty(p,q) = \max\{|x(p)-x(q)|, |y(p)-y(q)|\}$,
Manhattan distance $L_1(p,q) = |x(p)-x(q)| + |y(p)-y(q)|$,
squared Euclidean distance $L_2^2(p,q) = (x(p)-x(q))^2 + (y(p)-y(q))^2$.[1]

All these distances functions have been recently considered by Wang and Zhang [10] in the context of line-constrained k-median problem. They designed efficient algorithms based on a reduction to the minimum weight k-link path problem. However, their L_1 and L_∞ solutions work only in the special case of a horizontal facility line.

We provide a different dynamic programming formulation of the problem that works for any facility line χ in L_1 and L_∞. The new formulation can also be seen as a minimum weight k-link path in a DAG, where the weights are Monge. However, looking up the weight of an edge in this DAG is more expensive. We show how to implement edge lookups in $\mathcal{O}(\log n)$ after $\mathcal{O}(n \log n)$ time and space preprocessing which then allows us to apply the SMAWK algorithm [1] or, if $k = \Omega(\log n)$, the algorithm of Schieber [9] to obtain the following complexities.

Metric	Facility line	Time complexity
Wang and Zhang [10]		
L_1	horizontal	$\min\{\mathcal{O}(nk), n2^{\mathcal{O}(\sqrt{\log k \log\log n})} \log n\}$
L_∞	horizontal	$\min\{\mathcal{O}(nk \log n), n2^{\mathcal{O}(\sqrt{\log k \log\log n})} \log^2 n\}$
Our results		
L_1	general	$\min\{\mathcal{O}(nk \log n), n2^{\mathcal{O}(\sqrt{\log k \log\log n})} \log n\}$
L_∞	general	$\min\{\mathcal{O}(nk \log n), n2^{\mathcal{O}(\sqrt{\log k \log\log n})} \log n\}$

In L_∞, our general solution is faster than the one given by Wang and Zhang for the special case of horizontal facility line. We also provide a specialized procedure solving the problem for $k = 1$ in linear time.

2 Preliminaries

A basic tool for speeding up dynamic programming is the so-called Monge property. It can often be used to improve the time complexity by an order of magnitude, especially in geometric problems.

[1] This is not a metric.

Definition 1. *A weight function w is concave Monge if, for all $a < b$ and $c < d$, $w(a, c) + w(b, d) \leq w(b, c) + w(a, d)$.*

Dynamic programming can often be visualized as finding the row minima in a $n \times n$ matrix. Naively, this takes $\mathcal{O}(n^2)$ time. However, Aggarwal et al. [1] showed how to decrease the time complexity to $\mathcal{O}(n)$ if the matrix has the so-called total monotonicity property, which is often established through the Monge property. Their method is usually referred to as the SMAWK algorithm. There is a deep connection between SMAWK and other methods for speeding up dynamic programming, such as the Knuth-Yao inequality used for building optimal binary search trees, as observed by Bein et al. [3].

Let D be a DAG on n nodes $0, 1, \ldots, n-1$ with a concave Monge weight function $w(i, j)$ defined for $0 \leq i < j < n$ that corresponds to the weight of the edge $\langle i, j \rangle$. A minimum diameter path in D is a path from 0 to $n-1$ with the minimum weight. Galil and Park showed how to find such path in optimal $\mathcal{O}(n)$ time using the SMAWK algorithm [5]. A minimum weight k-link path is a minimum weight path from 0 to $n-1$ consisting of exactly k edges (links).

Lemma 2. *Minimum weight k-link path can be found in $\mathcal{O}(nk)$ and, for $k = \Omega(\log n)$, $n2^{\mathcal{O}(\sqrt{\log k \log\log n})}$ time.*

Proof. To obtain $\mathcal{O}(nk)$ time complexity, we iteratively compute minimum weight 1-link, 2-link, ..., $(k-1)$-link and finally k-link paths from 0 to every other node. This can be seen as k layers of dynamic programming, each requiring only $\mathcal{O}(n)$ time thanks to the SMAWK algorithm. Alternatively, $n2^{\mathcal{O}(\sqrt{\log k \log\log n})}$ time algorithm for $k = \Omega(\log n)$ was given by Schieber [9]. □

The weights of the edges in our DAG will be computed on-the-fly with orthogonal queries. We will use the following tool: preprocess a given set of n weighted points in a plane for computing the sum of the weights of all points in a given query range $[x, +\infty] \times [y, +\infty]$. We call this problem orthogonal range sum. The following is well-known.

Lemma 3. *There exists a data structure for the orthogonal range sum problem that can be built in $\mathcal{O}(n \log n)$ time and answers any query in $\mathcal{O}(\log n)$ time.*

Proof. We convert the points into a sequence by sorting them according to their x-coordinates (without losing the generality, these coordinates are all distinct) and writing down the corresponding y-coordinates. The y-coordinates are further normalized by replacing with the ranks on a sorted list of all y-coordinates (again, we assume that they are all distinct). Hence we obtain a sequence of length n over an alphabet $[n]$, where each character has its associated weight. We build a wavelet tree [6] of this sequence in $\mathcal{O}(n \log n)$ time. Each node of the wavelet tree is augmented with an array of partial sums of the prefixes of its subsequence. Given an orthogonal query, we first normalize it by looking at the sorted list of all x- and y-coordinates. Then we traverse the wavelet tree starting from the root and accumulate appropriate partial sums. The details can be found in [8]. □

3 Normalizing Problem Instances

L_1 and L_∞ metrics are equivalent, which can be seen by rotating the plane by 45°. Hence from now on we will work in L_1 metric. This simplification was not possible in the previous approach [10], since it required the facility line to be horizontal, which is no longer true after rotation.

We further modify the problem instance so that the line χ is expressed in a slope intercept form $y = ax$, where $a \in [0, 1]$, and all coordinates of points in P are non-negative. This is always possible by reflecting along the horizontal axis, then along the line $y = x$, and finally translating. Such transformations do not modify the distances in L_1, so computing the k-median solution Q for the transformed instance gives us the answer for the original instance. Because any solution Q can be transformed so that the x-coordinates of all facilities are distinct without increasing the cost, we will consider only such solutions and identify each facility with its x-coordinate.

4 Computing 1-median

Let $D(p, x)$ be the weighted distance between $p \in P$ and $(x, a \cdot x) \in \chi$:

$$D(p, x) = w(p) \cdot d(p, (x, a \cdot x))$$

Whenever we say that $p \in P$ is closer to coordinate x_i than x_j, we mean that $D(p, x_i) < D(p, x_j)$. For a set of points $A \subseteq P$, $D(A, x)$ is the sum of weighted distances:

$$D(A, x) = \sum_{p \in A} w(p) \cdot d(p, (x, a \cdot x))$$

The 1-median is simply $\min_{x \in \mathbb{R}} D(P, x)$.

A function $f : \mathbb{R} \to \mathbb{R}$ is convex if the line segment between any two points on its graph lies above or on the graph. Such functions have the following properties:

1. $f(x) = |x - y|$ is convex for any y.
2. if $f(x)$ is convex, then $g(x) = c \cdot f(x)$ is convex for any positive c.
3. if $f(x)$ and $g(x)$ are convex, then $h(x) = f(x) + g(x)$ is also convex.

Lemma 4. *For any point p, $D(p, x)$ is convex. For any set of points P, $D(P, x)$ is also convex.*

Proof. Consider any point $p \in P$. From the definition:

$$D(p, x) = w(p) \cdot L_1(p, (x, a \cdot x)) = w(p) \cdot (|x(p) - x| + |y(p) - a \cdot x|) .$$

This is a sum of absolute values functions multiplied by the (positive) weight of p. Hence by the properties of convex functions $D(p, x)$ is convex. Then $D(P, x)$ is also convex since it is a sum of convex functions over $p \in P$. □

Since $D(p, x)$ is convex, any of its local minima is a global minimum. Similarly to the function $f(x) = |x|$, it is only semi-differentiable. Its derivative $D'(p, x)$ is a staircase nondecreasing function, undefined for at most two values $x = x_1$ and $x = x_2$. We call x_1 and x_2 the *median candidates* and for convenience assume that $D'(p, x)$ is equal to its right derivative there. When $a = 0$ or $p \in \chi$, $D'(p, x)$ has exactly one median candidate $x_1 = x(p)$, that is the minimum. Otherwise, there are two median candidates $x_1 = x(p)$ and $x_2 = \frac{y(p)}{a}$. For $a \in (0, 1)$, x_1 is the only minimum, whereas for $a = 1$ every value in range $[x_1, x_2]$ is a minimum. Because the derivative of a sum of functions is the sum of their derivatives, $D'(P, x)$ can only change at a median candidate of some $p \in P$. This means that a minimum of $D(p, x)$ corresponds to one of at most $2n$ median candidates of P. In other words, there exists a solution $(x, y) \in \chi$, such that $x = x(p)$ or $y = y(p)$ for some $p \in P$. From now on, we use $\mathcal{M}(P)$ to denote the set of median candidates of P. $\mathcal{M}(P)$ can be computed in $\mathcal{O}(n)$ time by simply iterating over $p \in P$ and adding $x = x(p)$ and $x = \frac{y(p)}{a}$ to the result (note that this might give us a multiset, i.e., some median candidates might be included multiple times).

Theorem 5. *We can solve line-constrained* 1*-median problem in* $\mathcal{O}(n)$ *time.*

Proof. Because $D'(p, x)$ is nondecreasing, we can binary search for the largest x such that $D'(p, x) \leq 0$. Then we return x as the solution. In every step of the binary search we use the median selection algorithm [4] to narrow down the current search range $X = (x_{\text{left}}, x_{\text{right}})$. At the beginning of every step:

1. M is a multiset of all median candidates of P that are in X.
2. S contains all points from P with at least one median candidate in M.
3. $r = D'(P \setminus S, x)$ for some $x \in X$.

We select the median x_m of M and compute $D'(P, x_m)$. If $D'(p, x_m) > 0$, we continue the search in (x_{left}, x_m), and otherwise in (x_m, x_{right}), updating S and M accordingly. Eventually $x_{\text{left}} = x_{\text{right}}$ and we return x_{left}.

The key observation is that when a point p is removed from S, it does no longer have a median candidate within X and its $D'(p, x)$ remains constant in all further computations. This means that $D'(P \setminus S, x)$ is constant for all $x \in X$ and r can be updated after removing every point p from S in $\mathcal{O}(1)$ time. x_m can be found in $\mathcal{O}(|M|)$ time. Calculating $D'(P, x_m) = r + D'(S, x_m)$ then takes $\mathcal{O}(1 + |S|)$ time. For a point p to be in S, one of its median candidates must belong to M, so $|S| \leq |M|$. Hence the complexity of a single iteration is $\mathcal{O}(|M|)$. After each iteration the size of M decreases by a factor of two, so the running time is described by $T(n) = \mathcal{O}(n) + T(n/2)$, which solves to $\mathcal{O}(n)$. □

Theorem 6. *We can calculate* $D(P, x)$ *for every* $x \in \mathcal{M}(P)$ *in* $\mathcal{O}(n \log n)$ *time.*

Proof. The elements of $\mathcal{M}(P)$ can be sorted in $\mathcal{O}(n \log n)$ time, and we can assume that every point generates exactly two median candidates. Let $\mathcal{M}(P) = \{x_1, x_2, \ldots, x_{2n}\}$, where $x_i \leq x_{i+1}$ for all $i = 1, 2, \ldots, 2n - 1$. Recall that $D'(P, x) = D'(P, x_i)$ for any $x \in (x_i, x_{i+1})$. We compute $D(p, x_1)$ together with

$D'(P, x_1)$ in $\mathcal{O}(n)$ time. Then all other $D(p, x_i)$ are computed sequentially for $i = 2, 3, \ldots, 2n$ in $\mathcal{O}(1)$ time each using the formula:

$$D(P, x_i) = D(P, x_{i-1}) + D'(P, x_{i-1}) \cdot (x_i - x_{i-1})$$
$$D'(P, x_i) = D'(P, x_{i-1}) + 2 \cdot w(p) \cdot \sigma$$

where x_i is generated by the point p, $\sigma = 1$ if $x_i = x(p)$ and $\sigma = a$ otherwise. □

5 Computing *k*-median

Consider now any optimal solution Q of the k-median problem for the given set of weighted points P. For any facility $q \in Q$, let P_q be the set of points of P assigned to q. By interchanging the order of the summation, Q should minimize

$$\sum_{q \in Q} \sum_{p \in P_q} w(p) \cdot d(p, q).$$

Hence q must be an optimal solution of the 1-median problem for P_q. Since replacing q will not increase the sum of distances of points in $P \setminus P_q$, q can be chosen to be a median candidate of P_q. We deduce that there exists an optimal solution Q' such that

$$\bigwedge_{q \in Q'} q \in \mathcal{M}(P_q) \subseteq \mathcal{M}(P).$$

For $k \geq \min(n, |\mathcal{M}(P)|)$, every $p \in P$ can be assigned to its closest possible facility. Such an assignment can be easily computed in $\mathcal{O}(n)$ time. If we are required to return exactly k medians, then we add enough additional points to $\mathcal{M}(P)$. From now on, we assume that $k < \min(n, |\mathcal{M}(P)|)$. Thus there exists an optimal k-median solution, where all facilities are 1-median candidates of P.

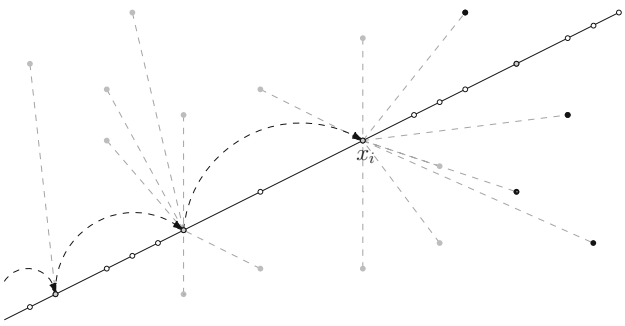

Fig. 1. Path in the DAG ending at the candidate x_i. Dashed lines represent current assignment of points from P to the closest chosen facility.

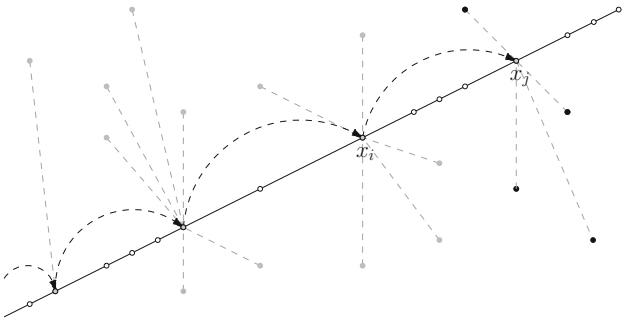

Fig. 2. We follow the edge $\langle i, j\rangle$. All (black) points now assigned to x_j were previously assigned to x_i, see Fig. 1.

By arranging all median candidates in a sequence according to their x-coordinates, we can view choosing k facilities as selecting a $(k+1)$-link path in a DAG between two artificial elements infinitely to the left and to the right of the sequence, called source and sink, respectively.

Imagine that we traverse the sequence from left to right while deciding if we should open a new facility at the current median candidate, see Fig. 1. Initially, all points are assigned to the artificial facility source and the cost of the current solution S is set to $+\infty$. If we decide to open a new facility at the current median candidate x_j, for every $p \in P$ we check if x_j is closer to p than the facility p is currently assigned to. If so, we reassign p to x_j, see Fig. 2.

We claim that $p \in P$ can be closer to x_j than the facility p is currently assigned to only if the currently assigned facility is the most recently chosen facility x_i, that is, the current solution does not contain any facilities between x_i and x_j. Assuming that the claim holds, we define the weight of an edge $\langle \mathsf{source}, i\rangle$ to be $D(P, x_i)$, and the weight of an internal edge $\langle i, j\rangle$ to be total decrease of the cost after giving each point $p \in P$ the possibility to switch from x_i to x_j. Finally, the weight of an edge $\langle j, \mathsf{sink}\rangle$ is 0. Then selecting k medians corresponds to selecting an $(k+1)$-link from source to sink in the DAG. However, we need to show the claim. To this end we consider the following properties of convex functions:

Proposition 7. *For any convex function f and $a < b < c$:*

(a) If $f(c) < f(b)$ then $f(b) < f(a)$.
(b) If $f(c) < f(a)$ then $f(b) < f(a)$.

Proof. Assume otherwise for any of the two implications. This means that $f(a) \leq f(b) > f(c)$ and the segment AC where $A = (a, f(a))$ and $C = (c, f(c))$ lies below $f(b)$, contradicting the assumption that the function f is convex. □

Now we can prove the claim. Consider a point $p \in P$ such that its currently assigned facility is x_i and, for some $k > i$, facility x_k was not selected as a better option. Then, for any $j > k$, facility x_j cannot be a better option either, because

$x_i < x_k < x_j$ so by Proposition 7(a) $D(p, x_i) \leq D(p, x_k)$ implies $D(p, x_j) \geq D(p, x_k)$. This means that if x_i was the most recently opened facility and x_j is the current median candidate, opening a new facility at x_j changes the total cost by

$$\sum_{p \in P} \min(D(p, x_j) - D(p, x_i), 0).$$

Definition 8. *Let $x_1, x_2, \ldots, x_{n-1}, x_{2n}$ be the sorted sequence of median candidates of P. We define its median DAG over nodes $0, 1, \ldots, 2n, 2n+1$ with edge weight function $w(i, j)$ as follows:*

$$w(i,j) = \begin{cases} \infty & \text{if } i = 0 \text{ and } j = 2n+1, \\ 0 & \text{if } i > 0 \text{ and } j = 2n+1, \\ D(P, x_j) & \text{if } i = 0 \text{ and } j \in \{1, 2, \ldots, 2n\}, \\ \sum\limits_{p \in P} \min(D(p, x_j) - D(p, x_i)), 0) & \text{otherwise.} \end{cases}$$

The total cost of any k-median solution is equal to the sum of weights on its corresponding path of length $k+1$ between 0 and $2n+1$, so finding k-median reduces to finding the minimum weight $(k+1)$-link path in the median DAG.

Because a sum of Monge functions is also Monge, to prove that $w(i, j)$ is Monge we argue that $w_p(i, j)$ is Monge, where $w(i, j) = \sum\limits_{p \in P} w_p(i, j)$ and:

$$w_p(i,j) = \begin{cases} \infty & \text{if } i = 0 \text{ and } j = 2n+1, \\ 0 & \text{if } i > 0 \text{ and } j = 2n+1, \\ D(p, x_j) & \text{if } i = 0 \text{ and } j \in \{1, 2, \ldots, 2n\}, \\ \min(D(p, x_j) - D(p, x_i)), 0) & \text{otherwise.} \end{cases}$$

Proposition 9. *For any convex function f, if $a < b < c$ then:*

$$\min(f(c) - f(a), 0) \leq \min(f(c) - f(b), 0).$$

Proof. If $f(c) \geq f(b)$ then the right side of the equation is equal to 0 and left side is non-positive. If $f(c) < f(b)$ then by Proposition 7(a) also $f(b) < f(a)$, so

$$\min(f(c) - f(a), 0) \leq f(c) - f(a) < f(c) - f(b) = \min(f(c) - f(b), 0)$$

so the claim holds. □

Proposition 10. *For any convex function f, if $a < b < c$ then*

$$f(b) + \min(f(c) - f(a), 0) \leq f(c) + \min(f(b) - f(a), 0).$$

Proof. If $f(b) \geq f(a)$ then by Proposition 7(a) also $f(c) \geq f(b)$. Hence also $f(c) \geq f(a)$ and

$$f(b) + \min(f(c) - f(a), 0) = f(b) \leq f(c) = f(c) + \min(f(b) - f(a), 0)$$

so the property holds. Otherwise, $f(b) < f(a)$ and the property becomes

$$f(b) + \min(f(c) - f(a), 0) \leq f(c) + f(b) - f(a)$$

which is always true due to $\min(f(c) - f(a), 0) \leq f(c) - f(a)$. □

Proposition 11. *For any convex function f, if $a < b < c < d$ then*

$$\min(f(c)-f(a),0)+\min(f(d)-f(b),0) \leq \min(f(d)-f(a),0)+\min(f(c)-f(b),0).$$

Proof. If $f(d) \geq f(a)$, then

$$\min(f(d) - f(b), 0) \leq 0 = \min(f(d) - f(a), 0).$$

Combined with Proposition 9 applied to $a < b < c$ we obtain the claim. Otherwise, $f(d) < f(a)$ and by Proposition 7(b) applied to $a < c < d$ also $f(c) < f(a)$, so the property becomes

$$f(c) + \min(f(d) - f(b), 0) \leq f(d) + \min(f(c) - f(b), 0)$$

which holds by Proposition 10 applied to $b < c < d$. □

Theorem 12. *For any point p, $w_p(i, j)$ is concave Monge.*

Proof. Consider any $s, t, u, v \in [0, 2n + 1]$ such that $s < t < u < v$. We need to prove that for any $p \in P$:

$$w_p(s, u) + w_p(t, v) \leq w_p(s, v) + w_p(t, u).$$

Case 1. $s = 0$ and $v = 2n + 1$
Straightforward, since $w_p(s, v) = \infty$ and all other edges have finite weights.
Case 2. $s > 0$ and $v = 2n + 1$

$$\begin{aligned} w_p(s, u) + w_p(t, v) &= w_p(s, u) + 0 \\ &= \min(D(p, u) - D(p, s), 0) \\ &\overset{9}{\leq} \min(D(p, u) - D(p, t), 0) \\ &= 0 + w_p(t, u) \\ &= w_p(s, v) + w_p(t, u) \end{aligned}$$

Case 3. $s = 0$ and $v < 2n + 1$

$$\begin{aligned} w_p(s, u) + w_p(t, v) &= D(p, u) + \min(D(p, v) - D(p, t), 0) \\ &\overset{10}{\leq} D(p, v) + \min(D(p, u) - D(p, t), 0) \\ &= w_p(s, v) + w_p(t, u) \end{aligned}$$

Case 4. $s > 0$ and $v < 2n + 1$

$$\begin{aligned} w_p(s,u) + w_p(t,v) &= \min(D(p,u) - D(p,s), 0) + \min(D(p,v) - D(p,t), 0) \\ &\overset{11}{\leq} \min(D(p,v) - D(p,s), 0) + \min(D(p,u) - D(p,t), 0) \\ &= w_p(s,v) + w_p(t,u) \end{aligned}$$

So in all cases $w_p(s,u) + w_p(t,v) \leq w_p(s,v) + w_p(t,u)$ and hence $w_p(i,j)$ is concave Monge. □

In order to apply the known algorithms for finding minimum weight k-link path in the k-median problem, we need to answer queries for $w(i,j)$.

Lemma 13. *After $\mathcal{O}(n \log n)$ time and space preprocessing, we can answer queries for $w(i,j)$ in $\mathcal{O}(\log n)$ time per query.*

Proof. All edges from the source can be computed in $\mathcal{O}(n \log n)$ time via Theorem 6. All edges to sink have zero weight. It remains to show how to calculate the weight of an internal edge $\langle i,j \rangle$. Consider the set of points $p \in P$ that are closer to x_j than to x_i:

$$V(i,j) = \{(x,y) \in P : |x - x_i| + |y - a \cdot x_i| > |x - x_j| + |y - a \cdot x_j|\}$$

By definition, $w(i,j) = D(V(i,j), x_j) - D(V(i,j), x_i)$. We describe how to compute $D(V(i,j), x_i)$. $D(V(i,j), x_j)$ can be computed using the formula:

$$D(V(i,j), x_j) = D(P, x_j) - D(P \setminus V(i,j), x_j)$$

where $D(P, x_j)$ is the already preprocessed weight of the edge $\langle \mathsf{source}, j \rangle$, and $D(P \setminus V(i,j), x_j)$ can be calculated by rotating the plane by 180° and using the same method as the one described below.

First we argue that if $(x,y) \in V(i,j)$ then $x > x_i$. Otherwise

$$|y - a \cdot x_i| - |y - a \cdot x_j| > x_j - x_i \geq a \cdot x_j - a \cdot x_i \geq 0$$

and we obtain a contradiction in each of the three cases:

1. $y < a \cdot x_i$ then the inequality becomes $a \cdot x_i - a \cdot x_j > 0$ but $x_i < x_j$.
2. $y \in [a \cdot x_i, a \cdot x_j)$ then the inequality becomes $2y > 2a \cdot x_j$ but $y < a \cdot x_j$.
3. $y > a \cdot x_j$ then the inequality becomes $a \cdot x_j - a \cdot x_i > a \cdot x_j - a \cdot x_i$.

We partition $V(i,j)$ into $V_1(i,j)$ and $V_2(i,j)$ with a horizontal line $y = a \cdot x_i$:

$$\begin{aligned} V_1(i,j) &= V(i,j) \cap \{(x,y) : y \geq a \cdot x_i\} \\ V_2(i,j) &= V(i,j) \cap \{(x,y) : y < a \cdot x_i\}. \end{aligned}$$

The median candidate $(x_i, a \cdot x_i)$ is on the left and bottom of all points in $V_1(i,j)$ and on the left and top of all points in $V_2(i,j)$. Consider the minimum area rectangle enclosing P with sides parallel to the coordinate axes, and enumerate

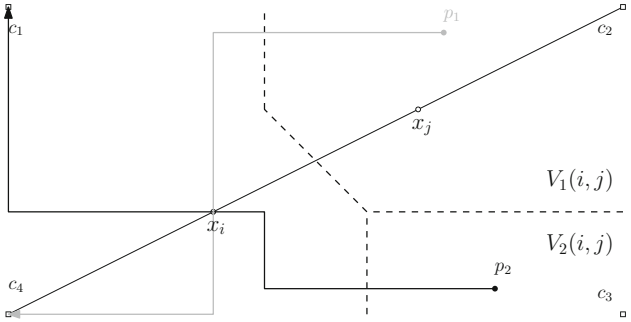

Fig. 3. Shortest route in L_1 from p_2 to c_1 and from p_1 to c_4 passing through the median candidate x_i.

its corners clockwise starting from the top left as c_1, c_2, c_3, c_4. In L_1 metric, one of the shortest routes from any point in $V_1(i,j)$ to the bottom left corner point c_4 goes via x_i, see Fig. 3. Therefore our desired sum of distances to x_i can be described in respect to c_4 as:

$$D(V_1(i,j), x_i) = \sum_{p \in V_1(i,j)} w(p) \cdot d(p, (x_i, a \cdot x_i))$$
$$= \left(\sum_{p \in V_1(i,j)} w(p) \cdot d(p, c_4) \right) - \left(d(c_4, (x_i, a \cdot x_i)) \cdot \sum_{p \in V_1(i,j)} w(p) \right).$$

Similarly, one of the shortest routes from any point in $V_2(i,j)$ to c_1 goes via x_i:

$$D(V_2(i,j), x_i) = \left(\sum_{p \in V_2(i,j)} w(p) \cdot d(p, c_1) \right) - \left(d(c_1, (x_i, a \cdot x_i)) \cdot \sum_{p \in V_2(i,j)} w(p) \right).$$

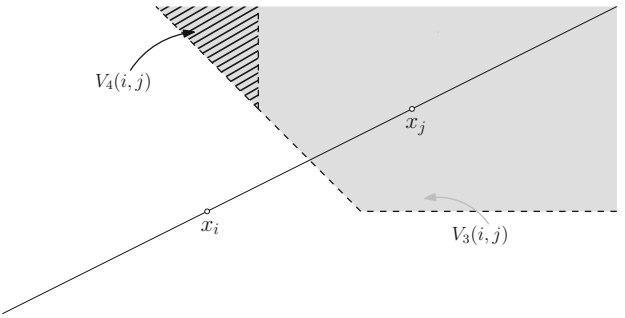

Fig. 4. V_1 represented as the gray V_3 minus the striped V_4.

The distances $d(c_1, (x_i, a \cdot x_i))$ and $d(c_4, (x_i, a \cdot x_i))$ can be computed in $\mathcal{O}(1)$ time. The expressions $\sum_{p \in V_2(i,j)} w(p) \cdot d(p, c_1)$ and $\sum_{p \in V_2(i,j)} w(p)$ can be evaluated in $\mathcal{O}(\log n)$ with orthogonal queries. To calculate $\sum_{p \in V_1(i,j)} w(p) \cdot d(p, c_4)$ and $\sum_{p \in V_1(i,j)} w(p)$, we represent $V_1(i,j)$ as $V_3(i,j) \setminus V_4(i,j)$, see Fig. 4 where $\delta_x = x_j - x_i$, $\delta_y = a(x_j - x_i)$ and

$$V_3(i,j) = \left\{ (x,y) \in P : y > ax_i \wedge \left(x + y > \frac{(\delta_x + \delta_y)}{2} \right) \right\}$$
$$V_4(i,j) = \left\{ (x,y) \in P : x \leq x_i + \delta_x - \delta_y \wedge \left(x + y > \frac{(\delta_x + \delta_y)}{2} \right) \right\}.$$

Now each of $V_2(i,j)$, $V_3(i,j)$ and $V_4(i,j)$ is defined by an intersection of two halfplanes. By transforming every point $p \in P$ into $(x(p)+y(p), y(p))$ for $V_3(i,j)$ and into $(x(p)+y(p), x(p))$ for $V_4(i,j)$, we can assume that the lines defining the halfplanes are parallel to the coordinate axes. Hence each sum can be calculated with orthogonal queries in $\mathcal{O}(\log n)$ time and $\mathcal{O}(n \log n)$ time and space preprocessing by Lemma 3. □

We reduced the line-constrained k-median problem in L_1 to the minimum k-link path problem. The weight of any edge can be retrieved in $\mathcal{O}(\log n)$ time by decomposing it into a constant number of orthogonal queries. By plugging in an appropriate algorithm for the minimum k-link path problem, we obtain the final theorem.

Theorem 14. *We can solve the line-constrained k-median problem in L_1 and L_∞ using $\mathcal{O}(kn \log n)$ time or, if $k = \Omega(\log n)$, $n 2^{\mathcal{O}(\sqrt{\log k \log \log n})} \log n$ time.*

References

1. Aggarwal, A., Klawe, M.M., Moran, S., Shor, P., Wilber, R.: Geometric applications of a matrix-searching algorithm. Algorithmica **2**(1–4), 195–208 (1987)
2. Bajaj, C.: The algebraic degree of geometric optimization problems. Discrete & Computational Geometry **3**(1), 177–191 (1988)
3. Bein, W., Golin, M.J., Larmore, L.L., Zhang, Y.: The Knuth-Yao quadrangle-inequality speedup is a consequence of total monotonicity. ACM Transactions on Algorithms (TALG) **6**(1), 17 (2009)
4. Blum, M., Floyd, R.W., Pratt, V.R., Rivest, R.L., Tarjan, R.E.: Time bounds for selection. J. Comput. Syst. Sci. **7**(4), 448–461 (1973)
5. Galil, Z., Park, K.: A linear-time algorithm for concave one-dimensional dynamic programming. Information Processing Letters **33**(6), 309–311 (1990)
6. Grossi, R., Gupta, A., Vitter, J.S.: High-order entropy-compressed text indexes. In: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms. pp. 841–850. Society for Industrial and Applied Mathematics (2003)
7. Megiddo, N., Supowit, K.J.: On the complexity of some common geometric location problems. SIAM journal on computing **13**(1), 182–196 (1984)

8. Navarro, G., Russo, L.M.S.: Space-Efficient Data-Analysis Queries on Grids. In: Asano, T., Nakano, S., Okamoto, Y., Watanabe, O. (eds.) ISAAC 2011. LNCS, vol. 7074, pp. 323–332. Springer, Heidelberg (2011)
9. Schieber, B.: Computing a minimum weight k-link path in graphs with the concave Monge property. Journal of Algorithms **29**(2), 204–222 (1998)
10. Wang, H., Zhang, J.: Line-Constrained k-Median, k-Means, and k-Center Problems in the Plane. In: Ahn, H.-K., Shin, C.-S. (eds.) ISAAC 2014. LNCS, vol. 8889, pp. 3–14. Springer, Heidelberg (2014)