Journal of Optimization Theory and Applications (2024) 200:178–193
https://doi.org/10.1007/s10957-023-02352-8

On Computing Medians of Marked Point Process Data Under Edit Distance

Noriyoshi Sukegawa[1] · **Shohei Suzuki**[2] · **Yoshiko Ikebe**[2] · **Yoshito Hirata**[3]

Received: 11 August 2022 / Accepted: 20 November 2023 / Published online: 20 December 2023

Abstract
In this paper, we consider the problem of computing a median of marked point process data under an edit distance. We formulate this problem as a binary linear program, and propose to solve it to optimality by software. We show results of numerical experiments to demonstrate the effectiveness of the proposed method and its application in earthquake prediction.

Keywords Integer programming · Marked point process · Median

Mathematics Subject Classification 90C10 · 90C90 · 37M10 · 60G55 · 86A15

1 Introduction

A time series of discrete events observed in continuous time is referred to as a point process. In particular, if each event is associated with some values called marks, then it is referred to as a marked point process (MPP). An example of the MPP is earthquake data since earthquakes occur discretely in continuous time and their magnitudes and epicenters, for example, naturally define their marks; see, e.g., [6, 15, 20, 24, 29] for other examples.

Communicated by Anita Schöbel.

✉ Noriyoshi Sukegawa
sukegawa@hosei.ac.jp

[1] Department of Advanced Sciences, Faculty of Science and Engineering, Hosei University, 3-7-2 Kajino-cho, Koganei-shi, Tokyo 184-8584, Japan

[2] Department of Information and Computer Technology, Faculty of Engineering, Tokyo University of Science, 6-3-1 Niijuku, Katsushika-ku, Tokyo 125-8585, Japan

[3] Institute of Systems and Information Engineering, University of Tsukuba, 1-1-1 Tennodai, Tsukuba-shi, Ibaraki 305-8573, Japan

In analyzing MPPs, it is common to use time windows to generate a large number of pieces, i.e., local patterns, of given data. This processing enables us to discuss properties of given data in terms of relationships among the pieces generated. In this setting, an edit distance, originated in [35] by Victor and Purpura, is often employed as a metric; see [31] for its general definition, and also, e.g., [8, 11, 28] for data analysis based on this distance.

In practice, distance is frequently inquired and hence its computation often forms a bottleneck of the whole computation, affecting the quality of analysis results. For this reason, algorithms for computing the Victor–Purpura-type edit distance (the VP distance) have been studied in the literature; see [1, 12, 14, 35]. See also, e.g., [9, 18, 19] for the Earth mover's distance and, e.g., [5, 34, 36] for the Wasserstein distance, which are related concepts.

However, algorithms for analyzing MPPs under the VP distance are limited in the existing literature [7, 16, 22, 27, 30]. In particular, median (or prototype) is useful when one wants to represent a set of MPPs by a single MPP that is typical and interpretable. Its applications include neuronal spike behavior [7], earthquake aftershocks [30], wildfire activity [27], and plant community [22]. Although algorithms for computing medians of MPPs have been devised in [7, 30, 33], they are heuristic. Against this background, in this paper, we devise a first exact method. It is based on integer programming, which enables us to apply state-of-the-art powerful software. Although optimization techniques have already been used in the literature by the nature of the VP distance, less attention has been paid to integer programming in this context.

1.1 Our Contribution

In this paper, we study the median computation for MPPs under the VP distance. Our contributions include:

- We first point out the existence of a well-structured median. We then use it to give a binary linear programming formulation, offering a first exact method for the median computation of MPPs.
- We conduct numerical experiments on random data, which show that the method has a potential to solve instances with hundreds of MPPs with thousands of events in total in a few minutes using software, Gurobi.
- We present an application of medians of MPPs in predicting earthquake behavior and show its validity using a real-world data set.

We note that the key ingredient for our formulation, the notion of candidates in Lemma 1, has already been pointed out in [32] (Theorem 4a) to develop a heuristic method to compute medians. We also note that although generalized metrics and barycenters are introduced in [10, 25], how to extend our current work to computing the related barycenters for MPPs is an open question.

1.2 Organization

The remainder of this paper is organized as follows. We first introduce definitions in Sect. 2. We then analyze the median computation in Sect. 3, and formulate it as a binary linear program in Sect. 4. Section 5 is devoted to numerical experiments. Concluding remarks are given in Sect. 6.

2 Preliminaries

2.1 Marked Point Process Data

Let $\mathbb{R}$ be the set of reals. An event is a point p in $\mathbb{R}^d$ where d is some positive integer. The first component is called the time of occurrence of the event, and each remaining $d-1$ component vector is the mark of the event. For example, if event represents earthquake data, we would typically have $d = 5$, with the components of the 4-dimensional mark representing the magnitude, the depth, and latitude and longitude of the epicenter. That is,

$$p = \begin{bmatrix} \text{time} \\ \text{magnitude} \\ \text{depth} \\ \text{latitude} \\ \text{longitude} \end{bmatrix}, \quad \text{mark of} \quad p = \begin{bmatrix} \text{magnitude} \\ \text{depth} \\ \text{latitude} \\ \text{longitude} \end{bmatrix}.$$

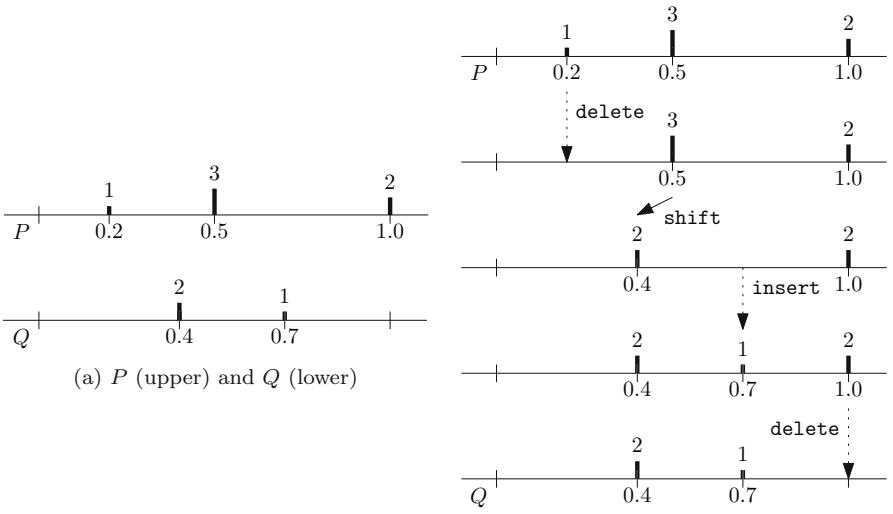
When $d = 1$, we are concerned only with occurrence times of some particular phenomenon (e.g., neuron activity), and the events will have no marks. We denote p_j to represent the jth component of event p.

A marked point process (MPP) P is defined as a finite multiset of events. Thus, $p = q$ can hold for two events p, q in a general MPP. Hereafter, we denote $|P|$ to represent the number of events in P, and also $\mathcal{U}$ to represent the set of all possible MPPs for some fixed d. The analysis of MPPs is important [21] because they naturally arise in many phenomena, such as earthquakes [13, 30], foreign exchange markets [11, 31] and their interactions [26], and floods [2]. In such analyses, edit distance plays a central role. We next define the edit distance of two MPPs.

2.2 Edit Distance for Marked Point Process Data

To define the Victor–Purpura-type edit distance (VP distance), we need first introduce transformations from P into Q, by means of the following three elementary operations on events: For $X \in \mathcal{U}$,

- the `shift` operation on event x of X is to replace it by an arbitrary event y in $\mathbb{R}^d$, which we call the `shift` destination of x,
- the `delete` operation on an event of X is to remove it from X, and
- the `insert` operation on an event of $\mathbb{R}^d$ is to add it to X.



(a) P (upper) and Q (lower)

(b) A transformation from P into Q

Fig. 1 Example of two MPP data P, Q and a transformation from P into Q

Note that $|X|$ is unchanged by the `shift` operation, while the `delete` and `insert`, respectively, decrease and increase the value by one.

We can transform any given $P \in U$ to any $Q \in U$ by a finite sequence of the above three operations. Indeed, a trivial transformation is to `delete` all events of P, then `insert` all events of Q. We now define costs of elementary operations. The costs of `delete` and `insert` are 1, i.e., a constant, while the cost of `shift` of event x to y is given by

$$\texttt{cost}(x, y) = \sum_{j=1}^{d} \lambda_j |x_j - y_j|,$$

where λ_j is a positive parameter for $j = 1, \ldots, d$. The cost of a transformation from P into Q, i.e., a finite sequence of `shift`, `delete`, and `insert` of events whose application to P yields Q, is defined as the sum of costs of the elementary operations. We say that a transformation from P into Q, is *optimal* if its cost attains the minimum cost among all such transformations.

For convenience, we write `cost`(T) to represent the cost of a transformation T. Also, we say that a transformation from P into Q is *simple* if

(C1) for each event of P, either `delete` or `shift` is applied exactly once,
(C2) for each event of Q, either `insert` is applied exactly once, or it is the `shift` destination of exactly one event of P,

and no other operations are involved in the transformation. Figure 1a shows, for $d = 2$,

two MPPs

$$P = \left\{ \begin{bmatrix} 0.2 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.5 \\ 3 \end{bmatrix}, \begin{bmatrix} 1.0 \\ 2 \end{bmatrix} \right\}, \quad Q = \left\{ \begin{bmatrix} 0.4 \\ 2 \end{bmatrix}, \begin{bmatrix} 0.7 \\ 1 \end{bmatrix} \right\},$$

where events are indicated by vertical bars. The values below the bars show the times of occurrence, and those above the bars correspond to the value of the (1-dimensional) mark. Figure 1b shows a simple transformation from P into Q that involves one `shift`, one `insert`, and two `delete` operations.

Proposition 1 *For any two MPPs P and Q, there always exists an optimal simple transformation from P into Q.*

Proof It suffices to show that we need only consider transformations in which `shift` is applied at most once to each event of P. To this end, suppose that `shift` is applied twice to some event, first from p to q, then from q to r. The cost associated with these two `shift` operations is

$$\begin{aligned} \texttt{cost}(p,q) + \texttt{cost}(q,r) &= \sum_{j=1}^{d} \lambda_j (|p_j - q_j| + |q_j - r_j|) \\ &\geq \sum_{j=1}^{d} \lambda_j |p_j - r_j| \\ &= \texttt{cost}(p,r), \end{aligned}$$

thus unifying the two `shift` operations into one operation from p to r would incur no increase in cost. This completes the proof of the proposition since any `shift` on an inserted event and any `delete` on a shifted event can be replaced by one `insert` and one `delete`, respectively, without increasing the cost. □

We can now define the edit distance.

Definition 1 [31, 35] The VP distance $d(P, Q)$ of two MPPs P, Q is the minimum cost required for transforming P into Q using the three operations `shift`, `delete`, and `insert` of events.

The value of $d(P, Q)$ is well defined since it is always attained by the cost of a simple transformation from P into Q from Proposition 1, and there are only finitely many simple transformations for fixed P, Q. The following proposition follows immediately from the definition of edit distance.

Proposition 2 *The edit distance is a metric.*

Proof For any three MPPs P, Q, R, one has

(i) $d(P, Q) = 0$ if and only if $P = Q$ by the positivity of the parameters,
(ii) $d(P, Q) = d(Q, P)$ by the symmetry of the costs, and

(iii) $d(P, R) \leq d(P, Q) + d(Q, R)$ since concatenating any transformation from P into Q and one from Q into R yields a transformation from P into R whose cost is exactly the sum of those of the two.

□

Proposition 3 *The VP distance is a metric on $\mathcal{U}$ for fixed d.*

Proof For any $P, Q, R \in \mathcal{U}$,

(i) $d(P, Q) = 0$ if and only if $P = Q$ from positivity of the parameters,
(ii) $d(P, Q) = d(Q, P)$ by symmetry of the costs, and
(iii) $d(P, R) \leq d(P, Q) + d(Q, R)$ because concatenating two transformations, one from P into Q and the other from Q into R, yields a transformation from P into R whose cost equals the sum of those of the two.

□

2.3 Bipartite-Graph Model for Computing Edit Distance

We now explain how to compute the value of $d(P, Q)$. This is accomplished by solving an assignment problem on a bipartite graph [14]. This bipartite graph is defined as follows:

– Let $V_P^i = \{v_p^i \mid p \in P\}$ and $V_Q^i = \{v_q^i \mid q \in Q\}$ for $i = 0, 1$. Then, set

$$V_P = V_P^0 \cup V_Q^1 \quad \text{and} \quad V_Q = V_P^1 \cup V_Q^0.$$

Thus, $|V_P| = |V_Q| = |P| + |Q|$. We say that each vertex v_x^1 is dummy.
– Let $E = \{\{u, v\} \mid u \in V_P, v \in V_Q\}$. Then, $(V_P \cup V_Q, E)$ is an undirected complete bipartite graph, which we denote by G_{PQ}. Each edge $\{v_x^i, v_y^j\}$ has a cost, denoted by $\texttt{cost}(\{v_x^i, v_y^j\})$, defined as

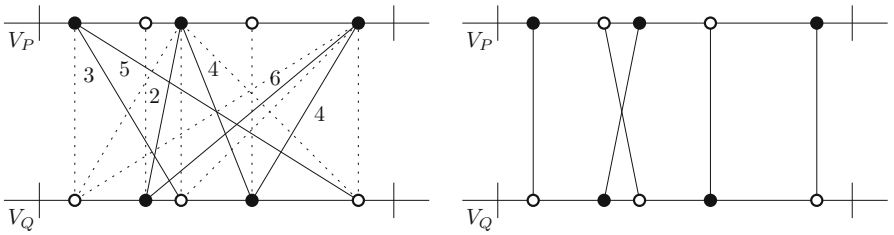
$$\begin{cases} \texttt{cost}(x, y) & \text{if } \ i = 0, j = 0\,(\texttt{shift}), \\ 1 & \text{if } \ i = 1, j = 0, \text{i.e., } v_x^i \text{ is dummy } (\texttt{insert}), \\ 1 & \text{if } \ i = 0, j = 1, \text{i.e., } v_y^j \text{ is dummy } (\texttt{delete}), \\ 0 & \text{if } \ i = 1, j = 1, \text{i.e., } v_x^i \ \text{ and } \ v_y^j \text{ are both dummy.} \end{cases}$$

For $F \subseteq E$, we denote by $\texttt{cost}(F)$ the sum of costs of the edges in F.

Figure 2a shows the graph G_{PQ} for two MPPs P, Q depicted in Fig. 1a. Black (white) circles correspond to non-dummy (dummy) vertices. We show costs corresponding to the `shift` edges next to them assuming $\lambda_1 = \lambda_2 = 10$. The `delete` and `insert` edges are indicated by dotted lines, and edges with 0 cost are omitted.

For a perfect matching M in G_{PQ} let T_M be a simple transformation from P into Q in which

– for each $p \in P$, `delete` is applied if $\{v_p^0, u\} \in M$ for some $u \in V_Q^1$ and `shift` from p to q is applied if $\{v_p^0, v_q^0\} \in M$ for some $q \in Q$, and



(a) The bipartite graph G_{PQ} for two MPPs P, Q depicted in Fig. 1(a)

(b) The perfect matching induced by the transformation depicted in Fig. 1(b)

Fig. 2 The bipartite-graph model of [14] for computing VP distance

– for each $q \in Q$, `insert` is applied if $\{u, v_q^0\} \in M$ for some $u \in V_P^1$ and `shift` from p to q is applied if $\{v_p^0, v_q^0\} \in M$ for some $p \in P$.

Then, $\texttt{cost}(M) = \texttt{cost}(T_M)$ holds by the definitions of the edge costs. Conversely, we observe that for any simple transformation T from P into Q, there is a perfect matching M such that $T = T_M$, which implies the following.

Proposition 4 [14] *For any two MPPs $P, Q \in \mathcal{U}$, let M^* be a minimum-cost perfect matching in G_{PQ}. Then, $\texttt{cost}(M^*) = d(P, Q)$.*

The perfect matching shown in Fig. 2b corresponds to the simple transformation shown in Fig. 1b.

3 Medians of Marked Point Process Data Under Edit Distance

3.1 Problem Description and Toy Examples

The median computation is a problem of finding, for a given set $\{P_1, \ldots, P_k\}$ of MPPs with $P_h \in \mathcal{U}$ for $h = 1, \ldots, k$, a single MPP X in $\mathcal{U}$ that minimizes the sum of VP distances from $\{P_1, \ldots, P_k\}$, i.e.,

$$f(X) \equiv \sum_{h=1}^{k} d(X, P_h),$$

over $\mathcal{U}$. Such an X is called a *median*. Hereafter, the given set of MPPs is referred to as an instance.

The median computation has a simple structure when $k = 2$. To see this, let $\{P_1, P_2\}$ be any instance in this case. Then, both P_1 and P_2 are medians of $\{P_1, P_2\}$ because for any X in $\mathcal{U}$, we have

$$\begin{aligned} f(X) &= d(X, P_1) + d(X, P_2) \\ &= d(P_1, X) + d(X, P_2) \\ &\geq d(P_1, P_2), \end{aligned}$$

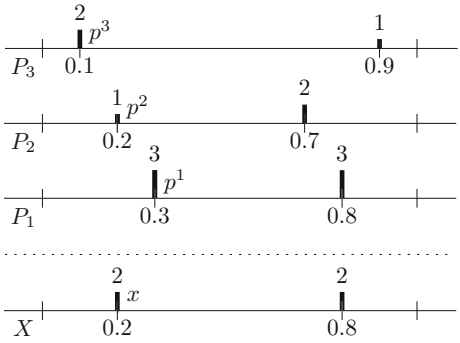


Fig. 3 An instance of the median computation for $d = 2$ and $k = 3$ with $|\mathcal{P}| = 6$ and $|\mathcal{C}| = 9$

and $f(P_1) = f(P_2) = d(P_1, P_2)$ since d is a metric. In general, any minimizer of f among all MPPs in an instance is referred to as a *medoid*. Hence, in this example, P_1 and P_2 are medoids as well as medians.

An instance where medoids and medians have no intersection is depicted in Fig. 3. Assuming that λ_1 and λ_2 are sufficiently small, any median consists of exactly two events, and neither `insert` nor `delete` is used since their costs are high. This means that X shown in the bottom is the unique median. Observe, for example, that event x of X is such that x_1 is the one-dimensional median of $\{p_1^h\}_{h=1,2,3}$, i.e., $\{0.1, 0.2, 0.3\}$ while x_2 is that of $\{p_2^h\}_{h=1,2,3}$, i.e., $\{1, 2, 3\}$ so that the total cost of `shift` is minimized for x.

3.2 Well-Structured Medians

Motivated by the above observation, we here point out that there always exists a median with a simple structure. Let $\mathcal{P}$ denote the set of events in a given instance $\{P_1, \ldots, P_k\}$, i.e., $\mathcal{P} = P_1 \cup \cdots \cup P_k$. Then, we call c in $\mathbb{R}^d$ a candidate if for each $j = 1, \ldots, d$, there is p in $\mathcal{P}$ with $c_j = p_j$, and denote by $\mathcal{C}$ the set of candidates. In other words, letting $\mathcal{A}_j$ be the set of values for the jth component of the events in the instance, i.e., $\{p_j \mid p \in \mathcal{P}\}$, $\mathcal{C}$ can be written as $\mathcal{C} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_d$. Note that $\mathcal{P} \subseteq \mathcal{C}$ and $\mathcal{C}$ is finite since each P_h is finite for $h = 1, \ldots, k$. For the instance shown in Fig. 3, we have $|\mathcal{P}| = 6$ while $|\mathcal{C}| = 18$ because $\mathcal{A}_1 = \{0.1, 0.2, 0.3, 0.7, 0.8, 0.9\}$ and $\mathcal{A}_2 = \{1, 2, 3\}$. Note that the two events of X belong to $\mathcal{C}$ but not to $\mathcal{P}$. The following lemma claims that one can construct a median for any instance by selecting events from $\mathcal{C}$ (allowing duplication).

Lemma 1 *Let $\{P_1, \ldots, P_k\}$ be any instance. Then, there exists a median such that all of its events belong to $\mathcal{C}$. We call such a median well-structured.*

Proof Take any median X and fix a simple transformation from X into P_h with cost $d(X, P_h)$ for each $h = 1, \ldots, k$. Let x be any event of X and H_x be the set of its `shift` destinations. Note that H_x is nonempty; otherwise, removing x from X decreases the value of f, a contradiction. The set X remains a minimizer of f even if x is replaced

by y as long as y minimizes

$$g(y) \equiv \sum_{p \in H_x} \texttt{cost}(y, p) = \sum_{p \in H_x} \sum_{j=1}^{d} \lambda_j |y_j - p_j| = \sum_{j=1}^{d} \lambda_j \left[\sum_{p \in H_x} |y_j - p_j| \right]$$

over $\mathbb{R}^d$. We see that y minimizes g if each y_j minimizes the term in the square bracket, which can be accomplished by setting y_j to p_j for some $p \in H_x$, i.e., a one-dimensional median. This implies that there exists a minimizer of g that belongs to $\mathcal{C}$, and hence completes the proof. □

4 A Binary Linear Programming Model

The binary linear programming model for the median computation proposed in this paper is a direct consequence of Lemma 1. We create, for each $c \in \mathcal{C}$, an integer variable m_c to represent the number of events in a solution that are identical to c, i.e., the multiplicity of c. We also create, for each $c \in \mathcal{C}$ and $p \in \mathcal{P}$, a binary variable s_{cp} which takes one if p is the `shift` destination of c. Finally, we create, for each $p \in \mathcal{P}$, a binary variable ι_p (σ_p) which takes one if p is inserted (p is the `shift` destination of some event). Then, the median computation can be formulated as

$$\text{(P)} \quad \min. \quad \sum_{p \in \mathcal{P}} \iota_p + \sum_{h=1}^{k} \sum_{c \in \mathcal{C}} \left[m_c - \sum_{p \in P_h} s_{cp} \right] + \sum_{c \in \mathcal{C},\, p \in \mathcal{P}} \texttt{cost}(c, p) s_{cp} \tag{1}$$

$$\text{s. t.} \quad \iota_p + \sigma_p = 1 \qquad (p \in \mathcal{P}) \tag{2}$$

$$\sum_{x \in \mathcal{C}} s_{cp} = \sigma_p \qquad (p \in \mathcal{P}) \tag{3}$$

$$\sum_{p \in P_h} s_{cp} \le m_c \qquad (c \in \mathcal{C}, h = 1, \ldots, k) \tag{4}$$

$$m_c \in \mathbb{N} \qquad (c \in \mathcal{C})$$

$$s_{cp} \in \{0, 1\} \qquad (c \in \mathcal{C},\ p \in \mathcal{P})$$

$$\iota_p \in \{0, 1\} \qquad (p \in \mathcal{P})$$

$$\sigma_p \in \{0, 1\} \qquad (p \in \mathcal{P})$$

where c_{cp} represents $c(c, p)$ for each $c \in \mathcal{C}$ and $p \in \mathcal{P}$, and $\mathbb{N}$ denotes the set of nonnegative integers.

In order to obtain the median X corresponding to a solution of (P), pick each element c with multiplicity m_c. The transformation from X to each P_h is obtained as follows: For each $p \in P_h$,

(i) if $\iota_p = 0$ and $\sigma_p = 1$, then take any event x of X that is identical to a candidate c with $s_{cp} = 1$ and shift it to p (where the unique existence of such c is guaranteed from (3) and the existence of such x from (4)),
(ii) if $\iota_p = 1$ and $\sigma_p = 0$, then p is inserted (in this case, there is no c with $s_{cp} = 1$ from (3)),

and the events in X that have no `shift` destination after the above procedure are deleted. Thus, the resulting transformation is simple.

The number of events that are deleted in the transformation from X into P_h constructed above is given by

$$\sum_{c \in \mathcal{C}} \left[m_c - \sum_{p \in P_h} s_{cp} \right],$$

thus the second term in the objective function coincides with the total cost concerning `delete`. The first and third terms in the objective function coincide with the total costs concerning `insert` and `shift`, respectively. Therefore, the value of the objective function (when minimized) coincides with $f(X)$ for any X that is represented by the m variables.

Theorem 1 *The binary linear programming problem (P) correctly formulates the median computation.*

Proof Straightforward from the discussion so far. □

Remark 1 One can replace the constraint of $m_c \in \mathbb{N}$ by $m_c \in \mathbb{R}$ because m_c is minimized in the objective function and the value of m_c is bounded from below by an integer from Eq. (4).

5 Numerical Experiments

In this section, we report numerical results. We used Python to implement the formulation and ran it on the Intel Core i7-1165G7@ 2.80 GHz with 16.0 GB using optimization software, Gurobi ver. 10.0.0.

We consider two settings. The first setting evaluates the computation time required for the formulation to solve instances to optimality where randomly generated data are employed. The second setting demonstrates the effective use of median in predicting the behavior of earthquakes where medoids are used as a benchmark.

5.1 Computation Time for Randomly Generated Data

Table 1 shows the results. Here, we consider 18 cases by changing the values for the dimension of the events, d, and the number of MPPs in an instance, k, which correspond to the rows. For $d = 1$, events have no marks while for $d = 3$, each event has a two-dimensional vector as its mark. For each case, ten instances are generated in the following way:

Table 1 Computation time required for randomly generated instances

d	k	Computation time (s)			$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{P}\rvert \cdot \lvert\mathcal{C}\rvert$
		Min	Avg	Max	Avg	Max
1	5	< 0.1	< 0.1	< 0.1	57	5000
	10	< 0.1	0.1	0.1	97	10,000
	20	0.1	0.2	0.2	223	20,000
	40	0.3	0.4	0.5	426	40,000
	80	0.8	0.9	1.1	872	80,000
	160	2.0	2.1	2.3	1728	160,000
2	5	0.1	0.1	0.2	54	25,000
	10	0.2	0.3	0.3	93	50,000
	20	0.6	1.2	2.5	195	100,000
	40	9.2	16.8	40.9	424	200,000
	80	7.4	294.1	814.5	807	400,000
	160	25.5	593.4	2060.5	1757	800,000
3	5	0.5	1.2	3.9	52	125,000
	10	1.0	5.3	11.4	101	250,000
	20	8.0	70.5	411.8	215	500,000
	40	300.4	NA	> 3600.0	415	1,000,000
	80	–	–	–	–	–
	160	–	–	–	–	–

- The first component, i.e., time of occurrence, of every event is selected from $\{0, 1, \ldots, 49\}$ uniformly at random,
- If $d \geq 2$, then the jth component for $j \geq 2$, i.e., a mark, of every event is selected from $\{1, \ldots, 5\}$ at uniformly random, meaning that there are 5 (25) possibilities for the mark vectors if $d = 2$ ($d = 3$), and
- The number of events in each MPP is selected from $\{1, \ldots, 20\}$ uniformly at random, meaning that it is 10.5 on average and hence the total number of events in the input, i.e., $|\mathcal{P}|$ is $10.5k$ on average and at most $20k$.

Each component of the events is normalized to be contained in $[0, 1]$ and λ_1 and λ_2 are set to one. The minimum, average, and maximum of the computation time for the ten instances are shown for each case.

We see from Table 1 that for each fixed d, the computation time grows as $|\mathcal{P}|$ increases. However, for $d = 1, 2$, instances with more than a thousand events are solved within 10 min on average. On the other hand, for $d = 3$, we often failed to solve an instance within an hour even for $k = 40$, and hence the results for larger instances are omitted. One reason is the fact that the number of the s variables is $|\mathcal{C}| \cdot |\mathcal{P}|$ and hence can be $50 \cdot 5^{d-1} \cdot 20k$ for the instances we tested, which is 1,000,000 for $d = 3$ and $k = 40$ as shown in the rightmost column. Here, note, in particular, that even when each component of the events is binary, we have $|\mathcal{C}| = O(2^d)$.

Nevertheless, we confirmed that the gap between upper and lower bounds in each computation tends to get small at early stages, which may suggest that the proposed

Table 2 Comparison with the naive sampling approach for ten instances for $d = 2$ and $k = 10$ where n denotes the number of samples

No	Objective value					Computation time (s)		
	Ours	Naive				Ours	Naive	
			$n = 10$		$n = 100$		$n = 10$	$n = 100$
1	53.96	57.94	(7.4%)	56.68	(5.0%)	0.43	0.23	2.12
2	63.10	64.42	(2.1%)	63.88	(1.2%)	0.24	0.21	1.90
3	41.40	42.78	(3.3%)	42.10	(1.7%)	0.21	0.20	1.73
4	56.98	59.60	(4.6%)	58.86	(3.3%)	0.26	0.19	1.89
5	45.80	47.22	(3.1%)	46.46	(1.4%)	0.16	0.17	1.65
6	44.84	46.34	(3.3%)	45.62	(1.7%)	0.20	0.19	1.79
7	63.04	64.28	(2.0%)	63.72	(1.1%)	0.27	0.20	1.96
8	59.96	60.90	(1.6%)	60.36	(0.7%)	0.43	0.20	1.86
9	40.74	41.68	(2.3%)	41.68	(2.3%)	0.18	0.18	1.73
10	43.88	45.62	(4.0%)	45.62	(4.0%)	0.18	0.17	1.63
Avg			(3.7%)		(2.2%)	0.26	0.19	1.83

formulation can be used in heuristic approaches. For example for an instance with $d = 3$ and $k = 40$, a solution with a guaranteed gap 1.50% is found within a minute. The average number of events involved in a median was approximately 10.5.

Remark 2 It is possible to impose bounds on the number of events if interested in a median with as few events as possible, or use $\mathcal{P}$ instead of $\mathcal{C}$, which substantially reduces the number of the s variables, if interested in approximating the median with events in a given input (although in both cases, we cannot guarantee that results are medians in the exact sense).

Finally, we look at a brute-force algorithm mentioned in [32]; see Sect. 3.2.2 of that paper for a heuristic method. This algorithm investigates all the $\binom{|\mathcal{C}|}{M}$ possibilities for a fixed M, and hence its computation time is much longer than that of ours in general although it is an exact method if ran for several promising values of M. Note that even for an instance with $|\mathcal{C}| = 43$, there are approximately 1.9×10^9 possibilities when $M = 10$ for example.

A naive approach for overcoming this drawback would be to sample, say, n, solutions from all the possibilities, and choose one that minimizes f among them. The comparison results for the ten instances for $d = 2$ and $k = 10$ are summarized in Table 2 where we set M to 10, the expected number of events in an MPP. The objective values are the values of f. In the brackets, we show relative errors. When $n = 10$, the naive one runs a little bit faster than ours but the relative error can be 7.4% in the worst case and when $n = 100$, the naive one is much slower than ours and there is no significant improvement on the worst-case relative error as well as the average relative error.

5.2 Application in Earthquake Prediction

We apply our method to analyze MPPs with one-dimensional marks adapted from real-world earthquake data. The task is to predict the days and magnitudes of earthquake occurrences of magnitude 5.0 or more. The datasets generated and/or analyzed during the current study are available from Japan Meteorological Agency.[1]

We focused on the earthquakes occurring in the period from January 1, 2000 to December 31, 2017, within the region of longitude 140°E–149°E and latitude 36°N–42°N, the area in the vicinity of the Tohoku Oki Earthquake of 2011, covering the points where the Pacific Plate and the North American Plate overlap. The time unit of events was set to 1 day. For each day, we have an event for each earthquake whose magnitude is 5.0 or greater in our preprocessed dataset. For each such event, we associate its magnitude as the one-dimensional mark. As in the previous experiment, each component of the events is normalized to be contained in [0, 1].

From this data, we created 216 MPPs by splitting the 18 year time period into time windows of 1 month each. Thus, each MPP corresponds to 1 month, and the number of events in each MPP is at most 31. We also allowed processes with zero events, as they stand for months in which no earthquakes occurred.

5.2.1 Problem Setting

First, we divided the set of MPPs obtained above into two parts. The data of 2000/1/1–2011/12/31 was treated as training data, and that of 2012/1/1–2017/12/31 as test data. For each MPP in the test data, we predict the pattern of earthquakes occurring in the following month by a procedure described below where k is a fixed parameter:

`Procedure`(k):

Step 1. For a given MPP, choose the k closest MPPs (with respect to the VP distance) from the training data.

Step 2. Extract the set of MPPs corresponding to the following month of the k MPPs chosen in Step 1, and compute their median. This median serves as the prediction for the following month.

Step 3. Compute VP distance between the MPP of the following month in the test data and its prediction computed in Step 2.

For example, suppose that we are to use the MPP of January 2012 to predict that of February 2012 when $k = 3$. In Step 1, we compute VP distance between the MPP of January 2012 and all MPPs in the training data, and choose the 3 MPPs having the smallest distances. Suppose that they are of June 2003, November 2008, and May 2001. Then, in Step 2 we extract the MPPs of July 2003, December 2008 and June 2001, and compute their median. Finally in Step 3, we compute VP distance between this median, and the MPP of February 2012.

[1] https://www.jma.go.jp/jma/en/menu.html.

Table 3 Prediction accuracy and the total computation time of `Procedure` (k) for $k = 2, \ldots, 10$

k	Prediction accuracy			Computation time (s)		
	$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 2.0$	$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 2.0$
1	1.000	1.000	1.000	128	116	126
2	0.635	0.587	0.718	129	114	126
3	0.565	0.539	0.645	130	115	127
4	0.503	0.512	0.591	131	115	127
5	0.478	0.488	0.590	132	116	128
6	0.526	0.506	0.574	132	117	130
7	0.518	0.504	0.571	133	118	131
8	0.517	0.497	0.567	136	120	133
9	0.514	0.498	0.575	137	131	135
10	0.526	0.500	0.573	139	124	136

5.2.2 Results

We evaluate the prediction accuracy of `Procedure` (k) in terms of the total sum of VP distances computed in Step 3, i.e., the gap between the actual MPPs in the test data and those predicted by the procedure. The benchmark is `Procedure` (1) because it corresponds to the method proposed in [13] (see also [11] for a similar method used for exchange forecasts). For this reason, we look at, for $k \geq 2$, its (relative) prediction accuracy defined as

$$\frac{\text{The prediction accuracy of } \texttt{Procedure}(k)}{\text{The prediction accuracy of } \texttt{Procedure}(1)}.$$

For example, if this value is 0.5, then it means that the proposed procedure brings 50% improvement over the alternative one proposed in [13].

The results are summarized in Table 3 where we applied the procedure to all MPPs from January 2012 to November 2017. December 2017 which has no data for the following month was excluded from the analysis, thus the number of data used in prediction was 71. The computation time shown in the table is the total computation time required for dealing with all these 71 instances. We varied the value of k as $k = 1, 2, \ldots, 10$ and that of parameters for the cost of `shift` λ_1 and λ_2 as $\lambda_1 = \lambda_2 = 0.5, 1.0, 2.0$.

We see from Table 3 that large values of k do not necessarily lead to good predictions. This is because if k is unnecessarily large, then the set considered in Step 2 can include MPPs with large VP distances, which can drag down the precision of the computed median. The best prediction accuracy for cases with $\lambda = 0.5, 1.0, 2.0$is attained at $k = 5, 5, 8$, respectively, which are indicated by underlines. We also observe that the computation time is not so sensitive to the changes in k as well as λ. We note that the sizes of the instances we solved here are at most those of the instances tested for $d = 2$ and $k = 10$ in the previous section, and hence the computation here was very quick.

6 Concluding Remarks

In this paper, we considered the problem of computing a median of marked point processes data under the edit distance, originated by Victor and Purpura in 1997. Our main result is the binary linear programming formulation for this problem. Numerical results showed that through our formulation, medians of thousands of events can be computed in reasonable time by making use of the software Gurobi. This study may shed light on the value of optimization approaches in analyzing marked point process data; see, e.g., [3, 4, 17, 23] for such attempts for other data analyses. In contrast to the current study, developing efficient heuristic algorithms is also an important challenge.

Acknowledgements The authors are very grateful to the two anonymous reviewers for their instructive comments and suggestions. The research of NS is partially supported by Grant-in-Aids for Scientific Research (no. JP20K19747). We thank the Japan Meteorological Agency for providing the earthquake dataset used in this study.

References

1. Aronov, D.: Fast algorithm for the metric-space analysis of simultaneous responses of multiple single neurons. J. Neurosci. Methods **124**(2), 175–179 (2003)
2. Banerjee, A., Goswami, B., Hirata, Y., Eroglu, D., Merz, B., Kruths, J., Marwan, R.: Recurrence analysis of extreme event-like data. Nonlinear Process. Geophys. **28**(1), 213–229 (2021)
3. Bertsimas, D., Dunn, J.: Optimal classification trees. Mach. Learn. **106**(7), 1039–1082 (2017)
4. Bertsimas, D., King, A., Mazumder, R.: Best subset selection via a modern optimization lens. Ann. Stat. **44**(2), 813–852 (2016)
5. Chizat, L., Roussillon, P., Léger, F., Vialard, F.-X., Peyré, G.: Faster Wasserstein distance estimation with the Sinkhorn divergence. Adv. Neural Inf. Process. Syst. **33**, 2257–2269 (2020)
6. Descombes, X., Zerubia, J.: Marked point process in image analysis. IEEE Signal Process. Mag. **19**(5), 77–84 (2002)
7. Diez, D.M., Schoenberg, F.P., Woody, C.D.: Algorithms for computing spike time distance and point process prototypes with application to feline neuronal responses to acoustic stimuli. J. Neurosci. Methods **203**(1), 186–192 (2012)
8. Eroglu, D., McRobie, F.H., Ozken, I., Stemler, T., Wyrwoll, K.-H., Breitenbach, S.F., Marwan, N., Kurths, J.: See-saw relationship of the Holocene East Asian-Australian summer monsoon. Nat. Commun. **7**(1), 1–7 (2016)
9. Gottschlich, C., Schuhmacher, D.: The shortlist method for fast computation of the earth mover's distance and finding optimal solutions to transportation problems. PLOS ONE **9**(10), e110214 (2014)
10. Heinemann, F., Klatt, M., Munk, A.: Kantorovich–Rubinstein distance and barycenter for finitely supported measures: foundations and algorithms. Appl. Math. Optim. **87**(1), 4 (2023)
11. Hirata, Y., Aihara, K.: Timing matters in foreign exchange markets. Physica A **391**(3), 760–766 (2012)
12. Hirata, Y., Aihara, K.: Edit distance for marked point processes revisited: an implementation by binary integer programming. Chaos Interdiscip J Nonlinear Sci **25**(12), 123117 (2015)

13. Hirata, Y., Iwayama, K., Aihara, K.: Possibility of short-term probabilistic forecasts for large earthquakes making good use of the limitations of existing catalogs. Phys. Rev. E **94**(4), 042217 (2016)
14. Hirata, Y., Sukegawa, N.: Two efficient calculations of edit distance between marked point processes. Chaos Interdiscip. J. Nonlinear Sci. **29**(10), 101107 (2019)
15. Holden, L., Sannan, S., Bungum, H.: A stochastic marked point process model for earthquakes. Nat. Hazards Earth Syst. Sci. **3**(1/2), 95–101 (2003)
16. Junqueira Saldanha, M.H., Hirata, Y.: Solar activity facilitates daily forecasts of large earthquakes. Chaos **32**(6), 061107 (2022)
17. Lee, I.G., Yoon, S.W., Won, D.: A mixed integer linear programming support vector machine for cost-effective group feature selection: branch-cut-and-price approach. Eur. J. Oper. Res. **299**(3), 1055–1068 (2022)
18. Li, W., Ryu, E.K., Osher, S., Yin, W., Gangbo, W.: A parallel method for earth mover's distance. J. Sci. Comput. **75**(1), 182–197 (2018)
19. Ling, H., Okada, K.: An efficient earth mover's distance algorithm for robust histogram comparison. IEEE Trans. Pattern Anal. Mach. Intell. **29**(5), 840–853 (2007)
20. Mallet, C., Lafarge, F., Roux, M., Soergel, U., Bretar, F., Heipke, C.: A marked point process for modeling lidar waveforms. IEEE Trans. Image Process. **19**(12), 3204–3221 (2010)
21. Marwan, R.: Challenges and perspectives in recurrence analyses of event time series. Front. Appl. Math. Stat. **32**(4), 420–429 (2023)
22. Mateu, J., Schoenberg, F.P., Diez, D.M., González, J.A., Lu, W.: On measures of dissimilarity between point patterns: classification based on prototypes and multidimensional scaling. Biom. J. **57**(2), 340–358 (2015)
23. Miyashiro, R., Takano, Y.: Mixed integer second-order cone programming formulations for variable selection in linear regression. Eur. J. Oper. Res. **247**(3), 721–731 (2015)
24. Mohler, G.: Marked point process hotspot maps for homicide and gun crime prediction in Chicago. Int. J. Forecast. **30**(3), 491–497 (2014)
25. Müller, R., Schuhmacher, D., Mateu, J.: Metrics and barycenters for point pattern data. Stat. Comput. **30**(4), 953–972 (2020)
26. Nakano, S., Hirata, Y., Iwayama, K., Aihara, K.: Intra-day response of foreign exchange markets after the Tohoku-Oki earthquake. Physica A **419**(1), 203–214 (2015)
27. Nichols, K., Schoenberg, F.P., Keeley, J.E., Bray, A., Diez, D.: The application of prototype point processes for the summary and description of California wildfires. J. Time Ser. Anal. **32**(4), 420–429 (2011)
28. Ozken, I., Eroglu, D., Stemler, T., Marwan, N., Bagci, G.B., Kurths, J.: Transformation-cost time-series method for analyzing irregularly sampled data. Phys. Rev. E **91**(6), 062911 (2015)
29. Prigent, J.-L.: Option pricing with a general marked point process. Math. Oper. Res. **26**(1), 50–66 (2001)
30. Schoenberg, F.P., Tranbarger, K.E.: Description of earthquake aftershock sequences using prototype point patterns. Environ. Off. J. Int. Environ. Soc. **19**(3), 271–286 (2008)
31. Suzuki, S., Hirata, Y., Aihara, K.: Definition of distance for marked point process data and its application to recurrence plot-based analysis of exchange tick data of foreign currencies. Int. J. Bifurc. Chaos **20**(11), 3699–3708 (2010)
32. Tranbarger, K.E.: Point process prototypes, and other applications of point pattern distance metrics. PhD Thesis, University of California, Los Angeles (2005)
33. Tranbarger Freier, K.E., Schoenberg, F.P.: On the computation and application of prototype point patterns. Open Appl. Inform. J. **4**, 1 (2010)
34. Vallender, S.: Calculation of the Wasserstein distance between probability distributions on the line. Theory Probab. Appl. **18**(4), 784–786 (1974)
35. Victor, J.D., Purpura, K.P.: Metric-space analysis of spike trains: theory, algorithms and application. Netw. Comput. Neural Syst. **8**(2), 127–164 (1997)
36. Xie, Y., Wang, X., Wang, R., Zha, H. A fast proximal point method for computing exact Wasserstein distance. In: Uncertainty in Artificial Intelligence, pp. 433–453. PMLR (2020)