A Scalable Optimization Algorithm for Solving the Beltway and Turnpike Problems with Uncertain Measurements

C. S. Elder[1], Minh Hoang[2], Mohsen Ferdosi[1], and Carl Kingsford[1(✉)]

[1] Ray and Stephanie Lane Computational Biology Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA
{celder,mferdosi,carlk}@cs.cmu.edu
[2] Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA
qhoang@andrew.cmu.edu

Abstract. The Beltway and Turnpike problems entail the reconstruction of circular and linear one-dimensional point sets from unordered pairwise distances. These problems arise in computational biology when the measurements provide distances but do not associate those distances with the entities that gave rise to them. Such applications include molecular structure determination, genomic sequencing, tandem mass spectrometry, and molecular error-correcting codes (since sequencing and mass spec technologies can give lengths or weights, usually without connecting them to endpoints). Practical algorithms for Turnpike are known when the distance measurements are accurate, but both problems become strongly NP-hard under any level of measurement uncertainty. This is problematic since all known applications experience some degree of uncertainty from uncontrollable factors. Traditional algorithms cope with this complexity by exploring a much larger solution space, leading to exponential blowup in terms of both time and space. To alleviate both issues, we propose a novel alternating optimization algorithm that can scale to large, uncertain distance sets with as many as 100,000 points. This algorithm is space and time-efficient, with each step running in $\mathcal{O}(m \log(m))$ time and requiring only $\mathcal{O}(\sqrt{m})$ working space for a distance set of size m. Evaluations of this approach on synthetic and partial digest data showcase improved accuracy and scalability in the presence of uncertain, duplicated, and missing distances. Our implementation of the algorithm is available at https://github.com/Kingsford-Group/turnpikesolvermm.

Keywords: Inverse Problem · Optimization · Beltway · Turnpike

1 Introduction

The Turnpike problem is to reconstruct n unknown points on a line from all $\binom{n}{2}$ pairwise distances between them provided without labels. The Beltway problem is a variant where the points lie on a circle instead of a line. These problems

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024
J. Ma (Ed.): RECOMB 2024, LNCS 14758, pp. 201–216, 2024.
https://doi.org/10.1007/978-1-0716-3989-4_13

arise frequently in computational biology applications when the measurement (e.g. mass spectrometry or sequencing) provides distances but does not associate those distances with the entities that gave rise to them. In the tandem mass spec application, for example, when sequencing an unknown peptide $a_1 a_2 \dots a_n$, the "points" are the weights of the fragments $a_i, a_i a_{i+1}, a_i a_{i+1} a_{i+2}, \dots$ — reconstructing those points would suggest the identity of each amino acid a_i via its weight. However, the MS/MS measurement provides the weights of every fragment $a_i \dots a_j$ (which can be treated as "distances" between points i and j) without associating that measurement with i or j. Various heuristics for this problem are applied to structure estimation of biomolecules [13], *de novo* sequencing of linear and cyclic peptides [9,17], and reconstructing DNA sequences from their partially digested fragments [20,22]. Some versions of the problem provide additional labeling information and reduced distance subsets, such as the labeled partial digest problem that separates the endpoint distances from the all-pairs distance set and the simplified partial digest problem [4] that returns only a subset of the distances. Other variants of the Turnpike problem are applied to quantum phase estimation [26] and molecular error-correcting codes used for databases [11].

The Exact Turnpike variant of the problem, where all distances are observed without error, can be solved exactly via a backtracking algorithm that alternates between placing the largest remaining distance and matching derived and ground truth distances [21]. While there exist pathological cases for which it incurs exponential runtimes [25], this algorithm is generally efficient in practice with an expected run time of $\mathcal{O}(n^2 \log n)$ for random instances [20]. Various extensions to the backtracking algorithm have been proposed to improve both expected and worst case runtimes, including a variant based on breadth-first search [1], that are empirically faster. One extension efficiently solves known pathological instances [18]. In contrast, algorithms for Beltway are not efficient, with a worst case runtime of $\mathcal{O}(n^n \log n)$ that is often realized in practice [9]. Other approaches to Turnpike include a fixed-parameter tractable algorithm that works by factoring a polynomial and scales with the largest distance in the set [14]. Regrettably, this approach is highly susceptible to numerical precision errors [15]. This restricts its practical applicability whenever floating point arithmetic is used. A semidefinite relaxation also exists that is able to solve some instances but is known to be numerically unstable and suffer from runtimes far exceeding the backtracking algorithm in practice [13].

When the distance measurements are uncertain, we have the Noisy Turnpike and Noisy Beltway problems, which are both strongly NP-complete as demonstrated by a reduction from the three-partition problem [6]. Skiena et al. modified the backtracking approach to use intervals instead of points to accommodate measurement uncertainty [20], but these modifications lead to the consideration of exponentially many paths, limiting the algorithm's efficiency and practical applicability [13]. Pandurangan et al. assumed that the partial digestion results from both ends of the double stranded DNA sample are observed [19]. Fomin et al. performed the equivalent modifications for Noisy Beltway and

reduce the running time by removing redundant measurements, but as with the exact case, only very small Noisy Beltway instances can be solved with this algorithm [10]. More recently, Huang et al. model both the Noisy Turnpike and Noisy Beltway problems as probabilistic inference of the point assignments using discrete bins that quantize the input domain [13]. In this approach, the bin size is set to be smaller than the smallest distance, and hence it was assumed that no bin can contain more than one point. However, this only holds true when the observation error is sufficiently small in magnitude relative to the smallest distances. As such, the accuracy of this algorithm deteriorates in noisier instances. In addition, it also struggles to efficiently solve larger problem instances, with $n > 500$ out of reach at present.

In Sect. 2, we propose a novel approach to solving the Noisy Turnpike and Noisy Beltway problems using a bilevel optimization scheme that alternates between estimating the point-distance correspondence and recovering the original point set given this assignment. Our formulation's non-convex optimization landscape contains many saddle points and local optima. We accommodate for this by introducing a divide-and-conquer step to recursively correct small-scale mistakes that lead to low-quality solutions. Our algorithm runs in time $\mathcal{O}(n^2 \log n)$ for each step, with time dominated by a low-cost sorting step.

In Sect. 3, we empirically demonstrate the performance of our Minorization-Maximization algorithm (MM) in various synthetic and realistic biological settings, such as the partial digestion task [13]. Our algorithm arrives at highly accurate solutions even in extremely noisy observation conditions. We also demonstrate that the proposed algorithm runs more efficiently than previous approaches and empirically matches our theoretical runtime expectation. Most notably, the proposed algorithm can efficiently process partial digestion instances with up to a hundred thousand digested fragments, which is realistically on the scale of a whole genome and has never been achieved by previous methods. Moreover, we provide an extension of the method in Appendix C [8] to problem variants that provide additional labeling information and reduced distance sets. In summary, this algorithm advances the capacity to address both the Noisy Turnpike and Noisy Beltway problems, and thereby improves the accuracy and scalability of various biological tasks that make use of these formulations.

2 Method

2.1 Problem Setting

Let $m = n(n-1)/2$ and $D \in \mathbb{R}^m$ be a vector of pairwise distances between n points. We denote the ground truth vector containing the points to be recovered as $z \in \mathbb{R}^n$. Without loss of generality, we assume that $z_1 \leq \cdots \leq z_n$, $\sum_{k=1}^{n} z_k = 0$, and $\|z\|_2 = 1$; that is, the unknown points are named in sorted order, centered around zero, and have unit norm. These assumptions do not fundamentally change the problem, but are nonetheless important as they prevent trivial non-uniqueness. The first and second assumptions hold because the distance set is invariant to translation and permutation of the points, allowing

us to look for a centered, sorted solution vector z. The third assumption follows because we can construct a scaled distance set $\bar{D} = \sqrt{n}\,\|D\|_2^{-1}\,D$ from the original distances that generates $\bar{z} = z/\|z\|_2$, using the fact that z is centered:

$$\|D\|_2^2 = \sum_{i \le j}^{n} (z_j - z_i)^2 = n\|z\|_2^2 + \sum_{i,\,j}^{n} z_i z_j = n\|z\|_2^2 + \left(\sum_{k=0}^{n} z_k\right)^2 = n\|z\|_2^2 \,.$$

Let $\mathcal{Z}$ be the set of vectors in $\mathbb{R}^n$ that satisfy all three assumptions above (treating dimensions of vectors in $\mathcal{Z}$ as the point locations), and let $\mathcal{S}_m$ denote the set of all permutation matrices of m items. We use Turnpike to refer to both the Exact Turnpike and Noisy Turnpike variants, when statements apply to both. The Exact Turnpike problem is formalized as finding a vector $\hat{z} \in \mathcal{Z}$ such that $Q\hat{z} = PD$ for some $P \in \mathcal{S}_m$, and where $Q \in \mathbb{R}^{m \times n}$ is a fixed incidence matrix defined as follows. Each row in Q corresponds to a pair of indices $j > i$. For convenience, we let the function $\alpha(i, j)$ map the index pair (i, j) to its (arbitrary) row index in Q. The incidence matrix Q is constructed such that $Q_{\alpha(i,j),j} = 1$ and $Q_{\alpha(i,j),i} = -1$ are the only non-zero entries in $Q_{\alpha(i,j)}$. It follows that $Q_{\alpha(i,j)}\hat{z} = \hat{z}_j - \hat{z}_i$, and $Q\hat{z}$ contains all the pairwise distances generated by $\hat{z}$. Furthermore, if $\hat{z}$ recovers the ground truth z, then $Q\hat{z}$ must also be a permutation of D, which explains the role of P in the objective above. In the Noisy Turnpike case, which is the focus of this paper, exact recovery is not possible in general due to the corrupted observations. Therefore, the objective can be written in the form of an optimization task:

$$\hat{z} = \underset{z' \in \mathcal{Z}}{\operatorname{argmax}} \max_{P \in \mathcal{S}_m} \langle Qz', PD \rangle, \tag{1}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors. This is equivalent to minimizing the ℓ_2 distance between Qz' and PD because the norm of P, Q and D are constant, z' is normalized based on our previous assumptions, and optimality in the exact case will take place when $Qz' = PD$.

2.2 Minorization-Maximization Scheme for Solving Turnpike

We first observe that Eq. (1) is bilinear because fixing either P or z reduces the problem to a linear function. This motivates a bilevel minorization-maximization (MM) scheme [23] to optimize this objective. In particular, we relax our objective into two alternating subproblems. At iteration $t + 1$, for $t > 0$, the first subproblem fixes an estimation z^t and solves for

$$P_t = \underset{P \,\in\, S_m}{\operatorname{argmax}} \langle Qz^t, PD \rangle, \tag{2}$$

which has a closed-form solution, as shown in Proposition 1 below. This closed form is a variant of the rearrangement inequality [12] used in Lemma 1.

Lemma 1. *Let $y = \langle y_1, y_2, \ldots, y_n \rangle \in \mathbb{R}^n$ such that $y_1 \le y_2 \le \ldots \le y_n$. The objective $\underset{P \in S_m}{\operatorname{argmax}} \langle Px, y \rangle$ is solved by P_x such that $P_x x$ is in sorted order.*

Algorithm 1. Minorization-Maximization (MM)

Input: Distance vector D, initial estimate z^0, tolerance $\epsilon > 0$

$t \leftarrow 0$
$D \leftarrow D^{\uparrow}$ ▷ Replace D with its sorted equivalent $D^{\uparrow}$
while not converged **do**
  $P_t \leftarrow \Pi_{Qz^t}^{\top}$ ▷ Calculate and sort Qz^t using Alg. 2
  $z^{t+1} \leftarrow Q^{\top} P_t D$ ▷ Estimate the next point vector
  $t \leftarrow t + 1$
  converged $\leftarrow \|z^{t+1} - z^t\|_2 < \epsilon$
end while
return unit(z^t)

Proof. Let $P \in \mathcal{S}_m$ be a non-sorting permutation, meaning there exist indices $i < j$ where $(Px)_i > (Px)_j$. Notice that transposing elements i and j increases the objective because

$$((Px)_j - (Px)_i)(y_j - y_i) \leq 0 \implies (Px)_j y_j + (Px)_i y_i \leq (Px)_i y_j + (Px)_j y_i.$$

Thus the permutation that first applies P then transposes elements i and j is no worse than P. Iterating this argument leads to a sorting permutation that is also no worse than P. As the initial permutation was arbitrary, this shows that there exists a globally maximizing permutation that sorts x. To finish the proof, notice that any two sorting permutations P_1 and P_2 must have the same objective value since sortedness implies $P_1 x = P_2 x$. □

Proposition 1. *Let $\Pi^{\top}$ be a permutation that puts Qz into sorted order. The permutation Π is a globally maximizing solution to Eq. (2).*

Proof. Without loss of generality, we assume that D is sorted as a preprocessing step to the Noisy Turnpike problem. We can rewrite the expression in Eq. (2) as $\langle Qz^t, PD \rangle = \langle P^{\top}(Qz^t), D \rangle$. By Lemma 1, any permutation $\Pi^{\top}$ that sorts Qz must be a global maximizer of the right-hand side. Notice the transpose Π is the equivalent solution on the left-hand side, proving the claim. □

On the other hand, the second subproblem fixes an estimation for P_t, which is the closed-form solution derived above, and solves for:

$$z^{t+1} = \underset{\hat{z} \in \mathcal{Z}}{\operatorname{argmax}} \ \langle Q\hat{z}, P_t D \rangle \ \equiv \underset{\hat{z} \in \mathcal{Z}}{\operatorname{argmax}} \langle \hat{z}, Q^{\top} P_t D \rangle \ . \tag{3}$$

Since the inner product of two vectors is maximized when they are parallel and $\|\hat{z}\|_2 = 1$ by assumption, the maximum objective value is obtained when $\hat{z} = \text{unit}(Q^{\top} P_t D)$, where unit$(\cdot)$ scales a vector to unit norm.

As objectives (2) and (3) have closed-form solutions, they motivate a practical bilevel optimization routine described in Algorithm 1. Note that the unit projection does not affect the permutation in the next iteration, so we omit it until the vector is returned. We avoid storing both the incidence matrix and intermediate

distance vector by using implicit matrix multiplication and a problem-specific matching algorithm, Algorithm 2. The runtime of the optimization inner loop is derived in Proposition 2. In the same proposition, we also derive a memory efficient implementation that avoids storing intermediate values during optimization.

Lemma 2. *The priority queue in Algorithm 2 uses z^t interval order, i.e., $(i,j) \leq (i',j') \iff (z^t[j] - z^t[i]) \leq (z^t[j'] - z^t[i'])$. This ordering satisfies $i \leq i'$, $j' \leq j$ and implies $(i,j) \leq (i',j')$ when z^t is sorted.*

Proof. For $i \leq i'$ and $j' \leq j$, we have $z^t[i] \leq z^t[i']$ and $z^t[j'] \leq z^t[j]$, which implies that $z^t[j'] - z^t[i'] \leq z^t[j] - z^t[i]$. This is the definition of $(i',j') \leq (i,j)$. □

Proposition 2. *Upon termination of Algorithm 2, the vector z^{t+1} contains $Q^\top \Pi^\top_{Qz^t} D$. Moreover, the algorithm runs in $O(n^2 \log n)$ time and uses only $\mathcal{O}(n)$ nonnegative integers for non-constant storage, where n is the number of points.*

Proof. We first prove that the priority queue pops the t^{th} smallest distance during iteration t. To that end, we define the sequences $I^k = (k, k+t)_{t=1}^{n-1}$. We note that the sequences $I^1, \ldots, I^{n-1}$ partition the $\binom{n}{2}$ possible intervals. Lemma 2 establishes that these chains are in interval sorted order, and thus implies Algorithm 2 produces the smallest unseen interval at each iteration. This holds because the queue holds the smallest element from each sequence, and we add the next one until each sequence has been exhausted.

By the discussion above, during iteration t, (i,j) is the (potentially non-unique) t^{th} smallest interval. Thus the sorting permutation will send interval (i,j) to index t, and its transpose (i.e., inverse) will send index t to interval (i,j), implying $D[t]$ will be used as the (i,j) distance. When multiplying by $Q^\top$, the (i,j) distance entry contributes only to the i^{th} and j^{th} points. Specifically, the (i,j) entry is subtracted from $z^{t+1}[i]$ and added to $z^{t+1}[j]$, which is immediately performed in Algorithm 2's loop. Thus the algorithm terminates with $z^{t+1} = Q^\top P_t D$ since we initially zero it out and accumulate all the entries that contribute to it.

The algorithm performs $\mathcal{O}(n \log n)$ work before the main loop, which takes $\mathcal{O}(n^2 \log n)$ time because the priority queue takes $\mathcal{O}(\log n)$ time per iteration using standard implementations. This is unaffected by the constant-time interval comparison function. The priority queue uses the only non-constant memory, as it needs to store $\mathcal{O}(n)$ non-negative integers for the intervals. □

Proposition 3. *The outer loop of Algorithm 1 terminates within a finite number of steps. The inner-loop takes $\mathcal{O}(n^2 \log n)$ time and $\mathcal{O}(n)$ non-constant storage.*

Proof. For the first claim, notice that the sorting permutation in each iteration fully decides the point vector that is produced in the next step, meaning the set of possible output vectors is finite. We also know that the permutation and point

vector must improve the objective at each step. This means the algorithm cannot continue to make progress indefinitely and will terminate when $z^t = z^{t+1}$.

For the second claim, notice that steps 4 and 5 take $O(n^2 \log n)$ time and need only $\mathcal{O}(n)$ non-negative integers of non-constant storage by the result of Proposition 2. We only need to keep two n-dimensional floating point vectors to calculate step 7. □

In practice, we observed that Algorithm 1 converges quickly but is prone to becoming trapped in local maxima. To prevent this, we further propose a divide-and-conquer heuristic formally described in Algorithm 3. In particular, after each pass of Algorithm 1, we partition the estimation $\hat{z}$ into non-overlapping subsets $\hat{z}_l$ and $\hat{z}_r$. In our implementation, the median is used to form the partitions, but any rule works with this framework. No matter the choice, this segments the distance set D into three portions: (a) D_{ll} contains the distances among points in $\hat{z}_l$; (b) D_{rr} contains the distances among points in $\hat{z}_r$; and (c) D_{lr} contains the distances between pairs of points respectively in $\hat{z}_l$ and $\hat{z}_r$. Even though we do not have the ground truth assignment of the point-distance correspondence, we can use the estimated permutation matrix P to perform this segmentation.

Intuitively, if $\hat{z}$ and P are the optimal Turnpike solution, then subsequent applications of Algorithm 1 on (z_l, D_{ll}) and (z_r, D_{rr}) will not alter this solution. Otherwise, the recursive sub-routines will likely not get trapped in the same local maxima as the parent routine and will serve as a self-correcting mechanism for Algorithm 1 by returning the adjusted permutations P_l and P_r. At this point, we can adjust $\hat{z}_l$ and $\hat{z}_r$ by solving the following regression tasks:

$$\hat{z}_l^+ = \operatorname*{argmin}_{z_l'} \|Q_l z_l' - P_l D_{ll}\| \qquad \text{and} \qquad \hat{z}_r^+ = \operatorname*{argmin}_{z_r'} \|Q_r z_r' - P_r D_{rr}\|,$$

where Q_l and Q_r are the respective incidence submatrices corresponding to $\hat{z}_l$ and $\hat{z}_r$. To avoid storing the incidence matrices, we can use any matrix-free solver such as the conjugate gradient method [24] (see Alg. 4 in Appendix A [8] for the matrix-free oracles). As the adjusted estimation $\hat{z}^+ = (\hat{z}_l^+, \hat{z}_r^+)$ breaks away from potential local maxima, this routine is repeated until convergence, as described in Algorithm 3 below. We provide a visualization of how this improves solutions in Appendix B [8].

Remark. An alternative approach (which will be compared against our method in Sect. 3) to solving Eq. (1) applies Birkhoff's theorem [3], which states that the polytope $\mathcal{B}_m$ of $m \times m$ doubly stochastic matrices is the convex hull of $\mathcal{S}_m$. This motivates a relaxation of Eq. (1) to optimize for P on $\mathcal{B}_m$:

$$\hat{z} = \operatorname*{argmin}_{z' \in \mathcal{Z}} \max_{P \in \mathcal{B}_m} \langle Qz', PD \rangle \,, \tag{4}$$

which allows for a differentiable permutation learning framework that combines (a) stochastic gradient descent over the space of square matrices; and (b) projection onto $\mathcal{B}_m$ with the Sinkhorn operator [16]. In the case of the Turnpike and Noisy Turnpike problems, this approach requires the algorithm to optimize an $m \times m$ matrix, which holds an infeasibly large $\Theta(n^4)$ entries. We refer to this alternative as the "gradient descent" method in the results below.

Algorithm 2. $Q^{\top}$ applied to matched D

Input: m-Distance vector D; n-Point vectors z^t, z^{t+1}

$z^{t+1}[:] \leftarrow 0$ ▷ Zero out the vector
$z^t \leftarrow \text{sort}(z^t)$ ▷ Sort the incoming point vector
frontier $\leftarrow$ Min-Interval-Priority-Queue(n, z^t) ▷ z^t interval order, n interval allocation
for $i \in [1, \ldots, n-1]$ **do**
    enqueue(frontier, $(i, i+1)$)
end for
for $t \in [1, \ldots, m]$ **do**
    $(i, j) \leftarrow$ Min-Pop(frontier)
    $z^{t+1}[i] \leftarrow z^{t+1}[i] - D[t]$
    $z^{t+1}[j] \leftarrow z^{t+1}[j] + D[t]$
    if $j < n$ **then**
        enqueue(frontier, $(i, j+1)$)
    end if
end for

procedure Interval-Compare(z, (i_1, j_1), (i_2, j_2))
    ▷ Interval comparison function based on z
    return $(z[j_1] - z[i_1]) \leq (z[j_2] - z[i_2])$
end procedure

2.3 Extension to Other Variants

In the Beltway problem, we are given $n(n-1)$ unlabeled arc lengths (distances) between n points $p_1, \ldots, p_n$ on a circle. Note that we receive double the number of distances as in the Turnpike case because there are two different arcs between any two points (i.e., clockwise and counter-clockwise). The Beltway problem can be solved within our framework (Sect. 2.2) with minor modifications. We also extend our method to a general variant of the Turnpike problem that handles both labeled and missing distances. This extension captures the labeled partial digest problem [19] and simplified partial digest problem [4]. We refer to Appendix C [8] for the details of these extensions.

2.4 Initializer Sampling

The choice of an initializer for Algorithm 3 plays a critical role in achieving good convergence and overall performance. A well-chosen initializer can lead to faster convergence, improved stability, and a more accurate solution. Here, we consider three practical initializing schemes. The first scheme samples a random Gaussian vector and sorts it. Though efficient to implement, this scheme is unlikely to produce a good initializer if the ground set exhibits pathological features such as having spread-out point clusters. On the other hand, if the points are well-spread, this scheme often finds a close starting point. The second scheme provides a random permutation P_0 to the sub-problem in Eq. (2) and sets z^0 as its closed-form solution. This incorporates the combinatorial nature of Turnpike and potentially encourages more diverse exploration of the solution space.

Algorithm 3. Minorization-Maximization Divide-and-Conquer (MMDQ)

Input: Distance vector D, initial estimate z^0, tolerance ϵ

$D \leftarrow D^{\uparrow}$ ▷ Replace D with its sorted equivalent $D^{\uparrow}$
$t \leftarrow 0$
while not converged **do**
  $z^{t+1}, P_{t+1} \leftarrow \text{MM}(D, z^t, \epsilon)$ ▷ Alg. 1
  $z_l, z_r \leftarrow \textsc{Partition}(z^{t+1})$ ▷ as described above
  $D_{ll}, D_{rr}, D_{lr} \leftarrow \textsc{Segment}(z_l, z_r, D, P_{t+1})$ ▷ as above
  $P_l, \_ \leftarrow \text{MMDQ}(D_{ll}, z_l, \epsilon)$ ▷ recursive call on left set
  $P_r, \_ \leftarrow \text{MMDQ}(D_{rr}, z_r, \epsilon)$ ▷ recursive call on right set
  $z^{t+1} \leftarrow$ solve Eq. (4) ▷ "consensus" point set
  $t \leftarrow t + 1$
  converged $\leftarrow \|z^{t+1} - z^t\| < \epsilon$
end while
return P_t, z^t

Nevertheless, selecting random permutations does not guarantee proximity to the optimal solution or even proximity to a valid distance permutation. The final scheme is a greedy-search method inspired by the classical backtracking approach [20]. That is, we sequentially fit the largest distance in D onto a line segment configuration (i.e., placing a new point to the left or to the right end of the segment based on this distance). However, unlike the original formulation—which uses backtracking to find the optimal placement—we make greedy choices to generate an initializer that will be polished with our algorithm afterwards, thus avoiding potentially exponential runtime.

3 Empirical Results

Experimental Design. We assessed the performance of our proposed algorithm on the Turnpike, Beltway, and Labeled Partial Digest problems. As a baseline, we used synthetic data to validate our proposed algorithm's performance on uncertain measurements and compared it to the backtracking method [20], the distribution matching method [13], and our projected gradient descent baseline using the Gumbel-Sinkhorn relaxation [16]. To evaluate the performance of our method in genome reconstruction, we conducted a series of experiments that simulated the reconstruction of a DNA sequence from fragments generated by enzymes. All experiments were implemented in Python 3.10 using a C++20 library implementing the algorithm integrated with Python using PyBind11 and conducted on a computer equipped with 1.0 TB of RAM, two Intel Xeon E5-2699A v4 CPUs, and a GTX 3080 GPU.

Synthetic Data. Synthetic datasets were generated by sampling n points on the real line from three distributions: the Cauchy distribution, the standard normal distribution, and the uniform distribution on $[0, 1]$. The uniform distribution was chosen to align with the setting explored by Huang et al. [13]. The normal

distribution was chosen to generate point sets with tightly-clustered points. The Cauchy distribution was selected to generate point sets with varying scales, i.e., sets where some points are much larger than others. This is important to test since outlying values often pose a challenge for ℓ_2 optimization methods [5].

We examined sample sizes ranging from 50 to 2,000 points (in increments of 50) and three additional large sample sizes of 5,000, 10,000, and 100,000 to demonstrate the method's scalability. To simulate measurement uncertainty of magnitude $\epsilon = 10^{-k}$ for integer $k \in [1, 12]$, we added a Gaussian noise vector $g \sim \mathcal{N}(\mathbf{0}, \epsilon\mathbf{I})$ to the given vector of pairwise distances [7]. We rounded the distance to zero when the amount of uncertainty exceeded the magnitude of the distance, which simulates missing distances. We predicted the points for each set of distribution, size, and uncertainty for 10 independent test cases. We run each algorithm 10 times and output the best estimate, which we quantified with the ℓ_2 distance between the estimated and uncertain distance sets (the algorithms are deterministic, but the choice of initializer is random as described above). We recorded the mean absolute error (MAE) and mean squared error (MSE) between the estimated and ground point sets. The MAE is a continuous alternative to the binning distance [13] and is more suitable for our method since we do not explicitly assign points to bins. Since the distribution matching method produces bins as its output, we use the midpoint of each bin as the predicted point.

Study of Different Initialization Schemes. We investigated the three initialization strategies (Sect. 2.4) to select one for subsequent experiments. We boosted the Gaussian point vector and permutation point vector initializer by drawing n distinct samples that were scored by solving Problem 2 for each and taking the maximum value. The sample with the maximum score from each strategy was used as the initializer. We used the Gaussian initializer as the starting point for the greedy-search initializer. We tested the strategies across all settings described previously. Figure 1 shows the cosine distances between the estimated and uncertain distance vectors. Among the three approaches, the permutation strategy exhibited the worst similarity scores, with an average magnitude 13 times larger than that of the greedy-search strategy. The Gaussian strategy demonstrated an average error magnitude that was 8 times larger than the greedy-search initialization.

A better initial score does not necessarily guarantee a better reconstruction after optimization. To assess the efficacy of each initializer, we analyzed whether lower pre-optimization errors translated to reduced post-optimization errors. The cosine distance after optimization is also shown in Fig. 1. The permutation initialization had the highest errors and the greedy-search approach had the lowest errors, which is consistent with the pre-optimization cosine distance. We used the greedy-search initializer for our experiments since it exhibited the lowest post-optimization distance. The greedy-search strategy's lower error comes at a computational cost. Table 1 shows the median runtime for the greedy-search initializer, Gaussian initializer, and optimization loop across a representative set of problem sizes. For all sizes, the greedy-search initialization takes more time than running the optimization, whereas the Gaussian initialization strategy runs in

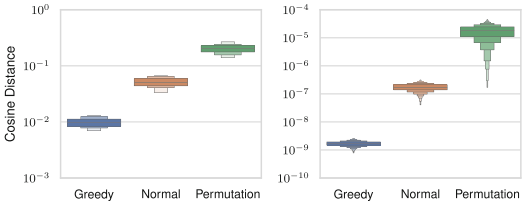

Fig. 1. Cosine distances between estimated distance vectors ($\hat{D}$) and ground distance vectors (D), before (left) and after (right) MM optimization under three different initialization schemes.

Table 1. Median runtimes (in seconds) for the MM optimizer (Opt.), Gaussian and Greedy initializations over different sample sizes.

Points	Opt.	Gauss.	Greedy
100	0.40	0.54	0.23
500	8.40	2.54	17.56
1000	16.49	4.63	36.49
1500	25.39	7.53	55.72
2000	42.38	18.53	84.06

an order of magnitude less time than the optimization. This is due to the inherently serial nature of the greedy-search initializer, which requires all previous steps to be considered first. This is in contrast to the optimization loop, which has a runtime dominated by sorting, which is parallelized.

Evaluating Noisy Turnpike Solutions on Synthetic Instances. We tested how accurately the MM (Sect. 2.2), backtracking, distribution matching [13], and gradient descent methods were able to reconstruct point sets. Table 2 shows the median MAE normalized by the uncertainty for a representative set of problem sizes and uncertainties. Each method had 1 h to solve each instance, with the exception of 10,000 and 100,000 point instances, which were given 90 and 6,000 min respectively. The backtracking method was able to solve instances with 1,000 or fewer points, but exhibited larger errors than our method. The gradient descent method solved all instances with 500 or fewer points with residual error that ranged between 10 and 1,000 times higher than the MM approach. The distribution matching method performed similarly to our method but could not

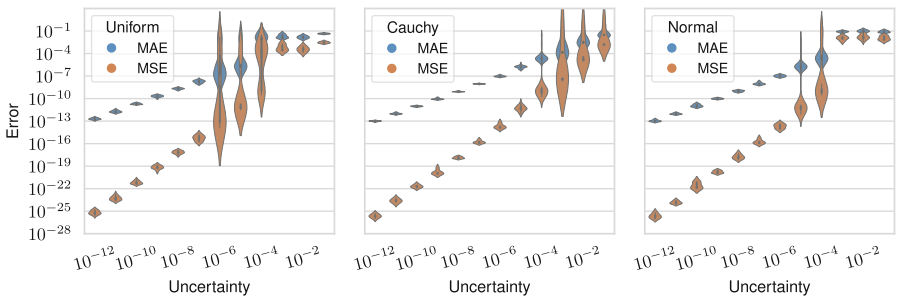

Fig. 2. Mean absolute error (blue) and mean squared error (orange) between the estimated and ground vectors across all levels of measurement uncertainty for Uniform, Cauchy and Normal data distributions. (Color figure online)

scale past 100 points. Our method was able to solve instances with 2,000 points with a median MAE that is 10 times lower than the uncertainty level up to a magnitude of 10^{-4}, after which the scaling becomes distribution dependent.

Table 2. Median MAE normalized by the magnitude of measurement uncertainty ϵ across different point set sizes and uncertainties. We compare our method (MM), distribution matching (DM), backtracking (BT), and gradient descent (GD) approaches. A dash indicates that a method did not finish solving any instances of this size due to either memory or runtime constraints.

$n/100$	10^{-6}				10^{-5}				10^{-4}			
	MM	DM	BT	GD	MM	DM	BT	GD	MM	DM	BT	GD
1	0.2	0.49	26.4	1270	0.19	0.51	26.6	186	2.09	0.68	56.7	1.54
2	0.1	—	26.4	462	0.14	—	26.5	39	1.54	—	55.7	5.72
5	0.11	—	36.4	312	0.09	—	31.5	40	0.94	—	34.7	3.08
10	0.06	—	36.4	—	0.04	—	36.5	—	0.08	—	36.7	—
50	0.08	—	—	—	0.08	—	—	—	0.86	—	—	—
100	0.08	—	—	—	0.11	—	—	—	0.82	—	—	—
1000	0.12	—	—	—	0.08	—	—	—	0.11	—	—	—

Table 3. Normalized MAE for 5 sizes and 4 uncertainty levels using the MM and gradient descent (GD) algorithms on simulated partial digestions of a cDNA.

n	MM (10^{-7})	GD (10^{-7})	MM (10^{-6})	GD (10^{-6})	MM (10^{-5})	GD (10^{-5})	MM (10^{-4})	GD (10^{-4})
10	0.148	0.087	0.159	0.096	0.168	0.145	0.232	0.284
15	0.157	0.072	0.150	0.078	0.140	0.160	0.202	0.274
20	0.214	0.031	0.186	0.032	0.123	0.068	0.224	0.231
38	0.113	0.043	0.128	0.094	0.135	0.112	0.178	0.243
54	0.101	0.053	0.102	0.078	0.186	0.203	0.146	0.581

Figure 2 shows our method's MAE and MSE over all settings plotted with respect to the magnitude of uncertainty. We observed that uncertainty in the distances correlated with reconstruction error, but the MAE is an order of magnitude lower than the uncertainty on average when the uncertainty is 10^{-4} or less. Instance size also affects the method's error scaling. As the sample size varied between 50 and 100,000 points, the median MAE shown in Table 2 demonstrates a downward trend for fixed error rates. This suggests the method scales at least as well as it does on small point sets as the number of points grows. This is because the distance measurement linear system is highly overdetermined, which makes it resilient to uncertainty [24]. Last, we report the mean and standard deviation of the runtimes of different solvers in Table 5. The MM mean runtime was lowest across all point sizes and MM is the only method that successfully solved the 5,000, 10,000, and 100,000 point instances.

Table 4. Normalized MAE for 10 sizes and 4 uncertainty magnitudes using the MM and gradient descent (GD) algorithms on simulated partial digestions of a linear genome. A dash indicates that the algorithm did not finish due to memory constraints or runtime constraints.

n	MM (10^{-7})	GD (10^{-7})	MM (10^{-6})	GD (10^{-6})	MM (10^{-5})	GD (10^{-5})	MM (10^{-4})	GD (10^{-4})
10	0.152	2.31×10^6	0.148	1.54×10^5	0.163	1.70×10^4	0.142	1.34×10^3
64	0.213	1.89×10^5	0.219	6.42×10^4	0.220	1.87×10^3	0.232	4.20×10^2
142	0.358	8.22×10^5	0.359	3.44×10^4	0.357	9.28×10^3	0.365	6.12×10^2
183	0.282	1.28×10^6	0.268	1.75×10^5	0.288	9.37×10^3	0.290	1.34×10^3
530	0.071	—	0.070	—	0.080	—	0.066	—
959	0.119	—	0.121	—	0.125	—	0.139	—
1209	0.119	—	0.132	—	0.127	—	0.115	—
1451	0.059	—	0.061	—	0.043	—	0.072	—
2669	0.048	—	0.039	—	0.050	—	0.048	—

Table 5. Runtime (in seconds) across point sizes for various Turnpike solvers.

$n/100$	MM	DM	BT	GD
1	0.8 ± 3.9	1680.3 ± 32.1	2.8 ± 10.2	16.3 ± 0.4
2	21.2 ± 13.0	—	53. ± 32.3	114.9 ± 0.5
5	204.3 ± 74.4	—	304.9 ± 20.3	4366.6 ± 8.6
10	552.3 ± 112.4	—	1052.5 ± 50.9	—
50	1992.1 ± 51.2	—	—	—
100	3543.8 ± 712.3	—	—	—
1000	5912.3 ± 52.4	—	—	—

Partial Digestion Experiments. We tested the effectiveness of our algorithms for reconstructing genomes via data generated by an enzyme that digests DNA into fragments at restriction sites [2]. The fragment lengths give the distances between all restriction sites, which are at unknown positions. The genome is assembled from the fragments after inferring the restriction site locations from the distances, a process equivalent to solving the Turnpike problem for linear genomes and the Beltway problem for circular genomes [13]. We simulated partial digestion instances to test our algorithms. For Turnpike instances, we used the human X chromosome's centromere, and for Beltway instances, we used the full genome of the bacteria *Carsonella ruddii*. In both cases, we used 15-base-long enzymes and simulated the digestion process by sampling the DNA sequence such that each restriction site occurred between 10 and 500 times. We obtained digested DNA fragments by splitting the sequence at all of its occurrences. We added a signed Poisson random vector to simulate when enzymes cut too many or too few bases, both frequent occurrences in practice [6].

The Turnpike experiments were performed with our method and the gradient descent baseline due to runtime constraints. The Beltway experiments were performed with our algorithm and the distribution matching algorithm, as

they are the only ones designed for uncertain Beltway instances. Table 4 shows normalized MAE for the Turnpike experiments on instances with 10–2,669 fragments. Our method recovered fragment locations with an MAE that scaled linearly to the uncertainty present in the measurements, performing orders of magnitude better than the gradient descent baseline. Table 3 shows normalized MAE for the Beltway experiments, which were performed on instances with 10–54 fragments. Our method performed competitively with the gradient descent approach.

Labeled Partial Digestion Experiment. Pandurangan et al. [19] performed a labeled partial digestion problem (LPDP) recovery experiment using the restriction sites of the enzyme HindIII on the bacteriophage λ. For each distance d, they simulated relative uncertainty of order $r \in [0, 1]$ by replacing d with a uniformly sampled integer in $[(1 - r)\, d, (1 + r)\, d]$. They varied r between 0% and 5% to mimic experimental settings, where 2% to 5% is expected.

Each experiment was repeated 100 times. A success is reported when the recovered distances were within the relative uncertainty of the ground truth set. We repeated this experiment using our base algorithm (MM) and our partition-update formulation (PMM) given in Appendix C [8]. Our base algorithm does not use additional labeling information. The results are shown in Table 6. We observe that our method performs competitively without additional labels and further improves when it is provided with the labels. All instances ran in less than 1 s across all uncertainty levels and across all solvers.

Table 6. Recovery success rate of our base solver (MM), our partition solver (PMM), and Pandurangan et al.'s solver [19] at various relative error levels.

r	MM	PMM	LPDP
0%	100%	100%	100%
1%	99%	99%	98%
2%	96%	97%	96%
3%	95%	96%	94%
4%	92%	94%	91%
5%	89%	92%	87%

4 Conclusion

Noisy Beltway and Noisy Turnpike are NP-hard problems that aim to recover a set of one-dimensional points based on a corrupted pairwise distances. These problems find application in widespread biological contexts. We introduced a novel optimization formulation and an alternating algorithm built from sorting and implicit matrix multiplication. This leads to an asymptotic runtime of $\mathcal{O}(n^2 \log n)$ time per iteration with $\mathcal{O}(n)$ auxiliary memory. To escape low-quality local optima, we introduced a divide-and-conquer step to fix common errors. We performed large-scale experiments with approximately 25 billion distances (equivalent to 100,000 points) to showcase the efficiency of our method. In contrast, previous methods are infeasible with as few as 125,000 distances (equivalent to 500 points). We also demonstrated the method's robustness and scalability in a variety of challenging situations, including large-scale uncertainty and distance duplication. Our algorithm efficiently solves large distance sets with

realistic levels of uncertainty, opening up new avenues of research into biological applications of Turnpike and computational geometry problems.

Acknowledgements. This work was supported in part by the US National Science Foundation [DBI-1937540, III-2232121], the US National Institutes of Health [R01HG012470] and by the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program. Disclosure of interests: C.K. is a co-founder of Ocean Genomics, Inc.

References

1. Abbas, M.M., Bahig, H.M.: A fast exact sequential algorithm for the partial digest problem. BMC Bioinform. **17**(19), 510 (2016)
2. Alizadeh, F., Karp, R.M., Weisser, D.K., Zweig, G.: Physical mapping of chromosomes using unique probes. J. Comput. Biol. **2**(2), 159–184 (1995)
3. Birkhoff, G.: Three observations on linear algebra. Univ. Nac. Tacuman, Rev. Ser. A **5**, 147–151 (1946)
4. Blazewicz, J., Burke, E., Kasprzak, M., Kovalev, A., Kovalyov, M.: Simplified partial digest problem: enumerative and dynamic programming algorithms. IEEE/ACM Trans. Comput. Biol. Bioinf. **4**, 668–680 (2007)
5. Boyd, S., Boyd, S.P., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
6. Cieliebak, M., Eidenbenz, S.: Measurement errors make the partial digest problem NP-hard. In: Farach-Colton, M. (ed.) LATIN 2004. LNCS, vol. 2976, pp. 379–390. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24698-5_42
7. Dokmanic, I., Parhizkar, R., Ranieri, J., Vetterli, M.: Euclidean Distance Matrices: essential theory, algorithms, and applications. IEEE Signal Process. Mag. **32**(6), 12–30 (2015)
8. Elder, C.S., Hoang, M., Ferdosi, M., Kingsford, C.: A scalable optimization algorithm for solving the beltway and turnpike problems with uncertain measurements. bioRxiv (2024)
9. Fomin, E.: Reconstruction of sequence from its circular partial sums for cyclopeptide sequencing problem. J. Bioinform. Comput. Biol. **13**(1), 1540008 (2015)
10. Fomin, E.: A simple approach to the reconstruction of a set of points from the multiset of pairwise distances in n^2 steps for the sequencing problem: III. Noise inputs for the beltway case. J. Comput. Biol. **26**(1), 68–75 (2019)
11. Gabrys, R., Pattabiraman, S., Milenkovic, O.: Mass error-correction codes for polymer-based data storage. In: 2020 IEEE International Symposium on Information Theory (ISIT), pp. 25–30, June 2020. ISSN 2157-8117
12. Hardy, G.H., Littlewood, J.E., Pólya, G.: Inequalities. Cambridge University Press, Cambridge (1952)
13. Huang, S., Dokmanić, I.: Reconstructing point sets from distance distributions. IEEE Trans. Signal Process. **69**, 1811–1827 (2021)
14. Lemke, P., Skiena, S.S., Smith, W.D.: Reconstructing sets from interpoint distances. In: Aronov, B., Basu, S., Pach, J., Sharir, M. (eds.) Discrete and Computational Geometry: The Goodman-Pollack Festschrift. Algorithms and Combinatorics, pp. 597–631. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-642-55566-4_27

15. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. **261**(4), 515–534 (1982)
16. Mena, G., Snoek, J., Linderman, S., Belanger, D.: Learning latent permutations with Gumbel-Sinkhorn networks. In: International Conference on Learning Representation, vol. 2018 (2018)
17. Mohimani, H., et al.: Multiplex de novo sequencing of peptide antibiotics. J. Comput. Biol. **18**(11), 1371–1381 (2011)
18. Nadimi, R., Fathabadi, H.S., Ganjtabesh, M.: A fast algorithm for the partial digest problem. Jpn. J. Ind. Appl. Math. **28**, 315–325 (2011)
19. Pandurangan, G., Ramesh, H.: The restriction mapping problem revisited. J. Comput. Syst. Sci. **65**(3), 526–544 (2002)
20. Skiena, S.S., Sundaram, G.: A partial digest approach to restriction site mapping. In: Proceedings. International Conference on Intelligent Systems for Molecular Biology, vol. 1, pp. 362–370 (1993)
21. Skiena, S.S., Smith, W.D., Lemke, P.: Reconstructing sets from interpoint distances (extended abstract). In: Proceedings of the Sixth Annual Symposium on Computational Geometry, SCG 1990, pp. 332–339, New York, NY, USA, May 1990. Association for Computing Machinery (1990)
22. Smith, H.O., Birnstiel, M.L.: A simple method for DNA restriction site mapping. Nucleic Acids Res. **3**(9), 2387–2398 (1976)
23. Sun, Y., Babu, P., Palomar, D.P.: Majorization-minimization algorithms in signal processing, communications, and machine learning. IEEE Trans. Signal Process. **65**(3), 794–816 (2017)
24. Wendland, H.: Numerical Linear Algebra: An Introduction. Cambridge University Press, Cambridge (2017)
25. Zhang, Z.: An exponential example for a partial digest mapping algorithm. J. Comput. Biol. **1**(3), 235–239 (1994)
26. Zintchenko, I., Wiebe, N.: Randomized gap and amplitude estimation. Phys. Rev. A **93**(6), 62306 (2016)