Improved complexity results for the robust mean absolute deviation problem on networks with linear vertex weights

Igor Averbakh$^{1,2}$, Oded Berman$^2$, and Marina Leal$^{2,3}$

$^1$ Department of Management, University of Toronto Scarborough, 1265 Military Trail, Toronto, Ontario M1C 1A4, Canada;
$^2$ Rotman School of Management, University of Toronto, 105 St. George Street, Toronto, Ontario M5S3E6, Canada;
$^3$ Department of Statistics and Operations Research, Faculty of Mathematics, Campus Reina Mercedes, 41011, University of Seville, Seville, Spain.

emails: averbakh@utsc.utoronto.ca, berman@rotman.utoronto.ca, marinalealpalazon@gmail.com

August 5, 2016

Abstract. In a recent paper [10], an algorithmic approach was presented for the robust (minmax regret) absolute deviation location problem on networks with vertex weights which are linear functions of an uncertain or dynamically changing parameter. In this paper, we present algorithmic and analytical improvements that significantly reduce the computational complexity bounds for the algorithm.

Key words: minmax regret, robust location, facility location under uncertainty, network algorithms.
1 Introduction

Significant research efforts have been devoted to facility location problems with uncertainty in data. In the last two decades, much attention has been focused on the minmax regret approach [1, 8, 9], where it is required to find a solution with an objective function value reasonably close to the optimal one for all possible realizations of data. For location problems on networks, there may be two types of uncertain parameters: vertex weights and/or edge lengths (transportation times). The literature on minmax regret location problems on networks is primarily focused on problems with uncertain vertex weights that may represent the sizes of the corresponding population centers, or their demands. Many facility location decisions are strategic in nature and correspond to locating facilities that are supposed to function for a long time, so uncertainty in vertex weights may reflect the population dynamics, or potential changes in demand. Models with uncertain edge lengths have fewer practical applications. Also, minmax regret versions of classical single-facility location problems such as 1-center or 1-median are strongly NP-hard on general networks if uncertainty in edge lengths is present [2], in contrast to single-facility problems with uncertain vertex weights that are typically polynomially solvable (e.g., [3, 4]). Thus, models with uncertain edge lengths have less potential for advanced algorithmic analysis.

The most commonly used approaches to representing uncertain data in minmax regret optimization are the interval data model, where it is assumed that each uncertain parameter can take on any value within its pre-specified uncertainty interval regardless of the values taken by other parameters, and the discrete scenario model, where a finite set of possible realizations of the vector of uncertain parameters is explicitly given as a part of the input [1, 8, 9]. One of the less studied approaches is the model where uncertain/changing parameters (e.g., vertex weights) are assumed to be linearly related, i.e. to be linear functions of a single parameter (e.g., time).

The literature on minmax regret facility location largely focuses on minmax regret versions of classical location objectives such as the median or center objectives. In [10], the minmax regret version of a location problem with an equity-based objective was considered; such an objective is an important service measure in the public sector. The mean absolute deviation is the weighted sum of the absolute deviations of customer-facility distances from the mean distance between the facility and the customers, and is a measure of “inequality” between the customers that was extensively studied for problems without uncertainty or dynamic evolution.
of the parameters [5, 7, 11, 12]. Lopez-de-los-Mozos et al. [10] consider this objective in the context of dynamic evolution (or linearly-correlated uncertainty) of vertex weights which are assumed to be linear functions of a changing parameter (time, for instance), and study the minmax and minmax regret versions of the problem. For the minmax regret version, they develop a polynomial time algorithm, which, in spite of being polynomial, cannot be considered practically useful due to very high order of complexity obtained by the analysis in [10] (slightly higher than \(O(n^8 \log n)\) for trees and slightly higher than \(O(m^3 n^8 \log n)\) for general networks, where \(n\) and \(m\) are the numbers of vertices and edges, respectively). In this paper, we present improvements that reduce the complexity bounds to \(O(m^2 n^3 (\log \log n)^2 \log n)\) for general networks, \(O(n^4 (\log^* n)^2 \log n)\) for trees, and \(O(n^3 (\log^* n)^2 \log n)\) for paths, where \(\log^* n\) is the iterated logarithm which is a function that grows extremely slowly and can be considered as a constant for practical purposes.

2 Notation and the problem

Consider a network \(N(V, E)\) with vertex set \(V = \{v_1, ..., v_n\}\) and the set \(E\) of \(m\) rectifiable edges. We use the following notation (consistent with [10]):

- \(l_e\) is the length of the edge \(e \in E\);
- \(N\) denotes the set of all points of the network, or the network itself;
- \(d(x, y)\) is the shortest-path distance in the network between points \(x\) and \(y\);
- \(w_i(t) = \alpha_i t + \beta_i \geq 0\) is the weight of vertex \(v_i\) which is a linear function of a real-valued parameter \(t\) (time, for instance) that can take any value from an interval \([t^-, t^+]\), for some constants \(t^-, t^+\), \(t^- \leq t^+\);
- \(W(t) = \sum_{i=1}^{n} w_i(t) = At + B\), where \(A = \sum_{i=1}^{n} \alpha_i\), \(B = \sum_{i=1}^{n} \beta_i\), is the total weight, which is assumed to be strictly positive for any \(t \in [t^-, t^+]\);
- \(M(x, t) = \frac{1}{W(t)} \sum_{i=1}^{n} w_i(t)d(v_i, x)\) is the dynamic median function, for \(t \in [t^-, t^+]\) and \(x \in N\);
- \(F(x, t) = \frac{1}{W(t)} \sum_{i=1}^{n} w_i(t)|d(v_i, x) - M(x, t)|\) is the dynamic mean absolute deviation (MAD) function;
\[ F^*(t) = \min_{x \in N} F(x, t). \]

Following [10], the minmax regret mean absolute deviation location problem (MMR-MAD) is

\[
\min_{x \in N} \max_{t \in [t^- , t^+]} (F(x, t) - F^*(t)).
\]

We also use some standard notation and facts from computational geometry:

- \( \log^* n \) is the iterated logarithm function (the minimum number of times that the logarithm
  (base 2) operator should be applied to \( n \) to obtain a number not greater than 1). \( \log^* n \)
  is a very slowly growing function and can be considered almost constant for practical
  purposes [14];
- \( \lambda_s(n) \) is the maximum length of a Davenport-Schinzel sequence of order \( s \) on \( n \) symbols.
  It is well known that for a set of \( x \)-monotone Jordan arcs with at most \( s \) intersections
  between any pair of arcs, its lower envelope has an \( O(\lambda_{s+2}(n)) \) complexity, and can be
  computed in \( O(\lambda_{s+1}(n) \log n) \) time ([14], Theorem 6.5). For any \( s \), \( \lambda_s(n) = O(n \log^* n) \)
  [15];
- \( \alpha(n) \) is the inverse of the Ackermann function which grows very slowly, \( \alpha(n) \leq 4 \) for any
  “practical” values of \( n \) [13].

For any integer \( i, j, i \leq j \), let \( [i : j] \) denote the set \( \{i, i+1, \ldots, j\} \).

Lopez-de-los-Mozos et al. [10] present an algorithm for MMR-MAD and analyze its complexity. Their analysis obtains the complexity bound \( O(mn^3 \lambda_6(m^2n^5) \log^* n \log n) \) (even though stated as \( O(mn^3 \lambda_6(n^5) \log^* n \log n) \) in [10], as we will discuss in Subsection 3.4) for general
networks, and the bound \( O(n^3 \lambda_6(n^5) \log^* n \log n) \) for trees.

### 3 A compressed description of the algorithm from [10]

The algorithm from [10] for MMR-MAD is essentially the same for a tree and for a general
network. On a tree, the problem restricted to a single edge is considered, and then the best of
the optimal solutions for different edges is chosen. On a general network, the problem restricted
to a primary region where all distance functions are linear is considered, and then the best of
the optimal solutions for different primary regions is chosen. Following [10], we describe the
algorithm for a tree and then briefly comment on extending it to a general network. So, suppose
that $N(V, E)$ is a tree. Thus, $|E| = n - 1$.

3.1 The breakpoint curves and initial domain partition.

For an edge $e = [u, v]$ with endpoint vertices $u$ and $v$, let any real number $x_e \in [0, l_e]$ denote
the point of $e$ such that the subedge $[u, x_e]$ has length $x_e$. Functions $d(v_i, x_e), i \in [1 : n]$ are
linear on $e$. Function $F(x_e, t)$ in the interior of the domain $D_e = [0, l_e] \times [t^-, t^+]$ is differentiable
everywhere except for the points of the breakpoint curves, where a breakpoint curve $B_{e,i}$ is
the set of points of $D_e$ that satisfy $d(v_i, x_e) - M(x_e, t) = 0$ for some $i \in [1 : n]$. Since
d$(v_i, x_e) = d(v_i, u) + \delta_i x_e$ where $\delta_i \in \{-1, 1\}$, each breakpoint curve $B_{e,i}$ can be defined by an
equation of the form $f_{e,i}(x_e, t) = 0$ where $f_{e,i}(x_e, t) = W(t)(d(v_i, x_e) - M(x_e, t)) = a_{e,i} x_e t + b_{e,i} + c_{e,i} x + d_{e,i}$ for some $a_{e,i}, b_{e,i}, c_{e,i}, d_{e,i} \in \mathbb{R}$ [10], so the breakpoint curves are hyperbolas
or straight lines. It is mentioned in [10] that two different breakpoint curves can intersect
in at most two points. In fact, this statement can be strengthened: two different breakpoint
curves $B_{e,i}$ and $B_{e,j}$ can intersect in at most one point since at the intersection point $(x_e, t)$ the
equality $d(v_i, x_e) = d(v_j, x_e)$ must hold, but this difference is not essential for the analysis in
[10]. The $n$ breakpoint curves in the domain $D_e$ define a partition $P(D_e)$ of the domain into
$O(n^2)$ full-dimensional cells $R_j, j \in [1 : r_e]$; the function $F(x_e, t)$ is differentiable in the interior
of each cell of this partition. The set $V(P(D_e))$ of vertices of $P(D_e)$ has cardinality $O(n^2)$. Computing
the arrangement of the $n$ breakpoint curves can be done in $O(n^2 \alpha(n))$ time and $O(n^2)$ space [6].

3.2 Phase 1: Obtaining $F^*(t)$.

For fixed $t \in [t^-, t^+]$ and $e \in E$, function $F(x_e, t)$ is piece-wise linear and convex on $[0, l_e]$ as a
function of $x_e$, with breakpoints that belong to the breakpoint curves. The minimum of $F(x, t)$
over all $x \in N$ for any fixed $t \in [t^-, t^+]$ is attained at some breakpoint curve in $D_e$ or at one of
the boundary segments $x_e = 0, x_e = l_e$ of $D_e$ for some $e \in E$ [10].

To obtain differentiable expressions for relevant functions, Lopez-de-los-Mozos et al. [10]
define the list $\Sigma = \{t_1, ..., t_\sigma\}$ of $t$-coordinates of all vertices of the initial partitions $P(D_e)$, $e \in E$, sorted in non-decreasing order; thus, $t_1 = t^-$, $t_\sigma = t^+$, $t_{k+1} \geq t_k$ for any $k = 1, ..., \sigma - 1$, $\sigma = |\Sigma| = O(n^3)$. Then, they consider intervals $[t_k, t_{k+1}]$, $k \in [1 : \sigma - 1]$. When $(x_e, t) \in e \times [t_k, t_{k+1}]$ moves along a breakpoint curve $f_{e,i}(x_e, t) = a_{e,i}x_et + b_{e,i}x_e + c_{e,i}t + d_{e,i} = 0$, $x_e$ can be expressed as a function of $t$, $x_e = \hat{f}_{e,i}(t)$, unless $a_{e,i} = b_{e,i} = 0$. They define $\hat{I}_{e,i}$ as the projection of the whole breakpoint curve $f_{e,i}(x_e, t) = a_{e,i}x_et + b_{e,i}x_e + c_{e,i}t + d_{e,i} = 0$, $(x_e, t) \in D_e$ on the $t$-edge of $D_e$ (if $a_{e,i} = b_{e,i} = 0$, this projection is a single point $t^{e,i} = -d_{e,i}/c_{e,i}$; in this special case, they define $\hat{f}_{e,i}(t) = x^{e,i}$ where $x^{e,i}$ is the minimizer of $F(x_e, t)$ over $x_e \in [0, l_e]$ for a fixed $t = t^{e,i}$). Each function $F(\hat{f}_{e,i}(t), t)$, $t \in \hat{I}_{e,i}$, has a single differentiable expression as the factor $(W(t))^{-2}$ multiplied by the quotient of two polynomials with maximum degrees bounded by 3 and 1, respectively, within any interval $[t_k, t_{k+1}]$ in $\hat{I}_{e,i}$ (but this expression can be different for different intervals); therefore, they consider these functions defined on different intervals $[t_k, t_{k+1}]$ separately (thereby each function $F(\hat{f}_{e,i}(t), t) \in \hat{I}_{e,i}$ is split into at most $|\Sigma| - 1 = \sigma - 1$ consecutive pieces), and point out that the function $F^*(t)$ is obtained from the lower envelope $L(t)$ of the collection of $(\sigma - 1)(n + 2)(n - 1) = O(n^5)$ functions

$$\{F(\hat{f}_{e,i}(t), t), t \in \hat{I}_{e,i}, i \in [1 : n], F(0, t), F(l_e, t)\}, e \in E, t \in [t_k, t_{k+1}], k \in [1 : \sigma - 1]\}. \quad (1)$$

These functions are partially defined on $[t^-, t^+]$, have differentiable expressions, and the number of intersection points between any two of them is at most four [10]. Then, invoking standard computational geometry results [14], the lower envelope $L(t)$ can be computed in $O(\lambda_5(n^5) \log n)$ time and consists of $O(\lambda_6(n^5))$ consecutive differentiable arcs.

### 3.3 Phases 2 and 3: Solving MMR-MAD.

Let $\tau = \{t'_1, ..., t'_r\}$ be the list of consecutive breakpoints of the lower envelope $L(t)$. Lopez-de-llos-Mozos et al. [10] consider a new (finer) partition $P'(D_e)$ of the domain $D_e$ which is obtained when the cells $\{R_j, j \in [1 : r_e]\}$ of the initial partition $P(D_e)$ are crossed by the set of lines $t = t'_k$, $t'_k \in \tau$. Obtaining the partition $P'(D_e)$ they consider as **Phase 2** of the algorithm. Let $R'_s$, $s \in [1 : r'_e]$ be the cells of $P'(D_e)$. Then, in the interior of each cell $R'_s$ of $P'(D_e)$, $F(x, t) - F^*(t)$ is a differentiable function. It is shown in [10] that for a fixed $x_e \in [0, l_e]$, the maximum of $F(x_e, t) - F^*(t)$ over $t \in (t^-, t^+)$ cannot be on a breakpoint curve, and, if it is in the interior of a cell $R'_s$ of $P'(D_e)$, it must satisfy

$$H_s(x_e, t) = \frac{\partial(F(x_e, t) - F^*(t))}{\partial t} = 0 \quad (2)$$
They showed that this equation can be represented as a bivariate polynomial equation in $x, t$ of maximum degree 3 in $t$ and 1 in $x$. Thus, when $(x_e, t)$ moves along the stationary curve $H_s(x_e, t) = 0$, isolating (if possible) $t$ from (2) can be done in closed form, $t = h_s(x_e)$, and values of $F(x_e, t) - F^*(t)$ over the stationary curve can be expressed as $F(x_e, h_s(x_e)) - F^*(h_s(x_e))$ for $x \in J_s$ where $J_s$ is the projection of $H_s(x_e, t) = 0$ on the $x$-edge. If it is not possible to isolate $t$ from (2), then the stationary curve is described as $x_e = x_e^s$, and $J_s = \{x_e^s\}$. In this special case, for consistency, they define $h_s(x_e) = t^*$ where $t^*$ is the value where $\max_t \{F(x_e^s, t) - F^*(t) : (x_e^s, t) \in R^s\}$ is attained.

Lopez-de-los-Mozos et al. [10] state that the partition $P'(D_e)$ has $r^e_\epsilon = O(n^2\lambda_6(n^5))$ cells since the initial partition $P(D_e)$ has $O(n^2)$ cells and $|\tau| = g = O(\lambda_6(n^5))$, and point out that $\max_{t \in [t^* - \epsilon, t^* + \epsilon]} \{F(x_e, t) - F^*(t)\}$ as a function of $x_e \in [0, l_e]$ is the upper envelope $U'_e(x_e)$ of the collection of functions

$$\{F(x_e, h_s(x_e)) - F^*(h_s(x_e)), \ x_e \in J_s, \ s \in [1 : r^e_\epsilon], \ F(x_e, t^}_k - F^*(t^}_k), \ k \in [1 : |\tau|]\}. \quad (3)$$

Obtaining the upper envelope $U'_e(x)$ they consider as **Phase 3** of the algorithm. Since $r^e_\epsilon = O(n^2\lambda_6(n^5))$, and the number of intersection points of any two functions from the collection (3) is bounded from above by a constant that does not depend on $n$, they obtain complexity $O(n^2\lambda_6(n^5) \log^* n \log n)$ for constructing $U'_e(x)$ and solving MMR-MAD restricted to a single edge $e$, which results in $O(n^3\lambda_6(n^5) \log^* n \log n)$ complexity for solving MMR-MAD over the whole tree. We note that since $\lambda_6(n^5)$ grows (slightly) faster than $n^5$, the complexity is (slightly) higher than $O(n^8 \log n)$.

### 3.4 MMR-MAD on a general network.

For a general network $N$, a point $x_e$ of an edge $e = [u, v]$ is called a **bottleneck point** with respect to $v_i \in V$, if the distance from $x_e$ to $v_i$ is the same via vertex $u$ as via vertex $v$. The bottleneck points partition the edge into $O(n)$ **primary regions** over which the distance function $d(x_e, v_j)$ is linear for any $v_j \in V$, and thus a primary region of an edge of the general network is similar to an edge of a tree. Lopez-de-los-Mozos et al. [10] point out that MMR-MAD restricted to a primary region in a general network can be solved and analyzed using the same procedure as MMR-MAD restricted to an edge of a tree. They claim that this results in $O(mn^3\lambda_6(n^5) \log^* n \log n)$ complexity for solving MMR-MAD on the whole network.
since there are $O(mn)$ primary regions. However, there is an error in this argumentation:
appearently they did not take into consideration that applying the same logic in the case of
a general network would result in the estimate $O(mn^3)$ for the cardinality of the list $\Sigma$ in
Phase 1 (instead of $O(n^3)$ for the tree case), because there are $O(mn)$ primary regions that
contribute points to this list instead of $O(n)$ edges of a tree. Also, instead of $n - 1$ edges
e $\in E$ in the case of a tree, for a general network we would have $O(mn)$ primary regions
in (1). Following the logic of [10], function $F(x,t)$ over each breakpoint curve is split into
$O(|\Sigma|) = O(mn^3)$ consecutive pieces, there are $O(n)$ breakpoint curves in each primary region
and $O(mn)$ primary regions. Therefore, a direct extension of the analysis of [10] from the tree
case to the general network case would result in the bound $O(m^2n^5)$ for the cardinality of the
collection (1) in the case of a general network and in the bound $O(\lambda_6(m^2n^5))$ for the number
of arcs of the lower envelope $L(t)$. In turn, the estimate of the cardinality of the list $\tau$ in Phase
2 would be $O(\lambda_6(m^2n^5))$ instead of $O(\lambda_6(n^5))$, which, following the logic of [10], would result
in the bound $O(mn^3\lambda_6(m^2n^5)\log^* n \log n)$ for the complexity of their algorithm for a general
network (instead of $O(mn^3\lambda_6(n^5)\log^* n \log n)$ stated in [10]), which is (slightly) higher than
$O(m^3n^8 \log n)$.

### 4 Improvements for the tree case

In this section, we assume that $N$ is a tree.

#### 4.1 Improvement 1.

In this subsection, we present an algorithmic improvement. In the algorithm of [10], in Phase
1, $F^*(t)$ is obtained as the lower envelope $L(t)$ of the collection (1) of $O(n^3)$ differentiable
functions partially defined on $[t^-, t^+]$. Thus, as argued in [10], $L(t)$ has $O(\lambda_6(n^5))$ differentiable
arcs and breakpoints. Here we show that $F^*(t)$ can be obtained as the lower envelope of a much
smaller collection of functions.

To avoid ambiguity, if breakpoint curves $B_{e,i}$ and $B_{e,j}$ are identical, we consider that they
do not have intersection points.

Consider a breakpoint curve $B_{e,i}$ defined by $f_{e,i}(x_e, t) = a_{e,i}x_e + b_{e,i}x_e + c_{e,i}t + d_{e,i} = 0,$
\((x_e, t) \in [0, l_e] \times [t^-, t^+] = D_e\), and its projection \(\hat{I}_{e,i}\) on the \(t\)-edge of \(D_e\), and suppose that \(\hat{I}_{e,i}\) is not a single point. Then, as discussed in Subsection 3.2, when \((x_e, t)\) moves along the breakpoint curve, \(x_e\) can be expressed as a function of \(t\), \(x_e = \hat{f}_{e,i}(t)\). Let \(t_{1}^{e,i}, t_{2}^{e,i}, ..., t_{q_{e,i}}^{e,i}\) be the sorted distinct \(t\)-coordinates of the intersection points of \(B_{e,i}\) with the other breakpoint curves \(B_{e,j}\) of edge \(e\) or with the boundary of \(D_e\), \(t_{1}^{e,i} < t_{2}^{e,i} < ... < t_{q_{e,i}}^{e,i}\); \(q_{e,i} = O(n)\). Observe that function \(F(\hat{f}_{e,i}(t), t)\) on \(\hat{I}_{e,i}\) is differentiable everywhere except the points \(t_{k}^{e,i}\), \(k \in [1 : q_{e,i}]\); only at these points the differentiable expression for \(F(\hat{f}_{e,i}(t), t)\) may change. Consequently, to break \(F(\hat{f}_{e,i}(t), t)\) into pieces with differentiable expressions, it is sufficient to break it into \(O(n)\) pieces defined by the points \(t_{k}^{e,i}\), \(k \in [1 : q_{e,i}]\), instead of \(O(n^3)\) pieces defined by the list \(\Sigma\) as done in \([10]\).

If \(\hat{I}_{e,i}\) is a single point \(t^{e,i}\), we define \(\hat{f}_{e,i}(t)\) as done in \([10]\) for this case, i.e., \(\hat{f}_{e,i}(t) = x^{e,i}\) where \(x^{e,i}\) is a minimizer of \(F(x_e, t)\) over \(x_e \in [0, l_e]\) for a fixed \(t = t^{e,i}\), and for consistency of notation define \(q_{e,i} = 2\), \(t_{1}^{e,i} = t_{2}^{e,i} = t^{e,i}\).

We can treat the boundary segments \(x_e = 0\) and \(x_e = l_e\) of \(D_e\) similarly to the breakpoint curves, formally considering them as breakpoint curves with indices 0 and \(n + 1\). Each of the functions \(F(0, t)\) and \(F(l_e, t), t \in (t^-, t^+)\) is differentiable everywhere except the \(t\)-coordinates of the \(O(n)\) points where the breakpoint curves \(B_{e,i}, i \in [1 : n]\) intersect the corresponding segment of the boundary of \(D_e\). Let us define \(\hat{f}_{e,0}(t) = 0, \hat{f}_{e,n+1}(t) = l_e\), and let \(t_{1}^{e,0}, t_{2}^{e,0}, ..., t_{q_{e,0}}^{e,0}\) be the sorted distinct \(t\)-coordinates of the intersection points of the breakpoint curves \(B_{e,i}, i \in [1 : n]\), with the segment \(x_e = 0\) of the boundary of \(D_e\) plus the values \(t^-, t^+\), i.e., \(t^--t_{1}^{e,0} < t_{2}^{e,0} < ... < t_{q_{e,0}}^{e,0} = t^+\). Let \(t_{1}^{e,n+1}, t_{2}^{e,n+1}, ..., t_{q_{e,n+1}}^{e,n+1}\) be the values defined similarly for the segment \(x_e = l_e\) of \(D_e\). Define \(\hat{I}_{e,0} = \hat{I}_{e,n+1} = [t^-, t^+]\).

**Observation 1.** The function \(F^{*}(t)\) is obtained from the lower envelope \(L'(t)\) of the collection of functions

\[
\left\{ F(\hat{f}_{e,i}(t), t), t \in \hat{I}_{e,i} \cap [t_{k}^{e,i}, t_{k+1}^{e,i}], k \in [1 : q_{e,i} - 1], i \in [0 : n + 1], e \in E \right\}.
\]

Since \(q_{e,i} = O(n)\), and \(|E| = n - 1\), the collection has \(O(n^3)\) functions. All of the functions have differentiable expressions, are partially defined on \([t^-, t^+]\), and any pair of them intersect in at most four points for the reasons discussed in \([10]\) (see also Subsection 3.2). Consequently, \(L'(t)\) has \(O(\lambda_6(n^3))\) consecutive differentiable arcs and can be computed in \(O(\lambda_5(n^3) \log n)\) time.

Keeping the remainder of the computations and the analysis the same as in \([10]\), we obtain that the list \(\tau\) in Phase 2 will have cardinality \(O(\lambda_6(n^3))\), and the overall complex-
ity bound of the algorithm with Improvement 1 will be $O(n^3 \lambda_6(n^3) \log^* n \log n)$ instead of $O(n^3 \lambda_6(n^5) \log^* n \log n)$ in [10].

4.2 Improvement 2.

In this subsection, we show that the collection (4) defined in the previous subsection in fact has a smaller cardinality $O(n^2)$ instead of $O(n^3 \lambda_6(n^3) \log^* n \log n)$ in [10].

For any two points $b, c$ of the tree $N$, let $P(b, c)$ denote the unique path between $b$ and $c$.

**Observation 2.** For any $i, j \in [1 : n]$, the set of points $a \in N$ that are equidistant from $v_i$ and $v_j$ (i.e., $d(a, v_i) = d(a, v_j)$) is either a single point of $N$, or is a subtree of $N$.

**Proof.** If the middlepoint of the path $P(v_i, v_j)$ is an interior point of some edge, this is the only point of $N$ equidistant from $v_i$ and $v_j$. If the middlepoint of $P(v_i, v_j)$ is a vertex $v_k$, then the set of points equidistant from $v_i$ and $v_j$ is the set of points $a \in N$ such that $v_k \in P(a, v_i)$, $v_k \in P(a, v_j)$ which is a subtree of $N$. □

An intersection point $(x_e, t)$ between breakpoint curves $B_{e,i}$ and $B_{e,j}$ is called a proper intersection point if $x_e \neq 0, x_e \neq l_e$, i.e. $x_e$ is an interior point of $e$. The following result shows that for any $i, j \in [1 : n]$, at most one edge $e$ of the tree $N$ contains a proper intersection point of the breakpoint curves $B_{e,i}$ and $B_{e,j}$.

**Lemma 1.** For any $i, j \in [1 : n]$, there is at most one proper intersection point of the breakpoint curves $B_{e,i}$ and $B_{e,j}$ over all edges $e$ of the tree $N$.

**Proof.** If $(x_e, t)$ is an intersection point between $B_{e,i}$ and $B_{e,j}$, $x_e$ must be equidistant from $v_i$ and $v_j$. If there is only one point of $N$ that is equidistant from $v_i$ and $v_j$, the statement of the lemma is straightforward. If there is a subtree of points equidistant from $v_i$ and $v_j$, then for any edge $e$ of this subtree $B_{e,i}$ and $B_{e,j}$ are identical and thus do not contribute intersection points. Other edges of $N$ do not have interior points equidistant from $v_i$ and $v_j$ and thus do not contribute proper intersection points between $B_{e,i}$ and $B_{e,j}$. The lemma is proven. □
Theorem 1. The collection (4) defined in the previous subsection has $O(n^2)$ functions.

Proof. For any $i \in [1 : n]$ and any $e \in E$, $B_{e,i}$ has a constant number (no more than 4) of intersection points with the boundary of $D_e$, thus for a fixed $i \in [1 : n]$ all $B_{e,i}$, $e \in E$ together have $O(n)$ intersection points with the boundaries of $D_e$, $e \in E$. Any non-proper intersection point between $B_{e,i}$ and $B_{e,j}$ is also an intersection point of these curves with the boundary of $D_e$. Using Lemma 1, we have that there are $O(n^2)$ proper intersection points between breakpoint curves over all $e \in E$, and $O(n^2)$ intersection points between breakpoint curves and the boundaries of the corresponding domains $D_e$. Also, we get that $\sum_{e \in E} q_{e,i} = O(n)$ for any $i \in [1 : n]$, and $\sum_{e \in E} q_{e,i} = O(n^2)$ for $i = 0$ and $i = n + 1$. The statement of the theorem follows. □

Keeping the remainder of the computations and the analysis the same as in [10], we obtain that the list $\tau$ in Phase 2 will have cardinality $O(\lambda_6(n^2))$, and the overall complexity bound of the algorithm with Improvements 1 and 2 will be $O(n^3 \lambda_6(n^2) \log^* n \log n)$ instead of $O(n^3 \lambda_6(n^5) \log^* n \log n)$ in [10].

Remark. The discussion above implies that the initial domain partitions $P(D_e)$, $e \in E$ altogether have $O(n^2)$ vertices and edges. Since each edge of each partition is adjacent to at most two cells, this implies that the partitions $P(D_e)$, $e \in E$ altogether have $O(n^2)$ cells. This observation will be used later in Subsection 4.4.

4.3 Improvement 3.

Improvements 1 and 2 focused on Phase 1. Now we focus on Phase 2. Lopez-de-los-Mozos et al. [10] estimate the number of cells in the partition $P'(D_e)$ obtained in Phase 2 as the product of the estimate of the cardinality of the list $\tau$ which is $O(\lambda_6(n^5))$ in [10] and the number of cells of $P(D_e)$ which is $O(n^2)$, obtaining an estimate $O(n^2 \lambda_6(n^5))$ for the number of cells in $P'(D_e)$. Using the same logic, in our case we would obtain a bound $O(n^3 \lambda_6(n^2))$ for the number of cells in $P'(D_e)$, because with Improvements 1 and 2, $|\tau| = O(\lambda_6(n^2))$. However, using a different argument, we will obtain a better bound $O(n \lambda_6(n^2))$ for the number of cells in $P'(D_e)$.

Again, let $\tau = \{t'_1, \ldots, t'_g\}$ be the list of consecutive breakpoints of the lower envelope $L'(t)$, $|\tau| = O(\lambda_6(n^2))$ according to the discussion above. Partition $P'(D_e)$ of $D_e$ is obtained as discussed in Subsection 3.3. Observe that each line $t = t'_k$, $t'_k \in \tau$ has $O(n)$ intersection points.
with the breakpoint curves in $D_e$. (If the line $t = t'_k$ is itself a breakpoint curve in $D_e$, we consider that it does not have intersection points with itself, and we can ignore the line as it does not contribute new cells to $P'(D_e)$). These intersection points partition the segment \{\(t = t'_k, \ x_e \in [0, l_e]\)\} of $D_e$ into $O(n)$ subsegments which we will call $\tau$-segments. Thus, there are $O(n\lambda_6(n^2))$ $\tau$-segments in $D_e$. Observe that each $\tau$-segment is an edge of at most two cells of $P'(D_e)$. Consequently, at most $O(n\lambda_6(n^2))$ cells of $P'(D_e)$ are adjacent to some $\tau$-segments. Observe that if a cell $R'_s$ of $P'(D_e)$ is not adjacent to any $\tau$-segment, then $R'_s$ is one of the cells of the original partition $P(D_e)$. Since there are $O(n^2)$ cells in $P(D_e)$, the number of cells of $P'(D_e)$ is $O(n\lambda_6(n^2)) + O(n^2) = O(n\lambda_6(n^2))$.

Keeping the remainder of the computations and the analysis the same as in [10], we obtain that the collection of functions \((3)\) will have cardinality $O(n\lambda_6(n^2))$, and the overall complexity bound of the algorithm with Improvements 1, 2 and 3 will be $O(n^2\lambda_6(n^2) \log^* n \log n)$ instead of $O(n^3\lambda_6(n^5) \log^* n \log n)$ in [10].

### 4.4 Improvement 4: The case of a path.

In this subsection, we assume that $N$ is a path network. Since a path is a special case of a tree, Improvements 1-3 are still valid. A path can be viewed as embedded in the number line. In addition to the notation $x_e$ that represents a point of an edge $e$, we will also use notation $x$ to represent the point of the path that is $x$ units away from its leftmost point.

Observe that for any fixed $t' \in [t^-, t^+]$, function $M(x, t')$ is continuous piece-wise linear convex as a function of $x$, with breakpoints at the vertices of $N$. For any $i \in [1 : n]$, function $d(v_i, x)$ is a convex tooth-function, i.e. continuous piecewise-linear convex function with at most two linear pieces and one breakpoint at $x = v_i$. This implies the following observation.

**Observation 3.** For fixed $t' \in [t^-, t^+]$ and $i' \in [1 : n]$, the equation $d(v_{i'}, x) - M(x, t') = 0$ over $x \in N$ has at most four isolated solutions, and also may hold over one or more full edges.

Consider some $t'$ from the list $\tau$ of breakpoints of the lower envelope $L'(t)$, and an $i' \in [1 : n]$. Suppose that the equation $d(v_{i'}, x) - M(x, t') = 0$ has only isolated solutions (at most four according to Observation 3). Then, the segments $\{t = t', \ x_e \in [0, l_e]\}$, $e \in E$ have at most four intersections with the breakpoint curves $B_{e,i'}, e \in E$ over all $n - 1$ domains $D_e$, $e \in E$. If the equation $d(v_{i'}, x) - M(x, t') = 0$ also holds over some full edges $e' \in E$, then for each
such edge $e'$ the segment $\{t = t', x \in [0 : l_e]\}$ coincides with the breakpoint curve $B_{e', t}$ and thus can be ignored as it does not contribute additional cells to $P'(D_{e'})$. Thus, the overall number of $\tau$-segments in all domains $D_e$, $e \in E$ that correspond to a fixed $t' \in \tau$ is $O(n)$. Since $|\tau| = O(\lambda_6(n^2))$, the overall number of $\tau$-segments in all domains $D_e$, $e \in E$ is $O(n\lambda_6(n^2))$. (Note that for the tree case, $O(n\lambda_6(n^2))$ was the estimate of the number of $\tau$-segments in just one domain $D_e$.) So, the number of cells that are adjacent to $\tau$-segments in all partitions $P'(D_e)$, $e \in E$ is $O(n\lambda_6(n^2))$. The number of cells not adjacent to any $\tau$-segments in all $P'(D_e)$, $e \in E$ is $O(n^2)$ since each such cell is a cell of some initial partition $P(D_e)$, and the total number of cells in all $P(D_e)$, $e \in E$ is $O(n^2)$ as derived in the discussion of Improvement 2 (see the remark at the end of Subsection 4.2). Consequently, the total number of cells in all $P'(D_e)$, $e \in E$ is $O(n\lambda_6(n^2))$. Let $C_e$ be the cardinality of the collection of functions (3) taking into account all the improvements; we obtain $\sum_{e \in E} C_e = O(n\lambda_6(n^2))$. Therefore, the overall complexity bound of the algorithm reduces to $O(n\lambda_6(n^2) \log^* n \log n)$ for the path case.

The following theorem summarizes the results of this section.

**Theorem 2.** a) MMR-MAD on a tree can be solved in $O(n^2 \lambda_6(n^2) \log^* n \log n) = O(n^4(\log^* n)^2 \log n)$ time.

b) MMR-MAD on a path can be solved in $O(n\lambda_6(n^2) \log^* n \log n) = O(n^3(\log^* n)^2 \log n)$ time.

This is a significant improvement over the complexity $O(n^3 \lambda_6(n^5) \log^* n \log n) = O(n^8(\log^* n)^2 \log n)$ obtained in [10].

5 **Improvements for the case of a general network**

In this section, we assume that $N$ is a general network. Thus, instead of domains $D_e$ that correspond to edges in the tree case, we deal with domains $D_{e,k}$ that correspond to primary regions, with $k$ being the index of the primary region on the edge $e$.

Improvement 1 is applicable without any changes, using primary regions of the general network instead of edges of a tree. Instead of breaking function $F(x, t)$ over a breakpoint curve of a primary region into $O(|\Sigma|) = O(mn^3)$ pieces, we break it into $O(n)$ pieces defined by the intersection points of this breakpoint curve with other breakpoint curves of the same primary region and the boundary of the corresponding domain. Each piece has a single differentiable
expression as a function of $t$. This results in $F^\ast(t)$ being represented as the lower envelope of a collection of $O(mn^2)$ functions (there are $O(mn)$ primary regions, $O(n)$ breakpoint curves for each primary region, and $F(x,t)$ over each breakpoint curve is split into $O(n)$ consecutive pieces), which leads to the estimate $O(mn^3\lambda_6(mn^3) \log^* n \log n)$ for the complexity of the overall algorithm for MMR-MAD. To emphasize that we are considering the general network case, we will call this Improvement 1G.

The idea of Improvement 2 is applicable in a modified form.

**Observation 4.** For any two vertices $v_i$, $v_j$ and an edge $e \in E$, the equation $d(v_i, x) = d(v_j, x)$ has at most one isolated solution on the edge $e$, and also may hold on one or more full primary regions of $e$; for such primary regions, the breakpoint curves that correspond to $v_i$ and $v_j$ coincide if exist.

Observation 4 implies that there is at most one intersection point of the breakpoint curves that correspond to $v_i$ and $v_j$ over all primary regions of $e$ and the corresponding domains $D_{e,k}$. So, the overall number of intersection points of breakpoint curves over all primary regions and the corresponding domains of an edge $e$ is $O(n^2)$. The overall number of intersection points between the breakpoint curves and the boundaries of all domains $D_{e,k}$ of an edge $e$ is also $O(n^2)$. This will result in $F^\ast(t)$ being represented as the lower envelope of $O(mn^2)$ functions, which leads to the estimate $O(mn^3\lambda_6(mn^2) \log^* n \log n)$ for the complexity of the overall algorithm for MMR-MAD. We will call this Improvement 2G.

Improvement 3 is applicable straightforwardly with the same logic, with domains $D_{e,k}$ instead of domains $D_e$, and will be called Improvement 3G. It reduces the overall complexity bound by a factor of $n$ to $O(mn^2\lambda_6(mn^2) \log^* n \log n)$.

An analog of Improvement 4, which will be called Improvement 4G, is applicable for every edge, because the primary regions of an edge are similar to edges of a path. However, there are some differences. Function $M(x,t')$ over an edge $e$ for a fixed $t' \in [t^-, t^+]$ is a continuous concave piecewise-linear function of $x$ (instead of a convex function $M(x,t)$ on a path) with breakpoints at the endpoints of primary regions; $d(v_i, x)$ over an edge $e$ for any $i \in [1:n]$ is a concave continuous piecewise-linear function (instead of a convex function $d(v_i, x)$ on a path) with at most two linear pieces and a breakpoint at an endpoint of a primary region; so we have the following analog of Observation 3.
Observation 5. For fixed $t' \in [t^-, t^+]$ and $i' \in [1:n]$, for any edge $e \in E$, the equation $d(v_{i'}, x) - M(x, t') = 0$ over $x \in e$ has at most four isolated solutions, and also may hold over one or more full primary regions of $e$.

All subsequent arguments are similar, with primary regions of an edge $e \in E$ and the corresponding domains $D_{e,k}$ instead of edges $e \in E$ and the corresponding domains $D_e$ for a path. Improvement 4G reduces the overall complexity bound by a factor of $n$. So, we obtain the following theorem that summarizes the results of this section.

**Theorem 3.** MMR-MAD on a general network can be solved in $O(mn\lambda_6(mn^2) \log^* n \log n) = O(m^2n^3(\log^* n)^2 \log n)$ time.

This is a significant improvement over the complexity bound $O(mn^3\lambda_6(m^2n^5) \log^* n \log n) = O(m^3n^8(\log^* n)^2 \log n)$ obtained using the analysis in [10].

6 Conclusion

In this paper, for the minmax regret mean absolute deviation single facility location problem MMR-MAD, we provided a further analysis of the polynomial time solution framework developed in [10]. We presented one algorithmic improvement (Improvement 1 or 1G) and three analytical improvements (Improvements 2-4 or 2G-4G) which significantly improve the computational complexity bounds for the algorithm. Specifically, for trees our complexity bound is $O(n^2\lambda_6(n^2) \log^* n \log n) = O(n^4(\log^* n)^2 \log n)$ as opposed to the bound $O(n^3\lambda_6(n^5) \log^* n \log n) = O(n^8(\log^* n)^2 \log n)$ obtained in [10]. For paths, our bound is further reduced to $O(n\lambda_6(n^2) \log^* n \log n) = O(n^3(\log^* n)^2 \log n)$. For general networks, our complexity bound is $O(mn\lambda_6(mn^2) \log^* n \log n) = O(m^2n^3(\log^* n)^2 \log n)$, as opposed to the bound $O(mn^3\lambda_6(m^2n^5) \log^* n \log n) = O(m^3n^8(\log^* n)^2 \log n)$ that follows from the analysis in [10].

We note that our improvements do not change in a major way the algorithmic framework of [10]; rather, they represent a deeper analysis of that framework. The complexity bounds obtained in [10] are polynomial but way too high to be of any practical use. Our complexity bounds, in our opinion, are already practical, as they show that the approach can be used for solving problems on networks of moderate size in times reasonable for strategic location applications. Thus, our results can also be viewed as a justification that the algorithmic framework
suggested in [10] is practical.

**ACKNOWLEDGEMENT.** The research of Igor Averbakh was supported by the Discovery Grant 238234-2012-RGPIN from the Natural Sciences and Engineering Research Council of Canada (NSERC). The research of Oded Berman was supported by the Discovery Grant 10547-2011-RGPIN from NSERC. The research of Marina Leal was supported by the Spanish Ministry of Science and Technology project MTM2013-46962-C2-1-P.

**References**


