Relaxations of the Seriation problem and applications to de novo genome assembly

Antoine Recanati

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences Lettres
PSL Research University

Préparée à l'École normale supérieure

Relaxations of the Seriation Problem and Applications to *de novo* Genome Assembly

Relaxations du problème de sériation et applications à l'assemblage de génome *de novo*.

École doctorale n°386

ÉCOLE DOCTORALE DE SCIENCES MATHÉMATIQUES DE PARIS CENTRE

Spécialité INFORMATIQUE

COMPOSITION DU JURY :

Alexandre d'Aspremont
Inria Paris, Directeur de thèse

Dominique Lavenier
IRISA/Inria Rennes, Rapporteur

Stéphane Vialette
LIGM, Université Paris Est,
Rapporteur

Thomas Brüls
CEA, Genoscope, Membre du Jury

Fajwel Fogel
Sancare, Membre du Jury

Jean-Philippe Vert
Google Brain, Mines ParisTech,
Directeur du Jury

Soutenue par Antoine Recanati
le 29 novembre 2018

Dirigée par
Alexandre d'Aspremont

ENS ÉCOLE NORMALE SUPÉRIEURE | PSL RESEARCH UNIVERSITY PARIS

Plus ça rate, et plus on a de chances que ça marche.

Second Principe Shadock

Résumé

Les technologies de séquençage d'ADN ne permettent de lire que de courts fragments, dont on ignore la position sur le génome. L'assemblage *de novo* vise à reconstituer une séquence d'ADN entière en mettant ces fragments bout-à-bout, tel un puzzle. Dans l'approche OLC (overlap-layout-consensus), on calcule le *chevauchement* entre fragments afin de les disposer en ordre (*réarrangement*), puis extraire une séquence *consensus*.

Le *réarrangement* peut s'écrire comme un problème combinatoire de *sériation*, où l'on réordonne des éléments comparable entre eux, de sorte que deux éléments adjacents sont similaires. Ce problème est résolu efficacement par un algorithme spectral en l'absence de bruit, mais il en va autrement des données génomiques réelles. En particulier, des régions du génome sont similaires bien qu'éloignées (séquences répétées), rendant l'assemblage problématique.

Les méthodes d'assemblage emploient des algorithmes hiérarchiques et gloutons pour désambiguïser les séquences répétées. Nous proposons ici une approche épurée où l'on réarrange tous les fragments «d'un coup» via la résolution de *sériation*.

Notre première contribution montre que l'emploi de la méthode spectrale pour le réarrangement s'intègre parfaitement dans le schéma OLC, produisant des résultats de qualité semblable aux méthodes standard. Cependant, du fait des séquences répétées, cette méthode produit des assemblages fragmentés (typiquement en quelques sous-séquences au lieu d'une).

La deuxième contribution est un prolongement de la méthode spectrale lié à la réduction de dimension sous conservation de distances, englobant les problèmes de sériation et de sériation circulaire (une variante où les éléments peuvent être ordonnés selon un cycle) dans un cadre unifié. Ce prolongement rend l'algorithme robuste au bruit et résout le problème de fragmentation de l'assemblage précédent.

Notre troisième contribution formalise la sériation robuste, où l'on souhaite réordonner des données bruitées. Nous décrivons des liens avec d'autres problèmes combinatoires, en particulier pour des matrices modèlisant les données réelles d'ADN. Nous proposons des algorithmes adaptés, améliorant expérimentalement la robustesse sur données synthétiques et réelles, bien que moins clairement que la deuxième contribution.

La quatrième contribution présente le problème de sériation avec duplication, motivé par l'assemblage de génomes cancéreux via des données de conformation spatiale, que nous tentons de résoudre avec un algorithme de projections alternées fondé en partie sur les méthodes de sériation robuste, sur données synthétiques.

Mots-clés

sériation, méthodes spectrales, optimisation combinatoire, relaxations convexes, permutations, permutaèdre, optimisation robuste, assemblage *de novo* , séquençage de troisième génération, Oxford Nanopore Technology, Overlap-Layout-Consensus, classement.

Abstract

In a sequencing experiment, we can only "read" small fragments (*reads*) of DNA due to physical limitations, whose location on the genome is unknown. *De novo* assembly aims to put them together to retrieve the full DNA sequence, like a jigsaw puzzle. The OLC approach computes pairwise Overlaps between reads to find their Layout, and then derive a Consensus sequence.

The layout can be cast as an instance of the Seriation combinatorial problem, seeking to reorder a set of elements based on their pairwise similarity, such that similar elements are nearby. In a noiseless setting, a spectral method can solve Seriation efficiently. Still, it often fails on noisy, real DNA data. Notably, assembly is challenged by repeated genomic regions (*repeats*) causing distant fragments to be similar.

Most assembly engines follow hierarchical, greedy schemes, including modules dedicated to detect and disambiguate repeats while constructing the output sequence. We explore a simpler approach using Seriation to lay out all reads at once.

Our first contribution is to show that the spectral method can be seamlessly integrated in an OLC framework, yielding competitive results compared to standard methods on real data. However, due to repeats, the method can only find fragmented assemblies (with a few large assembled fragments), *i.e.*, it does not succeed to layout all the reads together at once.

In our second contribution, we extend the spectral method using a multidimensional spectral embedding. It provides a unifying framework for seriation and circular seriation, a variant searching for a cyclic ordering of the data. This method significantly improves the robustness of the original algorithm on noisy data, and yields single-contig assembly of bacterial genomes.

As a third contribution, we introduce the Robust Seriation framework, formalizing the task of seriation on corrupted data. We outline the relation between (robust) seriation and other combinatorial problems, particularly for stylized matrices modeling DNA sequencing data. We propose dedicated algorithms that experimentally improve robustness on synthetic and real data, although they turn out to be more sensitive than the method constituting our second contribution.

In a fourth contribution, we introduce the problem of Seriation with Duplications, which is motivated by the application of assembling cancer genome from spatial conformation (Hi-C) data. We propose an alternated minimization algorithm that can utilize methods designed to solve Robust Seriation, and evaluate it on toy data.

Keywords

seriation, spectral methods, combinatorial optimization, convex relaxations, permutations, permutahedron, robust optimization, *de novo* genome assembly, third generation sequencing, Oxford Nanopore Technology, overlap-layout-consensus, layout problems, ordering.

Remerciements

Tout d'abord, merci à mon directeur de thèse, Alexandre d'Aspremont, avec qui ce fut une chance et un bonheur inestimables de travailler. Une chance, d'être guidé scientifiquement par ta vision d'ensemble et ta capacité bluffante à faire, en quelques secondes, le lien entre un problème nouveau et une méthode clé dont la plupart des gens ignoraient même l'existence. Un bonheur, d'être encadré avec tant d'enthousiasme, de dynamisme et de bienveillance.

Je remercie vivement Dominique Lavenier et Stéphane Vialette d'avoir accepté de rapporter cette thèse; c'est un honneur de présenter mes travaux devant vous et de bénéficier de vos observations. Je remercie également les autres membres du jury d'avoir accepté d'en faire partie, à commencer par son président, Jean-Philippe Vert, avec qui j'ai eu le privilège de travailler, dont je suis reconnaissant de la gentillesse et admiratif de la faculté à reformuler les problème les plus complexes avec transparence, et les présenter sous un autre angle qui y apporte un éclairage et des perspectives nouvelles. Merci à Thomas Brüls; avec qui ce fut aussi un bonheur de travailler, m'apportant une fraîcheur et une expertise précieuses, sous un oeil bienveillant. Merci enfin à Fajwel Fogel; j'ai été heureux de te côtoyer en commençant ma thèse et de reprendre le flambeau de certains de tes travaux, et je suis très enthousiaste à l'idée de travailler avec toi après ma thèse ! Je suis tout aussi enthousiaste de travailler avec Chloé-Agathe Azencott, que je remercie chaleureusement de m'accueillir au CBIO.

Ces trois années ont été enrichissantes pour moi, intellectuellement et humainement, et cela tient à l'environnement du laboratoire, qui ne serait pas ce qu'il est sans le directeur de l'équipe Sierra, Francis Bach, que je remercie de m'y avoir accueilli, et d'oeuvrer à ce climat exceptionnel. Merci à tous les autres membres de Willow-Sierra, avec qui j'ai partagé de très bons moments lors de diverses pauses (ou de travail commun !); et dont je considère aujourd'hui certains comme des amis. Merci notamment à mes co-bureaux formidables. Et désolé de ne pas faire de dédicaces nominatives dans ce paragraphe, mais je n'en pense pas moins !

Merci enfin à ma famille pour leur présence, leur importance et leur amour, et à Alice, pour les mêmes raisons, et d'autres encore.

Contents

Contributions and thesis outline

Chapter 1: In this introductory chapter, we present the two problems that come into play in this thesis. First, we introduce the Seriation problem, its mathematical formulation, and develop two key methods to tackle this combinatorial problem over permutations, that will be employed further in the manuscript : the spectral relaxation, and convex relaxations. Then, we present the problem of *de novo* genome assembly, and explain the challenge caused by repeated regions (*repeats*), before briefly describing the technology-specific data used in our experiments. We conclude this chapter by stating the major challenges arising when trying to apply seriation to genome assembly.

Chapter 2: In this applicative chapter, we set up to use the spectral method to compute the layout of the reads, and integrate this layout module in a straightforward, end-to-end Overlap-Layout-Consensus assembly pipeline. We test the method against real, third generation sequencing DNA data of bacterial and yeast genomes. The proposed method is shown to be competitive, thus validating the use of seriation for genome assembly. However, due to repeats, the spectral method cannot layout all reads at once and eventually produces fragmented assemblies. For the bacterial data-sets, the assembled contigs can sometimes be merged into a single contig seamlessly, but the final yeast assemblies remain fragmented.

Chapter 3: This chapter presents a simple, yet powerful enhancement of the spectral method, drawing a parallel between Seriation and the Spectral Clustering method, well known in the machine learning community. We propose a unifying framework for seriation and *circular* seriation. In the circular variant of seriation, one seeks to find a circular (as opposed to linear) ordering of the elements based on their pairwise similarity. It has diverse applications in bioinformatics. Notably, finding the layout of *circular* genomes (such as the bacterial genomes used in Chapter 2) fits in the circular seriation framework. In this chapter, we bring together results from machine learning and specific matrix theory that shed light on the mechanisms underpinning the spectral method for seriation, and allow us to prove theoretical guarantees for circular seriation analogous to those existing for linear seriation in the noiseless case. Importantly, we designed an algorithm leveraging these results. It is a straightforward extension of the baseline spectral method, yet we show that it yields a valuable gain in robustness through numerical

experiments. Remarkably, despite repeats, it correctly finds the layout of the bacterial genomes introduced in Chapter 2 in one shot (leading to a single contig).

Chapter 4: Here, we focus on the mathematical modeling of seriation through optimization problems, with the aim of finding algorithms that are by design more robust to the repeat-induced noise. We formalize the robust seriation problem and show that it is equivalent to a modified 2-SUM problem for a class of similarity matrices modeling those observed in DNA assembly. We explore several relaxations of this modified 2-SUM problem, and compare them empirically on synthetic data. The most salient and efficient methods are also evaluated on the bacterial genomes used in the previous chapters. One of them is able to correctly find the full layout of an *E. coli* genome in one shot from Oxford Nanopore reads. However, this method is experimentally more sensitive than the one presented in Chapter 3.

Chapter 5: This chapter introduces the problem of Seriation with Duplications. It is motivated by an application to cancer genome reconstruction, which is challenged by so-called *structural variations*. Namely, large portions of the genome, up to whole chromosomes, are duplicated or deleted, and new chromosomes are formed by fusing two pieces of chromosomes which are not connected in a normal genome. Hi-C (spatial conformation) data can be used to reconstruct the structure of such genome, but the duplications need to be addressed through a specific framework. After motivating and formalizing the problem of seriation with duplications, we propose an alternated minimization method, and evaluate it on synthetic data.

Chapter 6: This chapter concludes the thesis by summarizing our contributions, highlighting the key challenges addressed, and describing possible extensions and improvements of the present work.

Publications related to this manuscript are listed below.

- **Chapter 2** is based on the following publication, Antoine Recanati, Thomas Brüls, and Alexandre d'Aspremont. A spectral algorithm for fast de novo layout of uncorrected long nanopore reads. *Bioinformatics*, 2016. The software is available on https://github.com/antrec/spectrassembler.

- **Chapter 3** is based on the following report, Antoine Recanati, Thomas Kerdreux, and Alexandre d'Aspremont. Reconstructing latent orderings by spectral clustering. *arXiv preprint arXiv:1807.07122*, 2018a. A python package is available on https://github.com/antrec/mdso.

- **Chapters 4 and 5** are based on the following report, Antoine Recanati, Nicolas Servant, Jean-Philippe Vert, and Alexandre d'Aspremont. Robust seriation and applications to

cancer genomics. *arXiv preprint arXiv:1806.00664*, 2018b. The best performing method for Robust Seriation was added to the `mdso` package given above. Code for Seriation with Duplications is available on https://github.com/antrec/serdupli, although it is not properly documented and ready-to-use at the moment.

Chapter 1

Introduction

Genome sequencing plays central role in biological research, with applications ranging from evolutionary science to human disease research. The process of whole genome sequencing, that is, reading the genome of a (member of a) species, involves two distinct tasks. First, the actual *sequencing* consists in collecting signal from a biological sample through a physical experiment (in the laboratory). Then, the *assembly* aims to reconstruct the genome from this signal, typically with dedicated algorithms (on a computer).

Over the past quarter-century, increase in computational power has facilitated genome sequencing through the collection and processing of larger amounts of data. The advent of new, "high throughput" sequencing technologies lead to an even more dramatic increase in sequencing power (and reduction in cost). Some of these "sequencing revolutions" have changed the game of genome assembly, calling for adapted methods on the algorithmic front.

Seriation is a mathematical problem akin to solving a one-dimensional jigsaw puzzle. One of the key steps of genome assembly essentially boils down to solving seriation. In this thesis, we will present algorithmic efforts to solve seriation, and investigate their efficiency when applied to genome assembly. Let us introduce the two core components of this thesis: seriation (theory and algorithms), and *de novo* genome assembly (application).

1.1 Seriation

In the following, we present the seriation problem, and introduce a few formal definitions and related problems. Then, we briefly review methods that have been proposed to solve it.

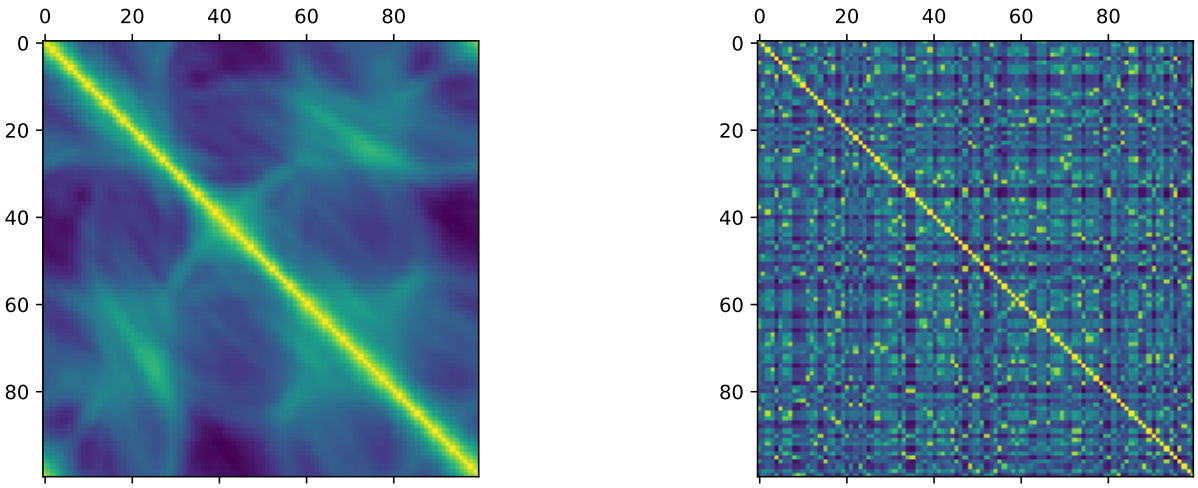
1.1.1 Presentation

The seriation problem seeks to recover a latent ordering from similarity information, such that similar elements are nearby in the final ordering. We typically observe a matrix measuring pairwise similarity between a set of n elements and assume they have a serial structure, *i.e.*,

they can be ordered along a chain where the similarity between elements decreases with their distance within this chain. In practice, we observe a random permutation of this similarity matrix, where the elements are not indexed according to that latent ordering. Seriation then seeks to find it back using only (local) pairwise similarity.

The problem was introduced in archaeology to find the chronological order of a set of graves [Robinson, 1951]. Each grave contained artifacts, assumed to be specific to a given time period. The number of common artifacts between two graves define their similarity, resulting in a chronological ordering where two contiguous graves belong to a same time period.

As a graphic illustration, let us consider the following example. The teapots dataset [Weinberger and Saul, 2006] is a collection of images of a rotating teapot, taken at angles regularly spaced between 0 and 360°. If the sequence of images is sorted by increasing angle, it constitutes a movie of the rotating teapot, making a full circle on itself. However, the collection of images is given unsorted. In order to recover the movie, we can compute the pairwise similarity between two images as the opposite of their ℓ_2 distance (the sum of the squared differences between the gray level in each pixel, if the image is black and white). Applying seriation to this set of similarities will output an ordering of the images where similar images are placed nearby, hopefully matching the ordering of the movie. Figure 1.1 shows the similarity matrix



(a) ordered (b) permuted

Figure 1.1: Similarity matrix between teapots images using the ℓ_2 distance, when the subscripts follow the ordering of the movie (1.1a), and when it is randomly permuted (1.1b).).

between images. Each entry (i, j) is the opposite of the sum of the squared pixel-wise distance. In Figure 1.1a, the subscripts are ordered by increasing rotation angle (linearly spaced between 0 to 360°). The similarity tends to decrease as we move away from the diagonal, *i.e.*, when the difference of angles of the teapot images increases. In Figure 1.1b, the subscripts are given in a random order, which is what is observed in practice. We still observe maximal values on the

main diagonal, which corresponds to self similarity between an image and itself. The diagonal of the matrix is invariant by permutation since the ℓ_2 distance between an image and itself is 0, for all images. The goal of seriation is to recover (1.1a) given (1.1b). As a qualitative result, Figure 1.2 shows a sub-sample of the ordered set of images found by seriation, on which we can see the rotative movement.

(a) equi-spaced

(b) consecutive

Figure 1.2: Five teapots images sampled along the ordering found by seriation. On the top Figure 1.2a, the images are uniformly sampled along the ordering (at positions 1, 21, 41, 61, 81 out of 100). The bottom Figure 1.2b displays five consecutive images in the ordering. The `mdso` software presented in Chapter 3 was used in this experiment.

The seriation problem has applications in DNA sequencing [Meidanis et al., 1998, Garriga et al., 2011] that we will develop throughout this manuscript. It also has applications in, *e.g.*, envelope reduction [Barnard et al., 1995] and bioinformatics [Atkins and Middendorf, 1996, Higgs et al., 2006, Cheema et al., 2010, Jones et al., 2012] (see Liiv [2010] for a more thorough overview of applications).

1.1.2 Notations

Let us introduce notations in order to formulate the problem mathematically.

Matrices and vectors. $\mathbf{S}_n$ is the set of real, symmetric matrices of dimension n, and $\mathbf{S}_n^+$ the set of non-negative, symmetric matrices of dimension n. The transpose of a matrix X is written X^T, and we use the notation $x^T y$ for the dot product between two vectors $x, y \in \mathbb{R}^n$ (which can be seen as a matrix of size $n \times 1$), and sometimes also the standard notation $\langle x, y \rangle$. $\mathbf{1}_n = (1, \ldots, 1)^T$ is the vector of size n with all ones. The sum of the entries of a vector $x \in \mathbb{R}^n$ can thus be written $x^T \mathbf{1}_n$. For a matrix X of dimension n, $\mathbf{diag}(X) \in \mathbb{R}^n$ is the vector constituting the main diagonal of X. However, if $x \in \mathbb{R}^n$ is a vector, $\mathbf{diag}(x)$ denotes the diagonal matrix whose main diagonal is x. We use $e_k = (0, \ldots, 0, 1, 0, \ldots, 0)^T$ for the k-th vector of the canonical basis

of $\mathbb{R}^n$. $\mathbf{I}_n$ is the identity matrix of dimension n. The sorted eigenvalues of a real, symmetric matrix $X \in \mathbf{S}_n$ are written $\lambda_1(X) \leq \cdots \leq \lambda_n(X)$. We often omit the comma separating the two subscripts of a matrix, *i.e.*we may use both notations $A_{i,j}$ and A_{ij}.

Permutations. The set of integers from 1 to n is written $\{1, \ldots, n\}$, or $[n]$ for short. Any permutation can be represented by a vector $\pi \in \mathbb{R}^n$ consisting in the rearrangement of the integers $1, \ldots, n$, where $\pi_i = j$ if and only if it moves the element at position j to position i. Such a permutation vector takes values in $[n]$ and each value appears once in the vector. For ease of reading, we will use both notations π_i and $\pi(i)$ to denote the i-th entry of the vector π. Alternatively, the same permutation can be represented by a permutation matrix $\Pi \in \{0,1\}^{n \times n}$ such that $\Pi_{ij} = 1$ if and only if $\pi_i = j$. The two representations are equivalent and relate through the equation $\Pi\mathbf{g} = \pi$, where $\mathbf{g} = (1, \ldots, n)^T$ denotes the identity permutation. The identity permutation matrix is the identity matrix $\mathbf{I}_n$. The matrix notation is convenient to write a matrix A whose entries have been permuted by π. $A\Pi^T$ is the matrix with coefficients $A_{i,\pi(j)}$, and $\Pi A \Pi^T$ is the matrix whose entry (i,j) is $A_{\pi(i),\pi(j)}$. The set of permutations of n elements is written $\mathcal{P}_n$, and this notation can refer to the set of permutation vectors or matrices, depending on the context. Also, whenever the dimension is clear from the context, we may omit the subscript n in $\mathcal{P}_n$, $\mathbf{1}_n$, etc.

1.1.3 Mathematical Formulation and Related Problems

The main structural hypothesis on similarity matrices related to seriation is the concept of R-matrix, defined hereafter, using the terminology introduced in Atkins et al. [1998].

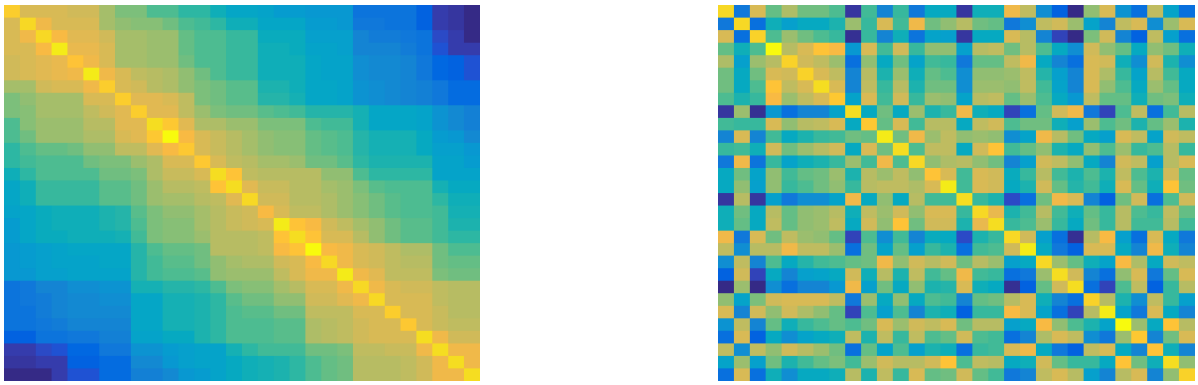
Definition 1.1.1. *We say that the matrix $A \in \mathbf{S}_n$ is an R-matrix (or Robinson matrix) iff it is symmetric and satisfies $A_{i,j} \leq A_{i,j+1}$ and $A_{i+1,j} \leq A_{i,j}$ in the lower triangle, where $1 \leq j < i \leq n$.*

These matrices are named after Robinson [1951]. The set of R matrices is written $\mathcal{R}$ in the following, and sometimes $\mathcal{L}_R$ in Chapter 3 (to emphasize that the underlying structure is Linear, as opposed to Circular). Their entries are non increasing when moving away from the diagonal in a given row or column. An equivalent formulation is to say that, given any triplet $(i,j,k) \in [n]^3$, with $i \leq j \leq k$, we have $A_{ij} \geq A_{ik}$ and $A_{jk} \geq A_{ik}$.

We say that a symmetric matrix A is pre-R if there exists a permutation matrix Π such that the matrix $\Pi A \Pi^T$ (whose entry (i,j) is $A_{\pi(i),\pi(j)}$) is an R-matrix. For such matrices, the seriation problem is to find a permutation that makes the matrix Robinson. Given a similarity matrix A that is pre-R, seriation can be written as a feasibility problem

$$\begin{array}{ll} \text{find} & \Pi \in \mathcal{P} \\ \text{such that} & \Pi A \Pi^T \in \mathcal{R}. \end{array} \qquad \text{(Seriation)}$$

Figure 1.3 illustrates these definitions.



(a) R-matrix (b) pre-R

Figure 1.3: An R-matrix, A (3.2a) and a permuted observation, $\Pi A \Pi^T$, with Π a permutation matrix (1.3b). Seriation seeks to recover the R-matrix (3.2a) from the pre-R matrix (1.3b).

Remark that the similarity matrix in Figure 1.1a is not an R-matrix. Indeed, although the similarity tends to decrease when moving away from the diagonal, it locally increases in some places. For instance, there are high similarity values between the last and the first images, given that the final orientation of the teapot is close to the initial. Yet, even when the strict (Seriation) problem is infeasible, we are still interested in finding an ordering such that (most) similar elements are placed nearby. This can be achieved by minimizing a well designed objective function. For instance, we can aim to minimize the number of anti-Robinson events, that is to say the number of violations of the two inequalities appearing in Definition 1.1.1. A discussion about such seriation criteria can be found in Hahsler [2017]. An objective function that will allow for spectral and convex relaxations in the following is the 2-SUM loss. The 2-SUM problem reads

$$\begin{array}{ll} \text{minimize} & \sum_{i,j=1}^n A_{ij}|\pi_i - \pi_j|^2 \\ \text{such that} & \pi \in \mathcal{P}_n. \end{array} \tag{2-SUM}$$

Note that it is equivalent to minimize $\sum_{i,j=1}^n A_{ij}|\pi_i - \pi_j|^2$ and $\sum_{i,j=1}^n A_{\sigma_i\sigma_j}|i-j|^2$ over the variable $\pi \in \mathcal{P}_n$ or $\sigma \in \mathcal{P}_n$, since the optimal permutation of one of these problems is the inverse permutation of the other ($\pi_* = \sigma_*^{-1}$). Intuitively, the 2-SUM problem lays similar elements nearby as it penalizes the similarity between two elements by their squared distance in the ordering. It is also a particular case of the Quadratic Assignment Problem [Koopmans and Beckmann, 1957], written

$$\min_{\pi \in \mathcal{P}_n} \sum_{i,j=1}^n A_{i,j} B_{\pi(i),\pi(j)} \tag{QAP(A,B)}$$

with $B_{ij} = |i-j|^2$. Laurent and Seminaroti [2015] showed that for pre-R matrices A, Seriation is equivalent to QAP(A,B) when $-B \in \mathcal{R}_n$, i.e. when B has increasing values when moving away from the diagonal, and has constant values across a given diagonal (*i.e.*, B is a Toeplitz matrix). This includes p-SUM problems, for $p > 0$, corresponding to $B_{ij} = |i-j|^p$. The case $p = 1$ is also known as the minimum linear arrangement problem (MLA) [George and Pothen, 1997]. For pre-R matrices, these problems are all equivalent and can be solved by a spectral algorithm in polynomial time, as we are about to see. However, when A is not pre-R, Seriation has multiple local solutions, and the spectral algorithm does not necessarily find a global optimum for 2-SUM, p-SUM or QAP(A,B) with B a Toeplitz, negated R matrix. In fact, these problems are NP-hard in general [Sahni and Gonzalez, 1976].

1.2 Optimization Strategies

We have seen that Seriation can be tackled by minimizing a loss function, such as the number of anti-Robinson events, the 2-SUM loss, or more generally some instances of QAP(A,B). Different methods can be used to perform the minimization.

Remark that the search space $\mathcal{P}_n$ is discrete and of cardinality $n!$, thus preventing the use of exhaustive search (*i.e.*, testing the values of the function over all possible permutations and pick the one with lowest score), even for small scale problems. To address this challenge, one may resort to relaxations of the problem. That is, replacing the hard, combinatorial problem with an easier, continuous one. By representing the permutation variable in a vector space such as $\mathbb{R}^n$, we can allow ("relax") the variable to take values in the space between some permutations, even though it does not represent a permutation anymore.

Spectral relaxations reformulate the problem into an eigen-problem, for which there exists efficient, polynomial time iterative algorithms. Convex relaxations let the variable take values in the convex hull of the initial set, meaning that instead of having a permutation variable $\pi \in \mathcal{P}$, we work with a variable x that can always be written $x = \sum_j \theta_j \pi^{(j)}$, with $\sum_j \theta_j = 1$, $\theta_j \geq 0$ and $\pi^{(j)} \in \mathcal{P}$ for all j. The loss function is also approximated by a convex function, if it is not already a convex function. Then, the arsenal of convex optimization, including first-order methods (such as gradient descent), can be used to solve the convex problem. These methods have convergence guarantees towards an optimal solution x_* of the convex problem. However, x_* is most often not in the initial search space $\mathcal{P}$, and its projection onto $\mathcal{P}$ may not be optimal for the initial problem.

1.2.1 Greedy Algorithms

Greedy methods typically solve sub-problems at a small neighborhood scale, with exhaustive-search like procedures, and add up the bricks together to form the output sequence. For instance, branch and bound methods have been proposed for small scale seriation [Brusco,

2002]. However, they are impractical for problems of size larger than $n \sim 100$. Other heuristics from combinatorial optimization, *e.g.*, simulated annealing and dynamic programming, have been proposed [Brusco et al., 2008], demonstrating very good experimental performance, but are still limited to small scale problems ($n \sim 100$) [Evangelopoulos et al., 2017a, Hahsler et al., 2008].

1.2.2 Spectral Relaxation

The 2-SUM loss has been extensively used since it can be written as a quadratic, which is convenient for optimization. The following relaxation is also at the core of spectral clustering, where one seeks to cluster the data instead of ordering it (see the tutorial of Von Luxburg [2007] for details). It has a major importance in this thesis. It will serve as a baseline in the experiments, for it is scalable and efficient. Also, in Chapter 3, we will investigate extensions of this method.

For any real symmetric matrix $A \in \mathbf{S}_n$, let $D = \mathbf{diag}(A\mathbf{1})$. $D_{ij} = 0$ if $j \neq i$, and $D_{ii} = \sum_{j=1}^{n} A_{ij}$. D is called the degree matrix of A. If A is the adjacency matrix of an undirected non-weighted graph, D_{ii} is the degree of the node i. Now, let $L = D - A$ be the Laplacian of A. For any vector $f \in \mathbb{R}^n$, we have

$$f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} A_{ij}(f_i - f_j)^2. \tag{1.1}$$

Indeed, observe that

$$\begin{array}{rcl} f^T L f & = & f^T D f - f^T A f \\ & = & \sum_{i=1}^n f_i^2 D_{ii} - \sum_{i,j=1}^n A_{ij} f_i f_j \\ & = & \sum_{i=1}^n f_i^2 (\sum_{j=1}^n A_{ij}) - \sum_{i,j=1}^n A_{ij} f_i f_j \\ & = & \sum_{i,j=1}^n A_{ij}(f_i^2 - f_i f_j) \\ & = & \frac{1}{2} \sum_{i,j=1}^n A_{ij}(f_j^2 + f_i^2 - 2 f_i f_j) \\ & = & \frac{1}{2} \sum_{i,j=1}^n A_{ij}(f_i - f_j)^2, \end{array} \tag{1.2}$$

where we have used the symmetry of A on the penultimate line, and the fact that the symbols of the subscripts i and j could be switched. Hence, 2-SUM can be written as a quadratic optimization problem on permutation vectors,

$$\begin{array}{ll} \text{minimize} & \pi^T L \pi \\ \text{such that} & \pi \in \mathcal{P}_n. \end{array} \tag{2-SUM (quad.)}$$

The spectral relaxation relies on the analysis of the spectrum of the Laplacian. Without loss of generality, we will consider in the following that the similarity matrices have non-negative entries. Remark that adding an offset to a matrix A to make it non-negative does not change

the optimal permutation in 2-SUM, since it translates into an offset in the objective function that is independent of the permutation. From Equation (1.1), we can see that when L is the laplacian of a (symmetric, non-negative) similarity matrix A, it has non-negative eigenvalues. Indeed, recall that the eigenvector associated to the smallest eigenvalue of a matrix $M \in \mathbf{S}_n$ is given by

$$f_1 \in \operatorname*{argmin}_{\|f\|_2=1} f^T M f, \tag{1.3}$$

and for $i > 1$, the i-th smallest eigenvector is given by

$$f_i \in \operatorname*{argmin}_{\|f\|_2=1,\, f^T f_j=0,\, j<i} f^T M f, \tag{1.4}$$

Now, observe that the right-hand-side of Equation (1.1) is always non-negative, hence no eigenvector f can have negative eigenvalue. Note also that $\mathbf{1}$ is an eigenvector of L, with associated eigenvalue 0. Indeed, if all f_i in Equation (1.1) are equal, then the right-hand-side is 0. To summarize, let $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, $\Lambda \triangleq \mathbf{diag}\,(\lambda_1, \ldots, \lambda_n)$, $\Phi = (\mathbf{1} = f_1, \ldots, f_n)$, be the eigendecomposition of $L = \Phi\Lambda\Phi^T$.

The difficulty in solving 2-SUM (quad.) does not come from the objective function (the unconstrained minimization of a quadratic is one of the easiest optimization problems), but from the constraint set (combinatorial). As observed by Lim and Wright [2014], a permutation vector can be characterized by the three following constraints,

$$\begin{aligned} &\pi(i) \in [n], \quad i = 1, \ldots, n && \text{(integer constraint)} \\ &\pi^T \mathbf{1} = n(n+1)/2 && \text{(sum constraint)} \\ &\|\pi\|_2^2 = n(n+1)(2n+1)/6. && \text{(norm constraint)} \end{aligned}$$

The membership to $\mathcal{P}$ can be equivalently enforced by the three previous constraints. The spectral relaxation of 2-SUM (quad.) studied in Atkins et al. [1998] basically drops the (integer constraint), as noted in Ding and He [2004]. Since $\mathbf{1}$ is in the nullspace of L, the objective in 2-SUM (quad.) does not vary by subtracting $((n+1)/2)\mathbf{1}$ to the permutation vectors. Hence, the (sum constraint) can be transformed into $\pi^T \mathbf{1} = 0$ in the 2-SUM problem. Finally, the quadratic 2-SUM objective is homogeneous, and we can therefore chose to rescale the permutation vectors and transform the (norm constraint) into $\|\pi\|_2^2 = 1$. All in all, the spectral relaxation of 2-SUM reads,

$$\begin{aligned} \text{minimize} \quad & f^T L f \\ \text{such that} \quad & f^T \mathbf{1} = 0,\ \|f\|_2 = 1. \end{aligned} \tag{Spectral Relax.}$$

Given that $\mathbf{1}$ is the first eigenvector of L, we recognize Equation (1.4) with $i = 2$. The solution f_* of the Spectral Relax. is the eigenvector associated to the second smallest eigenvalue of the laplacian of A. However, since we relaxed the integer constraint, the solution f_* is in general not a permutation. In order to recover a permutation from f_*, we can project f_* back onto the set of permutation vectors, *i.e.*, find $\tilde{\pi} \in \operatorname{argmin}_{\pi \in \mathcal{P}_n} \|f_* - \pi\|_2$. The computation of this projection actually boils down to sorting the entries of f_* : $\tilde{\pi}$ is such that $f_*(\tilde{\pi}(1)) \leq \ldots \leq f_*(\tilde{\pi}(n))$. Note that this projection π_* is not guaranteed to be the optimum of 2-SUM in general. If A is the adjacency matrix of a disconnected graph, *i.e.*, there are connected components in the graph with no edges in between, then f_* has constant values inside each connected component (and can be used for clustering [Von Luxburg, 2007]). This makes the projection degenerate, since there is no best way to sort a constant vector. However, Seriation and 2-SUM aim to find a global ordering integrating all the local similarities. If the graph is disconnected, say, in two clusters, then only the sub-orderings restricted to each cluster will matter. Therefore, we restrict ourselves to the case where A is the adjacency matrix of a connected graph (for any pair of nodes (i, j), there exists a path going from i to j). Then, the second smallest eigenvalue, called the Fiedler value, is positive : $\lambda_2 > 0$, and there are meaningful variations in f_*. Some values of f_* can still be equal, and Atkins et al. [1998] propose a method to deal with these degeneracies with so-called PQ-trees, but this situation is scarcely encountered in practice when dealing with real or noisy data.

We summarize the spectral method in Algorithm 1.1. A major result from Atkins et al.

Algorithm 1.1 Spectral ordering [Atkins et al., 1998]

Input: Connected similarity matrix $A \in \mathbb{R}^{n \times n}$
1: Compute Laplacian $L_A = \mathbf{diag}(A\mathbf{1}) - A$
2: Compute second smallest eigenvector of L_A, f_1
3: Sort the values of f_1
Output: Permutation $\sigma : f_1(\sigma(1)) \leq \ldots \leq f_1(\sigma(n))$

[1998] states that Algorithm 1.1 solves Seriation whenever it is feasible. This is formalized in Theorem 1.2.1

Theorem 1.2.1. *Atkins et al. [1998, Theorem 3.3] Let A be a pre-R matrix with a simple Fiedler value and whose Fiedler vector f_* has no repeated values. Let π be the permutation obtained by sorting the values of f_* by increasing value (the output of Algorithm 1.1), and Π the associated permutation matrix. Then, $\Pi A \Pi^T$ is an R-matrix.*

1.2.3 Convex Relaxations

The spectral relaxation is powerful, but it is intrinsically specific to the (2-SUM (quad.)) objective, and may perform poorly when the input matrix A is noisy. Convex relaxations are more flexible and come with theoretical guarantees, though they are often heavier computationally.

Formally, consider the constrained optimization problem,

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in C, \end{array} \tag{1.5}$$

in $x \in \mathbb{R}^d$, where f is a smooth (its gradient is Lipschitz continuous) convex function (*e.g.*, the (2-SUM) objective) and C is a closed set (*e.g.*, $\mathcal{P}$). The convex relaxation approach approximates the non-convex set C with a convex one. Specifically, it lets the variable be in a convex set that contains the original set C. The tightest relaxation consists in considering the *convex hull* of the original set. For the record, the convex hull of a set C, denoted $\text{hull}(C)$, is the set of all convex combinations of points in C,

$$\text{hull}(C) = \left\{\theta_1 x_1 + \ldots + \theta_k x_k \mid \theta_i \geq 0,\ i = 1, \ldots, k,\ \theta_1 + \ldots + \theta_k = 1\right\}.$$

It is the smallest convex set that contains C. The convex relaxation of (1.5) reads

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \text{hull}(C). \end{array} \tag{1.6}$$

Then, convex optimization methods can be used to solve (1.6). For instance, projected gradient descent is a first order iterative method. Given the current iterate x_t, it computes the gradient of the function at x_t, takes a step in the opposite direction of the gradient, and projects back onto the constraint set (since $x_{t+1/2}$ may no longer be in $\text{hull}(C)$), with,

$$\begin{aligned} x_{t+1/2} &= x_t - \gamma \nabla f(x_t), && \text{(gradient step)} \\ x_{t+1} &= \operatorname*{argmin}_{x \in \text{hull}(C)} \|x - x_{t+1/2}\|. && \text{(projection step)} \end{aligned}$$

In practice, although we can mathematically define $\text{hull}(C)$, it may be computationally challenging to perform the projection step.

Algorithm 1.2 Conditional gradient algorithm for constrained problem (1.5)

Inputs: Initial point $x_0 \in \text{hull}(C)$, target precision ε

for $t = 0, \ldots$ **do**

Solve linear minimization oracle

$$s_t = \underset{s \in \text{hull}(C)}{\text{argmin}} \langle \nabla f(x_t), s \rangle \tag{1.7}$$

Get estimated gap

$$\Delta_t = \langle x_t - s_t, \nabla f(x_t) \rangle \tag{1.8}$$

if $\Delta_t \leq \varepsilon$ **then** Stop **end if**

Set

$$x_{t+1} = x_t + \frac{2}{t+2}(s_t - x_t)$$

end for

Output: $\hat{x} = x_t$

The conditional gradient method (a.k.a Frank-Wolfe [Frank and Wolfe, 1956, Lacoste-Julien and Jaggi, 2015]), described in Algorithm 1.2, can circumvent this problem. It uses a linear minimization oracle (1.7) to produce a sequence of iterates that remain in $\text{hull}(C)$ *by construction*. When f is convex and smooth, Algorithm 1.2 has a guaranteed convergence rate of $O(1/t)$ towards a solution x^* of (1.6). It also provides an estimated duality gap (1.8), due to an inequality involving the convexity of f. Note that using Algorithm 1.2 is dependent upon the availability of an efficient linear minimization oracle to solve Equation (1.7). If this step can be performed efficiently (with a computationally cheap algorithm), then Algorithm 1.2 is likely to be efficient. The linear minimization oracle (LMO) depends solely on the set C. Indeed, in Equation (1.7), the gradient at x_t is given, hence the problem is no other but a linear program on $\text{hull}(C)$. For permutation problems, one can choose to represent permutations with vectors or matrices, resulting in two possible sets C.

Permutation matrices

The set of permutation matrices can be written with the following constraints,

$$\mathcal{P}_n = \left\{ \Pi \in \{0,1\}^{n \times n} \;\middle|\; \Pi \mathbf{1} = \mathbf{1},\ \Pi^T \mathbf{1} = \mathbf{1} \right\}. \tag{1.9}$$

The two stochastic constraints impose that in a permutation of the integers $1, \ldots, n$, each integer $i \in [n]$ appear once, and only once. The convex hull of $\mathcal{P}$ is obtained by relaxing the integer constraints $\Pi_{ij} \in \{0,1\}$ into $\Pi_{ij} \in [0,1]$. The resulting set, called the Birkhoff polytope, is defined as follows,

Definition 1.2.2. *The convex hull of the set of $n \times n$ permutation matrices, called the* Birkhoff polytope $\mathcal{B}_n$*, is the set of all doubly-stochastic $n \times n$ matrices:*

$$\mathcal{B}_n = \{X \in \mathbb{R}^{n \times n} \mid X \geq 0, X\mathbf{1} = \mathbf{1}, X^T\mathbf{1} = \mathbf{1}\}.$$

It is a polyhedron, *i.e.*, it is defined by linear constraints. Work akin to seriation involving $\mathcal{B}$ include that of Vogelstein et al. [2011], who used the conditional gradient Algorithm 1.2 to minimize the objective of QAP(A,B) over the Birkhoff polytope $\mathcal{B}$.

Each iteration of the algorithm involves solving a linear program in $\mathcal{B}$ (1.7), which is achieved using a Hungarian matching algorithm [Kuhn, 1955]. Again, the membership to $\mathcal{B}$ needs not to be enforced explicitly when using the conditional gradient algorithm, since the sequence of iterates remain in $\mathcal{B}$ by construction (using convex combinations of points in $\mathcal{B}$).

More recently, Fogel et al. [2013] proposed a convex relaxation of 2-SUM in $\mathcal{B}$. In its most basic form, it solves the following problem,

$$\begin{array}{ll} \text{minimize} & \mathbf{g}^T\Pi^T L\Pi\mathbf{g} \\ \text{subject to} & \Pi \in \mathcal{B}_n. \end{array} \tag{1.10}$$

The objective function is still the 2-SUM objective, but is written in the variable Π, and is a quadratic function of the variable. The constraints appearing in Definition 1.2.2 are linear, hence, the problem can be solved with standard optimization solvers such as MOSEK [Andersen and Andersen, 2000], using, *e.g.*, interior point methods. However, the variable Π has n^2 entries, which makes this approach limited to medium-scale problem. Refinements of this approach, using for instance the Frank-Wolfe algorithm, can make the problem scale to larger sizes [Fogel et al., 2013].

Permutation vectors

A permutation vector has only n entries (compared to n^2 for a permutation matrix). Therefore, a relaxation on the set of permutation vectors may be more scalable. We have seen earlier three constraints defining the set of permutation vectors (also written $\mathcal{P}$ in this subsection). Still, the (norm constraint) is not linear. Yet, convex optimization routinely solves problems with a quadratic objective and linear equality and inequality constraint, but quadratic constraints are often challenging, hence this formulation is not adapted to convex optimization. The set of

permutation vectors can also be defined as follows [Lim and Wright, 2014],

$$\begin{aligned} &\pi(i) \in [n], \quad i = 1, \dots, n && \text{(integer constraint)} \\ &\pi^T \mathbf{1} = n(n+1)/2 && \text{(sum constraint)} \\ &\sum_{i \in S} \pi_i \leq \sum_{i=1}^{|S|} (n+1-i) \text{ for all } S \subset [n] && \text{(partial sum constraints)} \end{aligned}$$

The convex hull of the set of permutation vectors is also obtained by relaxing the (integer constraint), and is defined as follows [Lim and Wright, 2014],

Definition 1.2.3. *The* permutahedron $\mathcal{PH}_n$*, the convex hull of the set of permutation vectors of size* n*, is*

$$\mathcal{PH}_n = \left\{ x \in \mathbb{R}^n \;\middle|\; \sum_{i=1}^{n} x_i = \frac{n(n+1)}{2}, \sum_{i \in S} x_i \leq \sum_{i=1}^{|S|} (n+1-i) \text{ for all } S \subset [n] \right\}.$$

Still, there are 2^n subsets S of $[n]$, hence it is impractical to enforce explicitly all 2^n inequalities from (partial sum constraints) in a convex optimization solver.

Lim and Wright [2014] proposed a convex relaxation of 2-SUM in $\mathcal{PH}$, using an extended formulation of $\mathcal{PH}_n$ from Goemans [2014], based on sorting networks. This formulation represents $\mathcal{PH}_n$ with $\Theta(n \log n)$ variables and constraints, instead of $\Theta(n^2)$ for permutation matrices. In a nutshell, the extended formulation states that a vector x belongs to $\mathcal{PH}_n$ if it constitutes the first n entries of a larger vector respecting some linear inequality constraints,

$$x \in \mathcal{PH}_n \quad \text{if} \quad x \in \{x^{\text{in}} \mid (x^{\text{in}}, x^{\text{rest}}) \in \mathcal{SN}_n\}, \tag{1.11}$$

where $\mathcal{SN}_n$ is a polyhedron, *i.e.*, the membership of $x^{\text{extd.}} = (x^{\text{in}}, x^{\text{rest}})$ to $\mathcal{SN}_n$ is enforced by linear inequalities. It enables solving the following relaxation,

$$\begin{aligned} \text{minimize} \quad & x^T L x \\ \text{subject to} \quad & x \in \mathcal{PH}_n \end{aligned} \tag{1.12}$$

by calling a convex optimization solver on a variable $x^{\text{extd.}}$ with linear inequality constraints, whose n first entry only are used in the objective.

Even more recently, Evangelopoulos et al. [2017a] attempted to solve 2-SUM with the conditional gradient Algorithm 1.2 in $\mathcal{PH}$. The key observation is that the LMO (1.7) can be computed efficiently here, as it boils down to sorting the entries of the gradient (which is a

vector of size n). Indeed, if y denotes the gradient of f at x_t, Equation (1.7) can be written as,

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{n} y_i s_i \\ \text{subject to} & s \in \mathcal{PH}_n. \end{array} \tag{1.13}$$

Now, observe that the minimum of a linear function on a polyhedron resides on a vertex, hence we can restrict the variable s to be in $\mathcal{P}_n$ instead of $\mathcal{PH}_n$. The permutation that minimizes (1.13) is the one with maximal weight (n) on the largest entry of y, with second maximal weight ($n-1$) on the second largest entry of y, etc. Formally, if π is the permutation that sorts the entries of s increasingly,

$$y_{\pi(1)} \leq y_{\pi(2)} \leq \cdots \leq y_{\pi(n)}$$

the solution s^* of Equation (1.13) is the inverse permutation of π, defined by,

$$s^*_{\pi(k)} = k, \text{ for } k = 1, \ldots, n.$$

Algorithm 1.2 can be implemented with $\text{hull}(C) = \mathcal{PH}_n$, where the LMO (1.7) consists in sorting the entries of the gradient as described previously, which has algorithmic complexity $O(n \log n)$.

Symmetry Breaking

A crucial issue with the convex relaxations presented above is that the optimum of the convex problems (1.10) and (1.12) are trivial and non-informative.

Indeed, recall that $\mathbf{1}$ is in the nullspace of L, and that L is positive semi-definite. The geometrical center of $\mathcal{PH}_n$, $\mathbf{c}_n = \frac{n+1}{2}\mathbf{1}_n$, therefore minimizes (1.12) (where the objective is 0). Similarly, the geometrical center of $\mathcal{B}_n$, $\mathbf{C}_n = \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$, minimizes (1.10). Yet, the respective centers of $\mathcal{PH}_n$ and $\mathcal{B}_n$ are at the same distance from any permutation (vector or matrix, respectively). Therefore, the task of projecting them back onto the set of permutations is totally degenerate.

This is essentially due to the following symmetry. The 2-SUM objective is invariant by flipping a permutation. For instance, $(1, \ldots, n)$ and $(n, \ldots, 1)$ yield equal score. Formally, the operator T_n defined by $T_n(\pi) = (n+1) - \pi$ leaves 2-SUM invariant. In order to overcome this issue, Fogel et al. [2013] augmented the convex relaxation (1.10) as follows.

- Introduce a *tie-breaking* constraint, $\pi_1 + 1 \leq \pi_n$, or in matrix form, $e_1^T \Pi \mathbf{g} + 1 \leq e_n^T \Pi \mathbf{g}$, to resolve ambiguity about the direction of the ordering.

- Add a penalty to the Froebenius norm of $P\Pi$, with $P = \mathbf{I}_n - \mathbf{C}_n = \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$, to push the solution away from the center $\mathbf{C}_n$.

Additionally, their convex formulation allows to incorporate ordering constraints of the form $\pi_i - \pi_j \leq \delta_k$, to leverage prior knowledge on the ordering (and, if this prior knowledge is consistent, help breaking the symmetry). Also, they average over several perturbations of $\mathbf{g}$ to gain in robustness. All in all, the enhanced problem (1.10) reads,

$$\begin{array}{ll} \text{minimize} & \frac{1}{p}\,\mathbf{Tr}(Y^T\Pi^T L_A \Pi Y) - \frac{\mu}{p}\|P\Pi\|_F^2 \\ \text{subject to} & D\Pi\mathbf{g} \leq \delta, \\ & \Pi\mathbf{1} = \mathbf{1},\ \Pi^T\mathbf{1} = \mathbf{1},\ \Pi \geq 0, \end{array} \qquad \text{(Matrix-Relax. 2-SUM)}$$

where $D\Pi\mathbf{g} \leq \delta$ contains the tie-breaking and *a priori* constraints, the second line of constraints imposes $\Pi \in \mathcal{B}_n$, each column of $Y \in \mathbb{R}^{n\times p}$ is a perturbed version of $\mathbf{g}$, and μ is a regularization parameter. Keeping $\mu < \lambda_2(L_A)\lambda_1(YY^T)$ ensures that (Matrix-Relax. 2-SUM) remains convex.

Lim and Wright [2014] adapted these improvements to the permutation vector formulation, yielding the following problem,

$$\begin{array}{ll} \text{minimize} & x^T L x - \mu\|Px\|_2^2 \\ \text{subject to} & Dx \leq \delta, \\ & x \in \mathcal{PH}_n. \end{array} \qquad \text{(Vector-Relax. 2-SUM)}$$

The objective of (Vector-Relax. 2-SUM) can also be written $x^T(L-\mu P)x$, making it clear that the objective remains convex when the regularization parameter is smaller than the Fiedler value, *i.e.*, $\mu < \lambda_2(L)$.

Finally, Evangelopoulos et al. [2017a] also use the above regularization, *i.e.*, they minimize the objective from (Vector-Relax. 2-SUM) in $\mathcal{PH}_n$ with the Frank-Wolfe Algorithm 1.2. However, rather than choosing a fixed value of the regularization parameter μ, they iteratively increase it in outer-loops of a continuation (a.k.a, graduated non-convexity) scheme. They start with $\mu < \lambda_2(L)$, and increase it until $\mu > \lambda_n(L)$. Their approach produces a sequence of solutions to sub-problems following a path from $\mathbf{c}_n$ to a permutation (*i.e.*, a vertex of $\mathcal{PH}_n$).

1.3 Applications to Genomics

Let us present the outline of *de novo* genome assembly, one of the key challenges (the *repeats*), and some specific sequencing technologies.

1.3.1 De novo Genome Assembly

DNA sequencing refers to the process of determining the nucleotide order of a given DNA fragment. There are four possible nucleotides (also called bases), Adenine, Cytosine, Thymine or Guanine, which we represent by their first letter, A, C, T, or G. DNA sequencing results in a linear sequence forming a string in the 4-letters alphabet {A,T,G,C}, *e.g.* 'AATCGCG'.

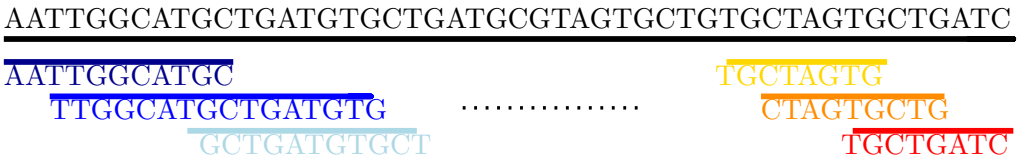


Figure 1.4: Illustration of the genome assembly process. The long, black sequence is the DNA strand we wish to sequence. The colored fragments (called *reads*) are the input for the assembly. However, these fragments are given "in a bag", *i.e.*, we ignore their positions and their order in the genome (they are correctly located in the Figure for illustration only). Thanks to the overlaps, we can recover the full sequence from the fragments.

In practice, we use a device that outputs an electronic signal from a DNA fragment. The sequence can then be deduced from the signal (this inference process is called basecalling). However, due to limitations inherent to the physicochemical process enabling us to "read" the DNA sequence, we can only access partial sub-fragments (called *reads* hereafter) extracted from the input DNA strand.

To overcome this limitation, the idea of shotgun sequencing is to clone the genome multiple times, and sequence pieces of the clones at random locations in the genome. Consequently, the genome is oversampled and all parts are covered by multiple reads with high probability. Hence, we have redundant information : there are overlaps between the reads, and we can assemble all the pieces together to retrieve the input DNA strand. For instance, the example sequence 'AATCGCA' could yield the three following sub-sequences in an experiment, {'AATC','ATCG','CGCA'}. We see that 'AATC' overlaps 'ATCG' and 'ATCG' overlaps 'CGCA', enabling us to reconstruct the full word. Nonetheless, we ignore the locations of the reads on the genome, and we must infer them from the overlap information, in the way of a jigsaw puzzle. A schematic illustration is given in Figure 1.4.

In some applications, we wish to sequence the genome of the member of a species for which we already have a reference genome (for instance, a human). Then, we can find the locations of the reads by mapping them to a reference before reconstructing the full sequence of the given individual. In contrast, *de novo genome assembly* refers to the task of reconstructing the whole DNA strand from the fragments (*reads*), sampled at random locations, without any reference. In a *de novo* assembly experiment, the *coverage* c is the sum of the lengths of the reads divided by the length of the genome. In average, if all reads had the same length and were sampled uniformly along the genome, a given read would overlap with c other reads.

The DNA has a double-strand structure, as shown on Figure 1.5. Each strand has an orientation determined by the direction in which DNA was replicated (from one end denoted 5' to the other, denoted 3'). The two strands are complementary, *i.e.*, one is the reverse complement of the other, where the reverse complement of a sequence $(s_1, \ldots, s_n)$ is defined as $(\bar{s}_n, \ldots, \bar{s}_1)$, with $\bar{A} = T$, $\bar{T} = A$, $\bar{C} = G$, and $\bar{G} = C$. In an assembly experiment, the reads may

come from either strand. Therefore, it is necessary to consider both possible orientations when computing the overlaps. Also, when using the Seriation framework, the orientation information is not taken into account in the similarity matrix, but the layout needs to be consistent with the orientation constraints (one read has only one global orientation).

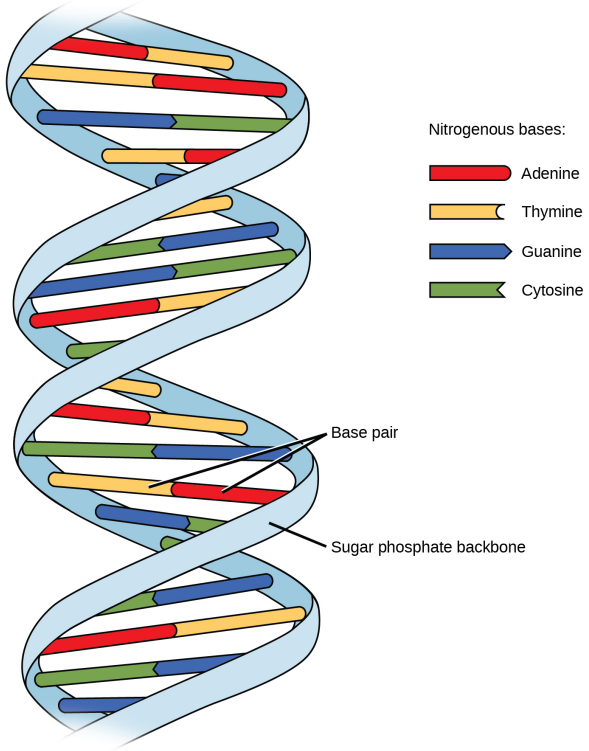


Figure 1.5: Double-helix 3D structure of DNA. The DNA has two strands, where each C base is bounded with G, and T with A. One strand is the reverse component of the other, *i.e.*, it is obtained by reading it backwards and replacing all Cs by Gs and all Ts by As (and vice-versa). In a genome assembly experiment, the reads may come from either strand. Therefore, one has to consider both orientations (strands) for each read when determining the layout. It adds consistency constraints on the layout, and is necessary to perform consensus. Figure from BCcampus on https://opentextbc.ca

Typical genome lengths are a few Mb (10^6 bases) for a bacteria, a few tens of Mb (10^7 bases) for a yeast, a few thousands Mb (10^9 bases) for the human genome, and can reach hundreds of Gb for some plants. The typical size of the reads is of a few hundreds bases for one of the major sequencing technologies (Illumina), and a few thousands to tens of kb (10^4 bases) for third generation sequencing technologies.

Overlap-Layout-Consensus (OLC) is a major assembly paradigm based on three main steps. First, compute the overlaps between all pairs of read. This provides a similarity matrix A, whose entry (i, j) measures how much reads i and j overlap (and is zero if they do not).

Then, determine the layout from the overlap information, that is to say find an ordering and positioning of the reads that is consistent with the overlap constraints. Finally, given the tiling of the reads obtained in the layout stage, the consensus step aims at determining the most likely DNA sequence that can be explained by this tiling.

Computing the overlaps between all pairs of reads involves $n(n-1)/2$ pairwise comparisons. Dynamic programming can be used to perform sequence alignment [Smith and Waterman, 1981] and provide an overlap score. However, such methods are impractical with $n \sim 10^4$ reads. Instead, methods based on hashing [Li, 2016, Berlin et al., 2015] can be used. In this work, we will use such software to compute the overlaps (sometimes called overlapper) as a black-box, providing, for all pairs of sequences, 1. an overlap score (length of the overlap, if any) and 2. overlap detailed information, *i.e.*, position of the overlap on each of the two sequences, and mutual orientation (a DNA strand can be read in two possible directions, as shown in Figure 1.5, and the orientation of an overlap indicates whether the two reads come from the same strand or from opposite strands).

The layout step, akin to solving a one dimensional jigsaw puzzle, is a key step in the assembly process, and fits in the framework of Seriation. In an ideal setting, a given read has a significant overlap with the next read, a smaller overlap with the one after, and so on, until it no longer overlaps the subsequent reads. For instance, on Figure 1.4, the leftmost read (darkblue) has a large overlap with the second read (blue), a small overlap with the third one (light blue), and does not overlap the following reads. Hence, the similarity matrix from an *ideal* genome assembly experiment is an R-matrix. Also, it is a sparse, banded matrix (it has non-zero values only within a band, corresponding to the maximal distance between two overlapping reads).

Finally, the consensus can be performed through multi-sequence alignment. In the above example with the 'AATCGCA' sequence, it could be recovered from the three reads as follows,

```
AATC
 ATCG
    CGCA
-------
AATCGCA
```

However, in practice, there are basecalling errors in the reads. There can be substitution errors (a nucleotide is replaced by another one), insertions (a nucleotide is added in the sequence), and deletions (a nucleotide is removed from the sequence). Performing the consensus therefore requires more than majority-vote like rules. Hopefully, efficient implementations of algorithms based on dynamic programming exist for multiple sequence alignment, with reasonable numbers and sizes of sequences to align [Lee et al., 2002, Sović et al., 2016].

1.3.2 Repeated Regions (repeats)

The basecalling errors represent a challenge in the assembly process, as they may induce errors in the overlap computation (leading to erroneous layout), and make the consensus derivation

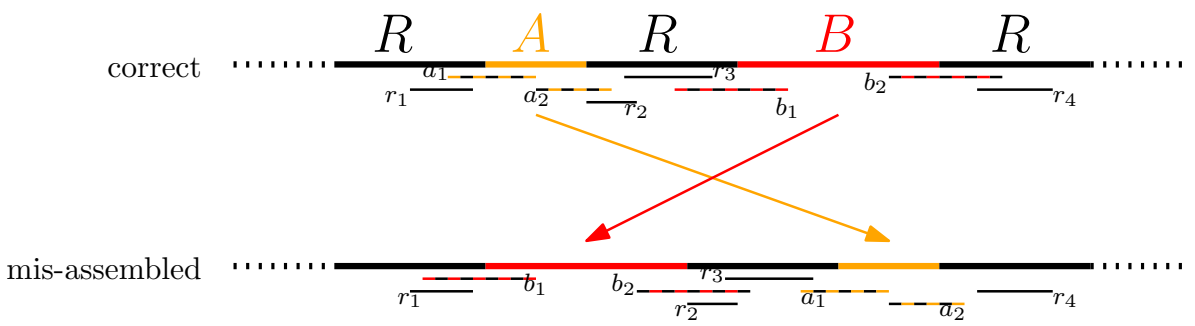


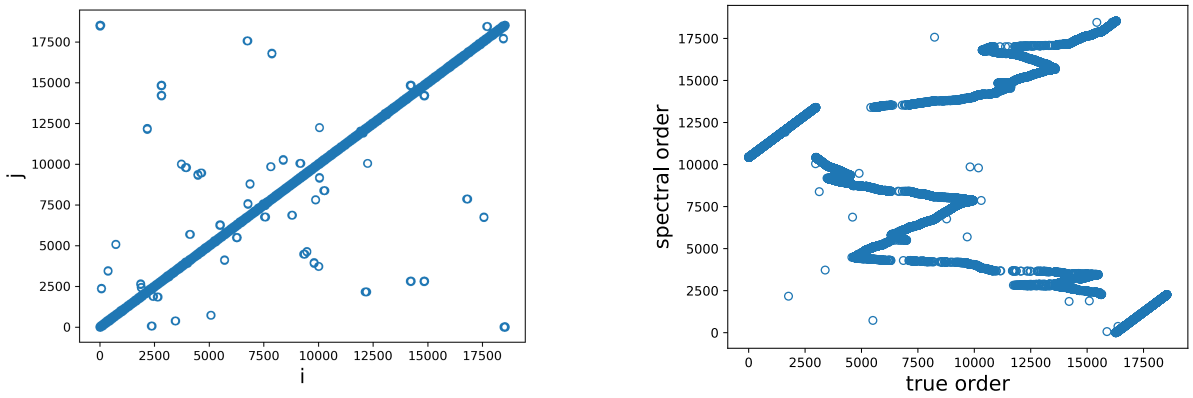
Figure 1.6: Genome rearrangement around a repeat that occurs in three places. The reads at the junction between one of the two sequences A, B and a repeat R have apparent overlaps with the two other repeats, resulting in two possible layouts consistent with the overlaps constraints (the correct and the mis-assembled).

more challenging. Yet an even more challenging issue is the presence of repeated regions (called *repeats*), *i.e.*, stretches of DNA that occur multiple times in near-identical copies throughout the genome. Repeats can be as long as a few thousands of nucleotides. As illustrated in Figure 1.6, a read that joins the end of a repeat and of another region in the genome will have apparent overlaps with the other repeats. For instance, on Figure 1.6, the read a_1 overlaps with r_1, but also r_3. Analogously, b_1 overlaps with r_3, but also r_1. In the end, the assembled layout [bottom] is consistent with the overlaps, yet it is incorrect (compared to the original sequence [top]).

From an algorithmic perspective, the repeats may compromise the application of Seriation to genome assembly. Indeed, they induce overlaps between reads that can be far apart in the genome. For instance, in Figure 1.6, b_2 overlaps with r_1, but not with b_1, a_2 nor a_1. Yet, b_2 is further away from r_1 than from a_2 and a_1. The resulting, correctly ordered similarity matrix therefore violates the Robinson property from Definition 1.1.1.

Figure 1.7a shows the overlap-based similarity matrix between reads from a bacterium (*Escherichia coli*). There are almost $n \simeq 20000$ reads, with a coverage $c \simeq 30X$. Therefore, with a repeat-free genome we would expect a similarity matrix Robinsonian and roughly banded with a bandwidth of order 30. Still, we observe a few out-of-band terms on Figure 1.7a, due to repeats. The resulting ordering found by the spectral Algorithm 1.1 shown in Figure 1.7b is corrupted.

A complete introduction to shotgun sequence assembly, presenting the repeats problem, assembly paradigms and engineering challenges in details can be found in Pop [2004], Nagarajan and Pop [2013].



(a) Similarity Matrix

(b) Ordering Found

Figure 1.7: Similarity matrix between reads from an *E. coli* genome (1.7a), and the ordering found with Algorithm 1.1 *vs* the true ordering (1.7b). If the ordering found was identical to the reference ordering, we would observe a straight line. The mis-assembly is imputable to the out-of-diagonal points observed in 1.7a.

1.3.3 Sequencing technologies

Several sequencing technologies exist and produce data with different characteristics, leading to technology-specific algorithmic paradigms. The most widely used sequencing technologies fall into the following categories (see Nagarajan and Pop [2013] for a more complete survey of sequencing tools and assembly algorithms).

Next Generation Sequencing (NGS)

What was formerly called Next Generation Sequencing (NGS) or High Throughout sequencing usually refers to several short-reads technologies that parallelize the sequencing process. One of the most widely used is commercialized by Illumina and based on the sequencing by synthesis process. It synthesizes and amplifies the reads with DNA polymerase, and uses imagery with fluorescent markers to read the (short) sub-sequences base per base. These technologies have dramatically reduced the cost of DNA sequencing in the late 1990s.

Typical data produced with Illumina are millions of reads of a few hundreds bases, with accuracy exceeding 99% (the read accuracy is the proportion of correctly sequenced bases, as opposed to the sequencing errors listed above [substitution, deletion, insertion]). Such sequencers provide additional information to be used in the assembly: the reads are given in pairs, and we know the distance between two paired reads (it is the same for all pairs and is substantially larger than the reads length). This adds structural constraints on the layout, and provides information of longer range than the overlaps.

Third Generation Sequencing

More recently, modifications to the sequencing by synthesis technology by Pacific Biosciences (PacBio) gave rise to Single Molecule Real Time Sequencing (SMRT), a method capable of producing reads tens of thousands nucleotides (10^4b) long. However, the reads accuracy dropped to $\sim 87\%$.

An even more recent long-reads technology, introduced by Oxford Nanopore Technology (ONT), is based on the observation of ion current when a DNA strand passes through a nanopore. It also produces reads of a few tens of thousands bases with lower accuracy than short-reads, although the reads accuracy tends to improve and is claimed to reach 92 to 97% now.

Despite its lower accuracy, such "long-reads" technology is highly valuable for *de novo* assembly, since the length of the reads is larger than most repeats, making them easier to resolve. Moreover, while short-reads assemblers have been enhanced with number of handcrafted heuristics throughout the years, long-reads assemblers are quite novel and the algorithmic design of dedicated assemblers is still burgeoning, as the technology keeps evolving. Therefore, the *de novo* assembly experiments conducted throughout this thesis will use long-reads.

1.3.4 State of the Art of Assembly Methods

Let us briefly review the key principles used in assembly computational tools.

Assembly paradigms

Most assembly methods rely on (at least) one of the following paradigms: greedy methods, De Bruijn graphs, and overlap-layout-consensus (OLC).

Greedy methods seek to reconstruct the sequence in a step-by-step fashion where only local information is used at each step. Given n reads to assemble, a prototypical greedy assembly algorithm starts with picking a first read (called `read 1`), and then searches for the read that has the largest overlap with `read 1` among the $n-1$ other reads. It then merges those two reads into a consensus sequence (called `seq 1`), before searching for the read that has the largest overlap with `seq 1` among the $n-2$ remaining reads, *etc.*

De Bruijn graphs (DBG) methods are based on the Eulerian path problem on a graph, which aims to find a path that visits every edge once. The application of DBG to genome assembly is detailed in, *e.g.*, Compeau et al. [2011]. Given a set of reads, we can construct a graph where the edges are the k-mers (sub-words of length k) appearing in the reads, and the nodes are the prefix and suffix corresponding k-1-mers. For instance, given the set of reads {'AATC','ATCG','CGCA'} virtually sequenced from the genome 'AATCGCA', and chosing k=3, the first read 'AATC' contains two 3-mers, 'AAT' and 'ATC'. Thus we add to the graph the edge 'AAT' between nodes 'AA' and 'AT' (prefix and suffix of 'AAT'), and the edge 'ATC'

between the node 'AT' and the new node 'TC'. Then, the read 'ATCG' contains two 3-mers, 'ATC' and 'TCG'. The edge and nodes corresponding to 'ATC' are already present in the graph (because this 3-mer was present in the read 'AATC', which overlaps with 'ATCG'). The 'TCG' 3-mer gives rise to two new nodes and a new edge in the graph. Doing the same operation with the read 'CGCA', we obtain the graph from Figure 1.8. A key computational benefit of De

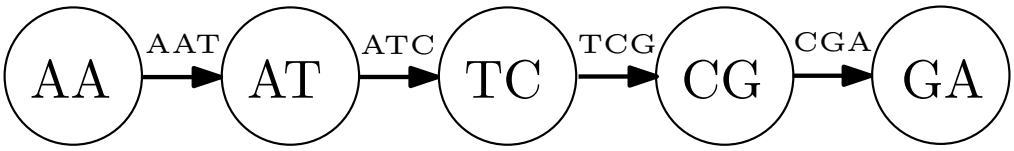


Figure 1.8: De Bruijn graph based on 3-mers from the reads 'AATC','ATCG','CGCA'. There is only one Eulerian path, yielding the consensus sequence 'AATCGCA'.

Bruijn graph methods is that although it is necessary that consecutive reads overlap in order to find a contiguous path in the graph, there is no need to compute the overlaps between the pairs of reads. However, such an exact k-mers based approach is sensitive to sequencing errors. Thus, De Bruijn graphs methods are suited to Next Generation Sequencing, as it provides large volumes (for which the pairwise alignment of reads can be prohibitive) of accurate (with few sequencing errors) data.

The Overlap-Layout-Consensus (OLC) paradigm has already been introduced in the present introduction. It consists in three steps. First, compute the overlaps between the reads. This can be done by computing pairwise alignment between all pairs of reads (with, *e.g.*, a dynamic programming algorithm [Smith and Waterman, 1981]), but more computationally efficient approaches based on hashing [Myers, 2014, Berlin et al., 2015, Li, 2016] can be used in practice. Then, the layout step searches for an ordering and positioning of the reads consistent with the overlap information. Notably, overlapping reads must be placed nearby in the layout found. For instance, a method akin to De Bruijn graphs could be applied to the layout step, where the nodes of the graph of interest are the reads, and edges represent the overlap (if any) between the reads. Then, the layout can be found by searching for a path that goes through all *nodes* once, where the sum of the weights along the path is maximized (Hamiltonian path problem). Finally, once the layout of the reads is obtained, the consensus step derives a sequence in a majority-vote fashion, with, *e.g.*, multiple sequence alignment.

Assembly pipelines

In practice, an approach such as DBG fails to uniquely assemble genomes in the presence of repeats. Hence, in order to meet the challenges caused by repeats and sequencing errors, assembly software consist in pipelines involving several components. Most NGS assembly pipelines involve the three following steps : 1. *Contig* generation; 2. *scaffolding*, 3. *finishing*.

The contigs are contiguous genomic fragments, *i.e.*, the result of the assembly of a subsequence of the whole genome. Typically, heuristics are used to identify reads originating from repeats, and the overlap graph is cut into separate, repeat-free connected components. In these repeat-free sub-graphs, a De Bruijn graph approach can be successfully applied to perform the partial assembly of the contigs. Thus, this first step assembles the "easy" regions with no ambiguities.

Then, the scaffolding consists in determining the layout of the contigs (which can be thought of as blocks of pieces of jigsaw puzzle, where a single piece would be a read) together, *i.e.*, determining their relative position and orientation. Although NGS data provides short reads, some longer-range additional *pairing* information is available. The scaffolding aims to lay out the contigs in a way that is consistent with these pairing constraints. It can be done, for instance, with a greedy method [Huson et al., 2002].

Finally, finishing seek to fill the gaps between the ordered contigs.

Although most methods for scaffolding do not explicitly handle the presence of repeats, some effort has been made to take them into account in an integer optimization framework. While Weller et al. [2015], Davot et al. [2018], Tabary et al. [2018] derived theoretical complexity and approximation bounds for solving the scaffolding problem in the presence of repeats, François et al. [2016], Francois et al. [2017] proposed a global mixed integer linear programming (MILP) approach to solve the scaffolding problem, by imposing overlap and pairing based constraints.

Due to the higher sequencing error-rate occurring in third generation sequencing data, most long-reads assembly pipelines, such as Canu [Koren et al., 2017], begin with a *correction* step. Although there is no *mate pair* or *pair end* side information with third generation data to be used in the scaffolding, the length of the reads allow to resolve contigs smaller than the read length, resulting in larger contigs than with NGS.

1.3.5 Hi-C: Spatial Conformation Data

Besides standard DNA sequencing techniques, an interesting recent development called Hi-C and based on the chromosome conformation capture (3C) technology allows to measure experimentally the frequency of physical interactions in 3D between all pairs of positions in the genome [Lieberman-Aiden et al., 2009a]. In short, if we split the full human genome into n bins (of typical length $10^4 - 10^6$ basepairs each), an Hi-C experiment produces an $n \times n$ interaction matrix A such that A_{ij} is the frequency of interactions between DNA fragments in bins i and j. It roughly proceeds as follows. First, freeze the DNA in its current 3D conformation, and collect pairs of DNA fragments that lie close to each other in this spatial conformation, thanks to a ligation process. For every such pair (k, l), each of the two fragments is then mapped to a reference genome, providing their positions, p_k and p_l. Finally, add $+1$ to the interaction matrix entry A_{ij} corresponding to the two bins i and j that respectively span p_k and p_l. This process is repeated to statistically obtain an average proximity (frequency) between two bins.

Interestingly, the frequency of 3D interactions tends to decrease with the distance between the fragments. Thus, the layout of the bins can be obtained by applying Seriation to Hi-C data (although this is not the original purpose of Hi-C data). The GRAAL assembler [Marie-Nelly et al., 2014] uses a probabilistic model of the interaction to compute the most likely genome structure from the contact maps.

1.3.6 10X Genomics

A recent development in sequencing technology commercialized by 10X genomics combines short-reads with so-called molecular barcoding, linking short-reads to long molecules to provide long-range information. In short, a barcode is associated to regions of DNA of large length (a few tens of kbp), and two reads that are close to each other on a DNA strand are likely to share several barcodes. This barcoding notably permits to call structural variants and distinguish between haplotypes for diploid genomes (such as the human genome). The Supernova assembler [Weisenfeld et al., 2017] is based on a short-reads assembly scheme, and the additional molecular barcode information is used to disambiguate the scaffolding.

1.4 Challenges

We have introduced a mathematical problem, seriation, and an applicative problem, *de novo* genome assembly. Although with idealistic data, the latter problem would fit seamlessly as an instance of the first, we have seen that in practice, it does not. For instance, applying the Spectral Algorithm 1.1 to a similarity matrix constructed from a real sequencing experiment yields a corrupted ordering, as one can see in Figure 1.7b. Yet, Theorem 1.2.1 guarantees that Algorithm 1.1 solves Seriation. Therefore, this experiment does not fit "as is" in the framework of Seriation. Let us highlight three key challenges when trying to apply seriation to *de novo* genome assembly.

- **Robustness**. As observed in Figure 1.7, the repeats induce a number of out-of-diagonal points on the similarity matrix (1.7a), which can be decomposed as the sum of a banded (and, in theory, Robinsonian) matrix, and a sparse noise matrix out of the band. This sparse noise suffices to make the spectral Algorithm 1.1 fail, although it has theoretical guarantees in the noiseless case. One of the key challenges is to design algorithmic schemes that are robust to variations from a noiseless R-matrix to a noisy observation of it. In fact, the repeats induce a specific kind of noise. Ideally we would like to be robust to it.

- **Scalability.** Typical *de novo* genome experiments with long reads involve similarity matrices of size $n \sim 10^4$. Many seriation algorithms are impractical at such a scale. Hence, a major challenge is to design algorithms for seriation that are scalable.

- **Dealing with multiple strands.** Another challenge that we have not mentioned so far is that many species are eukaryotes (have cells with a nucleus) and their genome has multiple chromosomes, *i.e.*, it is composed of several DNA strands. As a result, we do not wish to find one sequence, but several sequences (one per chromosome). However, there are overlaps between reads sampled from distinct chromosomes. Hence, the clustering-in-chromosomes step is non trivial, and cannot be done easily as a pre-processing, to break down the problem into several single-strand assembly problems.

Chapter 2

Application of the Spectral Method to Genome Assembly

This chapter presents a direct application of seriation to *de novo* assembly. From a mathematical perspective, it solely relies on the material introduced in 1.2.2. Specifically, the spectral Algorithm 1.1 is employed.

The main goal here was to get our hands dirty with real sequencing data, and make the proof of concept that seriation is an adequate framework for *de novo* assembly. To this end, we developed a pipeline that takes DNA reads (in fasta or fastq format) as input and provides an assembled sequence (or a set of sequences) in output. The overlaps between pairs of reads are computed with standard software. Algorithm 1.1 is applied to a similarity matrix built from the overlaps in a straightforward fashion. Then, given the layout found by seriation, a consensus sequence is derived via multiple sequence alignment, with dedicated software.

Despite the repeats issue presented in 1.3.2, this simple pipeline yields perhaps surprisingly good results. The challenge posed by repeats is addressed with the so-called bandwidth heuristic, which is a quick and dirty way to tackle the shortcomings of the spectral method. The underlying observation is that long overlaps are more likely to be true rather than repeat-induced overlaps. This seems to hold thanks to the length of the reads used here (third generation sequencing).

In the couple of years that followed this work, some related methods have emerged or evolved. For instance, we used the minimap [Li, 2016] tool to compute overlaps between reads, and GraphMap [Sović et al., 2016] to align the reads to a reference sequence in order to plot the layout versus a reference. Now, the minimap2 tool [Li, 2018] has been released and can perform both tasks. Also, the reads accuracy may have slightly improved, and the Oxford Nanopore preset modes of standard assemblers such as canu [Koren et al., 2017] incorporate more data-specific refinements. Still, the related work for *de novo* assembly of long reads outlined in the introduction of this chapter remains broadly accurate.

The content of this chapter is based on the following publication,
Antoine Recanati, Thomas Brüls, and Alexandre d'Aspremont. A spectral algorithm for fast de novo layout of uncorrected long nanopore reads. *Bioinformatics*, 2016.

Supplementary material for this chapter is given in Appendix Chapter A.

Chapter Abstract

Motivation: New long read sequencers promise to transform sequencing and genome assembly by producing reads tens of kilobases long. However, their high error rate significantly complicates assembly and requires expensive correction steps to layout the reads using standard assembly engines.
Results: We present an original and efficient spectral algorithm to layout the uncorrected nanopore reads, and its seamless integration into a straightforward overlap/layout/consensus (OLC) assembly scheme. The method is shown to assemble Oxford Nanopore reads from several bacterial genomes into good quality (∼99% identity to the reference) genome-sized contigs, while yielding more fragmented assemblies from the eukaryotic microbe *Sacharomyces cerevisiae*.
Availability and implementation: https://github.com/antrec/spectrassembler.

Contents

2.1 Introduction

De novo whole genome sequencing seeks to reconstruct an entire genome from randomly sampled sub-fragments whose order and orientation within the genome are unknown. The genome is oversampled so that all parts are covered multiple times with high probability.

High-throughput sequencing technologies such as Illumina substantially reduced sequencing cost at the expense of read length, which is typically a few hundred base pairs long (bp) at best. Yet, *de novo* assembly is challenged by short reads, as genomes contain repeated sequences resulting in layout degeneracies when read length is shorter or of the same order than repeat length [Pop, 2004].

Recent long read sequencing technologies such as PacBio's SMRT and Oxford Nanopore Technology (ONT) have spurred a renaissance in *de novo* assembly as they produce reads over 10kbp long [Koren and Phillippy, 2015]. However, their high error rate (∼15%) makes the task of assembly difficult, requiring complex and computationally intensive pipelines.

Most approaches for long read assembly address this problem by correcting the reads prior to performing the assembly, while a few others integrate the correction with the overlap detection phase, as in the latest version of the Canu pipeline [Koren et al., 2017] (former Celera Assembler [Myers et al., 2000]).

Hybrid techniques combine short and long read technologies: the accurate short reads are mapped onto the long reads, enabling a consensus sequence to be derived for each long read and thus providing low-error long reads (see for example Madoui et al. [2015]). This method was shown to successfully assemble prokaryotic and eukaryotic genomes with PacBio [Koren et al., 2012] and ONT [Goodwin et al., 2015] data. *Hierarchical assembly* follows the same mapping and consensus principle but resorts to long read data only, the rationale being that the consensus sequence derived from all erroneous long reads matching a given position of the genome should be accurate provided there is sufficient coverage and sequencing errors are reasonably randomly distributed: for a given base position on the genome, if 8 out of 50 reads are wrong, the majority vote still yields the correct base. Hierarchical methods map long reads against each other and derive, for each read, a consensus sequence based on all the reads that overlap it. Such an approach was implemented in HGAP [Chin et al., 2013] to assemble PacBio SMRT data, and more recently by Loman et al. [2015], to achieve complete *de novo* assembly of *Escherichia coli* with ONT data exclusively.

Recently, Li [2016] showed that it is possible to efficiently perform *de novo* assembly of noisy long reads in only two steps, without any dedicated correction procedure: all-vs-all raw read mapping (with minimap) and assembly (with miniasm). The miniasm assembler is inspired by the Celera Assembler and produces unitigs through the construction of an assembly graph. Its main limitation is that it produces a draft whose error rate is of the same order as the raw reads.

Here, we present a new method for computing the layout of raw nanopore reads, resulting

in a simple and computationally efficient protocol for assembly. It takes as input the all-vs-all overlap information (*e.g.* from minimap, MHAP [Berlin et al., 2015] or DALIGNER [Myers, 2014]) and outputs a layout of the reads (*i.e.*, their position and orientation in the genome). Like miniasm, we compute an assembly from the all-vs-all raw read mapping, but achieve improved quality through a coverage-based consensus generation process, as in nanocorrect [Loman et al., 2015], although reads are not corrected individually in our case.

The method relies on a simple spectral algorithm akin to Google's PageRank [Page et al., 1999] with deep theoretical underpinnings, described in §2.2.1. It has successfully been applied to consecutive-ones problems arising in physical mapping of genomes [Atkins and Middendorf, 1996], ancestral genome reconstructions [Jones et al., 2012], or the locus ordering problem [Cheema et al., 2010], but to our knowledge has not been applied to *de novo* assembly problems. In §2.2.2, we describe an assembler based on this layout method, to which we add a consensus generation step based on POA [Lee et al., 2002], a multi-sequence alignment engine. Finally, we evaluate this pipeline on prokaryotic and eukaryotic genomes in §3.4, and discuss possible improvements and limitations in §2.4.

2.2 Methods

2.2.1 Layout computation

We lay out the reads in two steps. We first sort them by position, *i.e.*, find a permutation π such that read $\pi(1)$ will be positioned before read $\pi(2)$ on the genome. Then, we iteratively assign an exact position (*i.e.*, leftmost basepair coordinate on the genome) to each read by using the previous read's position and the overlap information.

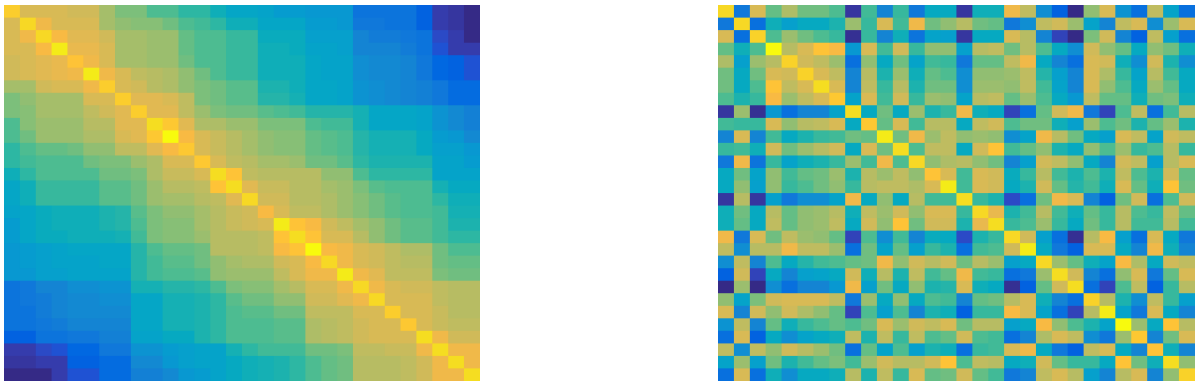
The key step is the first one, which we cast as a seriation problem, *i.e.*, we seek to reconstruct a linear order between n elements using unsorted, pairwise similarity information [Atkins et al., 1998, Fogel et al., 2013]. Here the n elements are the reads, and the similarity information comes from the overlapper (*e.g.* from minimap).

The seriation problem and the spectral relaxation have been discussed in the introductory Chapter 1. For self-containment, we briefly recall the formulation leading to the spectral relaxation. Given a pairwise similarity matrix A_{ij}, and assuming the data has a serial structure, *i.e.*that there exists an order π such that $A_{\pi(i)\pi(j)}$ decreases with $|i-j|$, seriation seeks to recover this ordering π (see Figure 1.3 in Chapter 1, repeated here in Figure 2.1 for an illustration). If such an order π exists, it minimizes the 2-SUM score,

$$\text{2-SUM}(\pi) = \sum_{i,j=1}^{n} A_{ij} \left(\pi(i) - \pi(j)\right)^2, \tag{2.1}$$

and the seriation problem can be solved as a minimization over the set of permutation vectors

[Fogel et al., 2013]. In other words, the permutation π should be such that if A_{ij} is high (meaning that i and j have a high similarity), then $\big(\pi(i) - \pi(j)\big)^2$ should be low, meaning that the positions $\pi(i)$ and $\pi(j)$ should be close to each other. Conversely, if $A_{ij} = 0$, the positions of i and j in the new order may be far away without affecting the score.



(a) R-matrix (b) pre-R

Figure 2.1: A similarity matrix reordered with the spectral algorithm. The original matrix (left) has values that decrease when moving away from the diagonal. It is randomly permuted (right), and the spectral algorithm will find back the original ordering.

When using seriation to solve genome assembly problems, the similarity A_{ij} measures the overlap between reads i and j. In an ideal setting with constant read length and no repeated regions, two overlapping reads should have nearby positions on the genome. We therefore expect the order found by seriation to roughly match the sorting of the positions of the reads.

The problem of finding a permutation over n elements is combinatorial. Still, provided the original data has a serial structure, an exact solution to seriation exists in the noiseless case [Atkins et al., 1998] using spectral clustering, and there exist several convex relaxations allowing explicit constraints on the solution [Fogel et al., 2013].

The exact solution is directly related to the well-known spectral clustering algorithm. Indeed, for any vector $\mathbf{x}$, the objective in (2.1) reads

$$\sum_{i,j=1}^{n} A_{ij} \left(x_i - x_j\right)^2 = x^T L_A x, \quad L_A = \mathbf{diag}(A\mathbf{1}) - A$$

where L_A is the Laplacian matrix of A. This means that the 2-SUM problem amounts to

$$\min_{\pi} \pi^T L_A \pi$$

where π is a permutation vector. Roughly speaking, the spectral clustering approach to seriation relaxes the constraint "π is a permutation vector" into "π is a vector of $\mathbb{R}^n$ orthogonal to the constant vector $\mathbf{1} = (1, ..., 1)^T$" with fixed norm. As we have seen in the Introduction, up to

a dilatation and a shift of the set of permutation vectors, this only amounts to relaxing the integer constraints on permutation vectors. The problem then becomes

$$\min_{\{\mathbf{1}^T\pi=0,\,\|\pi\|_2=1\}} \pi^T L_A \pi$$

This relaxed problem is an eigenvector problem. Finding the minimum over normalized vectors x yields the eigenvector associated to the smallest eigenvalue of L_A, but the smallest eigenvalue, 0, is associated with the eigenvector $\mathbf{1}$, from which we cannot recover any permutation. However, if we restrict x to be orthogonal to $\mathbf{1}$, the solution is the second smallest eigenvector, called the Fiedler vector. A permutation is recovered from this eigenvector by sorting its coefficients: given $\mathbf{x} = (x_1, x_2, ..., x_n)$, the algorithm outputs a permutation π such that $x_{\pi(1)} \leq x_{\pi(2)} \leq ... \leq x_{\pi(n)}$. This procedure is summarized in Algorithm 1.1, repeated here as Algorithm 2.1.

Algorithm 2.1 Spectral ordering [Atkins et al., 1998]

Input: Connected similarity matrix $A \in \mathbb{R}^{n\times n}$

1: Compute Laplacian $L_A = \mathbf{diag}(A\mathbf{1}) - A$

2: Compute second smallest eigenvector of L_A, $\mathbf{x}^*$

3: Sort the values of $\mathbf{x}^*$

Output: Permutation $\pi : {\mathbf{x}^*}_{\pi(1)} \leq {\mathbf{x}^*}_{\pi(2)} \leq ... \leq {\mathbf{x}^*}_{\pi(n)}$

In fact, [Atkins et al., 1998] showed that under the assumption that A has a serial structure, Algorithm 2.1 solves the seriation problem exactly, *i.e.*, recovers the order π such that $A_{\pi(i)\pi(j)}$ decreases with $|i-j|$. This means that we solve the read ordering problem by simply solving an extremal eigenvalue problem, which has low complexity (comparable to Principal Component Analysis (PCA)) and is efficient in practice (see Supplementary Figure A.1 and Table A.1).

Once the reads are reordered, we can sequentially compute their exact positions (basepair coordinate of their left end on the genome) and orientation. We assign position 0 and strand "+" to the first read, and use the overlap information (position of the overlap on each read and mutual orientation) to compute the second read's position and orientation, etc. More specifically, when computing the position and orientation of read i, we use the information from reads $i-1, ..., i-c$ to average the result, where c roughly equals the coverage, as this makes the layout more robust to misplaced reads. Note that overlappers relying on hashing, such as minimap and MHAP, do not generate alignments but still locate the overlaps on the reads, making this positioning step possible. Thanks to this "polishing" phase, we would still recover the layout if two neighboring reads were permuted due to consecutive entries of the sorted Fiedler vector being equal up to the eigenvector computation precision, for example.

2.2.2 Consensus generation

We built a simple assembler using this layout idea and tested its accuracy. It is partly inspired by the nanocorrect pipeline of Loman et al. [2015] in which reads are corrected using multiple alignments of all overlapping reads. These multiple alignments are performed with a Partial Order Aligner (POA) [Lee et al., 2002] multiple-sequence alignment engine. It computes a consensus sequence from the alignment of multiple sequences using a dynamic programming approach that is efficient when the sequences are similar (which is the case if we trim the sequences to align their overlapping parts). Specifically, we used SPOA, a Single Instruction Multiple Data implementation of POA developed in Vaser et al. [2016].

The key point is that we do not need to perform multiple alignment using all reads, since we already have a layout. Instead, we can generate a consensus sequence for, say, the first 3000 bp of the genome by aligning the parts of the reads that are included in this window with SPOA, and repeat this step for the reads included in the window comprising the next 3000 bp of the genome, etc. In practice, we take consecutive windows that overlap and then merge them to avoid errors at the edges, as shown in Figure 2.2. The top of the figure displays the layout of the reads broken down into three consecutive overlapping windows, with one consensus sequence generated per window with SPOA. The final assembly is obtained by iteratively merging the window $k{+}1$ to the consensus formed by the windows $1, \ldots, k$.

The computational complexity for aligning N sequences of length L with POA, with an average divergence between sequences ϵ, is roughly $O(mNL^2)$, with $m \simeq (1+2\epsilon)$. With 10% of errors, m is close to 1. If each window of size L_w contains about C sequences, the complexity of building the consensus in a window is $O(mCL_w^2)$. We compute L_g/L_w consensus windows, with L_g the length of the genome (or contig), so the overall complexity of the consensus generation is $O(mCL_gL_w)$. We therefore chose in practice a window size relatively small, but large enough to prevent mis-assemblies due to noise in the layout, $L_w = 3\text{kbp}$.

2.2.3 Overlap-based similarity and repeats handling

In practice, we build the similarity matrix A as follows. Given an overlap found between the i-th and j-th reads, we set A_{ij} equal to the overlap score (or number of matches, given in tenth column of minimap or fourth column of MHAP output file). Such matrices are sparse: a read overlaps with only a few others (the number of neighbors of a read in the overlap graph roughly equals the coverage). There is no sparsity requirement for the algorithm to work, however sparsity lowers RAM usage since we store the $n \times n$ similarity matrix with about $n \times C$ non-zero values, with C the coverage. In such cases, the ordered similarity matrix is band diagonal.

Unfortunately, the correctly ordered (sorted by position of the reads on the reference sequence) similarity matrix contains outliers outside the main diagonal band (see Figure 2.3a)

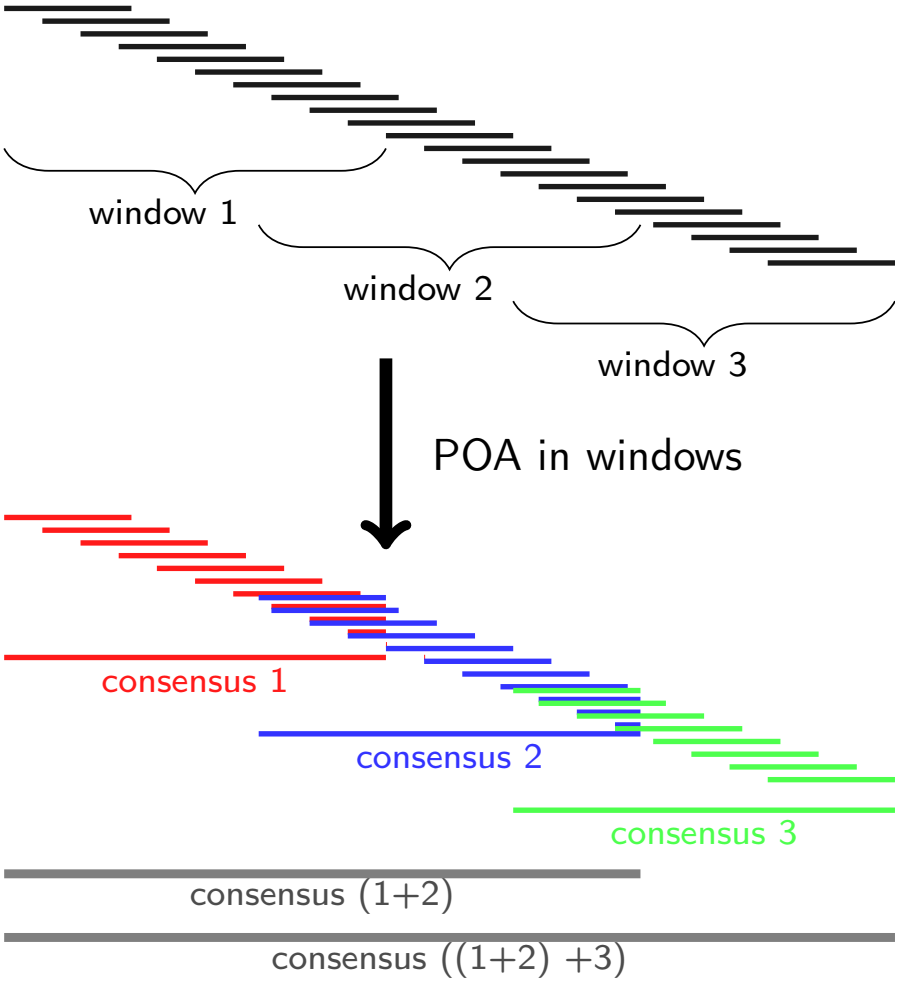
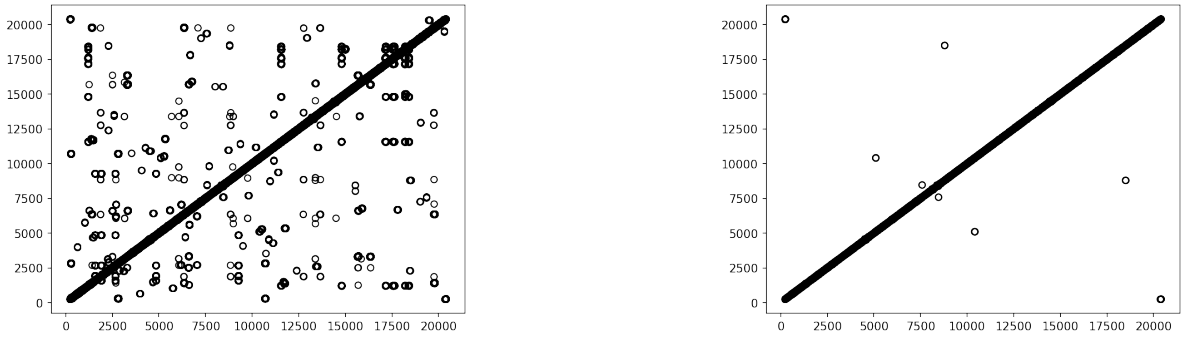


Figure 2.2: Consensus generation. Given the layout, the genome is sliced into overlapping windows, and a consensus is computed in each window. The final consensus is then obtained by merging the consensus windows.

that corrupt the ordering. These outliers are typically caused by either repeated subsequences or sequencing noise (error in the reads and chimeric reads), although errors in the similarity can also be due to hashing approximations made in the overlap algorithm. We use a threshold on the similarity values and on the length of the overlaps to remove them. The error-induced overlaps are typically short and yield a low similarity score (*e.g.*, number of shared min-mers), while repeat-induced overlaps can be as long as the length of the repeated region. By weighting the similarity, the value associated to repeat-induced overlaps can be lowered. Weighting can be done with, *e.g.*, the `-weighted` option in MHAP to add a tf-idf style scaling to the MinHash sketch, making repetitive k-mers less likely to cause a match between two sequences, or with default parameters with minimap.

In the Supplementary Material presented in Chapter A, we describe experiments with real,



(a) raw similarity

(b) thresholded at 90%

Figure 2.3: Similarity matrix for *E. coli* ONT sequences before (left) and after (right) thresholding. The positions of the reads were obtained by mapping to the reference genome with GraphMap [Sović et al., 2016].

corrected and simulated reads to assess the characteristics of such overlaps and validate our method. Figure A.2 shows that although the overlap scores and lengths are lower for outliers than for inliers on average, the distributions of these quantities intersect. As shown in Figure A.3, the experiments indicate that all false-overlaps can be removed with a stringent threshold on the overlap length and score. However, removing all these short or low score overlaps will also remove many true overlaps. For bacterial genomes, the similarity graph can either remain connected or be broken into several connected components after a threshold-based outlier removal, depending on the initial coverage. Figure A.3 illustrates the empirical observation that the coverage needs to be above 60x to keep the graph connected while removing all outliers. Most outliers can be similarly removed for real and synthetic data from *S. cerevisiae*, although a few outliers, probably harboring telomeric repeats, remain at the ends of chromosomes after thresholding.

There is thus a tradeoff to be reached depending on how many true overlaps one can afford to lose. With sufficient coverage, a stringent threshold on overlap score and length will remove both repeat-induced and error-induced overlaps, while still yielding a connected assembly graph. Otherwise, aggressive filtering will break the similarity graph into several connected components. In such a case, since the spectral algorithm only works with a connected similarity graph, we compute the layout and consensus separately in each connected component, resulting in several contigs. To set the threshold sufficiently high to remove outliers but small enough to keep the number of contigs minimal, we used a heuristic based on the following empirical observation, illustrated in Supplementary Figure A.4. The presence of outliers in the correctly (based on the positions of the reads) ordered band diagonal matrix imparts an increased bandwidth (maximum distance to the diagonal of non zero entries) on the matrix reordered with the spectral algorithm. We can therefore run the spectral algorithm, check the bandwidth in the reordered matrix, and increase the threshold if the bandwidth appears too

large (typically larger than twice the coverage).

In practice, we chose to set the threshold on the overlap length to 3.5kbp, and removed the overlaps with the lowest score [in the first 40%-quantile (respectively 90% and 95%) for C≤60X (resp. 60X≤C≤100X and C≥100X)]. As indicated in Algorithm 2.2, we let these threshold values increase if indicated by the bandwitdh heuristic.

Finally, we added a filtering step to remove reads that have non-zero similarity with several sets of reads located in distant parts of the genome, such as chimeric reads. These reads usually overlap with a first subset of reads at a given position in the genome, and with another distinct subset of reads at another location, with no overlap between these distinct subsets. We call such reads "connecting reads", and they can be detected from the similarity matrix by computing, for each read (index i), the set of its neighbors in the graph $\mathcal{N}_i = \{j : A_{ij} > 0\}$. The subgraph represented by A restricted to $\mathcal{N}_i$ is either connected (there exists a path between any pair of edges), or split into separate connected components. In the latter case, we keep the overlaps between read i and its neighbors that belong to only one of these connected components (the largest one).

Algorithm 2.2 OLC assembly pipeline

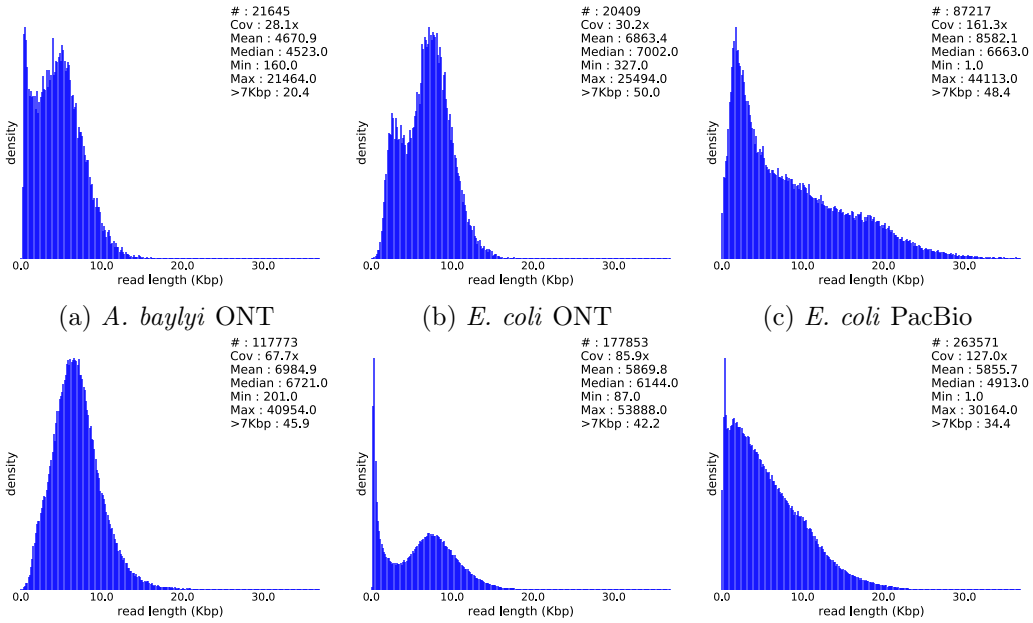
Input: n long noisy reads
1: Compute overlaps with an overlapper (*e.g.* minimap or MHAP)
2: Construct similarity matrix $S \in \mathbb{R}^{n \times n}$ from the overlaps
3: Remove outliers from S with a threshold on values S_{ij}, on overlap length, and removal of connecting reads (as explained in §2.2.3)
4: **for all** Connected component A of S **do**
5: Reorder A with spectral algorithm (Algorithm 2.1)
6: **if** bandwidth of $A_{reordered} \geq 2\times$ Coverage **then**
7: set higher threshold on A and try again
8: **end if**
9: Compute layout from the ordering found and overlaps
10: Partition the length of the contig into small windows
11: Compute consensus in each window with SPOA
12: Merge consecutive windows with SPOA
13: **end for**
Output: Contig consensus sequences

2.3 Results

2.3.1 Data

We tested this pipeline on ONT and PacBio data. The bacterium *Acinetobacter baylyi ADP1* and the yeast *Saccharomyces cerevisiae S288C* were sequenced at Genoscope with Oxford Nanopore's MinION device using the R7.3 chemistry, together with an additional dataset

of *S. cerevisiae S288C* using the R9 chemistry. Only the 2D high quality reads were used. The *S. cerevisiae S288C* ONT sequences were deposited at the European Nucleotide Archive (http://www.ebi.ac.uk/ena) where they can be accessed under Run accessions ERR1539069 to ERR1539080. We also used the following publicly available data: ONT *Escherichia coli* by Loman et al. [2015] (http://bit.ly/loman006 - PCR1 2D pass dataset), and PacBio *E. coli* K-12 PacBio P6C4, and *S. cerevisiae W303 P4C2*. Their key characteristics are given with the assembly results in Table 2.1, and read length histograms are given in Figure 2.4. For each dataset, we also used the reads corrected and trimmed by the Canu pipeline as an additional dataset with low error-rate. The results on these corrected datasets are given in Supplementary Figures A.6 and A.7 and Tables A.2 and A.4.



(a) *A. baylyi* ONT (b) *E. coli* ONT (c) *E. coli* PacBio

(d) *S. cerevisiae* ONT R7.3 (e) *S. cerevisiae* ONT R9 (f) *S. cerevisiae* PacBio

Figure 2.4: Read length histograms of the raw datasets.

2.3.2 Layout

Bacterial genomes

minimap was used to compute overlaps between raw reads (we obtained similar results with MHAP and DALIGNER). The similarity matrix preprocessed as detailed in Section 2.2.3 yielded a few connected components for bacterial genomes. The reads were successfully ordered in each of these, as one can see in Figure 2.5 for *E. coli*, and in Figure A.6 for the other datasets.

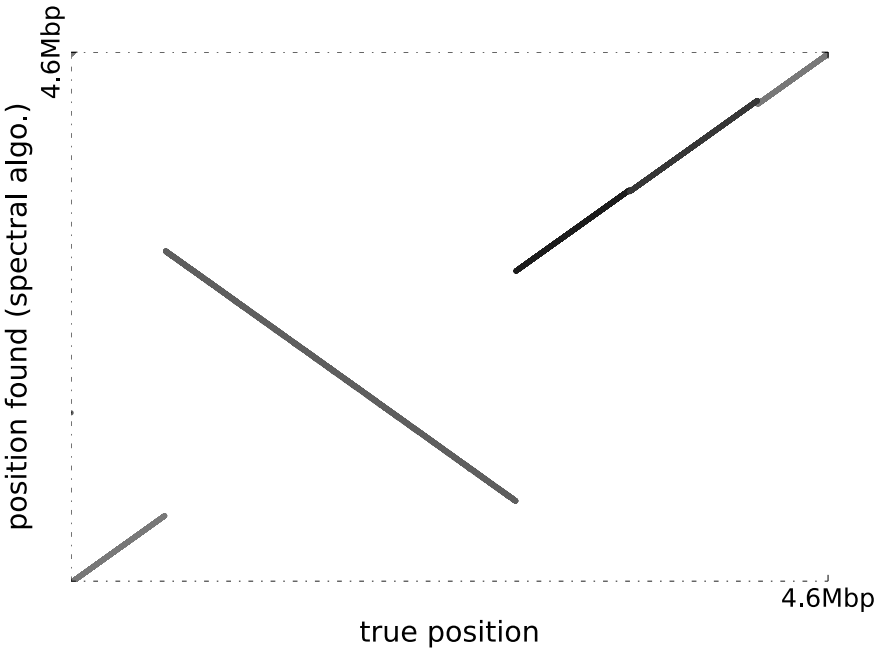


Figure 2.5: Ordering of the reads computed with the spectral algorithm vs true ordering (obtained by mapping the reads to the reference genome with GraphMap) for the *E. coli* ONT dataset. All contigs are artificially displayed on the same plot for compactness. There are two equivalent correct orderings for each contig : (1,2,...,n) and (n, n-1, ..., 1), both yielding the same 2-SUM score (2.1) and leading to the same consensus sequence (possibly reverse complemented).

Eukaryotic genome

For the *S. cerevisiae* genome, the threshold on similarity had to be set higher than for bacterial genomes because of a substantially higher number of repetitive regions and false overlaps, leading to a more fragmented assembly. Most of them are correctly reordered with the spectral algorithm, see Figure 2.6 and Supplementary Figure A.7.

2.3.3 Consensus

Recovering contiguity

Once the layout was established, the method described above was used to assemble the contigs and generate a consensus sequence. For the two bacterial genomes, the first round of layout produced a small number of connected components, each of them yielding a contig. Sufficient overlap was left between the contig sequences to find their layout with a second iteration of the algorithm and produce a single contig spanning the entire genome. The number of contigs in the yeast assemblies can be reduced similarly. The fact that the first-pass contigs overlap even though they result from breaking the similarity graph into several connected components might

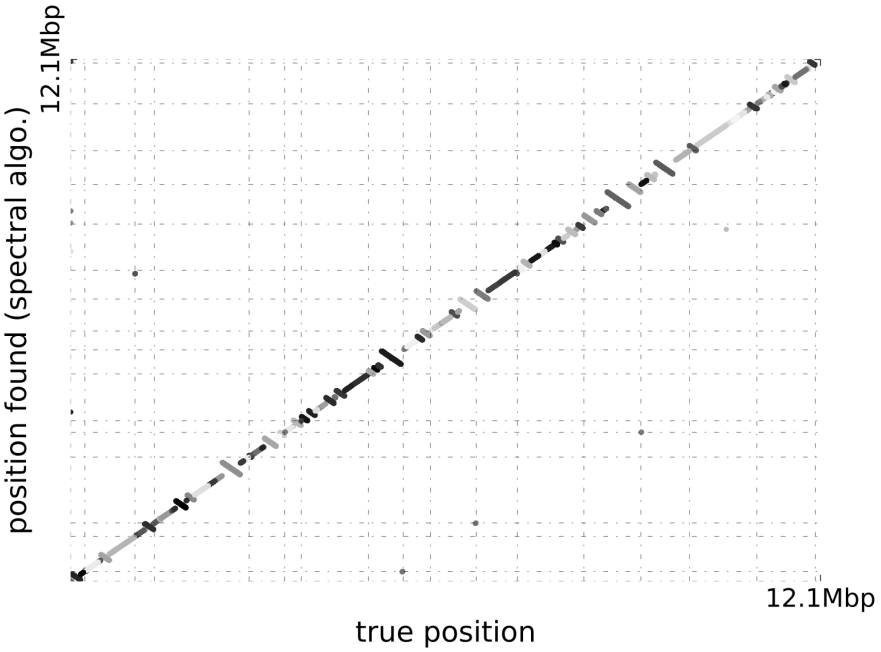


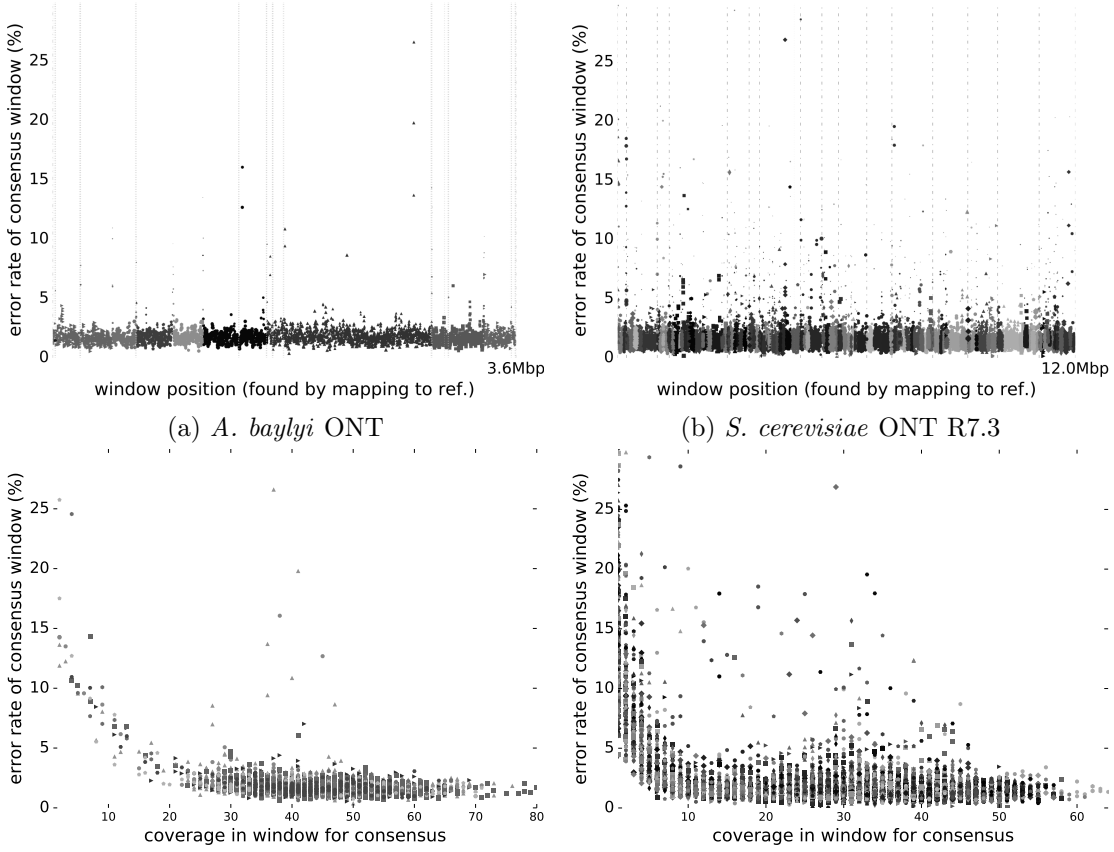
Figure 2.6: Ordering of the *Saccharomyces cerevisiae* ONT R7.3 reads identified with the spectral algorithm vs true ordering (obtained by mapping the reads to the reference genome with GraphMap and concatenating the ordering found in each chromosome). The different chromosomes are separated by grid lines.

seem counter-intuitive at first sight. However, note that when cutting an edge A_{ij} results in the creation of two contigs (one containing i and the other j), the sequence fragment at the origin of the overlap between the two reads is still there on both contigs to yield an overlap between them in the second iteration. Alternatively, we found the following method useful to link the contigs' ends: 1. extract the ends of the contig sequences, 2. compute their overlap with minimap, 3. propagate the overlaps to the contig sequences, 4. use miniasm with all pre-selection parameters and thresholds off, to just concatenate the contigs (see Supplementary Material §A.5).

Consensus quality evaluation

We first investigated the quality of the consensus sequences derived in each window. Figures 2.8 and A.5 highlight the correcting effect of the consensus. Figure 2.7 provides hints about what causes inaccurate consensus windows, and suggests that the error-rate in the consensus windows depends mainly on the local coverage. The top plots examine the error-rate in the consensus windows according to their position (and whether they are located on a repeat). Most of the windows with a high error rate are positioned at the ends of the contigs to which they belong. We also observed that repeats are often positioned at the edge between two contigs, though this does not seem to be the determinant factor. The bottom plots represent the error-rate

in the windows against their estimated coverage, defined as the total length of sequences used to perform the multiple alignment in the window normalized by the length of the consensus sequence. Overall, one can see that the windows with high error rate are the ones with low coverage. Nevertheless, especially for the yeast genomes, there are also several windows with high values for both error-rate and coverage. Manual inspection of these reveals that they usually do not span repeated regions, but their high error-rates arise from imperfections in the layout.



(a) *A. baylyi* ONT (b) *S. cerevisiae* ONT R7.3

(c) *A. baylyi* ONT (d) *S. cerevisiae* ONT R7.3

Figure 2.7: Error-rates in consensus windows *versus* position of the windows on the reference genome (a,b). The dashed lines represent the location of repeats for *A. baylyi*, and the separation between chromosomes for *S. cerevisiae*. The size of each scatter marker is proportional to the coverage of the window. The (c,d) panel represents the error-rates in consensus windows *versus* the coverage of the windows. The error-rate was computed with the `errorrates.py` script from `samtools`, using the mapping obtained from GraphMap.

We then compared our results to those obtained with other long reads assemblers : Miniasm, Canu and Racon [Vaser et al., 2016]. Racon takes a draft assembly, the raw reads, and a mapping of the reads to the draft assembly as input. We used it with the draft assembly produced by Miniasm (as done by Vaser et al. [2016]). We label this method "Miniasm+Racon"

in our results. We also used Racon with the draft assembly derived by our method ("Spectral+Racon" method), using Minimap to map the raw reads to the draft assemblies before using Racon. Racon's use here can be seen as a polishing phase for the sequences outputted by the spectral method and Miniasm. To keep both assemblers on an equal footing, we compared Spectral+Racon to two iterations of Miniasm+Racon (since one pass of Miniasm does not implement any consensus). A summary of assembly reports generated with DNAdiff [Kurtz et al., 2004] and QUAST [Gurevich et al., 2013] are given in Table 2.1 and Supplementary Table A.3. Briefly, the assemblies displayed between 98% and 99% average identity to their reference genome, with errors mostly consisting in deletions. Misassemblies were rare in reconstructed bacterial genomes but more frequent in assembled yeast genomes, where they mostly consisted in translocations and relocations caused by either deletions and/or misplaced reads in the layout. Canu clearly outperforms the spectral method on PacBio data, while both assemblers yield comparable results on the ONT datasets.

Table 2.1: Assembly results of the spectral method, compared to Miniasm, Canu and Racon, across the different datasets. For the spectral method, we give the results after contig merging (see §2.3.3); the number of contigs before this post-processing is given between parentheses. The best results in terms of average identity are highlighted in bold (but other metrics should also be used to compare the assemblies).

		Miniasm	Spectral	Canu	Miniasm+Racon	Miniasm+Racon (2 iter.)	Spectral+Racon
A. baylyi ONT R7.3 28x	Ref. size [bp]	3598621	3598621	3598621	3598621	3598621	3598621
	Total size [bp]	3531295	3551582	3513432	3564823	3566438	3551094
	Ref. chr. [#]	1	1	1	1	1	1
	Contigs [#]	5	1 (7)	1	5	5	1 (7)
	Aln. ref [bp]	3445457 (95.74%)	3596249 (99.93%)	3595082 (99.90%)	3596858 (99.95%)	3596854 (99.95%)	3598181 (99.99%)
	Aln. query [bp]	3379002 (95.69%)	3549290 (99.94%)	3513081 (99.99%)	3564455 (99.99%)	3566021 (99.99%)	3550742 (99.99%)
	Misassemblies [#]	0	0	2	2	2	0
	Avg. identity	87.31	98.17	97.59	98.18	98.36	**98.42**
E. coli ONT R7.3 30x	Ref. size [bp]	4641652	4641652	4641652	4641652	4641652	4641652
	Total size [bp]	4759346	4662043	4625543	4647066	4643235	4629112
	Ref. chr. [#]	1	1	1	1	1	1
	Contigs [#]	3	1 (4)	2	3	3	1 (4)
	Aln. ref [bp]	4355121 (93.83%)	4612515 (99.37%)	4638255 (99.93%)	4640127 (99.97%)	4640127 (99.97%)	4641457 (100.00%)
	Aln. query [bp]	4432658 (93.14%)	4623823 (99.18%)	4625535 (100.00%)	4642837 (99.91%)	4639816 (99.93%)	4628962 (100.00%)
	Misassemblies [#]	0	2	8	3	3	2
	Avg. identity	89.28	98.80	99.40	99.31	99.45	**99.46**
S. cerevisiae ONT R7.3 68x	Ref. size [bp]	12157105	12157105	12157105	12157105	12157105	12157105
	Total size [bp]	11813544	12213218	12142953	11926664	11926191	12167363
	Ref. chr. [#]	17	17	17	17	17	17
	Contigs [#]	29	71 (127)	36	29	29	71 (127)
	Aln. ref [bp]	11566318 (95.14%)	12043050 (99.06%)	12086977 (99.42%)	12084923 (99.41%)	12086556 (99.42%)	12061384 (99.21%)
	Aln. query [bp]	11236806 (95.12%)	12134480 (99.36%)	12089056 (99.56%)	11923058 (99.97%)	11918621 (99.94%)	12135284 (99.74%)
	Misassemblies [#]	0	7	34	18	19	11
	Avg. identity	89.00	98.00	98.33	98.49	**98.63**	98.61
S. cerevisiae ONT R9 86x	Ref. size [bp]	12157105	12157105	12157105	12157105	12157105	12157105
	Total size [bp]	11734150	11795644	12217497	12128279	12129086	11750114
	Ref. chr. [#]	17	17	17	17	17	17
	Contigs [#]	30	48 (85)	26	30	29	48 (85)
	Aln. ref [bp]	11947453 (98.28%)	11607131 (95.48%)	12126980 (99.75%)	12126663 (99.75%)	12127467 (99.76%)	11695983 (96.21%)
	Aln. query [bp]	11549494 (98.43%)	11668882 (98.93%)	12179843 (99.69%)	12118506 (99.92%)	12121202 (99.93%)	11717047 (99.72%)
	Misassemblies [#]	0	23	39	18	19	36
	Avg. identity	93.55	98.81	99.02	99.16	**99.20**	99.10
E. coli PacBio 161x	Ref. size [bp]	4641652	4641652	4641652	4641652	4641652	4641652
	Total size [bp]	4845211	4731239	4670125	4653228	4645420	4674460
	Ref. chr. [#]	1	1	1	1	1	1
	Contigs [#]	1	2 (6)	1	1	1	2 (6)
	Aln. ref [bp]	4437473 (95.60%)	4617713 (99.48%)	4641652 (100.00%)	4641551 (100.00%)	4641500 (100.00%)	4641652 (100.00%)
	Aln. query [bp]	4601587 (94.97%)	4705704 (99.46%)	4670125 (100.00%)	4653140 (100.00%)	4645420 (100.00%)	4673065 (99.97%)
	Misassemblies [#]	0	5	4	4	4	4
	Avg. identity	89.13	98.63	**99.99**	99.64	99.91	99.87
S. cerevisiae PacBio 127x	Ref. size [bp]	12157105	12157105	12157105	12157105	12157105	12157105
	Total size [bp]	12266420	12839034	12346258	12070971	12052148	12695031
	Ref. chr. [#]	17	17	17	17	17	17
	Contigs [#]	30	90 (136)	29	30	30	90 (136)
	Aln. ref [bp]	11250453 (92.54%)	11917823 (98.03%)	12091868 (99.46%)	12023040 (98.90%)	12024968 (98.91%)	12002816 (98.73%)
	Aln. query [bp]	11396172 (92.91%)	12456415 (97.02%)	12304982 (99.67%)	12045088 (99.79%)	12027812 (99.80%)	12485128 (98.35%)
	Misassemblies [#]	0	57	76	61	59	68
	Avg. identity	88.29	98.41	**99.87**	99.43	99.72	99.54

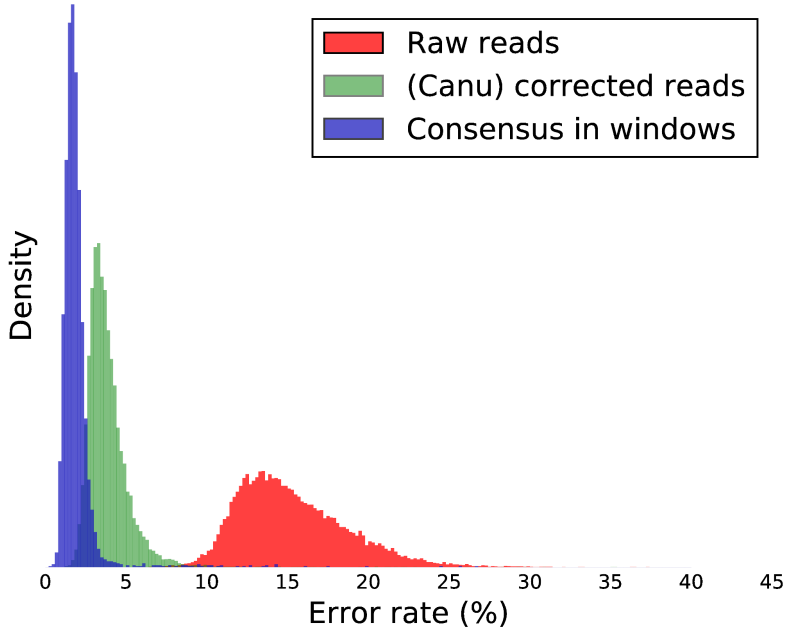


Figure 2.8: Error rate of consensus window sequences, compared to the raw and corrected (with the Canu correction and trimming modules) reads for the *A. baylyi* ONT dataset. The error rates were computed by mapping the sequences to the *A. baylyi* reference genome. Histograms for the other datasets are available in Supplementary Figure A.5.

Optical mapping

After the first iteration of the bacterial genome assembly pipeline, overlaps between the first-pass contigs were sufficient to find their layout. It should be anticipated however that not all overlaps might be apparent in some cases, e.g. if too many reads were removed during the preprocessing step. One attractive option is to use optical mapping [Aston et al., 1999] to layout the contigs. We had such an optical map available for the *A. baylyi* genome, and implemented the algorithm of Nagarajan et al. [2008] to map the contigs to the restriction map, which led to the same layout as the one identified from our two-round assemblies (data not shown), thus providing a "consistency check" for the layout. We suggest in Table 2.2 and Figure 2.9 that optical maps could be particularly valuable for the ordering of contigs from more structurally complex eukaryotic genomes such as *S. cerevisiae*.

Table 2.2 displays assembly results for the following experiment. We divided a set of reads from *S. cerevisiae* into subsets, according to the chromosome membership of each read (obtained by mapping the reads to a reference genome). We then ran the method on each chromosome-specific dataset separately. The assembled contigs were evaluated with QUAST and DNAdiff for each chromosome (only a subset of the QUAST descriptive statistics is shown here). This experiments sheds light on how our method would behave if there were no repeats between chromosomes, or if we knew to which chromosomes some reads belong to thanks to, *e.g.*, optical mapping. Figure 2.9 provides results from another experiment designed to evaluate the extent

Table 2.2: Assembly of each chromosome of *S. cerevisiae* (for each chromosome, we used the subset of reads from the *S. cerevisiae* ONT R7.3 dataset that were mapped to it).

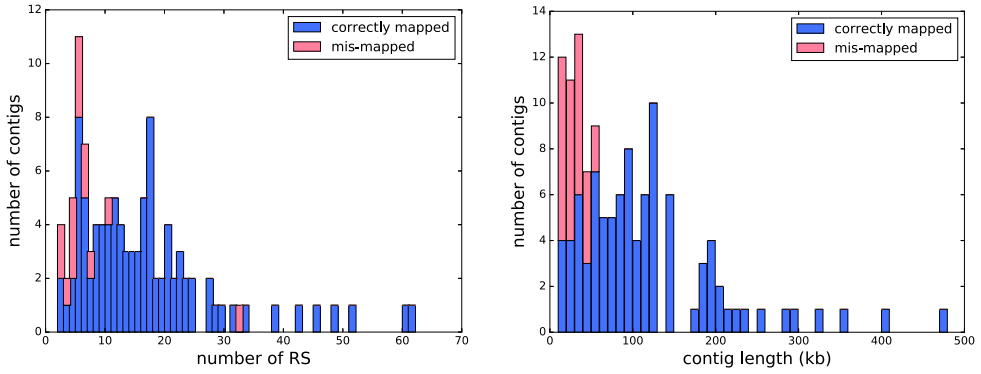
Chr.	Ref size [bp]	Contigs [#]	Aln. bp ref [bp]	Aln. bp query [bp]	Misassem--blies [#]	Avg. identity [%]
I	230218	1	228273(99.16%)	225845(98.43%)	0	98.21
II	813184	1	806340(99.16%)	797624(98.91%)	0	98.17
III	316620	4	313707(99.08%)	326011(93.47%)	3	98.33
IV	1531933	6	1519577(99.19%)	1539642(99.04%)	0	98.24
V	576874	1	574944(99.67%)	575037(99.30%)	3	98.37
VI	270161	3	270161(100.00%)	285160(98.97%)	0	98.36
VII	1090940	8	1088278(99.76%)	1115166(98.37%)	0	98.09
VIII	562643	2	556839(98.97%)	561348(99.48%)	2	98.22
IX	439888	2	437971(99.56%)	443785(97.81%)	0	98.38
X	745751	2	740696(99.32%)	738859(99.16%)	0	98.35
XI	666816	2	665942(99.87%)	667003(99.46%)	0	98.35
XII	1078177	5	1067559(99.02%)	1084233(98.50%)	2	98.27
XIII	924431	4	922948(99.84%)	937417(99.58%)	1	98.12
XIV	784333	2	779066(99.33%)	783072(99.35%)	0	98.41
XV	1091291	3	1089941(99.88%)	1088832(99.49%)	0	98.34
XVI	948066	11	942078(99.37%)	1015108(97.50%)	1	97.83
Chrmt.	85779	5	65196(76.00%)	69107(80.98%)	-	90.32

to which optical mapping could improve long-range anchoring of the 127 *S. cerevisiae* ONT R7.3 contigs and provide an alternative consistency check of the assembly. A restriction map was generated *in silico* from the reference *S. cerevisiae* genome with the BamHI restriction site (GGATCC), yielding one map per chromosome. Note that this simulated optical map represents a best-case scenario since real optical measurements lack some precision and are obtained through an error-prone assembly process. We used the same algorithm to layout the contigs with optical mapping as we had with the *A. baylyi* genome [Nagarajan et al., 2008]. Some contigs were correctly mapped by this process, while some others were not. Figure 2.9 shows histograms of the correctly and mis-mapped contigs according to the number of occurrences of the restriction site in the contigs, and to the length of the contigs. We observe that all contigs longer than 60kbp are correctly mapped.

2.4 Discussion

We have shown that seriation based layout algorithms can be successfully applied to *de novo* genome assembly problems, at least for genomes harboring a limited number of repeats.

In a similar vein to the recent report about the miniasm assembly engine [Li, 2016], our work confirms that the layout of long reads can be found without prior error correction, using only overlap information generated from raw reads by tools such as minimap, MHAP or DALIGNER. However, unlike miniasm, which does not derive a consensus but instead concatenates the reads into a full sequence, we take advantage of read coverage to produce contigs with a consensus quality on par with that achieved by assembly pipelines executing dedicated



(a) *A. baylyi* ONT (b) *S. cerevisiae* ONT R7.3

Figure 2.9: Histograms of the number of contigs as a function of the number of distinct restriction sites (RS) appearing in their sequence (a) or contig length (b). For a given number of RS occurrences (a) or contig length (b), the blue part of the bar shows the fraction of contigs correctly aligned to the theoretical restriction map, whereas the red part corresponds to the complementary fraction of unperfectly aligned contigs.

error-correction steps. The results of Table 2.1 appear promising. For example, our assembler combined with Racon yields among the highest average identities with the reference for the ONT datasets. In terms of speed however, our pipeline is clearly outperformed by Miniasm, but also by Miniasm+Racon, the latter improving overall accuracy. Still, compared to approaches implementing error correction steps, we gain significant speed-ups by highly localizing the error correction and consensus generation processes, which is made possible by knowledge of the layout. We believe that tools such as Miniasm and Racon are implemented in a much more efficient way than our own, but the layout method itself is efficient (see Supplementary Table A.1) and is known to be scalable as it relies on the same algorithmic core as Google's PageRank.

The main limitation of our layout algorithm is its sensitivity to outliers in the similarity matrix, hence the need to remove them in a pre-processing phase. Higher coverage and quality of the input reads, both expected in the near future, would likely improve the robustness of our pipeline. Still, for eukaryotic genomes, we found that some outliers require additional information to be resolved (see Supplementary Figure A.3), which could be provided in the future by extracting topological information from the assembly graph.

In the meantime, our pipeline behaves like a draft generating assembler for prokaryotic genomes, and a first-pass unitigger for eukaryotic genomes. Importantly, the overall approach is modular and can integrate other algorithms to increase layout robustness or consensus quality, as illustrated here by the integration of Racon as an optional polishing module.

Our original contribution here consists in the layout computation. The spectral OLC assembler we built on top of it could be enhanced in many ways. We have shown that the spectral algorithm is suited to find the layout for bacterial genomes, even though there is room left for

performance improvements on repeat-rich eukaryotic genomes.

For these eukaryotic genomes, it could make sense to use the spectral algorithm jointly with other assembly engines (*e.g.* Miniasm or Canu), to check the consistency of connected components before they are assembled. Our consensus generation method is coarse-grained for now and does not take into account statistical properties of ONT sequencing errors. Nevertheless, the three components (O, L and C) of the method being independent, an external and more refined consensus generation process could readily be plugged after the overlap and layout computations to further improve results and increase accuracy.

Chapter 3

Multi-dimensional Spectral Ordering : Reconstructing Linear Orderings via Spectral Embedding

In the previous chapters, we have observed that due to repeats, the spectral method (Algorithm 1.1) fails to reorder full similarity matrices correctly into a single contig (see Figure 1.7b). Yet, Algorithm 2.2, at the core of the method presented in Chapter 2, uses a simple iterative thresholding procedure leveraging the fact that the largest overlaps are scarcely due to repeats to adapt Algorithm 1.1. It yields correct but fragmented assemblies.

In this chapter, we explore an extension of the spectral method, that was at first motivated by the following experimental observation. While Figure 1.7b plots the first (non-trivial) eigenvector of the Laplacian -the Fiedler vector, we can also take a look at the following eigenvectors. For instance, we can make a 3d scatter plot of the three eigenvectors associated to the three smallest non-zero eigenvalues. Interestingly, the points in this 3d scatter plot are roughly distributed along a curve with linear pieces bent in some points. Recall Theorem 3.2.1. It states that if there is an ordering of the points such that the pairwise similarity decreases within their distance along this ordering, then the spectral method finds it. These assumptions mean that we can embed the points on a line such that the similarity is monotonic with the distance within the line. The Fiedler vector then provides such a linear embedding (it is a 1d embedding of the points, with one real value per coordinate $i \in [n]$). Imagine we start with data satisfying the assumptions, but we add similarity between the first and the last elements in the chain so that the assumptions no longer hold. In the original linear embedding, the first and the last elements have high similarity but are placed far apart on the line. However, if we add one dimension to the embedding (an additional degree of freedom), and place ourselves on a plane, we can bend the line so that the first and the last elements are close to each other. Specifically, we can obtain a circular embedding such that two elements that are nearby on the

circle have high similarity. Thus, while the repeats make it impossible to find a linear ordering consistent with all pairwise similarity information, we can hope that the chain structure appears in a higher dimensional embedding, where the repeats may cause angles and loops in the chain.

A significant part of this chapter is devoted to existing work, since several results scattered in different fields (from theoretical to application-specific) provide intuition or partial results motivating our approach. Bringing them all together into a consistent frame is one of the contributions of this work.

A remark about the notations used in this Chapter. We have previously used the notation $\lambda_1(L) \geq \ldots \geq \lambda_n(L)$ to denote the eigenvalues of the (laplacian) matrix L of size n. Here, we will instead use $\lambda_0(L) \geq \ldots \geq \lambda_{n-1}(L)$, and the same indexing for the associated eigenvectors, as we will be interested only in the non-zero eigenvalues and the associated eigenvectors, hence we start the indexing to 1 from the second (which is the first non-zero) eigenvalue.

The content of this chapter is based on the following publication,
Antoine Recanati, Thomas Kerdreux, and Alexandre d'Aspremont. Reconstructing latent orderings by spectral clustering. *arXiv preprint arXiv:1807.07122*, 2018a.

Supplementary for this chapter is given in Appendix Chapter B.

Chapter Abstract

Spectral clustering uses a graph Laplacian spectral embedding to enhance the cluster structure of some data sets. When the embedding is one dimensional, it can be used to sort the items (spectral ordering). Empirically we found that a multidimensional Laplacian embedding enhances the latent ordering of the data, if any. This also extends to circular orderings, a case where unidimensional embeddings fail. We tackle the task of retrieving linear and circular orderings in a unifying framework, and show how a latent ordering on the data translates into a filamentary structure on the Laplacian embedding. We propose a method to recover it, illustrated with numerical experiments on synthetic data, real DNA third-generation sequencing data, and spatial conformation Hi-C data. The code and experiments are available at https://github.com/antrec/mdso.

Contents

3.1 Introduction

At the risk of being redundant, let us recall the seriation problem introduced in Chapter 1, before we present its generalization to *circular orderings*.

The seriation problem seeks to recover a latent ordering from similarity information. We typically observe a matrix measuring pairwise similarity between a set of n elements and assume they have a serial structure, *i.e.* they can be ordered along a chain where the similarity between elements decreases with their distance within this chain. In practice, we observe a random permutation of this similarity matrix, where the elements are not indexed according to that latent ordering. Seriation then seeks to find that global latent ordering using only (local) pairwise similarity.

Yet, in some applications, the latent ordering is circular. For instance, in *de novo* assembly of bacterial genomes, such as the *E. coli* and *A. baylyi* genomes encountered in Chapter 2, one has to reorder DNA fragments sub-sampled from a circular genome. The graphic illustration of *de novo* assembly shown in Chapter 1, Figure 1.4 was adequate for a linear strand of DNA, but Figure 3.1 is more appropriate for a circular genomes.

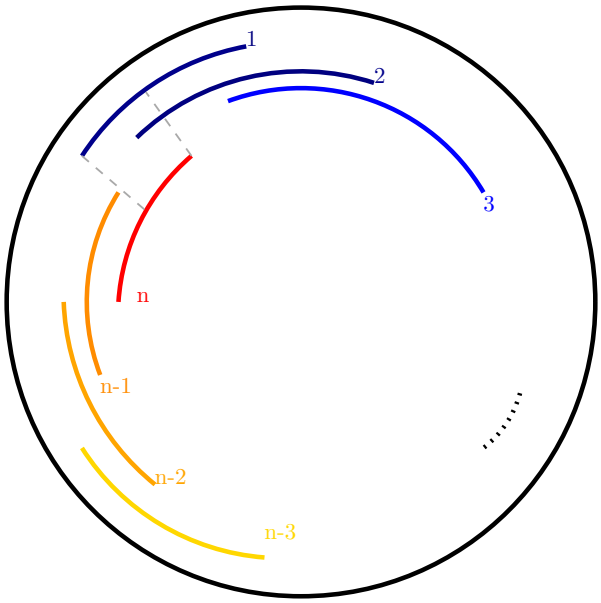


Figure 3.1: Illustration of the assembly process for a circular genome. The physical strand of DNA is circular. The DNA sequence is therefore represented as a circle (black). Reads are randomly sampled from this sequence, and we wish to infer their position (layout) from their pairwise overlaps. Given an ordering of the reads matching their position on the (linearized) genome, the first and the last reads are likely to overlap. Here, the first read (dark blue) overlaps with the last read (red).

Let us consider two other examples where one seeks to recover circular orderings. In biology, a cell evolves according to a cycle: a newborn cell passes through diverse states (growth, DNA-

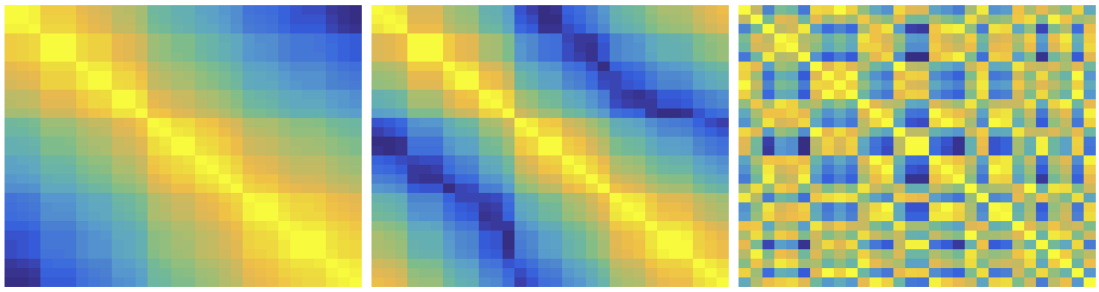
replication, *etc.*) before dividing itself into two newborn cells, hence closing the loop. Problems of interest then involve collecting cycle-dependent data on a population of cells at various, unknown stages of the cell-cycle, and trying to order the cells according to their cell-cycle stage. Such data include gene-expression [Liu et al., 2017], or DNA 3D conformation data [Liu et al., 2018]. In planar tomographic reconstruction, the shape of an object is inferred from projections taken at unknown angles between 0 and 2π. Reordering the angles then enables to perform the tomography [Coifman et al., 2008].

The main structural hypothesis on similarity matrices related to seriation is the concept of R-matrix, which we have introduced in Chapter 1 and repeat here, together with its circular counterpart.

Definition 3.1.1. *We say that $A \in \mathbf{S}_n$ is a R-matrix (or Robinson matrix) if it is symmetric and satisfies $A_{i,j} \leq A_{ik}$, for all triplets of indices (i,j,k) such that $|i-j| \geq |i-k|$.*

Definition 3.1.2. *We say that $A \in \mathbf{S}_n$ is a circular R-matrix if it is symmetric and satisfies $A_{i,j} \leq A_{ik}$, for all triplets of indices (i,j,k) such that $\mathcal{D}(|i-j|) \geq \mathcal{D}(|i-k|)$, where $\mathcal{D}(|i-j|) = \min(|i-j|, n-|i-j|)$.*

As a reminder, $\mathbf{S}_n$ is the set of real symmetric matrices of dimension n. The proximity matrix of points embedded on a line follows Definition 3.1.1, whereas that of points embedded on a circle (as in Figure B.1) follows Def 3.1.2. Figure 3.2 displays examples of such matrices.



(a) R-matrix (b) circular R-matrix (c) permuted R-matrix

Figure 3.2: From left to right, R-matrix (3.2a), circular R-matrix (3.2b), and a randomly permuted observation of a R-matrix (3.2c). Seriation seeks to recover (3.2a) from its permuted observation (3.2c).

In what follows, we write $\mathcal{L}_R^n$ (resp., $\mathcal{C}_R^n$) the set of R (resp., circular-R) matrices of size n, and $\mathcal{P}_n$ the set of permutations of n elements. A permutation can be represented by a vector π (lower case) or a matrix $\Pi \in \{0,1\}^{n \times n}$ (upper case) defined by $\Pi_{ij} = 1$ iff $\pi(i) = j$, and $\pi = \Pi\mathbf{g}$ where $\mathbf{g} = (1, \ldots, n)^T$. We refer to both representations by $\mathcal{P}_n$ and may omit the subscript n whenever the dimension is clear from the context. We say that $A \in \mathbf{S}_n$ is pre-$\mathcal{L}_R$ (resp., pre-$\mathcal{C}_R$) if there exists a permutation $\Pi \in \mathcal{P}$ such that the matrix $\Pi A \Pi^T$ (whose entry (i,j) is

$A_{\pi(i),\pi(j)}$) is in $\mathcal{L}_R$ (resp., $\mathcal{C}_R$). Given such A, Seriation seeks to recover this permutation Π,

$$\text{find} \quad \Pi \in \mathcal{P} \qquad \text{such that} \quad \Pi A \Pi^T \in \mathcal{L}_R \qquad \text{(Linear Seriation)}$$

$$\text{find} \quad \Pi \in \mathcal{P} \qquad \text{such that} \quad \Pi A \Pi^T \in \mathcal{C}_R \qquad \text{(Circular Seriation)}$$

A widely used method for Linear Seriation is the spectral relaxation presented in Chapter 1, 1.2.2, based on the graph Laplacian of the similarity matrix. It transposes Spectral Clustering [Von Luxburg, 2007] to the case where we wish to infer a latent ordering rather than a latent clustering on the data. Roughly speaking, both methods embed the elements on a line and associate a coordinate $f_i \in \mathbb{R}$ to each element $i \in [n]$. Spectral clustering addresses a graph-cut problem by grouping these coordinates into two clusters. Spectral ordering [Atkins et al., 1998] addresses Linear Seriation by sorting the f_i. The clustering method is closely related to the ordering, as noted in Ding and He [2004].

A graph-cut partitions the data in two clusters. When seeking to cluster data in K groups with $K > 2$, one can recursively iterate graph cuts in the sub-groups obtained at the previous iterations. However, most Spectral Clustering algorithms actually use a Laplacian embedding of dimension $d > 1$, denoted d-**LE** in the following, in order to find K clusters. Latent cluster structure is assumed to be enhanced in the d-**LE**, and the k-means algorithm [MacQueen et al., 1967, Hastie et al., 2009] seamlessly identifies the clusters from the embedding. In contrast, Spectral Ordering is restricted to $d = 1$ by the sorting step (there is no total order relation on $\mathbb{R}^d$ for $d > 1$). Still, the latent linear structure may emerge from the d-**LE**, if the points are distributed along a curve. Also, for $d = 2$, it may capture the circular structure of the data and allow for solving Circular Seriation. One must then recover a (circular) ordering of points lying in a $1D$ manifold (a curve, or filament) embedded in $\mathbb{R}^d$.

In Section 3.2, we review the Spectral Ordering algorithm and the Laplacian Embedding used in Spectral Clustering. We mention graph-walk perspectives on this embedding and how this relates to dimensionality reduction techniques. Finally, we recall how these perspectives relate the discrete Laplacian to continuous Laplacian operators, providing insights about the curve structure of the Laplacian embedding through the spectrum of the limit operators. These asymptotic results were used to infer circular orderings in a tomography application in, *e.g.*, Coifman et al. [2008]. In Section 3.3, we evidence the filamentary structure of the Laplacian Embedding, and provide theoretical guarantees about the Laplacian Embedding based method for Circular Seriation. We then propose a method in Section 3.4 to leverage the multidimensional Laplacian embedding in the context of Linear Seriation and Circular Seriation. In Section 3.5, we show that a perturbation analysis similar to that existing for Linear Seriation can be applied to Circular Seriation. We eventually present numerical experiments in Section 3.6 to illustrate how the spectral method gains in robustness by using a multidimensional Laplacian embedding.

3.2 Related Work

Let us recall the highlights of the spectral relaxation, which is the starting point of this work, before we review definitions and results involving higher-dimensional Laplacian embeddings.

3.2.1 Spectral Ordering for Linear Seriation

Linear Seriation can be addressed with a spectral relaxation of the 2-SUM combinatorial problem,

$$\text{minimize} \quad \sum_{i,j=1}^{n} A_{ij}|\pi_i - \pi_j|^2 \quad \text{such that} \quad \pi \in \mathcal{P}_n \tag{2-SUM}$$

Intuitively, the optimal permutation compensates high A_{ij} values with small $|\pi_i - \pi_j|^2$, thus laying similar elements nearby. As we have seen in Section 1.2.2, for any $f = \big(f(1), \ldots, f(n)\big)^T \in \mathbb{R}^n$, the objective of 2-SUM can be written as a quadratic,

$$\sum_{i,j=1}^{n} A_{ij}|f(i) - f(j)|^2 = f^T L_A f \tag{3.1}$$

where $L_A \triangleq \mathbf{diag}(A\mathbf{1}) - A$ is the graph-Laplacian of A. From (3.1), L_A is positive-semi-definite for A having non-negative entries, and $\mathbf{1} = (1, \ldots, 1)^T$ is an eigenvector associated to $\lambda_0 = 0$.

The spectral method relaxes the 2-SUM problem by dropping the integer constraint on permutation vectors $\pi \in \mathcal{P}_n$ and enforcing only norm and orthogonality constraints, $\|\pi\| = 1$, $\pi^T\mathbf{1} = 0$, to avoid the trivial solutions $\pi = 0$ and $\pi \propto \mathbf{1}$. It results in,

$$\text{minimize} \quad f^T L_A f \quad \text{such that} \quad \|f\|_2 = 1\,,\ f^T\mathbf{1} = 0. \tag{Relax. 2-SUM}$$

This is an eigenvalue problem on L_A solved by $f_{(1)}$, the eigenvector associated to $\lambda_1 \geq 0$ the second smallest eigenvalue of L_A. If the graph defined by A is connected (which we assume further) then $\lambda_1 > 0$. From $f_{(1)}$, one can recover a permutation by sorting its entries. The resulting algorithm, presented in Chapter 1 and applied to *de novo* assembly in Chapter 2, is recalled here in Algorithm 3.1. We also recall a key theoretical result related to it. For pre-$\mathcal{L}_R$ matrices, Linear Seriation is equivalent to 2-SUM [Fogel et al., 2013], and can be solved with Algorithm 3.1 [Atkins et al., 1998], as stated in Theorem 3.2.1.

Algorithm 3.1 Spectral ordering [Atkins et al., 1998]

Input: Connected similarity matrix $A \in \mathbb{R}^{n\times n}$

1: Compute Laplacian $L_A = \mathbf{diag}(A\mathbf{1}) - A$

2: Compute second smallest eigenvector of L_A, f_1

3: Sort the values of f_1

Output: Permutation $\sigma : f_1(\sigma(1)) \leq \ldots \leq f_1(\sigma(n))$

Theorem 3.2.1 (Atkins et al. [1998]). *If $A \in \mathbf{S}_n$ is a pre-$\mathcal{L}_R$ matrix, then Algorithm 3.1 recovers a permutation $\Pi \in \mathcal{P}_n$ such that $\Pi A \Pi^T \in \mathcal{L}_R^n$, i.e., it solves Linear Seriation.*

3.2.2 Laplacian Embedding

Let $0 = \lambda_0 < \lambda_1 \leq \ldots \leq \lambda_{n-1}$ be the eigenvalues of L_A, and $\Phi = (\mathbf{1}, f_1, \ldots, f_{n-1})$ the matrix whose column j is the eigenvector corresponding to the j-th smallest eigenvalue, which is the $j-1$ smallest, *non-zero* eigenvalue λ_{j-1}. We have the following decomposition, $L_A = \Phi \Lambda \Phi^T$, with $\Lambda \triangleq \mathbf{diag}\,(\lambda_0, \ldots, \lambda_{n-1})$.

Algorithm 3.1 embeds the data in 1D through the eigenvector f_1 (1-**LE**). It then uses this 1-d embedding to infer an ordering of the points. More generally, for any $d < n$, $\Phi^{(d)} \triangleq (f_1, \ldots, f_d)$ defines a d-dimensional embedding (d-**LE**)

$$\boldsymbol{y}_i = \big(f_1(i), f_2(i), \ldots, f_d(i)\big)^T \in \mathbb{R}^d, \quad \text{for} \quad i = 1, \ldots, n. \tag{d-\textbf{LE}}$$

The d-**LE** solves the following embedding problem, which is a generalization of 2-SUM to multi-dimensions,

$$\begin{array}{ll} \text{minimize} & \sum_{i,j=1}^n A_{ij} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2 \\ \text{such that} & \tilde{\Phi} = \left(\boldsymbol{y}_1^T, \ldots, \boldsymbol{y}_n^T\right)^T \in \mathbb{R}^{n \times d}\,,\ \tilde{\Phi}^T \tilde{\Phi} = \mathbf{I}_d\,,\ \tilde{\Phi}^T \mathbf{1}_n = \mathbf{0}_d \end{array} \tag{d-2SUM}$$

Indeed, thanks to the ortho-normality constraints, the objective of d-2SUM can also be written as a quadratic,

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}\left(\tilde{\Phi}^T L_A \tilde{\Phi}\right) \\ \text{such that} & \tilde{\Phi} \in \mathbb{R}^{n \times d}\,,\ \tilde{\Phi}^T \tilde{\Phi} = \mathbf{I}_d\,,\ \tilde{\Phi}^T \mathbf{1}_n = \mathbf{0}_d \end{array} \tag{d-2SUM'}$$

The reader can find a detailed derivation in Belkin and Niyogi [2003]. The 2-SUM intuition still holds: the d-**LE** lays similar elements nearby, and dissimilar apart, in $\mathbb{R}^d$. Other dimensionality reduction techniques such as Multidimensional scaling (MDS) [Kruskal and Wish, 1978], kernel PCA [Schölkopf et al., 1997], or Locally Linear Embedding (LLE) [Roweis and Saul, 2000] could be used as alternatives to embed the data in a way that intuitively preserves the latent ordering. However, guided by the generalization of Algorithm 3.1 and theoretical results that follow, we restrict ourselves to the Laplacian embedding.

Normalization and Scaling

Several variants of the Laplacian (and resulting spectral embeddings d-**LE**) exist in the literature, leading to diverse interpretations and experimental behaviors.

Given the weighted adjacency matrix $W \in \mathbf{S}_n$ of a graph, its Laplacian reads $L = D - W$, where $D = \mathbf{diag}(W\mathbf{1})$ has diagonal entries $d_i = \sum_{j=1}^n W_{ij}$ (degree of i). Normalizing W_{ij} by

$\sqrt{d_i d_j}$ or d_i leads to the normalized Laplacians,

$$L^{\text{sym}} = D^{-1/2} L D^{-1/2} = \mathbf{I} - D^{-1/2} W D^{-1/2} \qquad \text{(symmetric)}$$
$$L^{\text{rw}} = D^{-1} L = \mathbf{I} - D^{-1} W \qquad \text{(random-walk)}$$

They correspond to graph-cut normalizations (normalized cut or ratio cut). Moreover, L^{rw} has a Markov chain interpretation, where a random walker on edge i jumps to edge j from time t to $t+1$ with transition probability $P_{ij} \triangleq W_{ij}/d_i$. It has connections with diffusion processes, governed by the heat equation $\frac{\partial \mathcal{H}_t}{\partial t} = -\Delta \mathcal{H}_t$, where Δ is the Laplacian operator, $\mathcal{H}_t$ the heat kernel, and t is time [Qiu and Hancock, 2007]. These connections lead to diverse Laplacian embeddings backed by theoretical justifications, where the eigenvectors f_k^{rw} of L^{rw} are sometimes scaled by decaying weights α_k (thus emphasizing the first eigenvectors),

$$\tilde{\boldsymbol{y}}_i = \left(\alpha_1 f_1^{\text{rw}}(i), \ldots, \alpha_{d-1} f_d^{\text{rw}}(i)\right)^T \in \mathbb{R}^d, \quad \text{for} \quad i = 1, \ldots, n. \qquad (\text{(}\alpha\text{, d)-}\mathbf{LE})$$

Laplacian eigenmaps [Belkin and Niyogi, 2003] is a nonlinear dimensionality reduction technique based on the spectral embedding of L^{rw} (((α, d)-**LE**) with $\alpha_k = 1$ for all k). Specifically, given points $x_1, \ldots, x_n \in \mathbb{R}^d$, the method computes a heat kernel similarity matrix $W_{ij} = \exp - \left(\|x_i - x_j\|^2/t\right)$ and outputs the first eigenvectors of L^{rw} as a lower dimensional embedding. The choice of the heat kernel is motivated by connections with the heat diffusion process on a manifold, a partial differential equation involving the Laplacian operator. This method has been successful in many machine learning applications such as semi-supervised classification [Belkin and Niyogi, 2004] and search-engine type ranking [Zhou et al., 2004]. Notably, it provides a global, nonlinear embedding of the points that preserves the local structure.

The commute time distance CTD(i, j) between two nodes i and j on the graph is the expected time for a random walker to travel from node i to node j and then return. The full (α, d)-**LE**, with $\alpha_k = (\lambda_k^{\text{rw}})^{-1/2}$ and $d = n - 1$, satisfies CTD$(i, j) \propto \|\tilde{\boldsymbol{y}}_i - \tilde{\boldsymbol{y}}_j\|$. Given the decay of α_k, the d-**LE** with $d \ll n$ approximately preserves the CTD. This embedding has been successfully applied to vision tasks, *e.g.*, anomaly detection [Albano and Messinger, 2012], image segmentation and motion tracking [Qiu and Hancock, 2007].

Another, closely related dimensionality reduction technique is that of diffusion maps [Coifman and Lafon, 2006], where the embedding is derived to preserve diffusion distances, resulting in the (α, d)-**LE**, for $t \geq 0$, $\alpha_k(t) = (1 - \lambda_k^{\text{rw}})^t$.

Coifman and Lafon [2006], Coifman et al. [2008] also propose a normalization of the similarity matrix $\tilde{W} \leftarrow D^{-1} W D^{-1}$, to extend the convergence of L^{rw} towards the Laplace-Beltrami operator on a curve when the similarity is obtained through a heat kernel on points that are *non uniformly* sampled along that curve.

Finally, we will use in practice the heuristic scaling $\alpha_k = 1/\sqrt{k}$ to damp high dimensions,

as explained in Appendix B.2.5.

For a deeper discussion about spectral graph theory and the relations between these methods, see for instance Qiu and Hancock [2007] and Chung and Yau [2000].

3.2.3 Link with Continuous Operators

In the context of dimensionality reduction, when the data points $x_1, \dots, x_n \in \mathbb{R}^D$ lie on a manifold $\mathcal{M} \subset \mathbb{R}^d$ of dimension $K \ll D$, the graph Laplacian L of the heat kernel ($W_{ij} = \exp\left(-\|x_i - x_j\|^2/t\right)$) used in Belkin and Niyogi [2003] is a discrete approximation of $\Delta_{\mathcal{M}}$, the Laplace-Beltrami operator on $\mathcal{M}$ (a differential operator akin to the Laplace operator, adapted to the local geometry of $\mathcal{M}$). Singer [2006] specify the hypothesis on the data and the rate of convergence of L towards $\Delta_{\mathcal{M}}$ when n grows and the heat-kernel bandwidth t shrinks. Von Luxburg et al. [2005] also explore the spectral asymptotics of the spectrum of L to prove consistency of spectral clustering.

This connection with continuous operators gives hints about the Laplacian embedding in some settings of interest for Linear Seriation and Circular Seriation. Indeed, consider n points distributed along a curve $\Gamma \subset \mathbb{R}^D$ of length 1, parameterized by a smooth function $\gamma : \mathbb{R} \to \mathbb{R}^D$, $\Gamma = \{\gamma(s) \ : \ s \in [0, 1]\}$, say $x_i = \gamma(i/n)$. If their similarity measures their proximity along the curve, then the similarity matrix is a circular-R matrix if the curve is closed ($\gamma(0) = \gamma(1)$), and a R matrix otherwise. Coifman et al. [2008] motivate a method for Circular Seriation with the spectrum of the Laplace-Beltrami operator Δ_Γ on Γ when Γ is a closed curve. Indeed, Δ_Γ is simply the second order derivative with respect to the arc-length s, $\Delta_\Gamma f(s) = f''(s)$ (for f twice continuously differentiable), and its eigenfunctions are given by,

$$f''(s) = -\lambda f(s). \tag{3.2}$$

With periodic boundary conditions, $f(0) = f(1)$, $f'(0) = f'(1)$, and smoothness assumptions, the first eigenfunction is constant with eigenvalue $\lambda_0 = 0$. The remaining eigenvalues λ_m are double, associated to the eigenfunctions $f^{m,\cos}$ and $f^{m,\sin}$ given by, for $m = 1, \dots, \infty$,

$$\begin{aligned} \lambda_m &= (2\pi m)^2 \\ f_m^{\cos} &= \cos(2\pi m s) \\ f_m^{\sin} &= \sin(2\pi m s) \end{aligned}$$

Hence, the 2-**LE** given by the following equation should approximately lay the points on a circle, allowing for solving Circular Seriation [Coifman et al., 2008],

$$\left(f_1(i), f_2(i)\right) \approx \left(\cos(2\pi s_i), \sin(2\pi s_i)\right).$$

More generally, the 2d-**LE** given by the following equation is a closed curve in $\mathbb{R}^{2d}$,

$$\big(f_1(i),\dots,f_{2d+1}(i)\big)^T \approx \big(\cos(2\pi s_i),\sin(2\pi s_i),\dots,\cos(2d\pi s_i),\sin(2d\pi s_i)\big)\,.$$

If Γ is not closed, we can also find its eigenfunctions. For instance, with Neumann boundary conditions (vanishing normal derivative), $f(0)=1$, $f(1)=0$, $f'(0)=f'(1)=0$, the non-trivial eigenfunctions of Δ_Γ, f_m, associated to the eigenvalues λ_m, for $m=1,\dots,\infty$, are given by,

$$\lambda_m = (\pi m)^2$$
$$f_m = \cos(\pi m s)$$

The 1-**LE**, $f_1(i)\approx\cos(\pi s_i)$, respects the monotonicity of i, which is consistent with Theorem 3.2.1. Lafon [2004] invoked this asymptotic argument to solve an instance of Linear Seriation but seemed unaware of the existence of Atkin's Algorithm 3.1. Note that here too, the d-**LE**,

$$\big(f_1(i),\dots,f_d(i)\big)^T \approx \big(\cos(\pi s_i),\dots,\cos(d\pi s_i)\big)$$

follows a closed curve in $\mathbb{R}^d$, with endpoints.

These asymptotic results hint that the Laplacian embedding preserves the latent ordering of data points lying on a curve embedded in $\mathbb{R}^D$. However, these results are only asymptotic and there is no known guarantee for the Circular Seriation problem as there is for Linear Seriation. Also, the curve (sometimes called filamentary structure) stemming from the Laplacian embedding has been observed in more general cases where no hypothesis on a latent representation of the data is made, and the input similarity matrix is taken as is (see, *e.g.*, Diaconis et al. [2008] for a discussion about the horseshoe phenomenon).

3.2.4 Other embeddings

We focus on the Laplacian embedding since it naturally extends results from Atkins et al. [1998]. However, other methods can produce low-dimensional embeddings from a similarity (or distance) matrix, such as Multi-Dimensional Scaling (MDS) [Kruskal and Wish, 1978]. *Classical*-MDS uses the eigen-decomposition of the centered distance matrix, (it is also a spectral method). metric-MDS finds the embedding through the minimization of a stress function. t-SNE [Maaten and Hinton, 2008] minimizes the divergence between similarity-based probabilities to find a 2D or 3D embedding of the data. We experimentally compare the orderings found by our method when using these alternative embedding techniques.

Dimensionality reduction techniques, *e.g.*, kernel PCA [Schölkopf et al., 1997] and Locally Linear Embedding (LLE) [Roweis and Saul, 2000], take design matrices (high-dimensional embedding) as input, to produce the low-dimensional embedding, instead of distance/similarity

matrices, hence we do not consider those in the following.

3.2.5 Ordering points lying on a curve

Existing approaches for Seriation rely on either 1D or 2D embeddings, sorting coordinates (1D), or angles between two coordinates (2D) to reorder the points. Friendly [2002] sorts the angle between the coordinates of the 2D-MDS embedding to perform Linear Seriation. Coifman et al. [2008] use the 2-**LE** to perform Circular Seriation in a tomographic reconstruction setting, sorting the inverse tangent of the angle between the two components to reorder the points (Algorithm 3.2). Liu et al. [2018] use a similar approach to solve Circular Seriation in a cell-cycle related problem, but with the 2D embedding given by MDS. We are not aware of any method using higher-dimensional embeddings under Linear Seriation or Circular Seriation assumptions.

3.3 Spectral properties of some (circular) Robinson matrices

We have claimed that the d-**LE** enhances the latent ordering of the data and we now present some theoretical evidences. We adopt a point of view similar to Atkins et al. [1998], where the feasibility of Linear Seriation relies on structural assumptions on the similarity matrix ($\mathcal{L}_R$). For a subclass $\mathcal{C}_R^*$ of $\mathcal{C}_R$ (set of circular-R matrices), we show that the d-**LE** lays the points on a closed curve, and that for $d = 2$, the elements are embedded on a circle according to their latent circular ordering. This is a counterpart of Theorem 3.2.1 for Circular Seriation. It extends the asymptotic results motivating the approach of Coifman et al. [2008], shifting the structural assumptions on the elements (data points lying on a curve embedded in $\mathbb{R}^D$) to assumptions on the raw similarity matrix that can be verified in practice. Then, we develop a perturbation analysis to bound the deformation of the embedding when the input matrix is in $\mathcal{C}_R^*$ up to a perturbation. Finally, we discuss the spectral properties of some (non circular) $\mathcal{L}_R$-matrices that shed light on the filamentary structure of their d-**LE** for $d > 1$.

For simplicity, we assume $n \triangleq 2p + 1$ odd in the following. The results with $n = 2p$ even are relegated to the Appendix Chapter B, together with technical proofs.

3.3.1 Circular Seriation with Symmetric, Circulant matrices

Let us consider the set $\mathcal{C}_R^*$ of matrices in $\mathcal{C}_R$ that are circulant, in order to have a closed form expression of their spectrum. A matrix $A \in \mathbb{R}^{n\times n}$ is Toeplitz if its entries are constant on a given diagonal, $A_{ij} = b_{(i-j)}$ for a vector of values b of size $2n - 1$. A symmetric Toeplitz matrix A satisfies $A_{ij} = b_{|i-j|}$, with b of size n. In the case of circulant symmetric matrices, we also

have that $b_k = b_{n-k}$, for $1 \leq k \leq n$, thus symmetric circulant matrices are of the form,

$$A = \begin{pmatrix} b_0 & b_1 & b_2 & \cdots & b_2 & b_1 \\ b_1 & b_0 & b_1 & \cdots & b_3 & b_2 \\ b_2 & b_1 & b_0 & \cdots & b_4 & b_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ b_2 & b_3 & b_4 & \cdots & b_0 & b_1 \\ b_1 & b_2 & b_3 & \cdots & b_1 & b_0 \end{pmatrix}. \tag{3.3}$$

Where b is a vector of values of size $p+1$ (recall that $n = 2p+1$). The circular-R assumption (Def 3.1.2) imposes that the sequence $(b_0, \ldots, b_{p+1})$ is non-increasing. We thus define the set $\mathcal{C}_R^*$ of circulant matrices of $\mathcal{C}_R$ as follows.

Definition 3.3.1. *A matrix $A \in \mathbf{S}^n$ is in $\mathcal{C}_R^*$ iff it verifies $A_{ij} = b_{|i-j|}$ and $b_k = b_{n-k}$ for $1 \leq k \leq n$ with $(b_k)_{k=0,\ldots,\lfloor n/2 \rfloor}$ a non-increasing sequence.*

The spectrum of symmetric circulant matrices is known [Reichel and Trefethen, 1992, Gray et al., 2006, Massey et al., 2007], and for a matrix A of size $n = 2p+1$, it is given by,

$$\begin{aligned} \nu_m &= b_0 + 2\textstyle\sum_{k=1}^{p} b_k \cos\left(2\pi km/n\right) \\ y^{m,\cos} &= \tfrac{1}{\sqrt{n}}\left(1, \cos\left(2\pi m/n\right), \ldots, \cos\left(2\pi m(n-1)/n\right)\right) \\ y^{m,\sin} &= \tfrac{1}{\sqrt{n}}\left(1, \sin\left(2\pi m/n\right), \ldots, \sin\left(2\pi m(n-1)/n\right)\right). \end{aligned} \tag{3.4}$$

For $m = 1, \ldots, p$, ν_m is an eigenvalue of multiplicity 2 with associated eigenvectors $y^{m,\cos}$,$y^{m,\sin}$. For any m, $(y^{m,\cos}, y^{m,\sin})$ embeds the points on a circle, but for $m > 1$, the circle is walked through m times, hence the ordering of the points on the circle does not follow their latent ordering. The ν_m from equations (3.4) are in general not sorted. It is the Robinson property (monotonicity of (b_k)) that guarantees that $\nu_1 \geq \nu_m$, for $m \geq 1$, and thus that the 2-**LE** embeds the points on a circle *that follows the latent ordering* and allows one to recover it by scanning through the unit circle. This is formalized in Theorem 3.3.2, which is the main result of our paper, proved in Appendix B.3. It provides guarantees in the same form as in Theorem 3.2.1 with the simple Algorithm 3.2 that sorts the angles, used in Coifman et al. [2008].

Algorithm 3.2 Circular Spectral Ordering [Coifman et al., 2008]

Input: Connected similarity matrix $A \in \mathbb{R}^{n \times n}$

1: Compute normalized Laplacian $L_A^{\text{rw}} = \mathbf{I} - \left(\mathbf{diag}(A\mathbf{1})\right)^{-1} A$

2: Compute the two first non-trivial eigenvectors of L_A^{rw}, (f_1, f_2)

3: Sort the values of $\theta(i) \triangleq \tan^{-1}\left(f_2(i)/f_1(i)\right) + \mathbb{1}[f_1(i) < 0]\pi$

Output: Permutation $\sigma : \theta(\sigma(1)) \leq \ldots \leq \theta(\sigma(n))$

Theorem 3.3.2. *Given a permuted observation* $\Pi A \Pi^T$ *(*$\Pi \in \mathcal{P}$*) of a matrix* $A \in \mathcal{C}^*_R$*, the 2-***LE** *maps the items on a circle, equally spaced by angle* $2\pi/n$*, following the circular ordering in* Π*. Hence, Algorithm 3.2 recovers a permutation* $\Pi \in \mathcal{P}_n$ *such that* $\Pi A \Pi^T \in \mathcal{C}^*_R$*, i.e., it solves Circular Seriation.*

3.3.2 (Linear) Robinson Toeplitz matrices

In order to show Theorem 3.3.2, we have examined the spectrum of some circular-R matrices. Although only the 2-**LE** appears in Theorem 3.3.2, it is interesting to see that for any $d > 1$, the d-**LE** of matrices in $\mathcal{C}^*_R$ is a curve. Let us investigate how the latent linear ordering of Toeplitz matrices in $\mathcal{L}_R$ translates to the d-**LE**. Remark that from Theorem 3.2.1, the 1-**LE** suffices to solve Linear Seriation. Yet, for perturbed observations of $A \in \mathcal{L}_R$, the d-**LE** may be more robust to the perturbation than the 1-**LE**, as the experiments in Section 3.6 indicate. However, there is no closed form expression for the spectrum of (linear) R matrices in general, or even of Toeplitz R matrices, which are the analog of $\mathcal{C}^*_R$ for 2-SUM. Therefore, in the remainder of this Section, we will review spectral properties of specific standard Robinson matrices appearing in some applications, whose spectrum has been studied.

Tridiagonal Toeplitz matrices are defined by $b_0 > b_1 > 0 = b_2 = \ldots = b_p$. For $m = 0, \ldots, n-1$, they have eigenvalues ν_m with multiplicity 1 associated to eigenvector $y^{(m)}$ [Trench, 1985],

$$\begin{array}{rcl} \nu_m & = & b_0 + 2b_1 \cos\left(m\pi/(n+1)\right) \\ y^{(m)} & = & \left(\sin\left(m\pi/(n+1)\right), \ldots, \sin\left(mn\pi/(n+1)\right)\right), \end{array} \tag{3.5}$$

thus matching the spectrum of the Laplace operator on a curve with endpoints from §3.2.3 (up to a shift). This type of matrices can indeed be viewed as a limit case with points uniformly sampled on a line with strong similarity decay, leaving only the two nearest neighbors with non-zero similarity.

Kac-Murdock-Szegö (KMS) matrices are defined, for $\alpha > 0$, $\rho = e^{-\alpha}$, by $A_{ij} = b_{|i-j|} = e^{-\alpha|i-j|} = \rho^{|i-j|}$. For $m = 1, \ldots, \lfloor n/2 \rfloor$, there exists $\theta_m \in \big((m-1)\pi/n, m\pi/n\big)$, such that ν_m is a double eigenvalue associated to eigenvectors $y^{m,\cos}$,$y^{m,\sin}$,

$$\begin{array}{rcl} \nu_m & = & \frac{1-\rho^2}{1-2\rho\cos\theta_m+\rho^2} \\ y^{m,\cos} & = & \Big(\cos\big((n-2r+1)\theta_m/2\big)\Big)_{r=1}^{n} \\ y^{m,\sin} & = & \Big(\sin\big((n-2r+1)\theta_m/2\big)\Big)_{r=1}^{n} . \end{array} \tag{3.6}$$

Linearly decreasing Toeplitz matrices defined by $A^{lin}_{ij} = b_{|i-j|} = n - |i-j|$ have spectral properties analog to those of KMS matrices (trigonometric expression, interlacement, low frequency assigned to largest eigenvalue), but with more technical details available in Bünger

[2014]. This goes beyond the asymptotic case modeled by tridiagonal matrices.

Banded Robinson Toeplitz matrices typically include similarity matrices from DNA sequencing. Actually, any Robinson Toeplitz matrix becomes banded under a thresholding operation. Also, fast decaying Robinson matrices such as KMS matrices are almost banded. There is a rich literature dedicated to the spectrum of generic banded Toeplitz matrices [BoeÓttcher and Grudsky, 2005, Gray et al., 2006, Böttcher et al., 2017]. However, it mostly provides asymptotic results on the spectra. Notably, some results indicate that the eigenvectors of some banded symmetric Toeplitz matrices become, up to a rotation, close to the sinusoidal, almost equi-spaced eigenvectors observed in equations (3.5) and (3.6) [Böttcher et al., 2010, Ekström et al., 2017].

3.3.3 Spectral properties of the Laplacian

We have listed some spectral properties of typical similarity matrices. Let us conclude this section by remarking how the spectrum of a matrix relates to that of its Laplacian.

For circulant matrices A, L_A and A have the same eigenvectors since $L_A = \mathbf{diag}(A\mathbf{1}) - A = c\mathbf{I} - A$, with $c \triangleq \sum_{k=0}^{n-1} b_k$. For general symmetric Toeplitz matrices, this property no longer holds as $c_i = \sum_{j=1}^{n} b_{|i-j|}$ varies with i. Yet, for fast decaying Toeplitz matrices, c_i is almost constant except for i at the edges, namely i close to 1 or to n. Therefore, the eigenvectors of L_A resemble those of A except for the "edgy" entries.

Note that using the eigenvectors of A to embed the points boils down to classical (or, non-metric) multi-dimensional scaling (MDS) [Kruskal and Wish, 1978]. MDS is a dimensionality reduction method aiming to find an embedding of points that preserves the pairwise distances. Given a similarity matrix A, one can consider the distance matrix $D \triangleq \max(A) - A$, and apply MDS. Therefore, although the Laplacian embedding enjoys theoretical properties leading to Theorems 3.2.1, 3.3.2, in practice, classical MDS yields a similar embedding for R matrices with a fast decay.

3.4 Recovering Ordering on Filamentary Structure

We have seen that (some) similarity matrices A with a latent ordering lead to a filamentary d-**LE**. The d-**LE** integrates local proximity constraints together into a global consistent embedding. We expect isolated (or, uncorrelated) noise on A to be averaged out by the spectral picture. Therefore, we present Algorithm 3.3 that redefines the similarity S_{ij} between two items from their proximity within the d-**LE**.

3.4.1 The Algorithm

Basically, our algorithm fits the points by a line *locally*, in the same spirit as LLE, which makes sense when the data lies on a linear manifold (curve) embedded in $\mathbb{R}^K$. Note that Spectral

Ordering (Algorithm 3.1) projects all points on a given line (it only looks at the first coordinates $f_1(i)$) to reorder them. Our method does so in a local neighborhood, allowing for reordering points on a curve with several oscillations. We then run the basic Algorithms 3.1 (or 3.2 for Circular Seriation). Hence, the d-**LE** is eventually used to pre-process the similarity matrix.

Algorithm 3.3 Ordering Recovery on Filamentary Structure in $\mathbb{R}^K$.

Input: A similarity matrix $A \in \mathcal{S}_n$, a neighborhood size $k \geq 2$, a dimension of the Laplacian Embedding d.

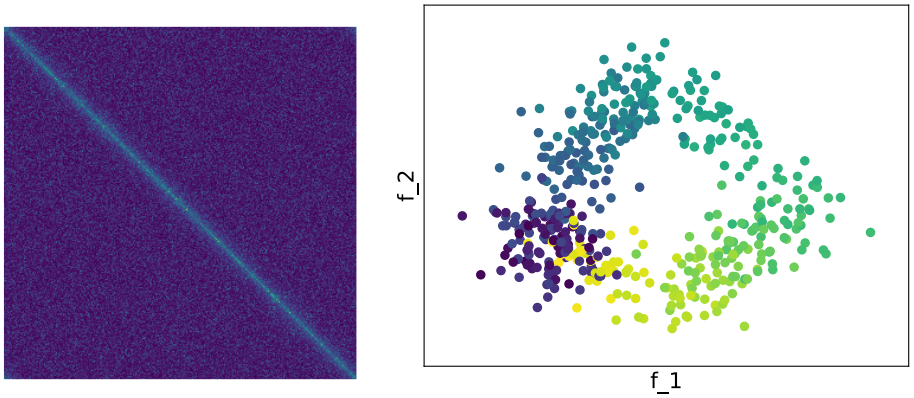
1: $\Phi = \left(\boldsymbol{y}_1^T, \ldots, \boldsymbol{y}_n^T\right)^T \in \mathbb{R}^{n \times d} \leftarrow$ d-**LE**(A) ▷ Compute Laplacian Embedding
2: Initialize $S = \mathbf{I}_n$ ▷ New similarity matrix
3: **for** $i = 1, \ldots, n$ **do**
4: $\quad V \leftarrow \{j \,:\, j \in k\text{-NN}(\boldsymbol{y}_i)\} \cup \{i\}$ ▷ find k nearest neighbors of $\boldsymbol{y}_i \in \mathbb{R}^d$
5: $\quad w \leftarrow \text{LinearFit}(V)$ ▷ fit V by a line
6: $\quad D_{uv} \leftarrow |w^T(\boldsymbol{y}_u - \boldsymbol{y}_v)|$, for $u, v \in V$. ▷ Compute distances on the line
7: $\quad S_{uv} \leftarrow S_{uv} + D_{uv}^{-1}$, for $u, v \in V$. ▷ Update similarity
8: **end for**
9: Compute σ^* from the matrix S with Algorithm 3.1 (resp., Algorithm 3.2) for a linear (resp., circular) ordering.

Output: A permutation σ^*.

In Algorithm 3.3, we compute a d-**LE** in line 1 and then a 1-**LE** (resp., a 2-**LE**) for linear ordering (resp., a circular ordering) in line 9. For reasonable number of neighbors k in the k-NN of line 4 (in practice, $k = 15$), the complexity of computing the d-**LE** dominates Algorithm 3.3. We shall see in Section 3.6 that our method, while being almost as computationally cheap as the base Algorithms 3.1 and 3.2 (roughly only a factor 2), yields substantial improvements. In line 7 we can update the similarity S_{uv} by adding any non-increasing function of the distance D_{uv}, *e.g.*, D_{uv}^{-1}, $\exp(-D_{uv})$, or $-D_{uv}$ (the latter case requires to add an offset to S afterwards to ensure it has non-negative entries, and is what we implemented in practice.) In line 9, the matrix S needs to be connected in order to use Algorithm 3.1, which is not always verified in practice (for low values of k, for instance). In that case, we reorder separately each connected component of S with Algorithm 3.1, and then merge the partial orderings into a global ordering by using the input matrix A, as detailed in Algorithm B.1, Appendix B.1.

3.4.2 Illustration of Algorithm 3.3

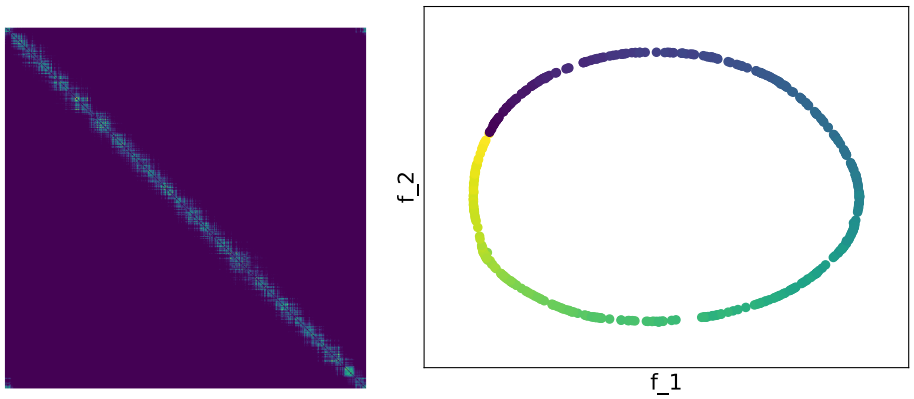
As a qualitative result, we provide a visual illustration of the method's behavior with a circular banded matrix in Figures 3.3 and 3.4. Given a matrix A (Figure 3.3a), Algorithm 3.3 computes the d-**LE**. The 2-**LE** is plotted for visualization in Figure 3.3b. Note that Algorithm 3.2 would directly infer the circular ordering from the 2-**LE** displayed in Figure 3.3b. Then, it creates a new matrix S (Figure 3.4a) from the local alignment of the points in the d-**LE**. Finally, from the new matrix S, it computes the 2-**LE** (Figure 3.4a), on which Algorithm 3.2 is eventually



(a) Noisy circular banded matrix A

(b) Noisy 2-**LE**

Figure 3.3: Noisy Circular Banded matrix (3.3a) and associated 2d Laplacian embedding (3.3b).



(a) Matrix S from Algorithm 3.3

(b) New 2-**LE**

Figure 3.4: Matrix S created through Algorithm 3.3 (3.4a), and associated 2d-Laplacian embedding (3.4b).

ran.

3.5 Perturbation analysis

The spectrum is a continuous function of the matrix. We can bound the deformation of the 2-**LE** under a perturbation of the matrix A using the Davis-Kahan theorem [Davis and Kahan, 1970], well introduced in [Von Luxburg, 2007, Theorem 7], yielding the following result.

3.5.1 Application of the Davis-Kahan Theorem

Proposition 3.5.1 (Davis-Kahan)**.** *Let L and $\tilde{L} = L + \delta L$ be the Laplacian matrices of $A \in \mathcal{C}_R^*$ and $A + \delta A \in \mathbf{S}^n$, respectively, and $V, \tilde{V} \in \mathbb{R}^{2\times n}$ be the associated 2-***LE*** of L and $\tilde{L}$, i.e., the concatenation of the two eigenvectors associated to the two smallest non-zero eigenvalues,*

written $\lambda_1 \leq \lambda_2$ for L. Then, there exists an orthonormal rotation matrix O such that

$$\frac{\|V_1 - \tilde{V}_1 O\|_F}{\sqrt{n}} \leq \frac{\|\delta A\|_F}{\min(\lambda_1, \lambda_2 - \lambda_1)} \ . \tag{3.7}$$

For circular matrices, we can derive a slightly finer result.

Proposition 3.5.2 (Davis-Kahan)**.** *Consider L a graph Laplacian of a R-symmetric-circular Toeplitz matrix A. We add a symmetric perturbation matrix H and denote by $\tilde{A} = A + H$ and $\tilde{L}$ the new similarity matrix and graph Laplacian respectively. Denote by $(p_i)_{i=1,\dots,n}$ and $(\tilde{p}_i)_{i=1,\dots,n}$ the 2-***LE** *coming from L and $\tilde{L}$ respectively. Then there exists a cyclic permutation τ of $\{1, \dots, n\}$ such that*

$$\sup_{i=1,\dots,n} ||p_{\tau(i)} - \tilde{p}_i||_2 \leq \frac{2^{3/2} \min(\sqrt{2}||L_H||_2, ||L_H||_F)}{\min(|\lambda_1|, |\lambda_2 - \lambda_1|)} \ , \tag{3.8}$$

where $\lambda_1 < \lambda_2$ are the first non-zeros eigenvalues of L.

Proof. For a matrix $V \in \mathbb{R}^{n \times d}$, denote by

$$\big|\big|V\big|\big|_{2,\infty} = \sup_{i=1,\dots,n} \big|\big|V_i\big|\big|_2 \ ,$$

where V_i are the columns of V. Because in $\mathbb{R}^n$ we have $||\cdot||_\infty \leq ||\cdot||_2$, it follows that

$$\begin{aligned} \big|\big|V\big|\big|_{2,\infty} &\leq \big|\big|\big(||V_i||\big)_{i=1,\dots,n}\big|\big|_2 = \sqrt{\sum_{i=1}^{n} ||V_i||_2^2} \\ &\leq \big|\big|V\big|\big|_F \ . \end{aligned}$$

We apply [Yu et al., 2014, Theorem 2] to our perturbed matrix, a simpler version of classical Davis-Kahan theorem [Davis and Kahan, 1970].

Let's denote by (λ_1, λ_2) the first non-zeros eigenvalues of L and by V its associated 2-dimensional eigenspace. Similarly denote by $\tilde{V}$ the 2-dimensional eigenspace associated to the first non-zeros eigenvalues of $\tilde{L}$. There exists a rotation matrix $O \in SO_2(\mathbb{R})$ such that

$$||\tilde{V} - VO||_F \leq \frac{2^{3/2} \min(\sqrt{2}||L_H||_2, ||L_H||_F)}{\min(|\lambda_1|, |\lambda_2 - \lambda_1|)} \ . \tag{3.9}$$

In particular we have

$$\begin{aligned} \big|\big|\tilde{V} - VO\big|\big|_{2,\infty} &\leq \big|\big|\tilde{V} - VO\big|\big|_F \\ \big|\big|\tilde{V} - VO\big|\big|_{2,\infty} &\leq \frac{2^{3/2} \min(\sqrt{2}||L_H||_2, ||L_H||_F)}{\min(|\lambda_1|, |\lambda_2 - \lambda_1|)} \end{aligned}$$

Finally because A is a R-symmetric-circular Toeplitz, from Theorem 3.3.2, the row of V are n ordered points uniformly sampled on the unit circle. Because applying a rotation is equivalent to translating the angle of these points on the circle. It follows that there exists a cyclic permutation τ such that

$$\sup_{i=1,\dots,n} ||p_i - \tilde{p}_{\tau(i)}||_2 \leq \frac{2^{3/2}\min(\sqrt{2}||L_H||_2, ||L_H||_F)}{\min(|\lambda_1|, |\lambda_2 - \lambda_1|)} ,$$

■

3.5.2 Exact recovery with noise for Algorithm 3.2

These results bounding the perturbation of the embedding allow to find conservative guarantees of ordering recovery with Algorithm 3.2, based on simple geometric reasoning. Indeed, when all the points remain in a sufficiently small ball around their original position on the circle, Algorithm 3.2 can exactly find the ordering. Let us start with a geometrical lemma quantifying the radius of the ball around each $(\cos(\theta_k), \sin(\theta_k))$ so that they do not intersect.

Lemma 3.5.3. *For $\boldsymbol{x} \in \mathbb{R}^2$ and $\theta_k = 2\pi k/n$ for $k \in \mathbb{N}$ such that*

$$||\boldsymbol{x} - (\cos(\theta_k), \sin(\theta_k))||_2 \leq \sin(\pi/n) , \tag{3.10}$$

we have

$$|\theta_x - \theta_k| \leq \pi/n ,$$

where $\theta_x = \tan^{-1}(\boldsymbol{x}_1/\boldsymbol{x}_2) + 1[\boldsymbol{x}_1 < 0]\pi$.

Proof. Let $\boldsymbol{x}$ that satisfies (3.10). Let's assume without loss of generality that $\theta_k = 0$ and $\theta_x \geq 0$. Assume also that $\boldsymbol{x} = \boldsymbol{e}_1 + \sin(\pi/n)\boldsymbol{u}_x$ where $\boldsymbol{u}$ is a unitary vector. A $\boldsymbol{x}$ for which θ_x is maximum over these constrained is such that $\boldsymbol{u}_x$ and $\boldsymbol{x}$ are orthonormal.

Parametrize $\boldsymbol{u}_x = (\cos(\gamma), \sin(\gamma))$, because $\boldsymbol{u}_x$ and $\boldsymbol{x}$ are orthonormal, we have $\cos(\gamma) = \sin(-\pi/n)$. Finally since $\theta_x \geq 0$, it follows that $\gamma = \pi/2 + \pi/n$ and hence with elementary geometrical arguments $\theta_x = \pi/n$.

■

Proposition 3.5.4 (Exact circular recovery under noise in Algorithm 3.2)**.** *Consider a matrix $\tilde{A} = \Pi^T A \Pi + H$ with A a $R-$circular Toeplitz (Π is the matrix associated to the permutation σ) and H a symmetric matrix such that*

$$\min(\sqrt{2}||L_H||_2, ||L_H||_F) \leq 2^{-3/2}\sin(\pi/n)\min(|\lambda_1|, |\lambda_2 - \lambda_1|) ,$$

where $\lambda_1 < \lambda_2$ are the first non-zeros eigenvalues of the graph Laplacian of $\Pi^T A \Pi$. Denote by $\hat{\sigma}$ the output of Algorithm 3.2 when having $\tilde{A}$ as input. Then there exists a cyclic permutation τ such that

$$\hat{\sigma} = \sigma^{-1} \circ \tau^{-1} \ . \tag{3.11}$$

Proof. We have

$$\Pi^T \tilde{A} \Pi = A + \Pi^T H \Pi \ .$$

L is the graph Laplacian associated to A and $\tilde{L}$, the one associated to $\tilde{A}$. Denote by $(p_i)_{i=1,\dots,n}$ and $(\tilde{p}_i)_{i=1,\dots,n}$ the 2-**LE** coming from L and $\tilde{L}$ respectively. $(\tilde{p}_{\sigma^{-1}(i)})_{i=1,\dots,n}$ is the 2-**LE** coming from the graph Laplacian of $\Pi^T \tilde{A} \Pi$.

Applying Proposition 3.5.2 with $\Pi^T \tilde{A} \Pi$, there exists a cyclic permutation such that

$$\sup_{i=1,\dots,n} ||\tilde{p}_{\sigma^{-1}(i)} - p_{\tau(i)}||_2 < \frac{2^{3/2} \min(\sqrt{2}||L_{H^\pi}||_2, ||L_{H^\pi}||_F)}{\min(|\lambda_1|, |\lambda_2 - \lambda_1|)} \ ,$$

with $H^\pi = \Pi^T H \Pi$, $\lambda_1 < \lambda_2$ the first non zero eigenvalues of A.

Graph Laplacian involve the diagonal matrix D_H. In particular we have that $D_{H^\pi} = \Pi^T D_H \Pi$. For the unnormalized Laplacian, it results in $L_{H^\pi} = \Pi^T L_H \Pi$. We hence have

$$\begin{aligned} \sup_{i=1,\dots,n} ||\tilde{p}_{\sigma(i)} - p_{\tau(i)}||_2 &< \frac{2^{3/2} \min(\sqrt{2}||L_H||_2, ||L_H||_F)}{\min(|\lambda_1|, |\lambda_2 - \lambda_1|)} \\ \sup_{i=1,\dots,n} ||\tilde{p}_i - p_{\tau \circ \sigma^{-1}(i)}||_2 &< \sin(\pi/n) \ . \end{aligned}$$

From Theorem 3.3.2, $p_i = \cos(2\pi i/n)$ for all i. It follows that for any i

$$||\tilde{p}_i - \cos(2\pi \tau \circ \sigma(i)/n)||_2 < \sin(\pi/n) \ .$$

Algorithm 3.2 recovers the ordering by sorting the values of

$$\theta_i = \tan^{-1}(\tilde{p}_i^1 / \tilde{p}_i^2) + 1[\tilde{p}_i^1 < 0]\pi \ ,$$

where $\tilde{p}_i = (\tilde{p}_i^1, \tilde{p}_i^2)$. Applying Lemma 3.5.3:

$$|\theta_i - 2\pi(\tau \circ \sigma^{-1})(i)/n| < \pi/n \quad \forall i \in \{1, \dots, n\},$$

so that

$$\theta_{\sigma^{-1} \circ \tau^{-1}(1)} \leq \cdots \leq \theta_{\sigma^{-1} \circ \tau^{-1}(n)} \ . \tag{3.12}$$

Finally $\hat{\sigma} = \sigma^{-1} \circ \tau^{-1}$. ■

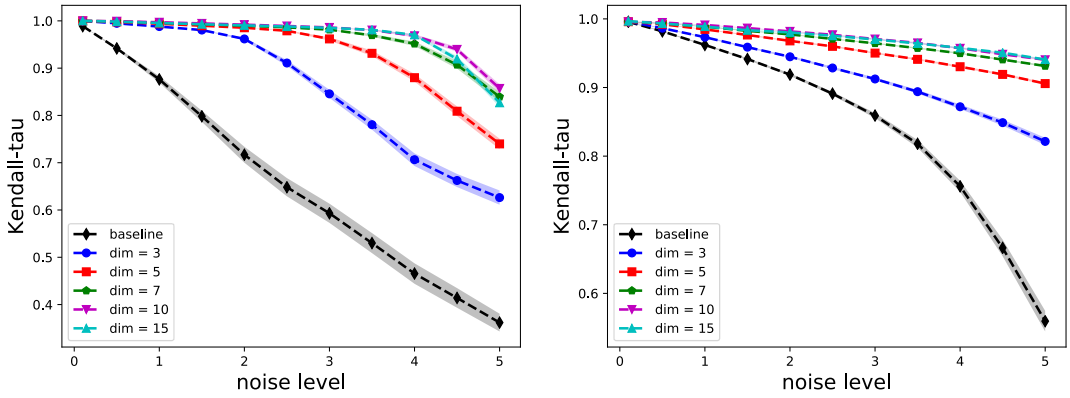
3.6 Numerical Results

In this section, we present results from synthetic experiments where we seek to reorder pre-R matrices corrupted with noise. They quantify the performance gain achieved by using the d-**LE** instead of the 1-**LE** (or 2-**LE**, for Circular Seriation), in terms of correlation between the ground-truth permutation (for which the noiseless matrix is R), and the permutation found by the algorithms. Then, we set out to use our method to determine the layout of reads in *de novo* assembly, *i.e.*, reordering overlap-based similarity matrices.

3.6.1 Synthetic Experiments

We performed synthetic experiments with noisy observations of Toeplitz matrices A, either linear ($\mathcal{L}_R$) or circular ($\mathcal{C}_R^*$). We added a uniform noise on all the entries, with an amplitude parameter a varying between 0 and 5, with maximum value of the noise $a\|A\|_F$. The matrices A used are either banded (sparse), with linearly decreasing entries when moving away from the diagonal, or dense, with exponentially decreasing entries (KMS matrices). We used $n = 500$, several values for the parameters k (number of neighbors) and d (dimension of the d-**LE**), and various scalings of the d-**LE** (parameter α in (α, d)-**LE**), yielding similar results (see sensitivity to the number of neighbors k and to the scaling (α, d)-**LE** in Appendix B.2.4). In an given experiment, the matrix A is randomly permuted with a ground truth permutation π^*. We report the Kendall-Tau scores between π^* and the solution of Algorithm 3.3 for different choices of dimension K, for varying noise amplitude a, in Figure 3.5, for banded (circular) matrices. For the circular case, the ordering is defined up to a shift. To compute a Kendall-Tau score from two permutations describing a circular ordering, we computed the best Kendall-Tau scores between the first permutation and all shifts from the second, as detailed in Algorithm B.2. The analog results for exponentially decaying (KMS) matrices are given in Appendix B.2.3, Figure B.3. For a given combination of parameters, the scores are averaged on 100 experiments and the standard-deviation divided by $\sqrt{n_{\text{exps}}} = 10$ (for ease of reading) is plotted in transparent above and below the curve. The baseline (in black) corresponds to the basic spectral method of Algorithm 3.1 for linear and Algorithm 3.2 for circular seriation. Other lines correspond to given choices of the dimension of the d-**LE**, as written in the legend.

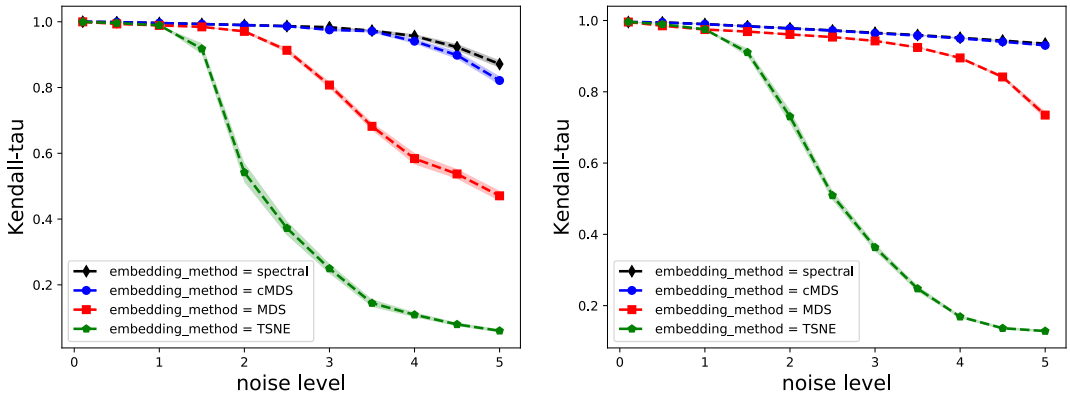
We observe that leveraging the additional dimensions of the d-**LE** unused by the baseline methods Algorithm 3.1 and 3.2 substantially improves the robustness of Seriation. For instance, in Figure 3.5a, the performance of Algorithm 3.3 is almost optimal for a noise amplitude going from 0 to 4, when it falls by a half for Algorithm 3.1. We illustrate the effect of the pre-processing of Algorithm 3.3 in Figures 3.3 and 3.4, Appendix 3.4.2.



(a) Linear Banded (b) Circular Banded

Figure 3.5: Kendall-Tau scores for Linear (3.5a) and Circular (3.5b) Seriation for noisy observations of banded, Toeplitz, matrices, displayed for several values of the dimension parameter of the d-**LE**(d), for fixed number of neighbors $k = 15$.

Finally, in Figure 3.6, we compare our Algorithm 3.3 when using alternative embedding methods mentioned in §3.2.4, namely classical MDS (denoted cMDS), metric-MDS (denoted MDS), and t-SNE, instead of the spectral (Laplacian embedding). The method performs similarly when used with a classical-MDS or spectral embedding, which is not surprising since both method rely on the spectral decomposition of slightly differently normalized similarity matrices. The spectral embedding outperforms the other techniques in these experiments, empirically justifying the choice of embedding made from theoretical considerations.



(a) Linear Banded (b) Circular Banded

Figure 3.6: Kendall-Tau scores for Seriation for noisy observations of Linear and Circular banded, Toeplitz, matrices, displayed for several embedding methods. All embeddings are of dimension $d = 8$ except t-SNE for which $d = 2$. The number of neighbors is set to $k = 15$.

3.6.2 Genome assembly experiment : bacterial genomes with ONT long-reads

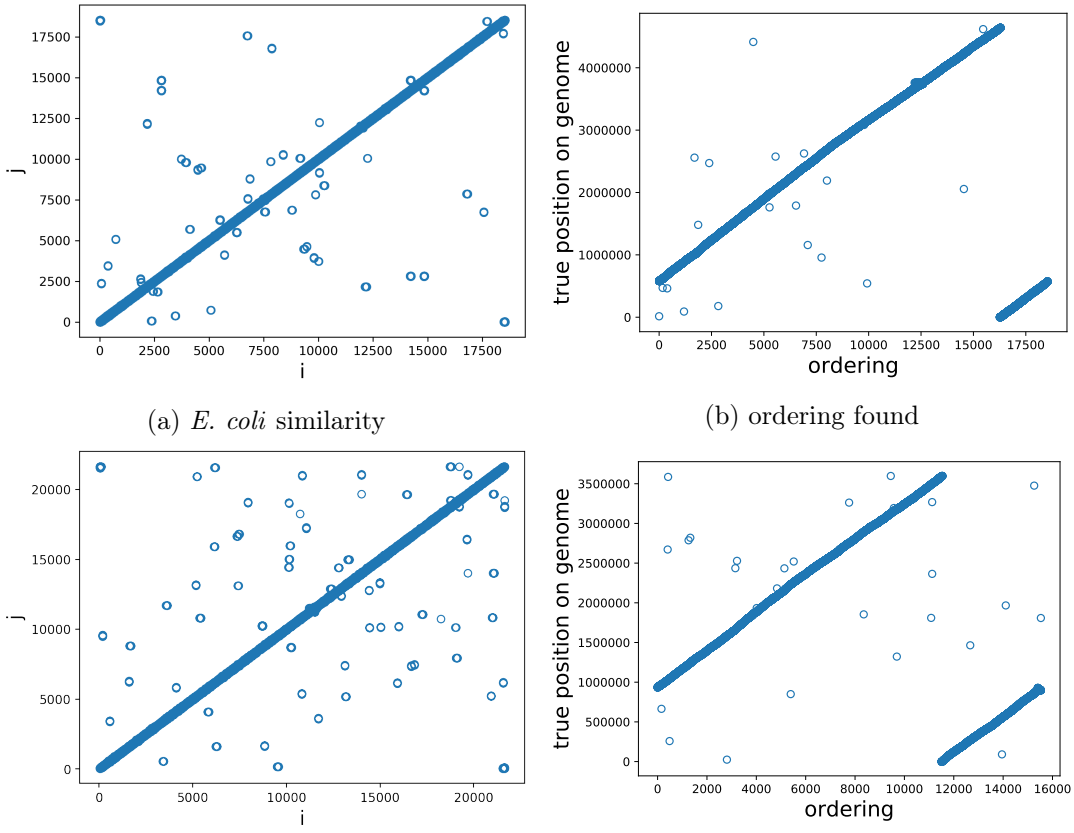
We tested the method on some of the datasets introduced in Chapter 2. Recall that in *de novo* genome assembly, a whole DNA strand is reconstructed from randomly sampled sub-fragments (called *reads*) whose positions within the genome are unknown. The genome is over-sampled so that all parts are covered by multiple reads with high probability. The Overlap-Layout-Consensus (OLC) assembly paradigm is based on three steps. First, compute the overlaps between all pairs of read. This provides a similarity matrix A, whose entry (i, j) measures how much reads i and j overlap (and is zero if they do not). Then, determine the layout from the overlap information, that is to say find an ordering and positioning of the reads that is consistent with the overlap constraints. This step, akin to solving a one dimensional jigsaw puzzle, is a key step in the assembly process. Finally, given the tiling of the reads obtained in the layout stage, the consensus step aims at determining the most likely DNA sequence that can be explained by this tiling. It essentially consists in performing multi-sequence alignments.

In the true ordering (corresponding to the sorted reads' positions along the genome), a given read overlaps much with the next one, slightly less with the one after it, and so on, until a point where it has no overlap with the reads that are further away. This makes the read similarity matrix Robinson and roughly band-diagonal (with non-zero values confined to a diagonal band). Finding the layout of the reads therefore fits the Linear Seriation framework (or Circular Seriation for circular genomes, as illustrated in Figure 3.1). In practice however, there are some repeated sequences (called *repeats*) along the genome that induce false positives in the overlap detection tool [Pop, 2004], resulting in non-zero similarity values outside (and possibly far away) from the diagonal band. The similarity matrix ordered with the ground truth is then the sum of a Robinson band matrix and a sparse "noise" matrix, as in Figure 3.7a. Because of this sparse "noise", the basic spectral Algorithm 3.1 fails to find the layout, as the quadratic loss appearing in 2-SUM is sensitive to outliers.

In Chapter 2, we have proposed the so-called bandwidth heuristic, which relies only on the baseline Algorithm 3.1 and iteratively breaks the overlap graph in connected components until the sub-components seem to contain no outlier. Instead, we show here that the simple multi-dimensional extension proposed in Algorithm 3.3 suffices to capture the ordering of the reads despite the repeats.

We used our method to perform the layout of an *E. coli* and *A. baylyi* bacterial genomes sequenced with the Oxford Nanopore Technology MinION device. Details on the data are given in Section 2.3.1 from Chapter 2. We computed the overlaps with the `minimap2` dedicated software [Li, 2018], as detailed in Appendix B.2.1.

The method only worked with a sufficient threshold on the input similarity matrix in a pre-processing step. Here, we used 50% for *E. coli* dataset, and 70% for *A. baylyi*. The new similarity matrix S computed from the embedding in Algorithm 3.3 was disconnected, resulting in several connected component instead of one global ordering (see Figure B.2b).



(a) *E. coli* similarity

(b) ordering found

(c) *A. baylyi* similarity

(d) ordering found

Figure 3.7: Overlap-based similarity matrix from *E. coli* (3.7a) and *A. baylyi* (3.7c) reads, and the ordering found with Algorithm 3.3 (for *E. coli* - 3.7b, and *A. baylyi* - 3.7d) versus the position of the reads within a reference genome obtained by mapping to a reference with minimap2. The genome being circular, the ordering is defined up to a shift, which is why we observe two lines instead of one in (3.7b and 3.7d).

However, the sub-orderings could be unambiguously merged into one in a simple way described in Algorithm B.1, resulting in the orderings shown in Figures 3.7b and 3.7d. In practice, the threshold on the input similarity can be set as high as possible as long as the resulting sub-orderings can be merged into one single component (yielding a single contig). This criterion leads to the results presented here, where the bacterial genomes are correctly reordered.

The Kendall-Tau score between the ordering found and the one obtained by sorting the position of the reads along the genome (obtained by mapping the reads to a reference with `minimap2`) is of 99.5% for the *E. coli* dataset, and 99.3% for *A. baylyi*, using Algorithm B.2 to account for the circularity of the genome.

3.6.3 Genome assembly using Hi-C data

Although the output of a DNA sequencing experiment consists in a linear representation of the genome (a string, *e.g.*, 'AAAT...GC'), or a collection of linear sequences (when there are several contigs or chromosomes), physical DNA has a non-linear spatial organization in the 3-D space (it can be thought of as a ball of yarn, or a plate of spaghetti). Hi-C is a chromosome conformation capture (3C) technique measuring the frequency of physical interactions between genomic loci that are nearby in 3-D space, though they can be separated by many nucleotides in the linear genome. Experiments indicate that the spatial proximity between genomic loci is not random, and provide valuable information in, *e.g.*, gene identification and regulation [Dekker et al., 2013].

Interestingly, the frequency of interactions between genomic loci (called bins) tend to decrease with their distance in the linear genome [Lieberman-Aiden et al., 2009b]. Also, interactions are more frequent within a given chromosome than between distinct chromosomes. Thus, we can use Hi-C data to find the layout of the bins in a *de novo* assembly experiment [Dudchenko et al., 2017]. We have seen that due to repeats, reads-overlap-based similarity matrices had a specific structure (banded Robinsonian matrices + sparse out-of-band noise) challenging seriation methods. Hi-C similarity matrices are also expected to be close to Robinsonian matrices, but with a different structure. The underlying stylized Robinsonian matrices are not banded but harbour a power-law decay when moving away from the diagonal, and the noise seem to have another structure that repeat-induced noise.

We performed experiments using synthetic and real Hi-C data. The synthetic data include similarity matrices of four synthetic genomes of lengths 100, 150, 300 and 1000 bins. In the true ordering, these similarity matrices follow a power-law decay, $1/|i-j+1|^2$, when moving away from the diagonal, with additive uniform noise.

There are also two synthetic similarity matrices modeling Hi-C data from a genome with multiple chromosomes, where the entries within a given chromosome (block of the matrix) are generated with the same rules as the single-stranded synthetic matrices described above, and the entries between distinct chromosomes are set with a low amplitude, sparse noise. The first one, called DL1, has 7 chromosomes of respective lengths 30, 40, 50, 70, 50, 40, 30 bins (total length 310 bins). The second one, called DL2, has 7 chromosomes of lengths 10, 20, 40, 80, 160, 320, 370 bins (total length 1000 bins).

Finally, we used four similarity matrices constructed from real data. The first one was built by mapping Hi-C reads of *Plasmodium knowlesi* to a reference genome split in 10kbp bins. The reference genome is made of 14 chromosomes, and the resulting matrix is of size 2014. The other matrices were built by mapping Hi-C reads of *Spodoptera frugiperda* to a genome assembly obtained with Pacbio long reads. The reference assembly is fragmented, and each of the three matrices Sf200, Sf669 and Sf846 is the restriction of the Hi-C reads that map to a given contig. However, these contigs may contain multiple chromosomes, or separated regions that

Table 3.1: Seriation results on synthetic Hi-C data from linear single-stranded genome.

		Spectral	mdso
$N = 100$	τ (%)	99.4	**99.7**
	Time (s)	0.74	0.18
$N = 150$	τ (%)	99.7	**100**
	Time (s)	0.093	0.23
$N = 300$	τ (%)	**100**	**100**
	Time (s)	0.42	0.70
$N = 1000$	τ (%)	**100**	**100**
	Time (s)	2.1	1.7

were wrongfully assembled together by the assembler. The respective lengths of the resulting matrices Sf200, Sf669 and Sf846 are 198, 284, and 461 bins.

We acknowledge Dominique Lavenier from the GenOuest group at INRIA Rennes for providing the synthetic and *Plasmodium knowlesi* data, and Fabrice Legeai, from the same group, for the *Spodoptera frugiperda* data.

Single-stranded synthetic data

In Table 3.1, we provide the Kendall-Tau (written τ) correlation score between the true permutation and the ordering obtained with the spectral baseline method (Algorithm 1.1) and the method introduced in this chapter (Algorithm 3.3), on the synthetic single-stranded frequency matrices of sizes 100, 150, 300, 1000. We also indicate the running time (ran on a 2014 Macbook Pro). We can see that although our method provides a marginal improvement on these matrices, the noise is sufficiently low for Algorithm 1.1 to be efficient.

3.6.4 Assembly of genomes with multiple chromosomes with Hi-C data

Let us consider the synthetic, multiple-chromosomes Hi-C data and the real data (which also contains separated fragments). For genomes with several chromosomes, we wish to find an ordering of the bins within any chromosome, rather than a global ordering with all chromosomes mixed. Yet, the chromosome assignment is not given in the Hi-C data. We can then attempt to cluster the bins into distinct chromosomes before reordering the elements in a given chromosome. If there was zero similarity between any two bins that span distinct chromosomes, then the input similarity matrix would be disconnected, and it would be trivial to find this cluster assignment. However, in practice we observe some high values of frequency of interactions between bins spanning distinct chromosomes, making the clustering step non trivial. In some cases, for instance here with the synthetic data and the *Plasmodium knowlesi* data, the user may know in advance the number of target chromosomes. In some others, for instance here

with the *Spodoptera frugiperda* data, or in a *de novo* perspective, that number is unknown.

Methods for clustering and ordering

Let us comment on the spectral method for seriation and clustering. From equations (3.1) and (Relax. 2-SUM), we can see that if the input similarity matrix is disconnected into K connected components, then the eigenvalue 0 has multiplity $K+1$, with associated eigenvectors $\mathbf{1}$ and the indicator vectors $\mathbb{1}_{C_k}$ of the K connected components C_k, $k = 1, \ldots, K$. Indeed, consider a given connected component C_k. By definition, $A_{ij} = 0$ if $i \in C_k$ and $j \notin C_k$. Thus, if $f = \mathbb{1}_{C_k}$ in the objective from equation (3.1), then for each pair (i, j), either $i, j \in C_k$ and $f(i) = f(j) = 1$, thus $f(i) - f(j) = 0$, or $i, j \notin C_k$ and $f(i) = f(j) = 0$, thus thus $f(i) - f(j) = 0$, or $i \in C_k, j \notin C_k$ and $A_{ij} = 0$. Therefore, all the products appearing in (3.1) are equal to zero. This is at the core of spectral clustering [Von Luxburg, 2007]. Therefore, for a matrix with K disconnected clusters, the K-**LE** only contains information regarding the clustering, but not the intra-cluster ordering, which is relegated to the higher-order eigenvectors. On the other hand, as we have seen in this chapter, if the similarity matrix is connected, then the eigenvalue 0 has a unique corresponding eigenvector, $\mathbf{1}$, and the Fiedler vector (the first non-zero eigenvector) is associated to the second smallest eigenvalue $\lambda_1 > 0$ and contains information about the ordering (remark that we have used the notation λ_1 for the second smallest eigenvector in this chapter, although the notation $\lambda_2 > 0$ is often used in the literature). When the similarity is "weakly" connected, λ_1 gets close to 0, the computation of the K-**LE** becomes less stable, and the ordering information contained in the K-**LE** gets diluted at the profit of clustering information. Therefore, we also include the t-SNE-embedding version of our method in the experiments, as t-SNE handles disconnected matrices seamlessly, where the clustering appears in the 2D embedding without losing the local serial structure, if any.

We consider the following methods to obtain a chromosome assignment and a reordering of the bins inside each chromosome. We denote the method presented in this chapter by mdso, standing for multi-dimensional spectral ordering.

Pre-processing. We found empirically that the following pre-processing of the similarity matrices enhanced the cluster structure and improved the results of the methods described below,

$$A_{ij} \leftarrow \frac{\sum_{h \in \text{k-NN}(i), l \in \text{k-NN}(j)} A_{hl}}{|\text{k-NN}(i)||\text{k-NN}(j)|}, \tag{3.13}$$

where k-NN(i) are the k-nearest neighbors of i (the bins with the top k similarity values with i). In practice we use $k = 15$. Appendix Figure B.8 illustrates the effect of this pre-preprocessing.

Spectral Clustering + Spectral Ordering (SC+SO). We first cluster the data from the similarity matrix (using no assumption regarding the Robinsonian structure) with spectral clustering, and then use Algorithm 1.1 in each cluster.

Spectral Clustering + mdso (SC+mdso). We first cluster the data from the similarity matrix

(using no assumption regarding the Robinsonian structure) with spectral clustering, and then use Algorithm 3.3 in each cluster.

mdso. We only run Algorithm 3.3 on the input similarity, hoping that the d-**LE** will capture both clustering and intra-cluster ordering information, leading to a new similarity matrix S computed in Algorithm 3.3 that has connected components corresponding to the chromosomes.

tSNE-mdso. Same as the previous method, except that we use t-SNE to compute the embedding instead of the Laplacian embedding, with $d = 2$. The behaviour of t-SNE does not change whether the input similarity is connected or disconnected. If it is disconnected, it will simply find an embedding with separate clusters, but keep the intra-cluster structure.

Remark that the two first methods (SC+SO and SC+mdso) require the user to provide the number of desired clusters as input. Therefore, we only use them for the synthetic data and the *Plasmodium knowlesi*, for which we know the number of chromosomes.

Evaluation of clustering and sub-orderings

We evaluate the quality of the clustering with respect to the ground truth chromosome assignments with two scores. Given two partitions into K clusters, $(\Omega_1, \dots, \Omega_K)$ and $(C_1, \dots, C_K)$, such that $\cup_{k=1}^K \Omega_k = \cup_{k=1}^K C_k = \{1, \dots, N\}$, and $\cap_{k=1}^K \Omega_k = \cap_{k=1}^K C_k = \emptyset$, the purity index defined by,

$$\text{purity}(\Omega, C) = \frac{1}{N} \sum_{k=1}^{K} \max_{j=1,\dots,K} |\Omega_k \cap C_j| \tag{3.14}$$

where $|\cdot|$ is the cardinal of a set. It takes value between 0 and 1, and is equal to 1 if and only if both partitions are equal (the higher, the better). This metric can only be used when we compare two partitions with the same number of clusters. When using spectral clustering, the user specifies the number of clusters, hence we can always find a clustering with as many clusters as the ground truth. Still, when using mdso or tSNE-mdso, we do not control the number of clusters found by the method. Therefore, we also use the following metric measuring the distance between two partitions $(\Omega_1, \dots, \Omega_K)$ and $\left(C_1, \dots, C'_K\right)$ (where K and K' may differ), as in Bach and Harchaoui [2008],

$$d^2(\Omega, C) = \frac{K + K'}{2} - \sum_{k=1}^{K} \sum_{k'=1}^{K'} \frac{|\Omega_k \cap C_{k'}|^2}{|\Omega_k||C_{k'}|} \tag{3.15}$$

It takes value between 0 and $\frac{K+K'}{2} - 1$ (and between 0 and $K - 1$ if the two partitions have the same number of clusters) (the lower, the better).

Finally, given two raking (in the form of a permutation) π_1 and π_2, we say that a pair $i < j$ is concordant if the two rankings agree, namely, $\pi_1(i) > \pi_1(j)$ and $\pi_2(i) > \pi_2(j)$, or $\pi_1(i) < \pi_1(j)$ and $\pi_2(i) < \pi_2(j)$. We say that they are discordant otherwise, namely, $\pi_1(i) > \pi_1(j)$ and

Table 3.2: Seriation results on synthetic Hi-C data from genomes with multiple chromosomes.

		SC+SO	SC+mdso	mdso	tSNE-mdso
DL1	# Chr.	7	7	7	7
	Purity	100	100	100	100
	Cluster dist.	0.0	0.0	0.0	0.0
	$\tilde{\tau}$ (%)	**96.7**	83.3	**96.7**	95.6
	Time (s)	0.079	0.24	0.31	3.77
DL2	# Chr.	7	7	7	7
	Purity	100	100	100	100
	Cluster dist.	0.0	0.0	0.0	0.0
	$\tilde{\tau}$ (%)	**99.5**	99.2	99.4	**99.5**
	Time (s)	0.19	0.53	1.45	15.9

$\pi_2(i) < \pi_2(j)$, or $\pi_1(i) < \pi_1(j)$ and $\pi_2(i) > \pi_2(j)$. Then, the Kendall-Tau rank correlation is defined by,

$$\tau = \frac{\text{number of concordant pairs} - \text{number of discordant pairs}}{n(n-1)/2}. \tag{3.16}$$

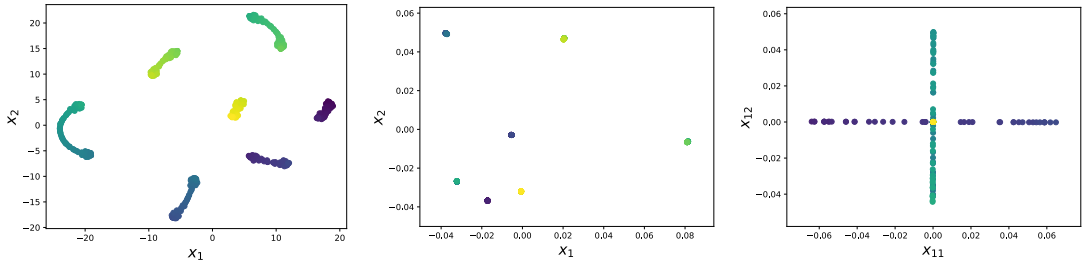
If we have clustered data, it makes no sense to compare global orderings including the concordance of pairs (i, j) where i and j are in two separate clusters. We therefore use the following definition of the weighted Kendall-Tau metric between two sets of local orderings in K clusters,

$$\tilde{\tau} = \frac{\sum_k \tau_k n_k(n_k-1)/2}{\sum_k n_k(n_k-1)/2} \tag{3.17}$$

where τ_k is the Kendall-Tau score between the two local orderings in the k-th cluster, and n_k is the number of points within the k-th cluster.

Results on data with reference clustering

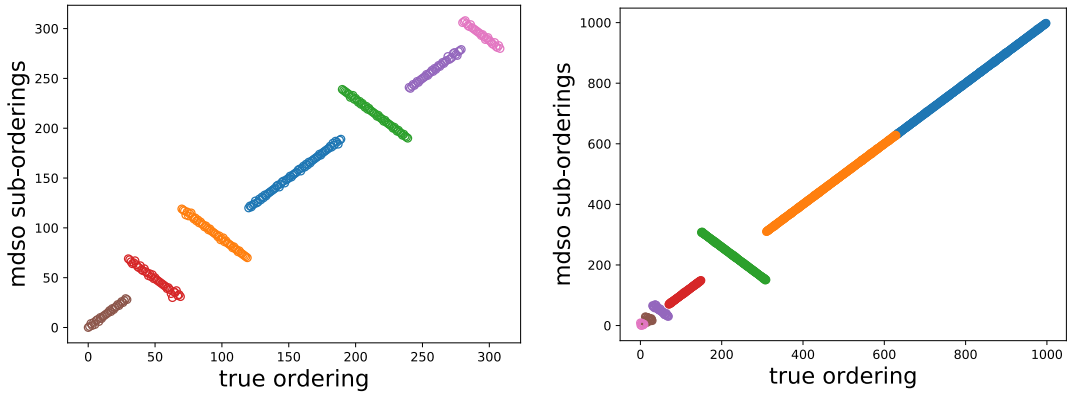
In Table 3.2 we compare these methods on the two synthetic datasets DL1 and DL2. We observe that on this data, the inter-chromosomes frequency is sufficiently low for all methods to recover the correct clustering. Notably, the spectral and t-SNE based embeddings used in the two versions of mdso contain both clustering and ordering information (multiple, separated filamentary structures). We illustrate this in Figure 3.8. The two first dimensions of the spectral embedding (Figure 3.8b) contain mostly clustering information, although zooming in allow to see that the points roughly follow a filament in each cluster. The higher order eigenvectors of the Laplacian (Figure 3.8c) contain partial ordering information. Note that Algorithm 3.3 leverages all these eigenvectors simultaneously in the line fitting procedure. We also display



(a) t-SNE (b) spectral, $d = 1, 2$ (c) spectral, $d = 11, 12$

Figure 3.8: t-SNE 2D embedding (3.8a), and two projections of the spectral embedding (3.8b, 3.8c) for the synthetic multiple chromosomes frequency matrix DL1. The colormap goes from dark blue to yellow with the absolute position of the bins, where the first bin is the first bin in the first chromosome, and the last bin is the last bin from the last chromosome (with an arbitrary ordering between chromosomes for illustrative purposes).

the sub-orderings found in the chromosomes for the mdso method in Figure 3.9.



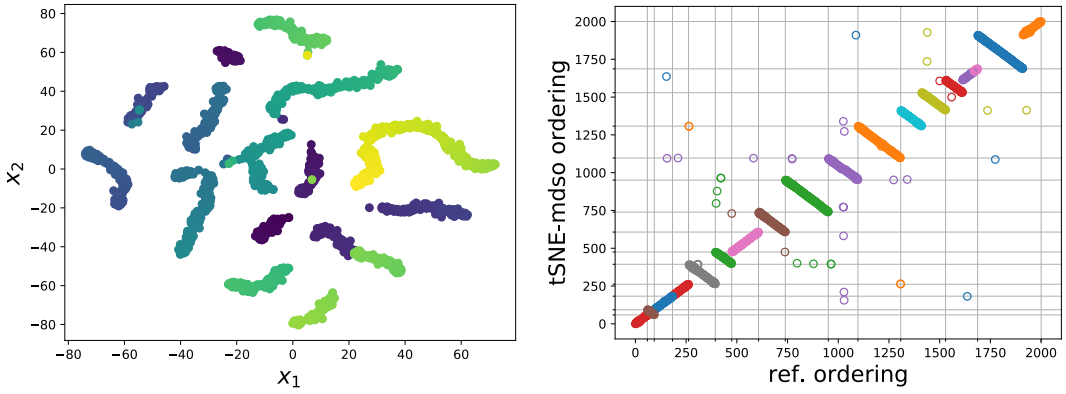
(a) DL1 (b) DL2

Figure 3.9: sub-orderings found by mdso on synthetic, multiple chromosomes Hi-C data.

In Table 3.3, we provide results on the *Plasmodium knowlesi* data. Interestingly, for the tSNE-mdso method, the weighted Kendall-Tau ($\tilde{\tau}$) metric varies significantly depending on the random initialization of t-SNE, with values ranging from about 60% to 92%. After investigation, it appears that the low values of $\tilde{\tau}$ are due to the fact that when the method finds a cluster which contains several non contiguous sets of bins (say, two distant chromosomes), even if the ordering within the two chromsomes is approximately correct, the way the sub-orderings are arranged together can make $\tilde{\tau}$ vary. We illustrate these variations in Figure 3.10 and Appendix Figure B.9, where we show the tSNE embedding and the resulting sub-orderings in two instances of the experiment, one leading to $\tilde{\tau} = 91.8$, and one to $\tilde{\tau} = 61.6$, where the low $\tilde{\tau}$ score is evidently due to the blue component spanning several chromosomes. In Table 3.3, we provide the results for tSNE-mdso in the following form : mean $\pm$ standard deviation, computed through 30 experiments with different random initialization of tSNE.

Table 3.3: Seriation results on real Hi-C data from the *Plasmodium knowlesi* genome

	SC+SO	SC+mdso	mdso	tSNE-mdso
# Chr.	14	14	8	16.7 ± 0.5
Purity	76.3	76.3	-	-
Cluster dist.	3.81	3.81	6.61	**2.83 ± 0.23**
$\tilde{\tau}$ (%)	77.3	69.6	18.9	**85.0 ± 11.7**
Time (s)	0.30	1.45	7.76	43.9 ± 2.8



(a) tSNE embedding

(b) sub-orderings

Figure 3.10: t-SNE embedding (3.10a), and resulting sub-orderings found with mdso (3.10b) on the *Plasmodium knowlesi* Hi-C data, in an experiment leading to a weighted Kendall-Tau score of 91.8%. The line ticks in (3.10b) delimitate the chromosomes.

Results on real data with no reference clustering

Here, we present results on the Hi-C data from *Spodoptera frugiperda* where the reference genome used was a fragmented assembly from Pacbio reads, with some mis-assemblies. The similarity matrices used here may contain fragments to separate (like chromosomes), but we ignore their locations and their number. Hence, we cannot provide the number of cluster to spectral clustering, and we cannot assess the quality of the clustering found by mdso and tSNE-mdso. Therefore, in Table 3.4, we only give the number of clusters and the weighted kendall-tau scores for mdso and t-SNE mdso, and we test the Spectral method without prior clustering. The results for tSNE-mdso are also averaged over 30 experiments, as with the *Plasmodium knowlesi* data. In Appendix Figure B.10, we show the pre-processed similarity matrices (with a logarithmic colormap for ease of reading), and the resulting orderings obtained with mdso.

3.6.5 Finding circular orderings with single-cell Hi-C data

Single-cell Hi-C data allows to capture part of the 3D architecture of DNA in the nucleus of individual cells. The similarity matrices used here are not Hi-C frequency matrices as in

Table 3.4: Seriation results on real Hi-C data from a *Spodoptera frugiperda* genome.

		Spectral	mdso	tSNE-mdso
Sf200	# Chr.	1	2	4.87 ± 0.7
	$\tilde{\tau}$ (%)	**92.7**	91.6	86.4 ± 3.0
	Time (s)	0.033	0.14	3.06 ± 0.12
Sf669	# Chr.	1	4	5.9 ± 0.5
	$\tilde{\tau}$ (%)	75.7	**88.2**	87.8 ± 2.3
	Time (s)	0.049	0.19	4.64 ± 0.07
Sf846	# Chr.	1	6	12.2 ± 0.45
	$\tilde{\tau}$ (%)	**95.8**	86.7	86.8 ± 0.6
	Time (s)	0.063	0.22	8.03 ± 0.66

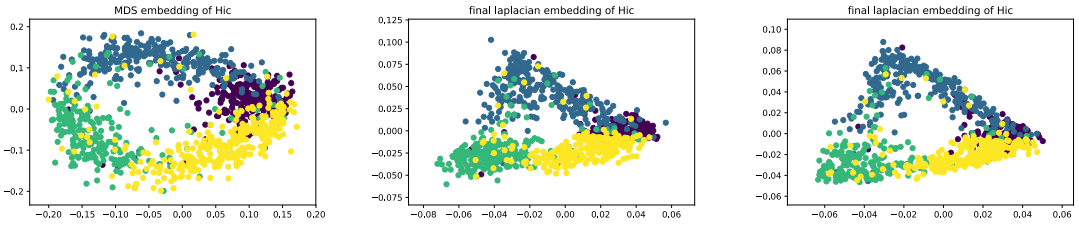
the previous subsection. Rather, given a set of n individual cells, Hi-C frequency interaction profiles were derived for each cell (such as the frequency matrices appearing above), and a pairwise similarity between cells was computed, based on the similarity between their frequency interaction profiles.

Liu et al. [2017] used it in order to cluster each cells according to four possible cell-cycle phases (G1, E-S, M-S or L-S/G2). Specifically, they applied classical MDS to HiCRep [Yang et al., 2017b] data, approximately embedding it onto a circle. They introduced a circular-ROC (CROC) measure to assess the ability of the embedding to distinguish between the four phases. This circle-like embedding reflects a latent ordering on the data, each capturing a cell architecture at a given stage of the cell life. Although we are ultimately interested in a clustering task, it can benefit from an embedding enhancing the latent ordering of the data, as Algorithm 3.3 produces.

Table 3.5: Comparison of CROC scores between MDS embedding, basic spectral embedding (Spec) and the pre-processing of our Multi-dimensional Spectral ordering method (Mdso) depending on the neighborhood parameter k. The score are only slighty better.

CROC	**G1**	**E-S**	**M-S**	**L-S/G2**	**avg**
MDS	0.938	0.966	0.917	**0.917**	0.936
Spec	0.932	0.951	0.922	0.886	0.923
Mdso$_5$	**0.943**	0.964	0.927	0.914	**0.937**
Mdso$_{10}$	0.943	**0.967**	0.921	0.905	0.934

Table 3.5 shows a comparison of the CROC score according to the embedding use. MDS and Spec performs similarly. Both embeddings (MDSO) resulting from Algorithm 3.3 lead to a better score with respect to the the baseline spectral Algorithm 3.2. Nevertheless there is no outstanding benefit in using the processing of Algorithm 3.3 instead of a simple MDS on that particular type of data. Figure 3.11 illustrates the MDS and spectral-Laplacian embeddings



(a) Linear KMS (b) Circular KMS (c) Circular KMS

Figure 3.11: Various embedding methods of HicRep data. Figure 3.11a is the first two dimension of Multi-dimensional Embedding while Figures 3.11b and 3.11c are the first two dimensions of the embedding resulting from Algorithm 3.3 for $k = 10$ and $k = 20$ respectively.

used in these methods.

3.7 Conclusion

In this chapter, we bring together results that shed light on the filamentary structure of the Laplacian embedding of serial data. It allows for tackling Linear Seriation and Circular Seriation in a unifying framework. Notably, we provide theoretical guarantees for Circular Seriation analog to those existing for Linear Seriation. These do not make assumptions about the underlying generation of the data matrix, and can be verified *a posteriori* by the practitioner. Then, we propose a simple method to leverage the filamentary structure of the embedding. It can be seen as a pre-processing of the similarity matrix. Although the complexity is comparable to the baseline methods, experiments on synthetic and real data indicate that this pre-processing substantially improves robustness to noise.

From a genome assembly application standpoint, generalizing the spectral Algorithm 3.1 to Circular Seriation provides a sounder model for laying out circular bacterial genomes. However, in practice, the repeat-induced overlaps also thwart Algorithm 3.2. Still, letting additional degrees of freedom in the d-**LE** enables the serial structure of the data to stand out although the repeats constrain some elements to remain close to each other in the embedding. The algorithm we propose leads to single-contig layouts for bacterial genomes.

Chapter 4

Robust Seriation

The work presented so far in this manuscript relies mainly on the spectral relaxation of the 2-SUM problem, introduced in Chapter 1. In Chapter 2, we use the existing method *as is* in the context of *de novo* genome assembly. In Chapter 3, we propose an enhancement of the spectral method, leading to a more efficient approach to overcome the presence of repeat-induced noise in the similarity matrices. Here, we explore another strategy, following the convex relaxations approaches to permutation problems proposed by, *e.g.*, Vogelstein et al. [2011], Fogel et al. [2013], Lim and Wright [2014, 2016], Evangelopoulos et al. [2017a]. We aim to model the similarity matrices arising in *de novo* assembly, and design algorithmic schemes that are robust to the specific, repeat-induced noise.

The content of this chapter is based on the following publication,
Antoine Recanati, Nicolas Servant, Jean-Philippe Vert, and Alexandre d'Aspremont. Robust seriation and applications to cancer genomics. *arXiv preprint arXiv:1806.00664*, 2018b

Supplementary material for this chapter is given in Appendix Chapter C.

Chapter Abstract

The seriation problem seeks to reorder a set of elements given pairwise similarity information, so that elements with higher similarity are closer in the resulting sequence. When a global ordering consistent with the similarity information exists, an exact spectral solution recovers it in the noiseless case and seriation is equivalent to the combinatorial 2-SUM problem over permutations, for which several relaxations have been derived. However, in applications such as DNA assembly, similarity values are often heavily corrupted, and the solution of 2-SUM may no longer yield an approximate serial structure on the elements. We introduce the robust seriation problem and show that it is equivalent to a modified 2-SUM problem for a class of similarity matrices modeling those observed in DNA assembly. We explore several relaxations of this modified 2-SUM problem and compare them empirically on both synthetic matrices and real DNA data.

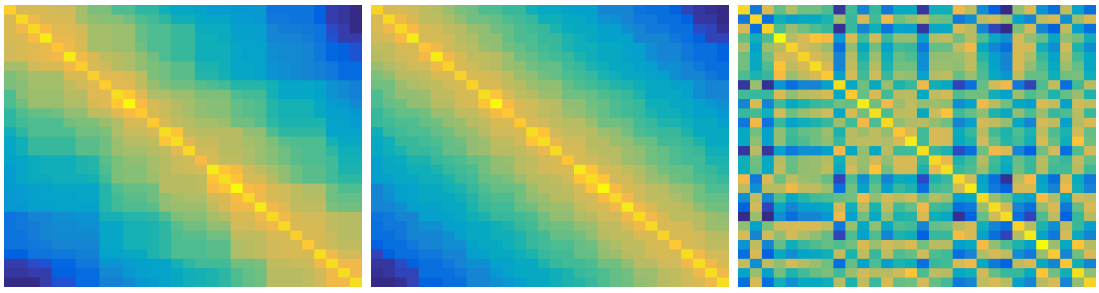
Contents

4.1 Introduction

In the seriation problem, we are given a similarity matrix between a set of n elements, which we assume to have a serial structure, *i.e.*, which can be ordered along a chain where the similarity between elements decreases with their distance within this chain. Among the applications of seriation, ranging from fields such as archeology [Robinson, 1951], to bioinformatics [Atkins and Middendorf, 1996, Cheema et al., 2010, Jones et al., 2012], the one of interest throughout this manuscript is genome assembly. We have introduced the Robinson structural hypothesis on similarity matrices underpinning Seriation in Chapters 1 and 3, in Definition 1.1.1. Here, we consider a stronger assumption, introduced below.

Definition 4.1.1. *We say that $A \in \mathbf{S}_n$ is a strong-R-matrix (or strong Robinson matrix) iff it is symmetric and satisfies $A_{ij} \leq A_{kl}$ for all (i, j, k, l) such that $|i - j| > |k - l|$.*

Here, $\mathbf{S}_n$ denotes the set of real symmetric matrices of dimension n. Definition 4.1.1 is more restrictive than the usual R-matrix property from Definition 1.1.1 (repeated in Definition 3.1.1), and used in Atkins et al. [1998], Fogel et al. [2013], which only requires the entries of the matrix to decrease when moving away from the diagonal *on a given row or column*. For strong-R matrices, we impose that the entries on a given diagonal are no greater than any entry located on the previous diagonals (see Figure 4.1).



(a) R-matrix (b) strong R-matrix (c) permuted strong-R

Figure 4.1: A R-matrix (4.1a) and its projection on the set of strong-R matrices (4.1b). A pre-strong-R matrix (4.1c) is a strong-R matrix up to a permutation of the rows and columns. If a matrix is pre-strong-R (4.1c), Seriation aims to find the permutation that makes it strong-R (4.1b).

In what follows, we write $\mathcal{R}_n^*$ the set of strong-R-matrices of size n, and $\mathcal{P}_n$ the set of permutations of n elements. A permutation can be represented by a vector π (lower case) or a matrix $\Pi \in \{0, 1\}^{n \times n}$ (upper case) defined by $\Pi_{ij} = 1$ iff $\pi(i) = j$, and $\pi = \Pi \mathbf{g}$ where $\mathbf{g} = (1, \ldots, n)^T$. We refer to both representations by $\mathcal{P}_n$ and may omit the subscript n whenever the dimension is clear from the context. We say that $A \in \mathbf{S}_n$ is pre-$\mathcal{R}^*$ if there exists a permutation $\Pi \in \mathcal{P}$ such that the matrix $\Pi A \Pi^T$ (whose entry (i, j) is $A_{\pi(i),\pi(j)}$) is a strong-R-

matrix, and the seriation problem seeks to recover this permutation Π, *i.e.*, solve

$$\begin{array}{ll} \text{find} & \Pi \in \mathcal{P} \\ \text{such that} & \Pi A \Pi^T \in \mathcal{R}^* \end{array} \tag{Seriation}$$

in the variable $\Pi \in \mathcal{P}$. This is illustrated in Figure 4.1. Given $A \in \mathbf{S}_n$, 2-SUM is an combinatorial problem over permutations, written

$$\begin{array}{ll} \text{minimize} & \sum_{i,j=1}^{n} A_{ij} |\pi_i - \pi_j|^2 \\ \text{such that} & \pi \in \mathcal{P}_n \end{array} \tag{2-SUM}$$

Remark that the search space $\mathcal{P}_n$ is discrete and of cardinality $n!$, thus preventing the use of exhaustive search or greedy branch and bound methods for Seriation or 2-SUM when n gets large [Hahsler et al., 2008]. Yet, for pre-$\mathcal{R}^*$ matrices, Seriation is equivalent to 2-SUM [Fogel et al., 2013], which can be solved exactly in polynomial time, using the spectral relaxation from Atkins et al. [1998], presented in Chapter 1 (Algorithm 1.1), and exploited in Chapter 2 in the context of genome assembly.

Problem 2-SUM is also a particular case of the Quadratic Assignment Problem [Koopmans and Beckmann, 1957], written

$$\min_{\pi \in \mathcal{P}_n} \sum_{i,j=1}^{n} A_{i,j} B_{\pi(i),\pi(j)} \tag{QAP(A,B))}$$

with $B_{ij} = |i-j|^2$. Laurent and Seminaroti [2015] showed that for pre-$\mathcal{R}^*$ matrices, Seriation is equivalent to QAP(A,B) when $-B \in \mathcal{R}_n^*$, i.e. when B has increasing values when moving away from the diagonal, and has constant values across a given diagonal (i.e. B is a Toeplitz matrix). This includes p-SUM problems, for $p > 0$, corresponding to $B_{ij} = |i-j|^p$. The case $p = 1$ is also known as the minimum linear arrangement problem (MLA) [George and Pothen, 1997]. For pre-$\mathcal{R}^*$ matrices, these problems are all equivalent and can be solved by the spectral algorithm of Atkins et al. [1998], described in Algorithm 1.1. However, when A is not pre-$\mathcal{R}^*$, the Seriation problem has multiple local solutions, and the spectral algorithm does not necessarily find a global optimum for 2-SUM, p-SUM or QAP(A,B) with B a Toeplitz, negated R matrix. In fact, these problems are NP-hard in general [Sahni and Gonzalez, 1976].

More recently, several relaxations have been proposed to tackle 2-SUM and QAP(A,B), although there are no approximation bounds in the general case [Lyzinski et al., 2016]. Vogelstein et al. [2011] used the Frank-Wolfe algorithm to minimize the objective of QAP(A,B) over the convex hull of the permutation matrices, namely the Birkhoff polytope $\mathcal{B}$. Fogel et al. [2013] presented a convex relaxation of 2-SUM in $\mathcal{B}$, and used a quadratic programming approach where the variable's membership to $\mathcal{B}$ is enforced through linear constraints (instead of the implicit projection of the Frank-Wolfe algorithm). Lim and Wright [2014] proposed a similar

relaxation in the convex hull of the set of permutation vectors, the Permutahedron $\mathcal{PH}_n$, represented with $\Theta(n \log n)$ variables and constraints, instead of $\Theta(n^2)$ for permutation matrices, thanks to an extended formulation by Goemans [2014]. All these relaxations for 2-SUM suffer from a symmetry problem, because flipping permutations leaves the objective unchanged, and the minimum of 2-SUM is achieved for a vector proportional to $\mathbf{1} = (1, \ldots, 1)^T$, which lies in the center of the convex hull of permutation vectors. To overcome this issue, constraints can be added to the problem, corresponding to either *a priori* kwowledge, or to pure "tie-breaking", *e.g.*, $\pi_1 + 1 \leq \pi_n$, ensuring that the center is excluded from the constraint set, thus breaking symmetry without loss of generality. Lim and Wright [2014] stated that a Frank-Wolfe algorithm could also be used for 2-SUM in $\mathcal{PH}$ if no other constraint but the tie-breaking was enforced, thanks to a specific linear minimization oracle, thus implicitly enforcing membership to $\mathcal{PH}$ without imposing the constraints from Goemans [2014]. Lim and Wright [2016] generalized the use of the representation of Goemans [2014] for $\mathcal{PH}$ to tackle QAP(A,B), with a coordinate descent algorithm and a continuation scheme to move away from the center of the convex hull of permutations. Evangelopoulos et al. [2017a] proposed a Frank-Wolfe algorithm in $\mathcal{PH}$ with a continuation scheme (instead of a tie-breaking constraint) to tackle 2-SUM and avoid the center. They also discussed problems of the form (QAP(A,B)) where $B_{ij} = \text{Pseudo-Huber}(|i-j|)$ [Evangelopoulos et al., 2017b], which helps in solving robust seriation as we will see below.

In Section 4.2, we introduce the robust seriation problem, motivated by applications to genome assembly. We show that for DNA data obeying a simple model which takes repeats into account, robust seriation is equivalent to Robust 2-SUM, which is a QAP problem similar to 2-SUM, where the squared distance to the diagonal that appears in the loss function is truncated. This truncated quadratic can be relaxed as a Huber loss. We present experiments to compare existing and new algorithmic approaches to solve this problem on two datasets: synthetic data following our simple model, and real data from an *E. coli* genome sequenced with third generation sequencing tools.

4.2 Robust Seriation

Classical Seriation is written as a feasibility problem: find the permutation that reorders the input matrix into an Robinson matrix. When A is pre-$\mathcal{R}^*$, solving 2-SUM yields this permutation. However, when A is not pre-$\mathcal{R}^*$, the matrix A reordered using the permutation that minimizes 2-SUM may be far from being R. Robust seriation seeks to find the closest pre-$\mathcal{R}^*$ matrix to A and reorder it, solving instead

$$\begin{array}{ll} \text{minimize} & \|S - \Pi A \Pi^T\| \\ \text{such that} & \Pi \in \mathcal{P}, \quad S \in \mathcal{R}^*. \end{array} \qquad \text{(Robust Seriation)}$$

where the variable $\Pi \in \mathcal{P}$ is a permutation matrix, the variable $S \in \mathcal{R}^*$ is a strong-R-matrix, and the norm is typically either the l_1 norm on components or the Froebenius norm.

4.2.1 Application of Seriation to Genome Assembly

De novo genome assembly has been presented in the Introduction, Section 1.3.1, and repeatedly throughout the chapters of this manuscript. As a reminder, it aims to reconstruct a DNA strand from fragments (reads) randomly sampled throughout the genome (and whose position on the genome is unknown). A common method is to compute the overlaps between all pairs of read, providing a similarity matrix A, whose entry (i, j) measures how much reads i and j overlap (and is zero if they do not). Then, we can determine the layout from the overlap information, that is to say find an ordering and positioning of the reads that is consistent with the overlap constraints.

In the true ordering (corresponding to the sorted reads' positions along the genome), a given read overlaps much with the next one, slightly less with the one after it, and so on, until a point where it has no overlap with the reads that are further away. This makes the read similarity matrix Robinson and roughly band-diagonal (with non-zero values confined to a diagonal band). Finding the layout of the reads therefore fits the Seriation framework. In practice, however, there are repeated sequences (*repeats*) along the genome that induce false positives in the overlap detection tool [Pop, 2004], resulting in non-zero similarity values outside (and possibly far away) from the diagonal band. The similarity matrix ordered with the ground truth is then the sum of a Robinson band matrix and a sparse "noise" matrix, as displayed in Figure 4.2a, which is a subset of the matrix shown in Figure 1.7a.

Repeats longer than the overlap length are perhaps the most fundamental issue in genome assembly as they lead to ambiguous reconstructions. For instance, recall the sequence RARBR, where A,B and R are sub-sequences, and R is repeated three times, illustrated in Figure 1.6 from Chapter 1. The overlap constraints arising from this sequence are identical to those of RBRAR, therefore the overlap constraints are not sufficient to uniquely determine the layout. Recently, long-reads sequencers such as PacBio's SMRT and Oxford Nanopore Technology (ONT) spurred a renaissance in assembly by enabling sequencing reads over 10kbp (kilo base-pairs) long, resolving many small repeats [Koren and Phillippy, 2015]. However, their error rate is high ($\sim 10\%$). Thus, many assemblers include a correction module in a preprocessing step, which can help in separating repeats when the repeated copies slightly differ [Pop, 2004]. They also use statistical models on the data generation in order to filter out the overlaps that are likely to be repeat-induced, and retrospectively inspect the overlap graph for potential errors in a greedy fashion, until the graph is "cleaned" and contains as few ambiguities for reconstruction as the model allows for [Koren et al., 2017, Li, 2016]. When there are ambiguities, the ambiguous reads are removed and the resulting assembly is fragmented.

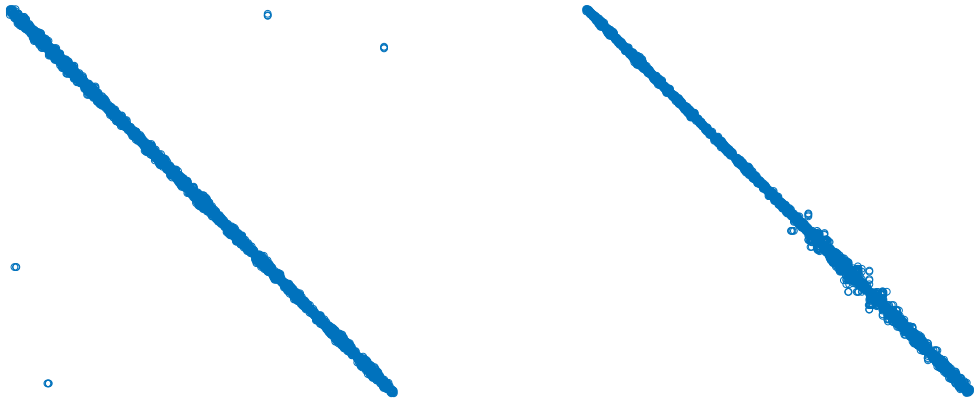
In contrast, the approach presented in Chapter 2 is simpler and more principled. Yet, the

presence of repeats often corrupts the ordering, as we illustrate in Figure 4.2, and previously in Figure 1.7b. To overcome this issue, the threshold-based method also ends up removing overlaps from the graph. Although it does not explicitly winnow out the repeats with a dedicated module, it eventually yields fragmented assemblies.

Here, we seek to apply Robust Seriation to genome assembly, dealing with the repeats in a principled manner. To this end, let us introduce stylized matrices modeling overlap-based similarity matrices arising in genome assembly. We write $\mathcal{M}_n(\delta, s)$ the set of matrices in $\{0,1\}^{n\times n}$ that are the sum of a band matrix of bandwidth δ and a sparse out-of-band matrix with s non-zero elements,

Definition 4.2.1. *$A \in \{0,1\}^{n\times n}$ belongs to $\mathcal{M}_n(\delta, s)$ iff it is symmetric and satisfies $A_{ij} = 1$ for all (i,j) such that $|i-j| \leq \delta$, and $nnz(A) = \left(n + (2n-1)\delta - \delta^2\right) + s$.*

Here $\mathrm{nnz}(A)$ is the number of non-zero elements of A, and the first term in the sum is the total number of elements in the bands. This means in particular $s \leq n^2 - \left(n + (2n-1)\delta - \delta^2\right)$ (the total number of non-zeros cannot exceed n^2). In this setting, we wish to find an ordering in which most pairs of similar elements are nearby. The 2-SUM objective can perform poorly here, since it strongly penalizes orderings with non-zero values far away from the diagonal, even when there is a small number of them, as we can see in Figure 4.2. Reducing this penalty on outliers is the goal of the robust seriation methods detailed below.



(a) Ground truth (b) 2SUM

Figure 4.2: Similarity matrix from a subset of Oxford Nanopore reads of *E. coli* in the ordering given by the ground truth position of the reads along the genome (4.2a, left), and the same matrix reordered by minimizing the 2SUM objective (4.2b, right), which pushes the out-of-diagonals terms close to the main diagonal and yields a corrupted ordering.

4.2.2 Robust 2-SUM

Given $A \in \mathbf{S}_n$, Robust Seriation seeks to find a pre-$\mathcal{R}^*$ matrix that is as close to A as possible. Instead of searching directly for a perturbation of A that is pre-$\mathcal{R}^*$, we search for a perturbation

of A that yields a low 2-SUM score, solving

$$\begin{array}{ll} \text{minimize} & \sum_{i,j=1}^{n} S_{ij}|\pi_i - \pi_j|^2 + \lambda\|A - S\|_1 \\ \text{such that} & \pi \in \mathcal{P}, \quad S \in \mathbf{S}_+. \end{array} \tag{R2S(λ)}$$

where $\mathbf{S}_+$ is the set of symmetric matrices with non-negative entries, and we use the l_1 norm on the difference between A and S to enforce sparsity in errors. Here, λ is a parameter that controls the deviation of S from A. The sum is separable and the minimization in S is closed form. Indeed, for a given (i, j), the function $S_{ij} \to S_{ij}\Delta_{ij}^2 + \lambda|S_{ij} - A_{ij}|$ is piecewise linear, with slope $\Delta_{ij}^2 - \lambda$ for $S_{ij} < A_{ij}$, and $\Delta_{ij}^2 + \lambda$ for $S_{ij} > A_{ij}$, and is therefore minimal at $S_{ij} = A_{ij}$ if $\Delta_{ij}^2 \leq \lambda$ and $S_{ij} = 0$ otherwise (recall that S_{ij} is constrained to be non-negative). Hence, R2S(λ) is equivalent to

$$\begin{array}{ll} \text{minimize} & \sum_{i,j=1}^{n} A_{ij} \min(\lambda, |\pi_i - \pi_j|^2) \\ \text{such that} & \pi \in \mathcal{P}. \end{array} \tag{R2SUM(λ)}$$

in the variable $\pi \in \mathcal{P}$. We now show that for stylized genome assembly similarity matrices, if the number of reads spanning repeated regions is controlled, then solving R2SUM(λ) also solves Robust Seriation.

Proposition 4.2.2. *For $s \leq s_{\text{lim}} \triangleq (n - \delta - 1)$ and $A \in \mathbf{S}_n$, if A can be permuted to belong to $\mathcal{M}_n(\delta, s)$, i.e., if there is $\Pi \in \mathcal{P}_n : \Pi A \Pi^T \in \mathcal{M}_n(\delta, s)$, then Π solves both Robust Seriation and R2SUM(λ) with parameter $\lambda = \delta^2$, and the ℓ_1 norm in Robust Seriation.*

Proof. Let δ, s be two positive integers such that $\delta \leq n$, $s \leq (n - \delta - 1)$. Without loss of generality, assume that $A \in \mathcal{M}(\delta, s)$, *i.e.*, $\Pi = \mathbf{I}$, the identity permutation (otherwise, we simply factor out the true permutation). First, let us observe that for $\lambda = \delta^2$, $\mathbf{I}$ is optimal for R2SUM(λ). Indeed, since $A \in \{0, 1\}^{n \times n}$, the objective in R2SUM(λ) is the sum of $\min(\delta^2, |\pi_i - \pi_j|^2)$ over all indexes (i, j) such that $A_{ij} = 1$. This sum can be split into two terms,

$$\begin{aligned} f_{\text{in}} &= \sum_{(i,j):A_{ij}=1\,,\,|\pi_i - \pi_j| \leq \delta} |\pi_i - \pi_j|^2, \\ f_{\text{out}} &= \sum_{(i,j):A_{ij}=1\,,\,|\pi_i - \pi_j| > \delta} \delta^2. \end{aligned}$$

For $\Pi = \mathbf{I}$, the number of terms in f_{in} is maximized since $A_{ij} = 1$ for all (i, j) such that $|i - j| \leq \delta$ ($A \in \mathcal{M}(\delta, s)$). The sum of the number of terms in f_{in} and f_{out} is equal to $\text{nnz}(A)$ and is invariant by permutation (therefore, the number of terms in f_{out} is also minimized for $\Pi = \mathbf{I}$) Since any term in f_{in} is smaller than any term in f_{out}, $\Pi = \mathbf{I}$ is optimal for R2SUM(λ) with $\lambda = \delta^2$.

Now, let us see that $\Pi = \mathbf{I}$ is also optimal for Robust Seriation. Given Π, optimizing over S in Robust Seriation yields $S_\Pi = \text{Proj}_{\mathcal{R}^*}(\Pi A \Pi^T)$, the projection of $\Pi A \Pi^T$ onto the set of

strong-R-matrices. Let us assume that we use the ℓ_1 norm in (Robust Seriation). Then, S_Π, the projection in ℓ_1 norm of the binary matrix $\Pi A \Pi^T$, is also binary, as we prove further in Lemma 4.2.3. A sparse, $\{0,1\}$ strong-R-matrix is necessarily of the form

$$\begin{cases} S_{ij} = 1 & \text{if } |i-j| \leq k, \\ S_{ij} = 0 & \text{if } |i-j| > k+1, \\ S_{ij} \in \{0,1\} & \text{for } |i-j| = k+1, \end{cases}$$

with the integer $k+1$ denoting the bandwidth of S. Given S_Π and the corresponding k, the distance between $\Pi A \Pi^T$ and S_Π appearing in Robust Seriation is separable (whether we use the l_1 or Frobenius norm, since $A \in \{0,1\}^{n \times n}$) and can be grouped into three terms, according to whether (i,j) is such that $|i-j| > k+1$, $|i-j| \leq k$ or $|i-j| = k+1$. The first term, $n_{\text{out}}(k) \geq 0$, equals the number of non-zero elements of $\Pi A \Pi^T$ such that $|i-j| > k+1$. The second, $n_{\text{in}}(k) \geq 0$, equals the number of zero elements of $\Pi A \Pi^T$ such that $|i-j| \leq k$. The third equals zero, because setting the (k+1)-th diagonal of S identical to the (k+1)-th diagonal of $\Pi A \Pi^T$ does not violate the R property of S_Π, and S_Π is by definition the strong-R-matrix that minimizes the distance to $\Pi A \Pi^T$. For any Π, if $k > \delta$, the number of non-zeros elements inside the band of width k being bounded by the number of non-zero elements of A, we have $n_{\text{in}}(k) \geq 2(n-\delta-1) - s \geq (n-\delta-1) \geq s$. Similarly, for $k \leq \delta$, $n_{\text{out}}(k) \geq s$. For $\Pi = \mathbf{I}$, as long as $k \leq \delta$, $n_{\text{out}}(k) \leq s$ decreases with k and $n_{\text{in}}(k) = 0$. For $k = \delta$, $n_{\text{in}}(k) = 0$ and $n_{\text{out}}(k) \leq s$ (it is equal to s minus the number of elements in the $\delta+1$-th diagonal). Thus, $\Pi = \mathbf{I}$ is optimal, and $k = \delta$. ■

Note that in practice, one has to chose the parameter λ without observing δ before trying to solve R2SUM(λ). Yet, for matrices A satisfying the hypothesis of Proposition 4.2.2, the number of non-zero values of A (which is observed even when A is permuted) provides a way to estimate δ. We compute it as the smallest integer δ such that the number of non-zero elements in a band matrix of size δ is larger than $\text{nnz}(A)$. Also remark that the proof of Proposition 4.2.2 is conservative: it only involves reasoning about the location of non-zero values of a vectorized version of $\Pi A \Pi^T$. Permuting rows and columns of a matrix adds constraints on the locations of these non-zero values that we did not take into account.

Lemma 4.2.3. *Given a binary symmetric matrix $S \in \{0,1\}^{n \times n}$, it has a binary projection in ℓ_1 norm onto the set of strong-R-matrices, that is to say, there exists a solution $R \in \{0,1\}^{n \times n}$ to the following problem,*

$$\begin{array}{ll} \textit{minimize} & \sum_{i,j=1}^{n} |R_{ij} - S_{ij}| \\ \textit{such that} & R \in \mathcal{L}_R. \end{array} \tag{R-proj}$$

Proof. Consider a given diagonal $0 \leq k \leq n-1$ in the lower triangle. The strong-R constraints are lower and upper bounds on the values of R_{ij} on the k-th diagonal. Let

$m_k \triangleq \min_{i,j\,:\,|i-j|=k} R_{ij}$, and $M_k \triangleq \max_{i,j\,:\,|i-j|=k} R_{ij}$. Recall that S has only ones and zeros on the k-th diagonal. From R-proj, R_{ij} has values in $[0,1]$. Clearly, a solution of R-proj satisfies,

$$R_{ij} = \begin{cases} M_{|i-j|}, & \text{if } S_{ij} = 1 \\ m_{|i-j|}, & \text{if } S_{ij} = 0. \end{cases}$$

Let $0 \leq p_k \leq n-k$ denote the number of ones on the k-th diagonal of S, and $0 \leq z_k = n-k-p_k$ the number of zeros on the k-th diagonal of S. Summing over all the diagonals of the matrix, the objective in R-proj can be written as,

$$\|S - R\|_1 = \; p_0\,(1 - M_0) + z_0\,(m_0 - 0) + 2\textstyle\sum_{k=1}^{n-1} p_k\,(1 - M_k) + z_k\,(m_k - 0) \tag{4.1}$$

where we have separated the main diagonal from the others that are coupled with their symmetric. Now, we have that $0 \leq m_k \leq M_k \leq 1$ for all $0 \leq k \leq n-1$. The strong-R constraints also require that $M_k \leq m_{k-1}$ for $1 \leq k \leq n-1$. The minimizer of R-proj saturates these constraints ($M_k = m_{k-1}$), and equation (4.1) can finally be written as,

$$\begin{aligned} \|S - R\|_1 \; &= p_0\,(1 - M_0) + z_0 m_0 + 2\textstyle\sum_{k=1}^{n-1} p_k\,(1 - m_{k-1}) + z_k m_k \\ &= p_0\,(1 - M_0) + (z_0 - 2p_1)\,m_0 + 2\textstyle\sum_{k=1}^{n-1} (z_k - p_{k+1})\,m_k + \sum_{k=1}^{n-1} p_k. \end{aligned}$$

where by convention $p_n \triangleq 0$. R-proj seeks to minimize this objective on the variables $(M_0, m_0, m_1, \ldots, m_{n-1})$, under the constraints $1 \geq M_0 \geq m_0$, $m_{k-1} \geq m_k$ for $1 \leq k \leq n-1$, and $m_k \geq 0$ for $0 \leq k \leq n-1$. All in all, we can rewrite R-proj as a linear program over the variable $x = (M_0, m_0, m_1, \ldots, m_{n-1}) \in \mathbb{R}^{n+1}$,

$$\begin{aligned} &\text{minimize} \quad c^T x \\ &\text{such that} \quad Ax \leq b\,, \;\; x \geq 0. \end{aligned}$$

where

$$A = \begin{pmatrix} -1 & & & & & \\ 1 & -1 & & & & \\ & 1 & -1 & & & \\ & & \ddots & \ddots & & \\ & & & 1 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad c = \begin{pmatrix} -p0 \\ (z_0 - 2p_1) \\ 2(z_1 - p_2) \\ \vdots \\ 2(z_{n-1} - p_n) \end{pmatrix}.$$

Now, observe that $b \in \mathbb{R}^{n+1}$ has integer entries, and that $A \in \mathbb{R}^{(n+1)\times(n+1)}$ is totally unimodular. It follows that it has an integral solution x^* [Papadimitriou and Steiglitz, 1998][Th. 13.3]. From the previous considerations, the corresponding matrix $R \in \mathcal{L}_R$ has entries in $\{0,1\}$. ∎

4.3 Robust Seriation Algorithms

We compare several methods to address the R2SUM(λ) problem. First, observe that the objective of R2SUM(λ) is not convex. In order to use convex optimization algorithms, it can be relaxed to its convex envelope, resulting in the following problem,

$$\begin{array}{ll} \text{minimize} & \sum_{i,j=1}^{n} A_{ij} h_\delta(|\pi_i - \pi_j|) \\ \text{such that} & \pi \in \mathcal{P}. \end{array} \qquad \text{(HuberSUM}(\delta)\text{)}$$

where $h_\delta(x)$ is the Huber function, which equals x^2 when $|x| \leq \delta$, and $\delta(2|x| - \delta)$ otherwise. In Figure 4.3 are shown plots of the square (ℓ_2), absolute value (ℓ_1), Huber, and truncated square loss functions appearing in (2-SUM), (1-SUM), (HuberSUM(δ)), and (R2SUM(λ)).

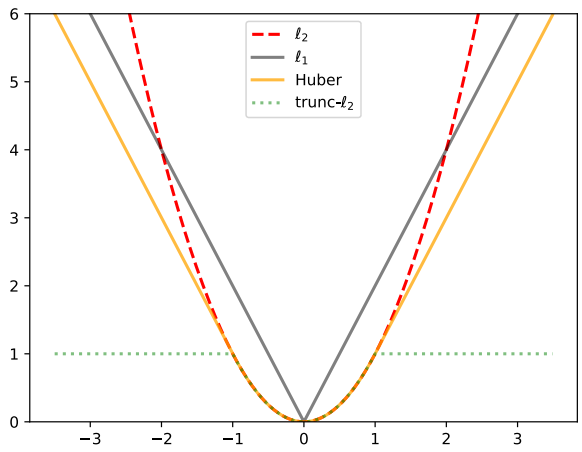


Figure 4.3: Plot of the square (ℓ_2), absolute value (ℓ_1), Huber, and truncated square losses, appearing respectively in the (2-SUM), (1-SUM), (HuberSUM(δ)), and (R2SUM(λ)) problems.

4.3.1 QAP solvers (FAQ and PHCD)

The first strategy is to directly minimize the objective of R2SUM(λ) using QAP solvers. Indeed, the problem matches QAP(A,B) with $B_{ij} = \min(\lambda, |i-j|^2)$. We test the aforementioned Vogelstein et al. [2011] and Lim and Wright [2016] methods for solving the QAP.

The first, which we refer to as FAQ [Vogelstein et al., 2011], uses the matrix representation of permutations with a relaxation in the convex hull of permutation matrices, $\mathcal{B}$, where the QAP(A,B) objective is optimized with the conditional gradient (a.k.a. Frank-Wolfe) algorithm, described in the Introduction Chapter, Algorithm 1.2, and repeated here in 4.1. Each step of Frank-Wolfe in $\mathcal{B}$ (4.2) involves an assignment problem solved with a Hungarian algorithm [Kuhn, 1955].

Algorithm 4.1 Conditional gradient algorithm for permutation problem $\min_{x\in\text{hull}(\mathcal{P})} f(x)$. ($\text{hull}(\mathcal{P})$ can be either $\mathcal{B}$ or $\mathcal{PH}$).

Inputs: Initial point $x_0 \in \text{hull}(\mathcal{P})$, target precision ε

for $t = 0, \ldots$ **do**

Solve linear minimization oracle

$$s_t = \underset{s\in\mathcal{P}}{\text{argmin}} \langle \nabla f(x_t), s \rangle \tag{4.2}$$

Get estimated gap

$$\Delta_t = \langle x_t - s_t, \nabla f(x_t) \rangle \tag{4.3}$$

if $\Delta_t \leq \varepsilon$ **then** Stop **end if**

Set

$$x_{t+1} = x_t + \frac{2}{t+2}(s_t - x_t)$$

end for

Output: $\hat{x} = x_t$

The latter, denoted PHCD [Lim and Wright, 2016] in the following, uses the sorting-network based representation of permutation vectors of Goemans [2014] and performs coordinate descent in the convex hull of permutation vectors $\mathcal{PH}$.

For completeness, we also used these QAP solvers in the experiments to solve 2-SUM (i.e. QAP(A,B) with $B_{ij} = |i-j|^2$), and HuberSUM(δ) ($B_{ij} = h_\delta(|i-j|)$).

4.3.2 Symmetry issue in the Permutahedron $\mathcal{PH}$

A typical convex relaxation work-flow involves relaxing both the objective function to its convex envelope, and relaxing the constrained set to its convex hull, in order to use of the arsenal of convex optimization, including scalable first order methods. Here, we seek to optimize the objective functions of 2-SUM and HuberSUM(δ), f_{2SUM} and f_{Huber}, on the convex hull of the set of permutation vectors $\mathcal{P}_n$, the polyhedron $\mathcal{PH}_n$.

Unfortunately, the solution of a relaxation $\tilde{x} \in \mathcal{PH}_n$ does not necessarily (and most of the time, not) lie in $\mathcal{P}_n$. To retrieve a solution in $\mathcal{P}_n$, one must project the relaxed solution $\tilde{x}$ onto the set of permutations $\mathcal{P}_n$, which may be challenging. Here, the flat vector $\mathbf{c}_n \triangleq \frac{n+1}{2}\mathbf{1}_n \in \mathcal{PH}_n$ minimizes f_{2SUM} and f_{Huber} in $\mathcal{PH}_n$. Indeed, all its entries being equal, $f_{\text{2SUM}}(\mathbf{c}_n) = f_{\text{Huber}}(\mathbf{c}_n) = 0$, which is optimal since these sums involve only non-negative terms. Yet, this optimum is non-informative. Any permutation $\pi \in \mathcal{P}_n$ has the same distance to $\mathbf{c}_n$, $d = \sum_{i=1}^n (\frac{n+1}{2} - i)^2$, thus projecting back $\mathbf{c}_n$ to $\mathcal{P}_n$ is completely degenerate.

This is illustrated in Figure 4.4, where $\mathcal{PH}_3$ is a salmon-colored hexagone centered around $\mathbf{c}_3$ (red circled dot), and whose vertices are the permutations. $\mathcal{PH}_3$ is represented on a planar figure

since $\mathcal{PH}_n$ lies in a hyperplane of dimension $n-1$, $\mathcal{H}_n = \{x \in \mathbb{R}^n | x^T \mathbf{1} = \frac{n(n+1)}{2}\}$. Indeed, all permutation vectors have the same set of elements, hence the same sum, and also the same norm, as one can see from the black dashed circle of fixed norm in Figure 4.4 on which all permutations lie. The symmetry of center $\mathbf{c}_n$, formally defined by $T_n(x) - \mathbf{c}_n = -(x - \mathbf{c}_n)$, is visible from the level lines of f_{2SUM} (blue ellipses). The objectives from 2-SUMand HuberSUM(δ) are invariant under the "flipping" operator T_n. For instance, the permutation $\pi = (1,3,2)^T$ and its symmetric $T_3(\pi) = (n+1)\mathbf{1} - \pi = (3,1,2)$ are on the same level line. This is the fundamental reason why the minimum of 2-SUM and HuberSUM(δ) lies in the center, making the basic convex relaxation in $\mathcal{PH}_n$ useless.

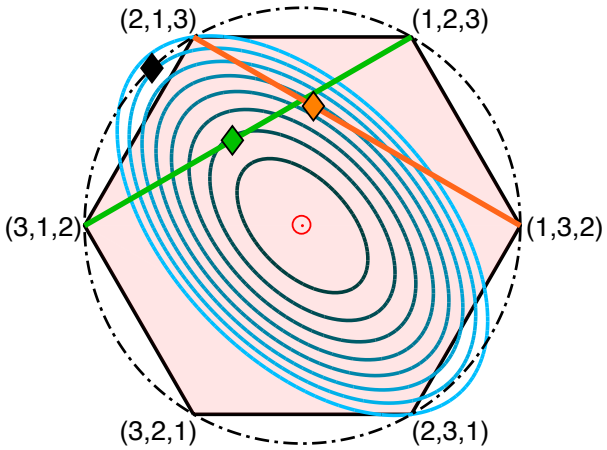


Figure 4.4: View of the 3-Permutahedron $\mathcal{PH}_3$ (filled polygon) in the 2D plane $\mathcal{H}_3$ (orthogonal to the vector $\mathbf{1}_3$ represented by the red pointing arrow (circled dot)). The blue ellipses are the level curves of f_{2SUM}. The black dashed circle represents the set of points having the same norm as the permutation vectors, and the black diamond is the minimizer of 2-SUM among them. The green (resp. orange) line is where the "good" (resp. "bad") tie-breaking constraint $\pi_2 + 1 \leq \pi_3$ (resp. $\pi_1 + 1 \leq \pi_3$) is active, and the green (resp. orange) diamond is the minimizer of f_{2SUM} on the corresponding constrained set, the triangle $\big((2,1,3),(1,2,3),(1,3,2)\big)$ [resp. $\big((3,1,2),(2,1,3),(1,2,3)\big)$]. The closest permutation to the green diamond is $(2,1,3)$, which is the correct solution (minimizer of f_{2SUM} on $\mathcal{P}_3$), but the orange diamond is closer to $(1,2,3)$ because of the anisotropy induced by the tie-breaking constraint. Figure adapted from Lim and Wright [2014].

To overcome this issue, Fogel et al. [2013], Lim and Wright [2014] employ two strategies. One is to add a penalty in the objective that increases towards to the center $\mathbf{c}$, *e.g.*, add the concave penalty $-\mu\|x - \mathbf{c}\|^2$ to the objective. The other one is to add constraints that keep the center $\mathbf{c}$ out of the feasible set, *e.g.*, add the tie-breaking constraint $\pi_1 + 1 \leq \pi_n$. This resolves the ambiguity about the direction of the ordering without removing any permutation from the search space (up to a flip), since, for any permutation $\pi \in \mathcal{P}$, either π satisfies the tie-breaking constraint, or its symmetric $T(\pi)$ does. On Figure 4.4, the tie-breaking constraint is active on the orange line, and the constrained set satisfying it is the top-right triangle of $\mathcal{PH}_3$,

$\big((2,1,3),(1,2,3),(1,3,2)\big)$. We consider methods employing both strategies in what follows.

4.3.3 Frank-Wolfe with tie-breaking constraint (FWTB)

The conditional gradient (Frank-Wolfe) Algorithm 4.1 is suited to optimization in $\mathcal{PH}_n$ since the linear minimization oracle (LMO) performed at each iteration (4.2) boils down to sorting the entries of a vector $g \in \mathbb{R}^n$ (hence, it has a computational complexity of $O(n \log n)$). Specifically, the LMO solves,

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n \pi_i g_i \\ \text{such that} & \pi \in \mathcal{PH}_n \end{array} \tag{LMO}$$

where g_i is the i-th entry of the gradient of the loss function. This linear form is minimized on a vertex of $\mathcal{PH}$, *i.e.* on a permutation π^*. Let $\sigma \in \mathcal{P}_n$ be a permutation that sorts the entries of g by decreasing order, such that $g_{\sigma_1} \geq \ldots \geq g_{\sigma_n}$, then π^* is defined by $\pi^*_{\sigma_1} = 1, \ldots, \pi^*_{\sigma_n} = n$.

The method (FWTB) adds a tie-breaking constraint (*e.g.*, $\pi_1 + 1 \leq \pi_n$) in order to break the symmetry and exclude the center c_n from the feasible set, as suggested by Lim and Wright [2014]. Yet, while Fogel et al. [2013], Lim and Wright [2014] proposed convex optimization methods that could incorporate any such linear constraint into the problem seamlessly, if one wants to use Frank-Wolfe in the restriction of $\mathcal{PH}$ where the tie-break is satisfied, the LMO has to be modified. The new LMO must solve,

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n \pi_i g_i \\ \text{such that} & \begin{cases} \pi \in \mathcal{PH}_n, \\ \pi_i + 1 \leq \pi_j. \end{cases} \end{array} \tag{LMO-tb}$$

where we let $1 \leq i \neq j \leq n$ be the tie-break indexes (in Fogel et al. [2013], Lim and Wright [2014], $i = 1$ and $j = n$). Lim and Wright [2014] propose an algorithm for solving LMO that preserves the $O(n \log n)$ complexity of the LMO. We describe in Algorithm 4.2 a slightly simplified version of theirs, for any tie-break indexes $1 \leq i \neq j \leq n$. We use the matlab-like notation $x(i)$ to denote x_i for ease of reading.

Proposition 4.3.1. *Algorithm 4.2 minimizes $g^T\pi$ over $\mathcal{PH}_n$ with tie-break $\pi(i) + 1 \leq \pi(j)$.*

Proof. Without loss of generality, let us assume for simplicity that g is already sorted by decreasing value. Let π^* be the solution of LMO. If $\pi^*_i + 1 \leq \pi^*_j$, then π^* is also solution of LMO-tb. Otherwise, the solution of LMO-tb will be a permutation π where the constraint is active : $\pi_i + 1 = \pi_j$ [Lim and Wright, 2014]. Let $k = \pi_i$. There are $n-1$ possible values for k : $\{1, \ldots, n-1\}$. For a given k, the vector $\tilde{\pi}_k$, the restriction of π to the $n-2$ indexes other than i and j is given by Smith's rule : it is the concatenation of the remaining values $\tilde{\pi}_k = (1, \ldots, k-1, k+2, \ldots, n)$ (given that g is sorted). Therefore, the permutation π optimal for LMO-tb is

Algorithm 4.2 Minimizing $g^T\pi$ over $\mathcal{PH}_n$ with tie-break $\pi(i)+1 \leq \pi(j)$.

$g', \sigma \leftarrow$ sort g in decreasing order (*i.e.*, $g(\sigma_1) \geq \ldots \geq g(\sigma_n)$)
for $k \leftarrow 1$ to $n-1$ **do**
  if $g'(k) < \frac{g(i)+g(j)}{2}$ **then**
    break
  end if
  $\sigma^{-1} \leftarrow \operatorname{argsort} \sigma$
  Set $\tilde{z} = (1, \ldots, k-1, k+2, \ldots, n)^T \in \mathbb{R}^{n-2}$
  $\pi(l) \leftarrow \tilde{z}(\sigma^{-1}(l))$ for $l \in \{1, \ldots, n\} \setminus \{i, j\}$
  $\pi(i) \leftarrow k$
  $\pi(j) \leftarrow k+1$.
end for
Output: A permutation $\pi^{(T)}$.

determined by k. Let us note $\tilde{g} \in \mathbb{R}^{n-2}$ the vector g without the two entries corresponding to indexes i and j, that is to say, if $i < j$, $\tilde{g} = (g_1, \ldots, g_{i-1}, g_{i+1}, \ldots, g_{j-1}, g_{j+1}, \ldots, n)$. To know the optimal value of k, let us observe the difference between the objective of LMO-tb for $k = K$ and $k = K+1$, with $1 \leq K \leq n-2$. For a given k, he objective in LMO-tb can be written as the sum $\tilde{g}^T\tilde{\pi}_k + kg_i + (k+1)g_j$. Let us write the tilde scalar product part first.

$$\begin{array}{rlllllll} \tilde{g}^T\tilde{\pi}_K & = 1\tilde{g}_1 + 2\tilde{g}_2 & + \ldots + & (K-1)\tilde{g}_{K-1} & + & (K+2)\tilde{g}_K & + & (K+3)\tilde{g}_{K+1} & + \ldots + n\tilde{g}_{n-2} \\ \tilde{g}^T\tilde{\pi}_{K+1} & = 1\tilde{g}_1 + 2\tilde{g}_2 & + \ldots + & (K-1)\tilde{g}_{K-1} & + & K\tilde{g}_K & + & (K+3)\tilde{g}_{K+1} & + \ldots + n\tilde{g}_{n-2} \end{array}$$

The difference between the objective values for $k = K$ and $k = K+1$ is therefore $\Delta_K = 2\tilde{g}_K - (g_i + g_j)$. Since we assumed g sorted by decreasing order, $\tilde{g}$ also is, and consequently, Δ_K decreases with K. The optimal K^* is therefore the smallest (first) index k for which $\tilde{g}_k < \frac{(g_i+g_j)}{2}$, and if $\tilde{g}_k \geq \frac{(g_i+g_j)}{2}$ for all $k \in \{1, \ldots, n-2\}$, then $K^* = n-1$. ■

4.3.4 Graduated Non-Convexity : Frank-Wolfe Algorithm with Concave Penalty (GnCR and HGnCR)

In Fogel et al. [2013], Lim and Wright [2014], the parameter μ controlling the amplitude of the penalty $-\mu \|x-c\|^2$ is bounded in order to keep the objective convex. Precisely, the objective function $f_{2\text{SUM}} = x^T L_A x$ is replaced by,

$$\tilde{f}(x) = x^T L_A x - \mu \|Px\|^2 = x^T (L_A - \mu P) x,$$

where $L_A = \mathbf{diag}(A\mathbf{1}) - A$ is the Laplacian of A and $P = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T$ projects on the subspace orthogonal to $\mathbf{1}$. To keep the problem convex, μ needs to be smaller than λ_2, the smallest non-zero eigenvalue of L_A. Still, for small values of λ_2, this may lead to solutions lying close to the center c up to numerical precision. Also, for f_{Huber} , the convexity is broken for any

positive value of μ.

Evangelopoulos et al. [2017a] proposed a graduated non-convexity scheme called GnCR to solve 2-SUM, where μ is gradually increased in outer iterations of the problem, starting with a small value ($\mu \leq \lambda_2$) preserving convexity, and moving towards high values of ($\mu \geq \lambda_{\max}$) , making the objective concave. This strategy aims at finding a sequence of solutions to the sub-problems that follow a path from near $\mathbf{c}_n$ (when the objective is convex) towards a permutation (when it is concave). To solve each subproblem, GnCR uses the Frank-Wolfe algorithm in $\mathcal{PH}_n$ without tie-breaking constraint. In Evangelopoulos et al. [2017b], the approach is extended to a pseudo-Huber loss, thus approximately solving HuberSUM(δ), with a method called HGnCR. We include both methods in the experiments.

4.3.5 Unconstrained Optimization in $\mathcal{H}_n$ with Iterative Bias (UBI)

We propose a method based on unconstrained optimization. We also add a penalty to f_{Huber} in order to avoid the center $\mathbf{c}$ and aim to minimize,

$$\tilde{f}_{\text{Huber}}(x) = f_{\text{Huber}}(x) - \mu h(x),$$

where h is a penalty function pushing away from c. In practice, we use a sigmoidal penalty,

$$h_w(x) = \Big(1 + \exp\big(-\langle x - c, w - c\rangle\big)\Big)^{-1}.$$

It breaks the symmetry by adding a bias in a given direction w. However, the penalty h becomes negligible compared to $f_{\text{Huber}}(x)$ when $\|x-c\|$ gets large, and the minimizer of $\tilde{f}_{\text{Huber}}(x)$ remains bounded. Up to a scaling of μ, it will lie in $\mathcal{PH}_n$. Hence, we can use *unconstrained* optimization to find a minimizer of $\tilde{f}_{\text{Huber}}$ in $\mathcal{PH}_n$ without enforcing the membership to $\mathcal{PH}_n$ explicitly.

Algorithm 4.3 Iterative scheme with biased unconstrained optimization in $\mathcal{H}_n$ (UBI).

Input: An objective function f, an initial bias direction $\pi^{(1)} \in \mathcal{P}_n$, an increasing bias function $h : \mathbb{R} \to \mathbb{R}$, a maximum number of outer iterations T, an optimization algorithm $\mathcal{A}$.

1: **for** $t = 1$ **to** T **do**

2: Compute

$$x_*^{(t+1)} \in \operatorname*{argmin}_{x \in \mathcal{H}_n} \left\{ f(x) - h_{\pi^{(t)}}(x) \right\} \quad \text{using algorithm } \mathcal{A}. \tag{4.4}$$

3: Set $\pi^{(t+1)} = \operatorname{argsort} x_*^{(t+1)}$

4: **end for**

Output: A permutation $\pi^{(T)}$.

We propose an iterative method where each outer iteration t solves a sub-problem biased towards the optimum $x^*_{(t-1)}$ found at the previous iteration, described in Algorithm 4.3. The

algorithm $\mathcal{A}$ used in practice in (4.4) is the LBFGS method, using the implementation from Schmidt [2005]. Figure 4.5 illustrates the iterative procedure. The colored crosses indicate the

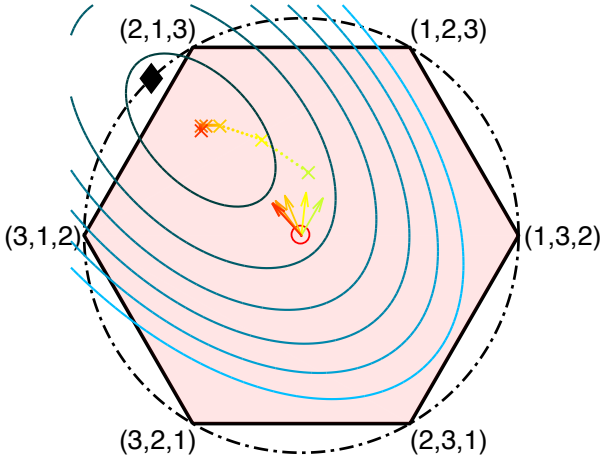


Figure 4.5: Illustration of Algorithm 4.3 in the 3-Permutahedron $\mathcal{PH}_3$ (filled polygon, same representation as in Figure 4.4). The colored crosses (from flashy yellow (right) to red (left)) represent the solutions $x_*^{(t)}$ obtained at the outer loops of the algorithm, and the associated colored arrows in the center point towards the associated bias that was used at iteration t. In blue are the level lines of $\left\{f(x) - h_{x_*^{(t)}}(x)\right\}$ with $f = f_{\text{2SUM}}$ and $t = 5$ (red arrow).

minima of the sequence of biased functions. The last one (with the level lines) is biased towards the optimum. Empirically, we found better results by using We could have used $h_{x_*^{(t)}}$ rather than $h_{\pi^{(t)}}$ in step 4.4 of Algorithm 4.3, but we empirically found better results with the latter option. Optimization of f in $\mathcal{H}_n$ is done through unconstrained optimization in $\mathbb{R}^{n-1}$ of the composition of f with an affine transformation described in the following.

We have seen in § 4.3.2 and Figure 4.4 that the set of permutation lie in a hyperplane of dimension $n-1$, $\mathcal{H}_n = \{x \in \mathbb{R}^n | x^T \mathbf{1} = \frac{n(n+1)}{2}\}$ (which simply means that all permutation vectors have the same sum). Thus, we compute a basis of $\mathcal{H}_n$ and use an affine transformation from $\mathbb{R}^{n-1}$ to $\mathcal{H}_n$ such that 0_{n-1} corresponds to $c_n = (n+1)/2\mathbf{1}_n \in \mathcal{H}_n$. In practice, we used, $U = (u^{(1)}, \ldots, u^{(n-1)}) \in \mathbb{R}^{n \times n-1}$, *e.g.*, $u^{(j)} = \frac{\tilde{u}^{(j)}}{\|\tilde{u}^{(j)}\|}$, with

$$\begin{cases} \tilde{u}_i^{(j)} = 0 & \text{if } i < j, \\ \tilde{u}_j^{(j)} = -j & \\ \tilde{u}_i^{(j)} = 1 & \text{if } i > j, \end{cases}$$

The vectors $\{u^{(j)}\}_{1 \leq j \leq n-1}$ are orthonormal and are all orthogonal to $\mathbf{1}_n$. Any point $x \in \mathcal{H}_n$ can be written as $x = \mathcal{A}(y) \triangleq Uy + c_n$ with $y \in \mathbb{R}^{n-1}$. For any $f : \mathbb{R}^n \to \mathbb{R}$, we define $f_{\mathcal{H}_n} : \mathbb{R}^{n-1} \to \mathbb{R}$ by $f_{\mathcal{H}_n}(y) = f(Uy + c_n)$ in order to perform unconstrained optimization (UBI) on $f_{\mathcal{H}_n}$. The intersection of the sphere Σ_n with $\mathcal{H}_n$, represented by the black dashed

circle in Figure 4.4, is the transformation of a sphere $\tilde{\Sigma}_{n-1}$ by $\mathcal{A}$.

4.3.6 Spectral relaxation for HuberSUM(δ)

Let us recall the spectral method introduced in Chapter 1, here in Algorithm 4.4. Roughly, it aims to minimize the 2SUM objective defined as,

$$\sum_{i,j=1}^{n} A_{ij} \left(x_i - x_j\right)^2 = x^T L_A x, \tag{2SUM}$$

over the set of permutations, by relaxing the integer constraints on permutations, leading to an eigenvalue problem. The spectral method minimizes $f_{2\text{SUM}}$ on a sphere of fixed norm by

Algorithm 4.4 Spectral ordering [Atkins et al., 1998]

Input: Connected similarity matrix $A \in \mathbb{R}^{n \times n}$
1: Compute Laplacian $L_A = \mathbf{diag}(A\mathbf{1}) - A$
2: Compute second smallest eigenvector of L_A, f_1
3: Sort the values of f_1
Output: Permutation $\sigma : f_1(\sigma(1)) \leq \ldots \leq f_1(\sigma(n))$

computing a second extremal eigenvector, thus resolving the center issue. As we have seen in Section 1.2.2, up to a translation and dilatation of this sphere, it is the sphere with fixed sum ($x^T\mathbf{1} = n(n+1)/2$) and fixed norm ($\|x\|_2^2 = n(n+1)(2n+1)/6$) on which all permutations lie. For $n = 3$, it is represented by the black dashed circle in Figure 4.4.

As we noted earlier, the spectral relaxation relies on the quadratic nature of the 2-SUM objective, and there is no general method for performing convex optimization on a sphere (we can seamlessly deal with linear equality constraints, but not quadratic constraints such as the fixed norm constraints). Optimizing HuberSUM(δ) over a sphere is therefore challenging. We propose to extend the spectral Algorithm 1.1 to HuberSUM(δ) through the variational form of the Huber loss (so-called η-trick). The absolute value of a real number $x \in \mathbb{R}$ can be expressed as the solution of an minimization problem over a real variable η,

$$2|x| = \min_{\eta \geq 0} \frac{x^2}{\eta} + \eta, \tag{4.5}$$

and the minimum is realized for $\eta^* = |x|$. Similarly, for any $\delta \geq 0$, the Huber function defined by

$$h_\delta(x) = \begin{cases} x^2 & \text{if } |x| \leq \delta, \\ \delta(2|x| - \delta) & \text{otherwise} \end{cases} \tag{4.6}$$

can be expressed, up to an affine transformation, as the minimum of the same function of η,

but on $[\delta, +\infty]$ instead of $[0, +\infty]$,

$$h_\delta(x) \simeq \min_{\eta \geq \delta} \frac{x^2}{\eta} + \eta. \tag{4.7}$$

Specifically, the equation for the Huber function as defined in (4.6) is,

$$h_\delta(x) = \min_{\eta \geq \delta} \delta \left(\frac{x^2}{\eta} + \eta - \delta \right), \tag{4.8}$$

and the minimum is realized for $\eta^* = \max(\delta, |x|)$, *i.e.*, $\eta^* = |x|$ when $|x| \geq \delta$, and $\eta^* = \delta$ otherwise. Using this variational form, we can write HuberSUM(δ) as an optimization problem over variables π and $\eta \in \mathbf{S}_n$,

$$\begin{array}{ll} \text{minimize} & \sum_{i,j=1}^n A_{ij} \left(\frac{(\pi_i - \pi_j)^2}{\eta_{ij}} + \eta_{ij} \right) \\ \text{such that} & \pi \in \mathcal{P}, \\ & \eta_{ij} \geq \delta, \text{ for all } i, j. \end{array} \tag{η-HuberSUM}$$

The objective in η-HuberSUM is jointly convex in (π, η) (sum and combination of linear functions with quadratic over linear). The constraint set for η is convex, and although $\mathcal{P}$ is not, it can be relaxed to $\mathcal{PH}$. However we found empirically that an alternate minimization scheme that is not based on convex optimization but rather exploits the efficiency of the spectral algorithm demonstrates good performances. We present it in Algorithm 4.5. We use the spectral

Algorithm 4.5 η-Spectral Alternate Minimization Scheme for HuberSUM(δ).

Input: A similarity matrix $A \in \mathbf{S}_n^+$, a maximum number of iterations T.

1: Set $\eta^{(1)} = \mathbf{1}_n \mathbf{1}_n^T$.

2: **for** $t = 1$ **to** T **do**

3: Compute

$$\pi^{(t)} \in \operatorname*{argmin}_{\pi \in \mathcal{P}} \sum_{i,j=1}^n A_{ij} \left((\pi_i - \pi_j)^2 / \eta_{ij}^{(t)} + \eta_{ij}^{(t)} \right),$$

i.e., $\pi^{(t)}$ is solution of (2-SUM) for the matrix $A \oslash \eta$ where $\oslash$ denotes the Hadamard (entrywise) division.

4: Compute

$$\eta^* \in \operatorname*{argmin}_{\eta \geq \delta} \sum_{i,j=1}^n A_{ij} \left((\pi_i^{(t)} - \pi_j^{(t)})^2 / \eta_{ij} + \eta_{ij} \right),$$

i.e., $\eta_{ij}^* \leftarrow \max(\delta, |\pi_i^{(t)} - \pi_j^{(t)}|)$, for all (i, j).

5: Update $\eta^{(t+1)} \leftarrow \gamma \eta^{(t)} + (1 - \gamma) \eta^*$.

6: **end for**

Output: A permutation $\pi^{(T)}$.

algorithm to (approximately) solve (2-SUM) in line 3. Here, γ is a parameter that controls the influence of the previous iterates of η, the case $\gamma = 0$ is just plain alternate minimization. In practice, we evaluate the objective of (HuberSUM(δ)) for A and $\pi^{(t)}$ at each iteration, and keep the iterate π with the lowest score.

4.3.7 First Order Optimization on Manifold

Finally, we used a manifold optimization toolbox [Boumal et al., 2014] as a black-box, to which we provide the expression of the objective and gradient of HuberSUM(δ) and ask for the minimum over the sphere (computed with a trust-regions algorithm). We refer to this method as Manopt, which is the name of the toolbox [Boumal et al., 2014]. We use the formulation of the hyperplane $\mathcal{H}_n$ through an affine transformation as with the UBI method, in order to use (Manopt) with $f_{\mathcal{H}_n}$ on a sphere in $\mathbb{R}^{n-1}$ in the experiments.

4.4 Numerical Results

In this section, we test the algorithms detailed above on both synthetic and real data sets.

4.4.1 Synthetic data

We performed experiments with matrices from $\mathcal{M}_n(\delta, s)$ with $n = 100, 200, 500$, $\delta = n/10, n/20$, and $s/s_{\text{lim}} = 0.5, 1, 2.5, 5, 7.5, 10$, with s is the number of out-of-band terms as in Definition 4.2.1 and $s_{\text{lim}} = (n-\delta-1)$ is the value appearing in Proposition 4.2.2, where R2SUM(λ) and Robust Seriation coincide when $s \leq s_{\text{lim}}$. In Table 4.1, we show the seriation results of the different methods described in Section 4.3. When an algorithm can be used for 2-SUM, but also with R2SUM(λ) (or HuberSUM(δ), respectively), we pre-pend -R (or -H, resp.) to its name in the R-2SUM (or Huber, resp.) corresponding row of the Table. In Table 4.2, we show the Kendall-τ score for different values of s/s_{lim}. For a given set of parameters (n, δ, s), we generated 100 experiments with random locations for the out-of-band entries. The results displayed in Tables 4.1 and 4.2 are averaged over these experiments, with the standard deviation given after the $\pm$ sign. The experiments with different values of n and δ exhibit similar trends, as one can see in Tables C.1 and C.2. Overall, η-Spect. finds the best ordering, and is also time efficient. Uncons is also competitive. Some methods such as HGnCR do not perform as good in average, but have a higher standard deviation over the 100 simulations. They actually perform well on most simulations, but fail on a few ones. Overall this results in a lower mean Kendall-τ score and a higher standard deviation.

Table 4.1: Kendall-τ, HuberSUM(δ), R2SUM(λ), Robust Seriation (with Froebenius norm) scores for the different methods for $n = 200$, $\delta = 20$, and $s/s_{\text{lim}} = 5$. The results are averaged over 100 instances of $A \in \mathcal{M}_n(\delta, s)$. The first six methods are used with the 2-SUM loss, the six middle ones with the HuberSUM(δ) loss, where δ was chosen following the rule described at the end of §4.2.2, and the two last middle ones with the R2SUM(λ) loss. and Some scores are scaled to simplify the table.

	Kendall-τ	Huber $\times 1e{-}6$	R2SUM $\times 1e{-}6$	Dist2R	2SUM $\times 1e{-}6$	Time (s)
spectral	0.86 ±0.06	7.76 ±0.61	2.67 ±0.19	73.6 ±5.3	7.7 ±0.4	3.54e-01
GnCR	0.87 ±0.15	7.21 ±0.40	2.47 ±0.17	67.6 ±4.7	7.5 ±0.3	6.99e-01
FAQ	0.89 ±0.08	7.19 ±0.31	2.46 ±0.14	67.6 ±4.1	**7.4** ±0.2	3.37e+00
LWCD	0.89 ±0.08	7.18 ±0.30	2.46 ±0.14	67.5 ±3.9	**7.4** ±0.2	2.99e+00
UBI	0.89 ±0.06	7.32 ±0.31	2.52 ±0.12	69.5 ±3.3	7.5 ±0.2	1.45e+00
Manopt	0.86 ±0.06	7.72 ±0.58	2.66 ±0.18	73.2 ±5.2	7.6 ±0.4	3.90e+00
η-Spectral	**0.97** ±0.00	**6.74** ±0.13	2.03 ±0.02	50.8 ±0.8	7.6 ±0.2	1.07e+00
HGnCR	0.89 ±0.22	6.91 ±0.52	2.11 ±0.26	53.6 ±8.6	7.7 ±0.4	9.06e+00
H-FAQ	0.95 ±0.08	6.84 ±0.32	2.01 ±0.08	49.0 ±3.9	7.7 ±0.3	4.28e-01
H-LWCD	0.94 ±0.09	6.88 ±0.34	2.03 ±0.11	49.7 ±5.0	7.7 ±0.3	3.00e+00
H-UBI	**0.97** ±0.00	**6.74** ±0.13	2.05 ±0.02	51.4 ±1.1	7.6 ±0.2	3.08e+00
H-Manopt	0.92 ±0.06	7.05 ±0.39	2.26 ±0.15	59.7 ±5.2	7.6 ±0.3	9.22e+00
R-FAQ	0.95 ±0.10	6.97 ±0.40	**1.99** ±0.08	**44.9** ±4.3	7.9 ±0.4	3.39e-01
R-LWCD	0.94 ±0.09	7.03 ±0.42	2.01 ±0.09	46.0 ±4.8	8.0 ±0.4	3.32e+00

4.4.2 Frank-Wolfe with Tie-Break (FWTB) is biased

We have not included the results of the FWTB method in the previous section, as it performed poorly. After investigation, we realized that the tie-break constraint actually introduces a bias in the problem, as we explain in the following. Let us focus on the 2-SUM problem. The loss function is homogeneous,

$$f_{2SUM}(tx) = \sum_{i,j} A_{ij}(tx_i - tx_j)^2 = t^2 \sum_{i,j} A_{ij}(x_i - x_j)^2 = t^2 f_{2SUM}(x).$$

Similarly, $f_{1SUM}(tx) = t f_{1SUM}(x)$ for $t > 0$. Hence, scaling down a given vector x, *e.g.*, letting $x \leftarrow \frac{1}{2}x$, reduces the objective function but does not add information about the optimal permutation (the projection on the set of permutations is the same for both vectors). What we are interested in is to find a direction x^* which is optimal compared to other vectors x of same norm. In the original problem over permutations, all permutation vectors have the same norm. In the spectral relaxations, we optimize over a sphere. However, when we relax to $\mathcal{PH}$, the most prominent descent direction of the function is towards the center. The tie-breaking constraint prevents iterates reaching the center, but it adds a bias in a given direction because not all points saturating the tie-breaking contraint have the same norm nor the same distance to the center. On the set of points in $\mathcal{PH}_n$ where the tie-break is active, *e.g.*, $\{x \in \mathcal{PH}_n \,|\, x_1 + 1 \leq x_n\}$,

Table 4.2: Kendall-τ score for different values of s/s_{lim}, for the same methods as in Table 4.1, and $n = 200$, $\delta = 20$.

	$s/s_{\text{LIM}} = 0.5$	$s/s_{\text{LIM}} = 1$	$s/s_{\text{LIM}} = 2.5$	$s/s_{\text{LIM}} = 5$	$s/s_{\text{LIM}} = 7.5$	$s/s_{\text{LIM}} = 10$
SPECTRAL	0.96 ±0.01	0.95 ±0.01	0.91 ±0.03	0.86 ±0.06	0.84 ±0.06	0.80 ±0.09
GNCR	0.98 ±0.00	0.96 ±0.04	0.93 ±0.07	0.87 ±0.15	0.81 ±0.20	0.80 ±0.18
FAQ	0.98 ±0.00	0.97 ±0.00	0.94 ±0.02	0.89 ±0.08	0.87 ±0.08	0.82 ±0.13
LWCD	0.98 ±0.00	0.97 ±0.00	0.94 ±0.02	0.89 ±0.08	0.87 ±0.08	0.82 ±0.13
UBI	0.97 ±0.00	0.96 ±0.01	0.92 ±0.03	0.89 ±0.06	0.86 ±0.07	0.82 ±0.12
MANOPT	0.97 ±0.00	0.95 ±0.01	0.91 ±0.03	0.86 ±0.06	0.84 ±0.06	0.80 ±0.09
η-SPECTRAL	0.99 ±0.00	0.99 ±0.00	0.98 ±0.00	**0.97** ±0.00	**0.96** ±0.00	**0.94** ±0.06
HGNCR	**1.00** ±0.00	0.99 ±0.00	**0.99** ±0.00	0.89 ±0.22	0.85 ±0.23	0.83 ±0.25
H-FAQ	**1.00** ±0.00	**1.00** ±0.00	0.99 ±0.01	0.95 ±0.08	0.94 ±0.09	0.91 ±0.13
H-LWCD	**1.00** ±0.00	**1.00** ±0.00	0.99 ±0.02	0.94 ±0.09	0.94 ±0.09	0.90 ±0.14
H-UBI	0.99 ±0.00	0.99 ±0.00	0.98 ±0.00	**0.97** ±0.00	**0.96** ±0.01	**0.94** ±0.03
H-MANOPT	**1.00** ±0.00	0.99 ±0.00	0.97 ±0.02	0.92 ±0.06	0.89 ±0.07	0.84 ±0.10
R-FAQ	**1.00** ±0.00	**1.00** ±0.00	0.99 ±0.04	0.95 ±0.10	0.94 ±0.10	0.90 ±0.15
R-LWCD	0.99 ±0.00	**1.00** ±0.00	0.99 ±0.04	0.94 ±0.09	0.94 ±0.10	0.90 ±0.16

the point $\tilde{c} = c + e_n - \frac{1}{n}\mathbf{1}$ has a squared distance to c and ℓ_2 norm : $\|\tilde{c} - c\|_2^2 \simeq 1$, $\|\tilde{c}\|_2^2 \simeq \frac{n^3}{4}$, whereas a permutation π that satisfies the constraints has a distance to c that scales in n^3 and a larger norm : $\|\pi - c\|_2^2 \simeq \frac{n^3}{12}$, $\|\pi\|_2^2 \simeq \frac{n^3}{3}$. Therefore, although the direction $\tilde{c}$ may not be optimal for 2-SUM (compared to other vectors of same norm), the minimizer of 2-SUM with tie-break may be closer to the direction of $\tilde{c}$ than to the optimal one. This is what we observe in Figure 4.4 for the bad (orange, top-right triangle) tie-break.

When n becomes large, this may actually lead to numerical precision issues. Indeed, the $n-1$ first entries of $\tilde{c}$ are equal. When the optimum x^* in the tie-break-constrained $\mathcal{PH}$ gets close to $\tilde{c}$, the variations among the $n-1$ first entries of x^* also shrink and the precision required to sort them (in order to project back onto the set of permutations) may become too high.

In Figure 4.4, we also display a good (green, top-left triangle) tie-break. In practice, although there are $\binom{n}{2}$ non-redundant choices for the indexes i and j constituting a tie-breaking constraint $\pi_i + 1 \leq \pi_j$, we can use the solution $\pi^{\text{spectr.}}$ of the cheap, spectral ordering (Algorithm 3.1) to find a good candidate tie-break. Specifically, chose $i \in \operatorname{argmin} \pi^{\text{spectr.}}$ and $j \in \operatorname{argmax} \pi^{\text{spectr.}}$.

The performances of FWTB with the naive ($i = 1$, $j = n$) and spectral-initialized tie-breaking strategies are compared to that of the basic spectral Algorithm 3.1 in Table 4.3 (a -I is appended to the algorithm name for the spectral-initialized tie-break results), with the same experimental setup as in Section 4.2 with matrices in $\mathcal{M}_n(\delta, s)$.

We can see that using a default tie-breaking constraint performs very poorly on average. Using the solution of the spectral algorithm to define the tie-breaking constraint significantly improves the performance compared to using a default tie-break. Still, it does not outperform

Table 4.3: Kendall-τ score for different values of s/s_{lim}, for the spectral method and Frank-Wolfe with default and initialized tie-breaks (-I variants), with $n = 200$, $\delta = 20$.

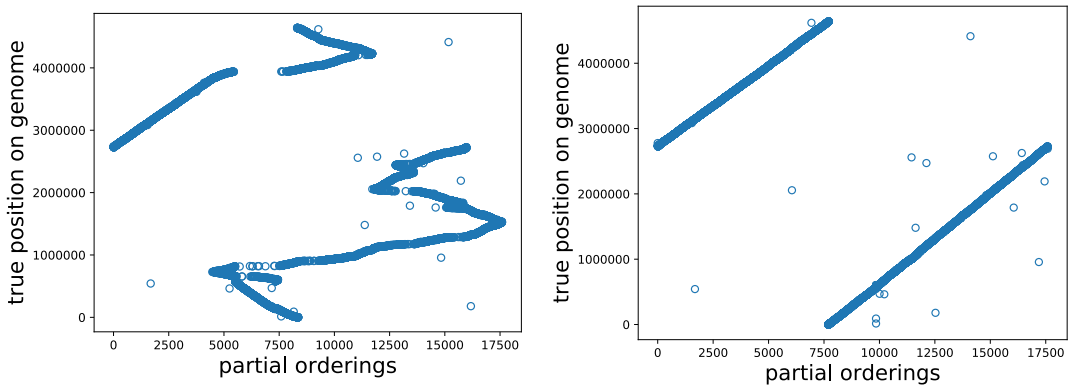
	$s/s_{\text{LIM}} = 0.5$	$s/s_{\text{LIM}} = 1$	$s/s_{\text{LIM}} = 2.5$	$s/s_{\text{LIM}} = 5$	$s/s_{\text{LIM}} = 7.5$	$s/s_{\text{LIM}} = 10$
SPECTRAL	0.96 ±0.01	0.95 ±0.01	0.91 ±0.03	0.86 ±0.06	0.84 ±0.06	0.80 ±0.09
FWTB	0.40 ±0.27	0.33 ±0.27	0.32 ±0.26	0.34 ±0.22	0.28 ±0.21	0.24 ±0.20
H-FWTB	0.50 ±0.31	0.36 ±0.27	0.32 ±0.25	0.32 ±0.22	0.25 ±0.21	0.22 ±0.18
FWTB-I	0.91 ±0.20	0.92 ±0.13	0.84 ±0.19	0.73 ±0.20	0.71 ±0.13	0.64 ±0.15
H-FWTB-I	0.98 ±0.01	0.94 ±0.13	0.86 ±0.15	0.70 ±0.18	0.63 ±0.15	0.57 ±0.17

the spectral algorithm except in a very low noise setting.

4.4.3 *E. coli* genome reconstruction

We performed experiments with two ONT bacterial data-sets introduced in Chapter 2, including reads sampled from an *Escherichia coli* genome [Loman et al., 2015], and from an *A. baylyi* genome [Recanati et al., 2016]. These data-sets are described in Section 2.3.1 and read-length histograms are given in Figure 2.4. Notably, 50% of the reads from the *E. coli* data-set are larger than 7kbp, whereas it is only the case for 20% of the *A. baylyi* reads. We used the `minimap2` tool [Li, 2018] with default ONT parameters to compute the overlaps between the reads. For each pair of reads for which `minimap2` found an overlap, we set the similarity value between those reads as the output (number of matching bases) from `minimap2`. This process defined a similarity matrix on which we tested our seriation methods. Among the methods that could scale to this size of problem $n \sim 10^4$, namely, the Frank-Wolfe based relaxations, UBI and η-Spectral, only η-Spectral gave satisfying results, which we report here. We performed a grid search on the threshold to set on the similarity matrix with 24 linearly spaced values varying between the 40% and 80% percentiles of all similarity entries. For each of them, we compute $\sqrt{\lambda} = \delta$ from the number of non-zero entries of the matrix as explained in 4.2.2, and kept the permutation yielding the best R2SUM(λ) score.

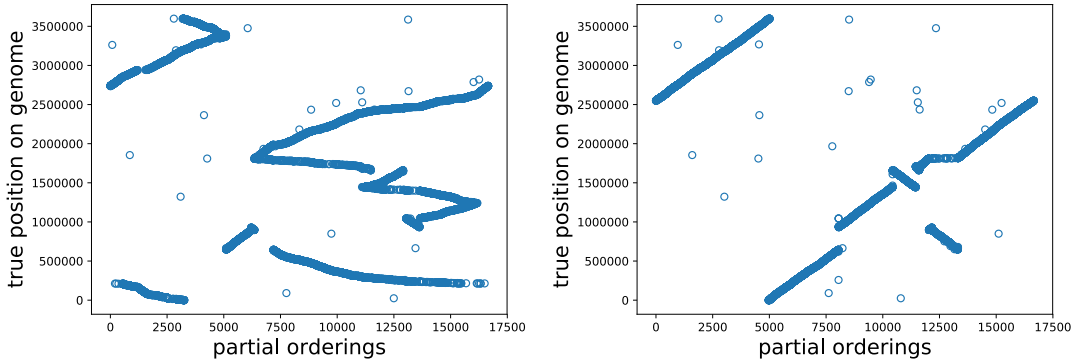
For the *E. coli* data, this method yielded correctly ordered reads, as one can see in Figure , with a Kendall-Tau score of 99.5% with the reference ordering obtained by mapping the reads to a reference genome with minimap2. In comparison, the spectral Algorithm 4.4 has a Kendall-Tau score of 32.6%. For the *A. baylyi* data, however, the method produced an ordering with mis-assemblies, as one can see on Figure , with a Kendall-Tau score of 90.3% (in comparison, Algorithm 4.4 has 41.5%). Here, we only assess the quality of the ordering, but we have seen in Chapter 2 that a correct layout lead to high quality assembly. The mis-ordered points on Figure are not scattered at random. We expect an assembly resulting from this layout to harbour a few large mis-assemblies, *i.e.*, large portions of genome mis-placed or reversed.



(a) Spectral

(b) η-Spectral

Figure 4.6: Ordering found with the spectral baseline Algorithm 4.4 (4.6a), and with the η-Spectral Algorithm 4.5 (4.6b) on the *E. coli* ONT data.



(a) Spectral

(b) η-Spectral

Figure 4.7: Ordering found with the spectral baseline Algorithm 4.4 (4.7a), and with the η-Spectral Algorithm 4.5 (4.7b) on the *A. baylyi* ONT data..

4.4.4 Genome assembly using Hi-C data

We evaluate the η-Spectral method on the real Hi-C data-sets introduced in Chapter 3, Section 3.6.4, where the weighted Kendall-Tau score $\tilde{\tau}$ to assess fragmented orderings, and the purity index and cluster distance to assess clustering, are defined. Table 4.4 shows the results for the *Plasmodium knowlesi* data, for which we have a ground truth chromosome assignment. Table 4.5 shows the results for the *Spodoptera frugiperda* data, for which we do not have such a ground truth clustering. We can see that the η-Spectral method improves upon the basic spectral method for the data-sets where the spectral method is already efficient, but that the method introduced in Chapter 3 combined with tSNE performs better on some data-sets harbouring a cluster structure, such as the *Plasmodium knowlesi* and Sf669 data.

Table 4.4: Seriation results on real Hi-C data from the *Plasmodium knowlesi* genome.

	SC+SO	SC+mdso	SC+η-SO	mdso	tSNE-mdso
# Chr.	14	14	14	8	16.7 ± 0.5
Purity	76.3	76.3	76.3	-	-
Cluster dist.	3.81	3.81	3.81	6.61	**2.83 ± 0.23**
$\tilde{\tau}$ (%)	77.3	69.6	79.3	18.9	**85.0 ± 11.7**
Time (s)	0.30	1.45	2.04	7.76	43.9 ± 2.8

Table 4.5: Seriation results on real Hi-C data from a *Spodoptera frugiperda* genome.

		Spectral	η-Spectral	mdso
Sf200	# Chr.	1	1	2
	$\tilde{\tau}$ (%)	92.7	**95.9**	91.6
	Time (s)	0.033	0.026	0.14
Sf669	# Chr.	1	1	4
	$\tilde{\tau}$ (%)	75.7	75.9	**88.2**
	Time (s)	0.049	0.61	0.19
Sf846	# Chr.	1	1	6
	$\tilde{\tau}$ (%)	95.8	**97.7**	86.7
	Time (s)	0.063	0.9	0.22

4.5 Conclusion

We introduced the Robust Seriation problem, which arises in *e.g. de novo* genome assembly. We show that for a class of similarity matrices modeling those observed in genome assembly, the problem of Robust Seriation is equivalent to a modified 2-SUM problem. This modified problem can be relaxed, with an objective function using a Huber loss instead of the squared loss present in 2-SUM. We adapt several relaxations of permutation problems to this 2-SUM problem with Huber loss and also introduce new relaxations, including the η-Spectral method, which is computationally efficient and performs best in our experiments. Notably, it successfully reorders a bacterial genome from third generation sequencing data.

Chapter 5

Seriation with Duplications

In this chapter, we introduce the problem of Seriation with Duplications. It is an extension of Seriation that differs from the problem of seriation with repeats. Here, given two duplicates, we do not observe the similarity values for any of the two duplicates, like we could with two repeated reads in genome assembly. Instead, we observe an aggregated similarity over all duplicates. It is motivated by an application to cancer genome assembly given Hi-C frequency data.

The content of this chapter is based on the following publication,
Antoine Recanati, Nicolas Servant, Jean-Philippe Vert, and Alexandre d'Aspremont. Robust seriation and applications to cancer genomics. *arXiv preprint arXiv:1806.00664*, 2018b

Supplementary material for this chapter is given in Appendix Chapter D.

Chapter Abstract

The seriation problem seeks to reorder a set of elements given pairwise similarity information, so that elements with higher similarity are closer in the resulting sequence. We introduce the problem of seriation with duplications, which is a generalization of Seriation motivated by applications to cancer genome reconstruction. In this context, we also aim to reorder a set of elements such that similar elements are nearby. However, some of these elements are identical copies, but we do not have access to the similarity information involving each of the copies. Instead, we observe a coarser-grained aggregated similarity, which is the sum over all the copies. We propose an alternated minimization scheme that involves seriation, and present preliminary results on synthetic data sets.

Contents

5.1 Introduction

The seriation problem has been studied throughout this thesis. As a reminder, it seeks to reorder a set of n elements given only pairwise similarity information. The resulting ordering should lay similar elements nearby. In practice, this translates to properties on the similarity matrix. We recall key definitions related to seriation from Chapter 4.

Definition 5.1.1. *We say that $A \in \mathbf{S}_n$ is a strong-R-matrix (or strong Robinson matrix) iff it is symmetric and satisfies $A_{ij} \leq A_{kl}$ for all (i, j, k, l) such that $|i - j| > |k - l|$.*

Here, $\mathbf{S}_n$ denotes the set of real symmetric matrices of dimension n. We write $\mathcal{R}_n^*$ the set of strong-R-matrices of size n, and $\mathcal{P}_n$ the set of permutations of n elements. A permutation can be represented by a vector π (lower case) or a matrix $\Pi \in \{0, 1\}^{n \times n}$ (upper case) defined by $\Pi_{ij} = 1$ iff $\pi(i) = j$, and $\pi = \Pi\mathbf{g}$ where $\mathbf{g} = (1, \ldots, n)^T$. We refer to both representations by $\mathcal{P}_n$ and may omit the subscript n whenever the dimension is clear from the context.

We say that $A \in \mathbf{S}_n$ is pre-$\mathcal{R}^*$ if there exists a permutation $\Pi \in \mathcal{P}$ such that the matrix $\Pi A \Pi^T$ (whose entry (i, j) is $A_{\pi(i),\pi(j)}$) is a strong-R-matrix, and the seriation problem seeks to recover this permutation Π, *i.e.*, solve

$$\begin{array}{ll} \text{find} & \Pi \in \mathcal{P} \\ \text{such that} & \Pi A \Pi^T \in \mathcal{R}^* \end{array} \qquad \text{(Seriation)}$$

in the variable $\Pi \in \mathcal{P}$.

Chapter 4 introduced the problem of Robust seriation, which seeks to find the closest pre-$\mathcal{R}^*$ matrix to A and reorder it, solving instead

$$\begin{array}{ll} \text{minimize} & \|S - \Pi A \Pi^T\| \\ \text{such that} & \Pi \in \mathcal{P}, \quad S \in \mathcal{R}^*. \end{array} \qquad \text{(Robust Seriation)}$$

where the variable $\Pi \in \mathcal{P}$ is a permutation matrix, the variable $S \in \mathcal{R}^*$ is a strong-R-matrix, and the norm is typically either the l_1 norm on components or the Froebenius norm.

The main challenge we have been confronted to when trying to perform genome assembly with Seriation is the presence of repeated regions throughout the genome. Let us say that a fragment of DNA is repeated in two separate locations in the genome. When computing pairwise alignments between sequences (to obtain a pairwise similarity), a read encompassing one of the repeated region can seem to overlap with reads encompassing the other.

In the former framework, each read has a distinct identifier, even though it may essentially contain a repeated sequence. Here, we are interested in a different, more complex problem, where we are not able to distinguish duplicates. If an element appears in two copies, we cannot tell whether one or the other copy is similar to another given element. Instead, we can only access a similarity value aggregated over the two duplicates (this will be formalized in

Section 5.2.2).

The chapter is organized as follows. In Section 5.2, we first motivate the problem with the application of cancer genome assembly through Hi-C data. Then, we formalize the problem, starting with an illustrative example. In Section 5.3, we propose an alternate minimization method to solve the problem of seriation with duplications. It involves solving robust seriation, for which we have proposed several algorithms in Chapter 4. Finally, we present numerical results on synthetic data in Section 5.4.

5.2 Seriation with Duplications

The reformulation of *de novo* sequencing as a (robust) seriation problem is based on the assumption that, up to noise, the bins can be reordered to form a long chain. While this hypothesis is relevant when a normal genome or chromosome is sequenced with long reads, it clearly fails to hold in an important case: cancer genomes. Indeed cancer cells typically harbour so-called *structural variations* where large portions of the genome, up to whole chromosomes, are duplicated or deleted, and where new chromosomes are formed by fusing two pieces of chromosomes which are not connected in a normal genome. For example, Figure 5.1 shows the 1D structure of a breast cancer cell line. Different colors correspond to DNA fragments normally in different chromosomes. Instead of 23 pairs of chromosomes with each pair in a single uniform color, expected in a normal cell, we observe various mosaics of colors indicating various duplication and fusion events.

5.2.1 Hi-C data

Reconstructing the 1D structure of a cancer genome from experimental data is an important problem. Besides standard DNA sequencing techniques, an interesting recent development called Hi-C and based on the chromosome conformation capture (3C) technology allows to measure experimentally the frequency of physical interactions in 3D between all pairs of positions in the genome [Lieberman-Aiden et al., 2009a]. In short, if we split the full human genome into n bins (of typical length $10^4 - 10^6$ basepairs each), an Hi-C experiment produces an $n \times n$ interaction matrix A such that A_{ij} is the frequency of interactions between DNA fragments in bins i and j. Interestingly, most 3D interactions take place between DNA fragments which are on the same chromosome, and the frequency of 3D interactions tends to decrease with the distance between the fragments when they are on the same chromosome; hence Hi-C data can be used to perform genome assembly, using *e.g.*, a seriation algorithm to obtain the layout [Korbel and Lee, 2013].

An Hi-C experiment roughly proceeds as follows. Freeze the DNA in its current 3D conformation, and collect pairs of DNA fragments that lie close to each other in this spatial conformation. For every such pair (k, l), map each of the two fragments to a normal reference

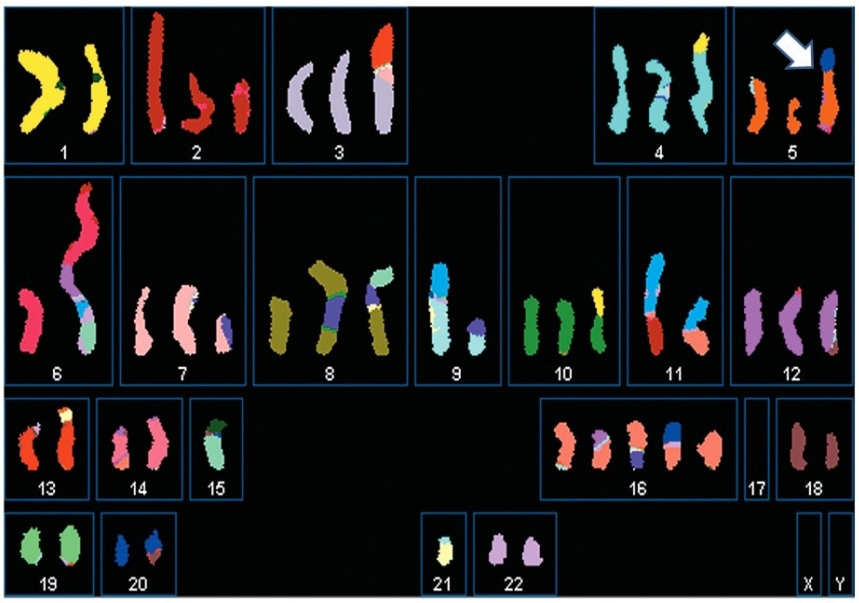


Figure 5.1: Structure of a typical cancer genome (breast cancer cell line). Instead of the standard 23 pairs of chromosomes, cancer cells often harbour large structural variants, such as changes in copy number and translocations. Reconstructing this 1D map from high-throughput Hi-C or sequencing data is an important problem that motivates the definition of seriation with duplications. Figure from Karp et al. [2015].

genome, providing their positions, p_k and p_l. Add $+1$ to the interaction matrix entry A_{ij} corresponding to the two bins i and j that respectively span p_k and p_l. This process is repeated to statistically obtain an average proximity (frequency) between two bins.

Because of duplications, deletions and translocations in cancer genome, each bin (defined according to a normal reference genome) may be included in several fragments of different chromosomes in a cancer genome, and it may therefore not be possible nor relevant to order the bins. Instead, since it is possible to estimate from Hi-C data the total number of DNA copies for each bin, it makes more sense to first associate to each bin a corresponding number of fragments (e.g. two fragments per bin in a normal diploid genome), and then reconstruct an ordering of fragments into a number of chains to estimate the 1D structure of a cancer genome (Figure 5.1).

The difficulty to apply a seriation algorithm is that Hi-C data provide *cumulative* information at the bin level, not at the fragment level. More precisely, if we denote S_{kl} the (unobserved) frequency of interactions between fragments k and l, respectively extracted from bins b_i and b_j, what Hi-C measures as interactions between b_i and b_j is the sum of $S_{k'l'}$ where k' and l' are fragments contained in b_i and b_j, respectively. This motivates the definition of the seriation with duplication problem formalized below.

5.2.2 Problem setting

For clarity, let us begin by an example with $n = 3$, $N = 4$. Consider a simplified reference genome split in 3 subsequences, $g = (\heartsuit, \diamondsuit, \clubsuit)$. In a cancer genome, the $\heartsuit$ sequence is duplicated and also appears at the end of the genome. Using the symbol $\heartsuit$ to denote the duplicated sequence of DNA, the cancer genome can be written $\tilde{g} = (\heartsuit, \diamondsuit, \clubsuit, \heartsuit)$. The true interaction matrix between the fragments $(\heartsuit, \diamondsuit, \clubsuit, \heartsuit)$ is a $\mathcal{L}_R$ matrix,

$$S_* = \begin{array}{c} \\ \heartsuit \\ \diamondsuit \\ \clubsuit \\ \heartsuit \end{array} \begin{array}{c} \begin{array}{cccc} \heartsuit & \diamondsuit & \clubsuit & \heartsuit \end{array} \\ \begin{pmatrix} 3 & 2 & 1 & 0 \\ 2 & 3 & 2 & 1 \\ 1 & 2 & 3 & 2 \\ 0 & 1 & 2 & 3 \end{pmatrix} \end{array}$$

Yet, interactions between $(\clubsuit, \heartsuit)$ and $(\clubsuit, \heartsuit)$ are both attributed to $(\clubsuit, \heartsuit)$ by the Hi-C experiment, resulting in the following observed interaction matrix and duplication count vector,

$$A = \begin{array}{c} \\ \heartsuit \\ \diamondsuit \\ \clubsuit \end{array} \begin{array}{c} \begin{array}{ccc} \heartsuit & \diamondsuit & \clubsuit \end{array} \\ \begin{pmatrix} 6 & 3 & 3 \\ 3 & 3 & 2 \\ 3 & 2 & 3 \end{pmatrix} \end{array}, \quad c = (2, 1, 1)^T.$$

Observing A, the sequence we wish to reconstruct is in fact $\pi_* = (1, 2, 3, 1)^T$.

Given a matrix $A \in \mathbf{S}_n$ of similarity between n bins, and a vector $c \in \mathbf{N}^n$ (the "counts" of the bins), with total $N = \sum_{i=1}^n c_i$, Seriation with Duplications aims at finding a sequence $\tilde{\pi} \in [1, n]^N$ of N integers such that i appears c_i times in $\tilde{\pi}$, at positions $L_i \subset [1, N]$ with $|L_i| = c_i$, and a matrix $S \in \mathcal{R}_N^*$ such that

$$A_{ij} = \sum_{k \in L_i, l \in L_j} S_{kl} \quad \text{for all } i, j \in [1, n].$$

Remark that if $c = \mathbf{1}_n$ (the vector of $\mathbb{R}^n$ with all entries equal to 1), the problem is equivalent to seriation and $\tilde{\pi}$ is a permutation vector.

To represent the subsets $\{L_i\}_{i \in [1,n]}$, we use assignment matrices $Z \in \{0, 1\}^{n \times N}$ such that $Z_{ik} = 1$ iff $k \in L_i$ (as in clustering problems). Such an assignment matrix is linked to the vector-based notation $\tilde{\pi} \in [1, n]^N$ from above through $\tilde{\pi} = Z^T(1, 2, \ldots, n)^T$. We write $\mathcal{Z}_c$ the set of assignment matrices for a given duplication count vector $c \in \mathbf{N}^n$,

$$\mathcal{Z}_c = \left\{ Z \in \{0, 1\}^{n \times N} \;\middle|\; Z\mathbf{1}_N = c \,,\; Z^T\mathbf{1}_n = \mathbf{1}_N \right\}$$

where $N = c^T \mathbf{1}_n$, and the constraints indicate that each bin $i \in [1, n]$ has c_i duplicates, and that each element $k \in [1, N]$ comes from one single bin. Observe that given an initial assignment matrix $Z_0 \in \mathcal{Z}_c$, any other $Z \in \mathcal{Z}_c$ can be expressed as Z_0 whose columns have been permuted, *i.e.* there exists $\Pi \in \mathcal{P}_N$ such that $Z = Z_0\Pi$. As in the Seriation formulation, the problem of Seriation with Duplications can be written

$$\begin{array}{ll} \text{find} & \Pi \in \mathcal{P}_N \ , \ S \in \mathcal{R}_N^* \\ \text{such that} & Z_0 \Pi S \Pi^T Z_0^T = A. \end{array} \tag{SD}$$

where Z_0 is an initial assignment matrix. Like Seriation, SD may not be feasible. The analog of Robust Seriation is then written

$$\begin{array}{ll} \text{minimize} & \|Z_0 \Pi S \Pi^T Z_0^T - A\| \\ \text{such that} & \Pi \in \mathcal{P}_N, \ S \in \mathcal{R}_N^*. \end{array} \tag{RSD}$$

Note again that if $c = \mathbf{1}_n$, then $N = n$, $Z_0 = \mathbf{I}_n$, and SD (respectively RSD) is equivalent to Seriation (resp. Robust Seriation).

5.3 Algorithms

5.3.1 Alternate projection for Seriation with Duplications

Let us assume that we are able to project on the set of pre-strong-R matrices, that is to say, given S, we can compute the couple $(\Pi_*, S_*) \in \mathcal{P} \times \mathcal{R}^*$ that minimizes $\|\Pi R \Pi^T - S\|$ (note that the projection on the set of pre-strong-R matrices is nothing but the Robust Seriation problem). We can then use alternationg projections to optimize (RSD) (although the set of pre-strong-R matrices is not convex, so convergence to a global optimum is not garanteed). We detail this method in Algorithm 5.1.

In fact, we can use any method presented in the previous chapter (Section 4.2) to solve the projection step 3 in Algorithm 5.1. In our experiments here, we use η-spectral and UBI, which are the most efficient, and spectral as a baseline. From the permutation Π_* obtained by, *e.g.*, solving HuberSUM(δ) with η-Spectral, we compute S_* by doing a ℓ_1 projection of $\Pi_* S^{(t)} \Pi_*^T$ onto $\mathcal{R}^*$ through linear programming. Indeed, the membership to $\mathcal{R}^*$ can be described by a set of linear inequalities. We can also add upper bounds on the matrix entries belonging to a given diagonal, if we have *a priori* knowledge on the law by which the entries decrease when moving away from the diagonal, which is the case for Hi-C genome reconstruction. We detail these steps in Section 5.3.3. Projecting onto the set of matrices satisfying linear equality constraints in step 4 can also be done with a convex programming solver, but the problem is actually separable on the values $(i, j) \in [1, n] \times [1, n]$ and has a closed form solution detailed in Section 5.3.3.

Algorithm 5.1 General Alternating Projection Scheme for Seriation with Duplications.

Input: A matrix $A \in \mathbf{S}_n$, a duplication count vector $c \in \mathbf{N}^n$, a maximum number of iterations T.

1: Set $N = \sum_{i=1}^{n} c_i$, $Z^{(0)} \in \mathcal{Z}_c$ and $S^{(0)} = Z^{(0)T}\,\mathbf{diag}(c^{-1})A\,\mathbf{diag}(c^{-1})^T Z^{(0)}$, *i.e.*, $S^{(0)}_{kl} = \frac{A_{ij}}{c_i c_j}$ with $k \in L_i$ and $l \in L_j$.
2: **while** $t \leq T$ **do**
3: Compute (Π_*, S_*), solution of (Robust Seriation) for $S^{(t)}$, and set
$S^{(t+\frac{1}{2})} \leftarrow S_*$
$Z^{(t+1)} \leftarrow Z^{(t)}\Pi_*$
4: Compute S_A, projection of $S^{(t+\frac{1}{2})}$ on the set of matrices that satisfy $Z^{(t+1)}SZ^{(t+1)T} = A$, and set
$S^{(t+1)} \leftarrow S_A$
5: $t \leftarrow t + 1$.
6: **if** $Z^{(t+1)} = Z^{(t)}$ **then**
7: **break**
8: **end if**
9: **end while**

Output: A matrix $S^{(T)}$, an assignment matrix $Z^{(T)}$

5.3.2 Algorithms for Robust Seriation

We have studied the (Robust Seriation) problem in Chapter 4 and evaluated various methods designed to solve it. One of the steps of Algorithm 5.1 coincides with (Robust Seriation). In this chapter, we will retain the three following methods evaluated in Chapter 4,

- Spectral (baseline method, Algorithm 4.4)
- η-Spectral (Algorithm 4.5)
- Unconstrained minimization in $\mathcal{PH}$ (UBI, Algorithm 4.3),

and refer the reader to Section 4.3 for details on these methods.

5.3.3 Algorithmic details

We now detail algorithmic solutions to several subproblems required by seriation with duplications.

Projection on $\mathcal{R}^*$ (step 3 of Algorithm 5.1)

In step 3 of Algorithm 5.1, we wish to compute (Π_*, S_*), solution of (Robust Seriation) for $S^{(t)}$. To do so, we can use one of the algorithms presented in Section 4.2. However, these algorithms do not address the problem of Robust Seriation directly. Rather, they seek to find a permutation that is optimal for a objective function which coincides with Robust Seriation for the specific class of $\mathcal{M}_n(\delta, s)$ matrices. Two problems arise then. First, in our Seriation

with Duplication setting (SD), the matrices may not fit the class $\mathcal{M}_n(\delta, s)$, especially when the matrix S to be recovered is dense (and not a band matrix). Second, the output of the algorithm is a permutation Π_*, but what we are really interested in step 3 of Algorithm 5.1 is the matrix $S_* \in \mathcal{R}_N^*$ that is the closest to $S^{(t)}$. To approximate $S_* \in \mathcal{R}_N^*$, we first use one of the methods introduced in Section 4.2 to find a permutation Π_* that makes $\Pi_* S^{(t)} \Pi_*^T$ as close to $\mathcal{R}_N^*$ as possible. Still, in general the permuted matrix $\Pi_* S^{(t)} \Pi_*^T$ will not be in $\mathcal{R}_N^*$. We then project $\Pi_* S^{(t)} \Pi_*^T$ onto $\mathcal{R}_N^*$, which is solved with linear programming. Indeed, the projection, in ℓ_1 norm for example of a matrix S, reads

$$\begin{array}{ll} \text{minimize} & \sum_{i,j=1}^{N} |R_{ij} - S_{ij}| \\ \text{such that} & R \in \mathcal{R}_N^*. \end{array} \tag{R-proj}$$

We can also use a Froebenius norm and consider the sum of squares instead of the absolute differences. We would then use quadratic programming, as we have then a quadratic objective with linear constraints. The constraint $R \in \mathcal{R}_N^*$ can indeed be written as linear constaints on R. Specifically, we consider the vectorized forms of S and R, $s, r \in \mathbb{R}^{N^2}$, which are the concatenation of the columns of S and R, respectively. Imposing $R \in \mathcal{R}^*$ is equivalent to saying that $r_u \leq r_v$ for all pairs of indexes (u, v) such that the corresponding subscripts for u are on a diagonal higher than those for v. There is one linear constraint per pair (u, v) (and there are $\frac{N(N-1)}{2}$ pairs), but we can reduce the number of constraints by adding slack variables $\{\lambda_k\}_{1 \leq k \leq N}$ and impose that for each element r_u on a given diagonal k, $1 \leq k \leq N-1$, $r_u \leq \lambda_{k+1}$ and $r_u \geq \lambda_k$. Finally, we can use *a priori* knowledge on how the values are supposed to decrease when moving away from the diagonal (*e.g.*, a power law $S_{ij} = |i-j|^{-\gamma}$ as in our experiments, which is consistent with the intra-chromosomal frequency observed in Lieberman-Aiden et al. [2009b]), to upper bound the values λ_k. We end up with the following optimization problem over the variable $(r, \lambda)^T$,

$$\begin{array}{ll} \text{minimize} & \|r - s\| \\ \text{such that} & C \begin{pmatrix} r \\ \lambda \end{pmatrix} \leq 0, \\ & 0 \leq \lambda \leq b \end{array} \tag{R-proj}$$

where the matrix C contains the strong-R constraints expressed between r and λ, and the vector $b \in \mathbb{R}^N$ contains upper bounds on the values of λ_k, *e.g.*, $b_k = k^{-\gamma}$.

Projection on duplication constraints (step 4 of Algorithm 5.1)

In step 4 of Algorithm 5.1, we wish to compute the projection of S on the set of matrices X that satisfy $ZXZ^T = A$, that is to say, solve the following optimization problem on variable

X,

$$\begin{array}{ll} \text{minimize} & \sum_{k,l=1}^{N} |S_{kl} - X_{kl}| \\ \text{such that} & ZXZ^T = A. \end{array} \qquad \text{(dupli-proj)}$$

The constraints impose that for each pair $(i,j) \in [1,n] \times [1,n]$, $A_{ij} = \sum_{k \in L_i, l \in L_j} X_{kl}$, where $L_i \subset [1,N]$ is the set of indexes assigned to i through the assignment matrix Z. The objective is also separable, since

$$\sum_{k,l=1}^{N} |S_{kl} - X_{kl}| = \sum_{i,j=1}^{n} \sum_{k \in L_i, l \in L_j} |S_{kl} - X_{kl}|$$

We can then solve separately, for each pair (i,j), the subproblem,

$$\begin{array}{ll} \text{minimize} & \sum_{k \in L_i, l \in L_j} |S_{kl} - X_{kl}| \\ \text{such that} & A_{ij} = \sum_{k \in L_i, l \in L_j} X_{kl}. \end{array} \qquad \text{(dupli-proj(i,j))}$$

For a given pair (i,j), L_i and L_j are known (through Z), and if we consider the vectorization (stacking of the columns into a single vector) of the submatrices X_{L_i,L_j} and S_{L_i,L_j}, denoted x and s respectively, and denote $a = A_{ij}$, the subproblem on the variable x reads

$$\begin{array}{ll} \text{minimize} & \|s - x\| \\ \text{such that} & x^T \mathbf{1} = a, \\ & x \geq 0. \end{array} \qquad \text{(dupli-proj(i,j))}$$

We impose non-negativity of the coefficients of X since this is part of the definition of similarity matrices. The above general problem of approximating a vector with a non-negative vector of fixed norm can be solved exactly when the norm is the ℓ_2 norm (this solution is optimal for the ℓ_1 norm too) with Algorithm 5.2.

5.4 Numerical Results

We performed synthetic experiments in which we generate the data as follows. We first build a strong-R matrix S of size N, and a random duplication count vector $c \in \mathbf{N}^n$ such that $N = \sum_{i=1}^{n} c_i$. We generate a random assignment matrix $Z \in \mathcal{Z}_c$, and the corresponding observed matrix $A = ZSZ^T$. We then test Algorithm 5.1 by providing it with A and c and comparing its output Z^{out} and S^{out} to the ground truth.

Specifically, we compute the relative Froebenius distance between S and S^{out}, $\mathrm{d2}R = \|S - S^{out}\|_F / \|S\|_F$, and we compute a distance between the assignment matrices as follows. For a given bin index $i \in [1,n]$ (*i.e.* a row Z_i), there are c_i locations for the non-zeros of the i-th row

Algorithm 5.2 Minimizing $\|s - x\|$ with non-negativity ($x \geq 0$) and sum ($x^T \mathbf{1} = a$) constraints.

Input: A target vector $s \in \mathbb{R}^p_+$, a value $a \geq 0$.

$s', \sigma \leftarrow$ sort s in decreasing order (*i.e.*, $s(\sigma_1) \geq \ldots \geq s(\sigma_n)$)
for $k \leftarrow 1$ to n **do**
  $\tilde{x}'(k) \leftarrow s'(k) + \frac{1}{k}(a - \sum_{i=1}^{k} s'(i))$
  if $\tilde{x}'(k) < 0$ **then**
    $k \leftarrow k - 1$
    break
  end if
end for
$x'(j) = s'(j) + \frac{1}{k}(a - \sum_{i=1}^{k} s'(i))$ for $j = 1, \ldots, k$
$x'(j) = 0$ for $j > k$
$x(\sigma_j) = x'(j)$ for $j = 1, \ldots, p$.

Output: A vector $x \in \mathbb{R}^p_+$.

of Z and of Z^{out} (which can also be viewed as two subsets L_i and L_i^{out} of $[1, N]$). To compute the distance between these positions, we first compute a matching between the elements of L_i and L_i^{out} using the Hungarian algorithm [Kuhn, 1955]. Then, we compute the distance between each matched pair of elements $(k, k^{out}) \in L_i \times L_i^{out}$, and store the average distance between matching pairs for row i. Supplementary Figures D.5 and D.6 illustrates this process. The average over all rows of this average distance is given in Table 5.1 as meanDist, and we also provide its standard deviation and median.

In the experiments, we built dense strong-R, Toeplitz matrices S where the entries follow a power law of the distance to the diagonal, $S_{kl} = |k-l|^{-\gamma}$, which is consistent with the observed frequency of intra-chromosomal interactions [Lieberman-Aiden et al., 2009b]. We used $N = 200$ and tried several values for the exponent γ and the ratio N/n, namely $\gamma \in \{0.1,\ 0.5,\ 1\}$ and $N/n \in \{1.33,\ 2,\ 4\}$. The results are shown in Tables 5.2, D.2, D.3, and some qualitative results are shown in Figure D.1. We also conducted experiments with sparse, band matrices $S \in \mathcal{M}_N(\delta, s)$ as in Section 4.2. The results are shown in Tables 5.1, D.4, D.5, and some qualitative results are shown in Figure D.2. The η-Spectral method works best for dense matrices, and is outperformed by H-UBI for maatrices in $\mathcal{M}_N(\delta, s)$. We observe that, as expected, the recovered assignment Z^{out} is closer to Z when N/n is smaller. However, the D2S scores and the qualitative Figures D.1 and D.2 suggest that for large N/n, the recovered matrix S^{out} may be close to S although the assignment is not well recovered. Intuitively, this means the problem is degenerate, with several assignment matrices roughly leading to the same matrix S, and the algorithm finds one of these.

Table 5.1: Results of synthetic experiments for Seriation with Duplications from matrices $S \in \mathcal{M}_N(\delta, s)$ with $n = 200$, $\delta = n/5$, $s = 0$, and various values of N/n, where the (Robust Seriation) problem is tackled with either Spectral, η-Spectral or H-UBI within Algorithm 5.1. From the output S^{out} and Z^{out} of Algorithm 5.1 and the ground truth S and Z from which the data A is generated, D2S is the relative Froebenius distance between S and S^{out}, Huber is the (HuberSUM(δ)) loss on S, meanDist, stdDist and medianDist are the average, standard deviation and median of the distance between the positions assigned to a index k by Z and Z^{out} (see main text for details). Time is the amount of CPU time elapsed until convergence of Algorithm 5.1.

N/n	METHOD	D2S	HUBER ($\times 1e{-}7$)	MEANDIST	STDDIST	TIME ($\times 1e{-}3s$)
1.33	SPECTRAL	0.53 $\pm$0.08	1.67 $\pm$0.33	11.8 $\pm$3.5	13.2 $\pm$1.7	7.45 $\pm$4.08
	η-SPECTRAL	0.12 $\pm$0.06	0.76 $\pm$0.06	0.8 $\pm$0.8	2.4 $\pm$2.2	2.85 $\pm$1.78
	H-UBI	**0.09** $\pm$0.06	**0.74** $\pm$0.05	**0.6** $\pm$0.6	1.8 $\pm$1.9	3.99 $\pm$2.76
2	SPECTRAL	0.38 $\pm$0.05	1.48 $\pm$0.26	10.3 $\pm$4.2	10.5 $\pm$2.8	1.30 $\pm$0.25
	η-SPECTRAL	0.21 $\pm$0.04	0.99 $\pm$0.12	4.1 $\pm$4.1	6.9 $\pm$3.9	0.50 $\pm$0.19
	H-UBI	**0.19** $\pm$0.05	**0.96** $\pm$0.14	**4.0** $\pm$5.8	6.2 $\pm$4.6	0.79 $\pm$0.31
4	SPECTRAL	0.29 $\pm$0.02	1.45 $\pm$0.09	18.4 $\pm$4.5	11.8 $\pm$3.1	1.34 $\pm$0.23
	η-SPECTRAL	0.22 $\pm$0.02	1.29 $\pm$0.06	16.3 $\pm$6.8	12.2 $\pm$5.1	0.61 $\pm$0.14
	H-UBI	**0.22** $\pm$0.02	**1.26** $\pm$0.06	**15.9** $\pm$7.2	12.0 $\pm$5.6	0.91 $\pm$0.25

5.5 Multiple chromosomes : Seriation+Clustering with Duplications

In the application motivating this problem, the cancer genome to be reconstructed has multiple chromosomes (which may harbour structural variants). In a Hi-C experiment, the inter-chromosome frequencies of interaction are significantly lower than the intra-chromosome frequencies [Lieberman-Aiden et al., 2009a]. Thus, if reordered correctly, the target similarity matrix S has a block-structure, and each block has a Robinsonian structure. Still, the methods we use for Seriation, and in particular the spectral methods, are not necessarily suited to reordering clustered (block) matrices. We therefore propose to add a clustering step in Algorithm 5.1 in order to leverage the cluster structure. This is summarized in Algorithm 5.3, where we project the current matrix S on the set of block matrices in line 4, and we reorder each cluster in line 5. The projection on block matrices is not exactly a clustering procedure, since we only want to find breakpoints between clusters, but two points can be in the same cluster only if they are contiguous in the current ordering Π_*. To find these breakpoints, we use an algorithm from Ding and He [2004] which seeks the minima of a measure called cluster crossing. For each point i, it is roughly defined as the sum along the anti-diagonal i in a bandwidth m,

$$\rho(i) = \sum_{j=1}^{m} A_{i-j,i+j}, \tag{5.1}$$

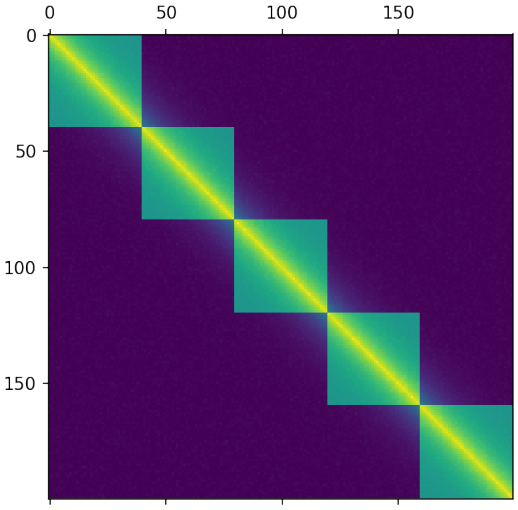
Table 5.2: Results of synthetic experiments for Seriation with Duplications from dense, strong-R matrices of size $n = 200$, with the same metrics and methods as in Table 5.1, with $\gamma = 0.5$.

N/n	METHOD	D2S	HUBER ($\times 1e{-}7$)	MEANDIST	STDDIST	TIME ($\times 1e{-}2s$)
1.33	SPECTRAL	0.25 ±0.04	1.36 ±0.03	6.1 ±1.8	7.9 ±1.6	8.74 ±4.85
	η-SPECTRAL	**0.15** ±0.02	**1.30** ±0.01	**2.2** ±0.7	3.7 ±1.1	6.12 ±4.84
	H-UBI	0.24 ±0.04	1.35 ±0.03	5.5 ±1.6	7.3 ±1.4	11.06 ±7.56
2	SPECTRAL	0.27 ±0.02	1.41 ±0.02	9.5 ±1.6	8.4 ±1.3	7.47 ±3.20
	η-SPECTRAL	**0.22** ±0.02	**1.37** ±0.02	**6.6** ±1.5	6.7 ±1.9	7.89 ±3.89
	H-UBI	0.26 ±0.02	1.40 ±0.02	9.0 ±1.5	8.1 ±1.2	10.09 ±4.90
4	SPECTRAL	0.18 ±0.01	1.35 ±0.01	14.4 ±2.8	8.7 ±2.7	6.53 ±1.90
	η-SPECTRAL	**0.18** ±0.01	**1.35** ±0.01	**14.3** ±2.9	8.9 ±2.9	7.59 ±2.28
	H-UBI	0.19 ±0.01	1.35 ±0.01	14.8 ±2.5	8.8 ±2.1	8.62 ±2.46

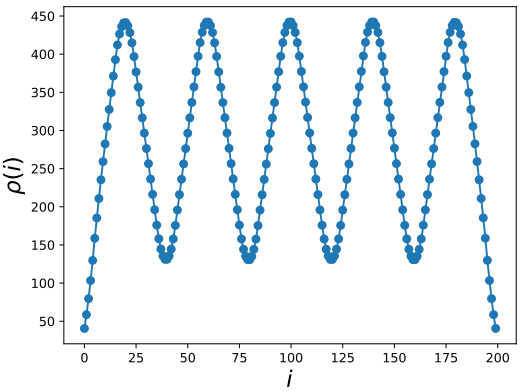
where m can be chosen according to the number of target clusters. If the similarity matrix has a cluster structure, then the cluster crossing ρ should have local minima at the boundaries between the clusters, as we can see in Figure 5.2b.

5.5.1 Numerical experiments with block + Robinson matrices

We conducted experiments similar to those of Section 5.4 where we start with matrices S that are the sum of a dense Robinson matrix and a block matrix, as the one displayed in Figure 5.2.



(a) similarity S



(b) cluster crossing ρ

Figure 5.2: Similarity matrix with Robinson + Block structure (5.2a), and the associated cluster-crossing curve that enables one to determine the breakpoints between the clusters (5.2b).

Algorithm 5.3 General Alternating Projection Scheme for Seriation+Clustering with Duplications.

Input: A matrix $A \in \mathbf{S}_n$, a duplication count vector $c \in \mathbf{N}^n$, a maximum number of iterations T.

Set $N = \sum_{i=1}^{n} c_i$, $Z^{(0)} \in \mathcal{Z}_c$ and $S^{(0)} = Z^{(0)T}\,\mathbf{diag}(c^{-1})A\,\mathbf{diag}(c^{-1})^T Z^{(0)}$, *i.e.*, $S^{(0)}_{kl} = \frac{A_{ij}}{c_i c_j}$ with $k \in L_i$ and $l \in L_j$.
while $t \leq T$ **do**
  Compute (Π_*, S_*), solution of (Robust Seriation) for $S^{(t)}$.
  Compute $S_{\text{Clus.}}$, projection of S_* on the set of block matrices.
  Reorder each block with (Robust Seriation), and update Π_* accordingly.
  Set $S^{(t+\frac{1}{2})} \leftarrow S_*$, and $Z^{(t+1)} \leftarrow Z^{(t)}\Pi_*$.
  Compute S_A, projection of $S_{\text{Clus.}}$ on the set of matrices that satisfy $Z^{(t+1)}SZ^{(t+1)T} = A$, and set
  $S^{(t+1)} \leftarrow S_A$
  $t \leftarrow t + 1$.
  if $Z^{(t+1)} = Z^{(t)}$ **then**
    break
  end if
end while

Output: A matrix $S^{(T)}$, an assignment matrix $Z^{(T)}$

In Table 5.3, we provide the results of such experiments with 5 clusters, where we use the η-Spectral algorithm to solve (Robust Seriation) in Algorithms 5.1 and 5.3. Table D.6 is the analog with 10 clusters, for which the results are similar. We observe that when N/n is suffiently small ($N/n = 1.33$), the clustering step clearly helps the algorithm to converge to a good minimum. However, the performance gain becomes marginal and both Algorithms 5.1 and 5.3 perform poorly with higher values of N/n.

5.6 Discussion

We have introduced the problem of Seriation with Duplications. It generalizes Seriation and can adapt the problem of performing genome assembly from Hi-C frequency matrices to genomes with structural variants. After formalizing the problem setting, we present an algorithmic scheme based on seriation and alternate projections between the set of duplications constraints and the set of Robinson matrices. We evaluate this method synthetic experiments on stylized matrices modeling Hi-C experiments from single-chromosome genomes, with duplications. Then, we outline the issues due to the presence of multiple chromosomes. We propose to modify the alternate projections algorithm to handle the cluster structure, and evaluate it on synthetic data, on which it perform moderately well.

Table 5.3: Results of synthetic experiments for Seriation+Clustering with Duplications from dense, strong-R matrices of size $n = 200$, with an additive block matrix with 5 clusters, with Algorithm 5.1 (that do not take the cluster structure into account), denoted SerDupli and Algorithm 5.3, denoted SerDuClus. Both are used with the η-Spectral method at step 3 of the alternate projections Algorithm. The results are averaged over 20 experiments and the standard deviation is given after the $\pm$ sign.

N/n	method	Huber	meanDist	stdDist
1.33	SerDupli	1.266e+07 $\pm$4.946e+06	23.9 $\pm$16.2	19.9 $\pm$12.2
	SerDuClus	**9.265e+06** $\pm$2.439e+06	**9.04** $\pm$10.4	8.9 $\pm$8.1
2	SerDupli	1.683e+07 $\pm$5.283e+06	38.8 $\pm$9.9	26.6 $\pm$7.1
	SerDuClus	**1.373e+07** $\pm$4.284e+06	**30.6** $\pm$12.8	20.2 $\pm$8.1
4	SerDupli	3.639e+07 $\pm$4.357e+06	42.0 $\pm$11.0	18.5 $\pm$4.9
	SerDuClus	**2.512e+07** $\pm$6.704e+06	**35.5** $\pm$9.1	15.5 $\pm$5.0

Chapter 6

Conclusion and Perspectives

6.1 Summary of the thesis

Throughout this thesis, we have studied the problem of seriation with the aim of solving an important genome assembly task. We explored existing techniques and develop new methods in order to meet the challenges arising with real genomic data.

In the opening chapter, we formalized the seriation problem mathematically. We recalled algorithmic challenges and detailed existing approaches constituting the basis of our work. Then, we presented genome assembly techniques, highlighted the key challenges, and reviewed major sequencing technologies.

In our first contribution, we made the proof of concept that the seriation framework was suited to *de novo* genome assembly. To this end, we applied a standard spectral, seriation algorithm to real *de novo* assembly problems, using third-generation sequencing data. We integrated our seriation module seamlessly in an end-to-end overlap-layout-consensus assembly scheme. This yielded competitive experimental results compared to state-of-the-art methods, although it was challenged by the repeated regions occurring in DNA sequences, leading to possibly fragmented assemblies.

In our second contribution, we borrowed from spectral graph theory and embedding techniques to propose an extension of the spectral method used in the first contribution (Chapter 2). It provided a unifying framework for seriation and circular seriation, a variant of the problem where instead of seeking for an ordering of data along a linear chain, we search for a cyclic ordering of the data, where the objects at the end of the cycle are similar to those at the beginning of it. We derived theoretical guarantees for circular seriation analog to those existing for linear seriation. Notably, our extended spectral method significantly improves the robustness of the original spectral method when the data is corrupted by noise. We evaluated it on several types of data, including third-generation sequencing data for *de novo* assembly, spatial conformation (Hi-C) data, and single-cell Hi-C data used in a cell-cycle ordering problem.

In our third contribution, we attempted to model the problem of performing seriation on

data corrupted by noise, and introduced the framework of robust seriation. We showed how several seriation-like optimization problems relate for stylized matrices modeling those observed in *de novo* assembly (including the noise). Then, we explored several algorithmic approaches, including recently developed methods for permutation problems, and new methods that we introduced, to tackle these problems. We compared experimentally this set of methods on synthetic and real sequencing data. Some of our methods are substantially more robust to noise than the basic spectral method, but the *de novo* assembly experiments do not support improvement over the spectral extension presented in Chapter 3.

Finally, our last contribution was to introduce the problem of seriation with duplications. It is an extension of the seriation problem, motivated by the assembly of cancer genomes from Hi-C data. We described how structural variations arising in cancer genomes lead to Hi-C frequency matrices which cannot be used for assembly in a standard seriation framework. We then formalized the problem setting, and proposed an alternate projection scheme to tackle it. We evaluated this method on synthetic data modeling a single-chromosome genome with structural variations. Then, we attempted to adapt the method to genomes with multiple chromosomes, and performed additional synthetic experiments.

6.2 Perspectives

The work presented in this manuscript calls for subsequent development, in both applications and theory. Let us list key future work directions.

i **Integration of new seriation methods in a full-assembly pipeline.** In Chapters 3 and 4, we introduced new algorithmic methods for the seriation problem with enhanced robustness, and evaluated them for the layout computation in *de novo* assembly experiments. However, we did not integrate them into the full assembly pipeline presented in Chapter 2. Thus, when testing the methods, we only evaluated ordering produced, but not the eventual output DNA assembly. Since our new methods produced accurately ordered layouts with fewer contigs than the spectral method used in Chapter 2, we can expect the resulting assembly to be of better quality, with a smaller number of mis-assemblies. Still, it would be interesting to quantify how improvements on the layout translate to the consensus produced.

ii **Extensions of Chapter 3.** In Chapter 3, we derived theoretical guarantees for circular seriation, but with more restrictive assumptions than those used for linear seriation in Atkins et al. [1998], which essentially rely on results specific to the ordering relation of real numbers. Moreover, in the linear seriation case, we provided insight about the curve structure of the Laplacian embedding through the study of the spectrum of specific Toeplitz matrices, but we could not generalize to general Robinson matrices. An interesting work direction is to search for theoretical guarantees about circular seriation under milder assumptions on the

input matrices. Another line of work concerns the normalization of the Laplacian (or of the similarity matrix itself). Indeed, we have explored the normalization proposed by Coifman et al. [2008], that roughly normalizes the similarity by the local density of points. It would be interesting to compare this method to normalizations of the similarity matrix able to make it closer to a Toeplitz matrix, such as Sinkhorn-Knopp normalization.

iii **10X Genomics data.** We have mostly dealt with third-generation sequencing data in our experiments. Such data contains long-read (tens of kbp), allowing for an easier resolution of the repeats than with short-reads, where using additional pair-end information is necessary to construct the layout through a scaffolding procedure. A recent development in sequencing technology commercialized by 10X genomics combines short-reads with so-called molecular barcoding, linking short-reads to long molecules to provide long-range information. In short, a barcode is associated to regions of DNA of large length (a few tens of kbp), and two reads that are close to each other on a DNA strand are likely to share several barcodes. This barcoding notably permits to call structural variants and distinguish between haplotypes for diploid genomes (such as the human genome). The Supernova assembler [Weisenfeld et al., 2017] is based on a short-reads assembly scheme, and the additional molecular barcode information is used to disambiguate the scaffolding. It would be of major interest to see whether the barcode information is sufficient to find the layout of the short-reads with seriation, without even computing overlaps. Specifically, we could define a pairwise similarity between reads as the number of barcodes they share, and apply a clustering/reordering method directly on it.

iv **Algorithms to perform Clustering+Seriation.** In Chapter 2, we attempted to assemble eukaryotic genomes with seriation methods. These genomes contain multiple chromosomes. Though, due to repeats occurring in distinct chromosomes, the read-overlap based similarity matrices contain non-zero values between reads coming from distinct chromosomes. We have shown through experiments related to optical mapping that having prior information about the chromosome assignment of the reads (given a read, know to which chromosome it belongs) improved the quality of the assembly. In practice, we do not have such information, and our methods need to fragment the assembly into many contigs to avoid mis-assemblies where contigs from distinct chromosomes are mixed together. Then, in Chapter 3, we conducted experiments with Hi-C frequency matrices having a block structure corresponding to distinct chromosomes. Although the inter-chromosomes similarity is generally smaller than the intra-chromosome similarity for Hi-C data, there are still some high similarity values between chromosomes. On synthetic data where the cluster structure is prominent, our method can naturally split the data in sub-orderings. Indeed, Algorithm 3.3 from Chapter 3 creates a new similarity matrix from a spectral embedding of the data. When there is a clear cluster structure in the data, it translates to the embedding and the new similarity matrix

can be disconnected into connected components corresponding to individual chromosomes. Still, in most cases, with real data involving separate chromosomes, dividing the reads into clusters can be crucial, yet it is not explicitly handled by our methods. An interesting line of future research would be to formulate the task of performing both clustering and ordering simultaneously as an optimization problem over permutations, and try to derive dedicated algorithms. A possible lead could be to follow the approach of Lim and Wright [2016] using the extended formulation of the Permutahedron [Goemans, 2014].

v **Seriation with duplication on real Hi-C data from cancer genomes.** In Chapter 5, we introduce the problem of seriation with duplications in order to assemble genomes with structural variants from Hi-C data. Yet, we only test our method on synthetic data. A key issue arising with real data is related to the previous point, namely that the genome is divided into distinct chromosomes. Still, in seriation with duplications, there is an additional level of complexity compared to regular seriation, for the clustering step also. Indeed, the cluster-structure appears on the hidden similarity matrix S, but not on the observed, cumulative matrix A. In Chapter 5, we consider adding a clustering step to the alternate projections scheme. However, this only enhances the results with few duplications, on synthetic, well-conditioned matrices that are the sum of a block matrix and a Robinson matrix. Regarding the previous item from this list, finding a principled method solving Clustering+Seriation could be used in the alternated projection scheme instead of regular seriation. Also, another issue with our proposed method is its algorithmic complexity. The projection on the set of R-matrices is done through a linear program and do not scale to large similarity matrices. Finally, Hi-C data from cancer genomes have additional structure that is not leveraged here. The structural variants consist of entire blocks of DNA that are duplicated and merged. Within a block, the ordering is the same as for the reference genome. Taking this structure into account would likely improve the method.

Appendix A

Supplementary Material for Chapter 2, Application of the Spectral Method to Genome Assembly

A.1 Running Times

In Table A.1, we give the running time of the methods evaluated in Chapter 2. Figure A.1 focuses on the runtime for the layout method (spectral algorithm) only.

A key *a posteriori* remark is that the implementation of the spectral method whose results are reported here is quite slow for large matrices, as one can see in Figure A.1. The hack proposed here was to use the Julia computing language [Bezanson et al., 2017] for large matrices. However, we noted during the experiments of Chapter 3 that a simpler solution could be used, while keeping all the code in python. Indeed, resorting to the `pyamg` solver instead of `arpack` in the eigenvalue computation of the Laplacian solved the issue and enabled a speedup of an order of magnitude for matrices with $n \sim 10^4$.

A.1.1 Total time

Table A.1 shows the run-time and peak memory for the previously compared methods, when run on a 24 cores Intel Xeon E5-2640 2.50GHz node. Runtime and Max mem correspond to the wall-clock and maximum resident set size fields of the unix `/usr/bin/time -v command`. The first column (Spectral Layout) displays the running time of the layout phase of our method in the following way: time to reorder contigs with the spectral algorithm (total time to get fine-grained layout); the total time for the layout (including the fine-grained computation of the position of the reads on a backbone sequence) is given between parentheses next to the time for the ordering. The second column gives the runtime for our full pipeline, including running minimap to obtain the overlaps. The runtime for Racon includes the time to map the reads

Table A.1: Running time for the different methods on the datasets presented in Section 2.3.1 (Chapter 2)

		Spectral Layout	**Spectral (full, +Minimap)**	**Canu**	**Minimap + Miniasm**	**Racon after Miniasm**	**Racon after Spectral**
A. baylyi ONT R7.3 28x	Runtime [h:mm:ss]	0:00:23 (0:00:59)	0:12:52	0:25:55	0:00:28	0:01:54	0:01:48
	Max mem [Gb]	1.966	1.966	3.827	1.499	0.756	0.484
E. coli ONT R7.3 30x	Runtime [h:mm:ss]	0:00:41 (0:01:25)	0:16:15	0:28:40	0:00:13	0:04:36	0:02:14
	Max mem [Gb]	1.216	1.216	4.655	2.099	0.879	0.645
S. cerevisiae ONT R7.3 68x	Runtime [h:mm:ss]	0:01:41 (0:07:60)	1:41:20	4:33:08	0:01:17	0:21:11	0:21:32
	Max mem [Gb]	12.208	12.208	4.015	8.506	2.376	2.325
S. cerevisiae ONT R9 86x	Runtime [h:mm:ss]	0:03:38 (0:09:28)	2:26:44	7:15:41	0:02:14	0:23:09	0:22:03
	Max mem [Gb]	32.928	32.928	3.986	12.397	2.966	2.775
E. coli PacBio 161x	Runtime [h:mm:ss]	0:05:19 (0:05:44)	1:32:13	0:51:32	0:01:16	0:16:51	0:18:18
	Max mem [Gb]	21.650	21.650	3.770	9.969	8.082	4.619
S. cerevisiae PacBio 127x	Runtime [h:mm:ss]	0:03:11 (0:07:01)	2:59:41	1:50:23	0:02:10	0:20:54	0:23:32
	Max mem [Gb]	32.184	32.184	3.810	16.881	4.290	4.307

to the backbone sequence with Minimap and to run Racon for the consensus (Racon requires a backbone sequence, obtained either with Miniasm or Spectral in the present experiments). Indeed, the Racon pipeline maps the reads to a draft sequence to get the layout and then computes consensus sequences in windows across the genome. Our pipeline instead directly computes the layout and then generates consensus sequences in windows across the genome (the latter task being embarassingly parallel). Canu is faster than our method on the PacBio datasets (probably at least because because we did not adapt our pipeline (as Canu does) to the much higher coverage, nor to the higher fraction of chimeric reads typical of PacBio data), but not on the ONT datasets. The memory for the spectral method can be allocated among several cores.

A.1.2 Runtime for layout only

The running time of the sole spectral method in Figure A.1 aims to show that although our full pipeline is not strikingly fast (as one can see in Table A.1, due to a somewhat naive implementation), the layout itself is fast to compute.

Given the results from Figure A.1, we implemented a call to Julia for matrices of size larger

than 3000 in the code since its eigenvector computation scales better for large matrices but has a non-negligible overhead for small matrices. However, as mentioned earlier, a simpler and more efficient solution is to switch to the `amg` solver instead of `arpack` in the eigenvalue computation (results not shown here).

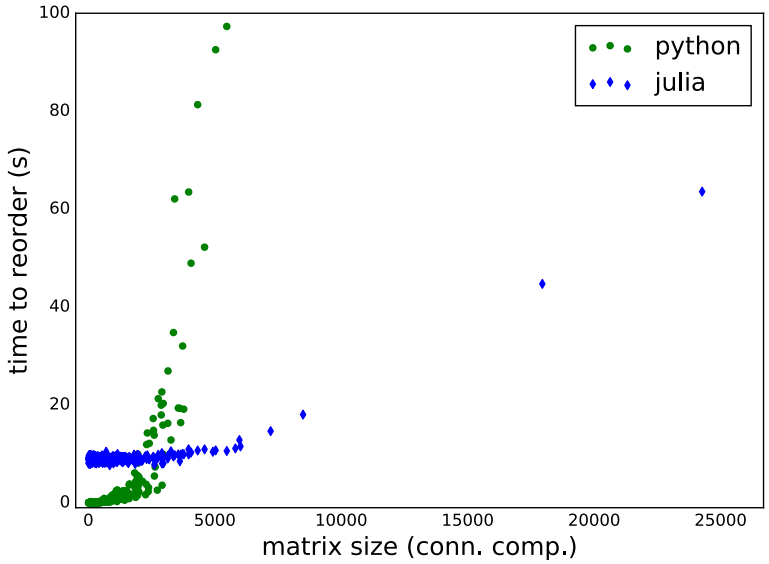
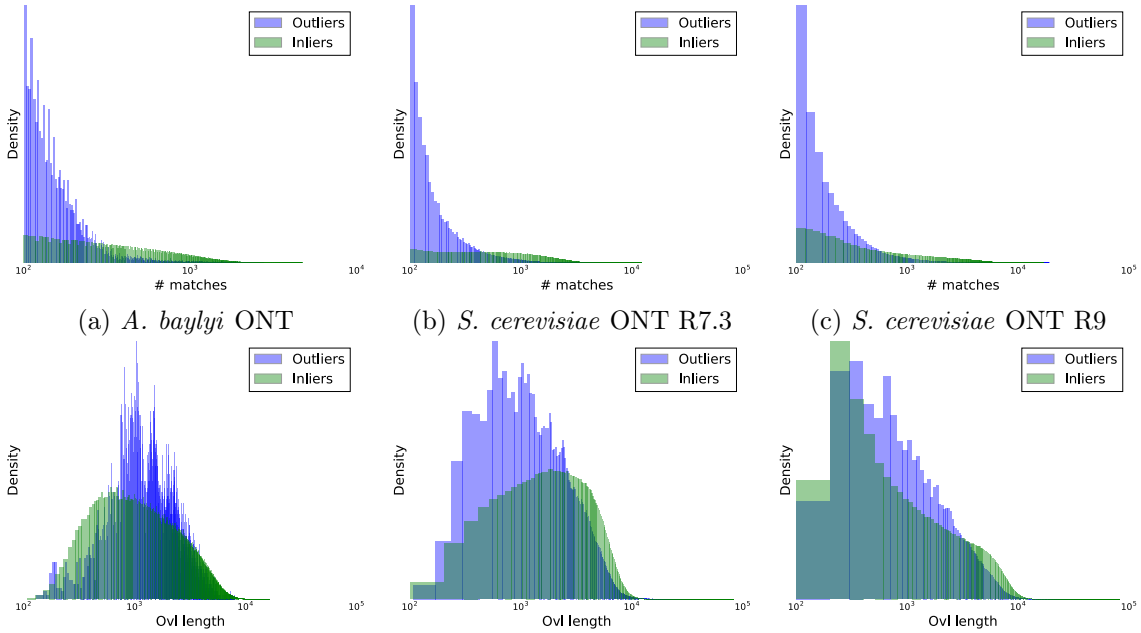


Figure A.1: Runtime of the spectral ordering algorithm in connected components of different sizes (across all datasets), with two solvers for the eigenvalues computations (scipy.sparse.eigsh and the eigs function from Julia [Bezanson et al., 2017]).

A.2 the Bandwidth Heuristic

We present some qualitative and quantitative results to support the bandwidth heuristic.

Figure A.2 shows the distributions of overlap length for the repeat-induced, and the true overlaps. Although these distributions intersect, the true-overlaps distribution has a longer tail, hence long and accurate overlaps are most likely not due to repeats.



(a) *A. baylyi* ONT (b) *S. cerevisiae* ONT R7.3 (c) *S. cerevisiae* ONT R9

(d) *A. baylyi* ONT (e) *S. cerevisiae* ONT R7.3 (f) *S. cerevisiae* ONT R9

Figure A.2: Histograms of overlap scores [number of matches from minimap] (a-c) and overlap lengths (d-f) for the ONT datasets, for outliers (blue) and inliers (green). The x-axis is in log scale. The mapping of the reads to the reference genome with GraphMap was used to label inliers and outliers.

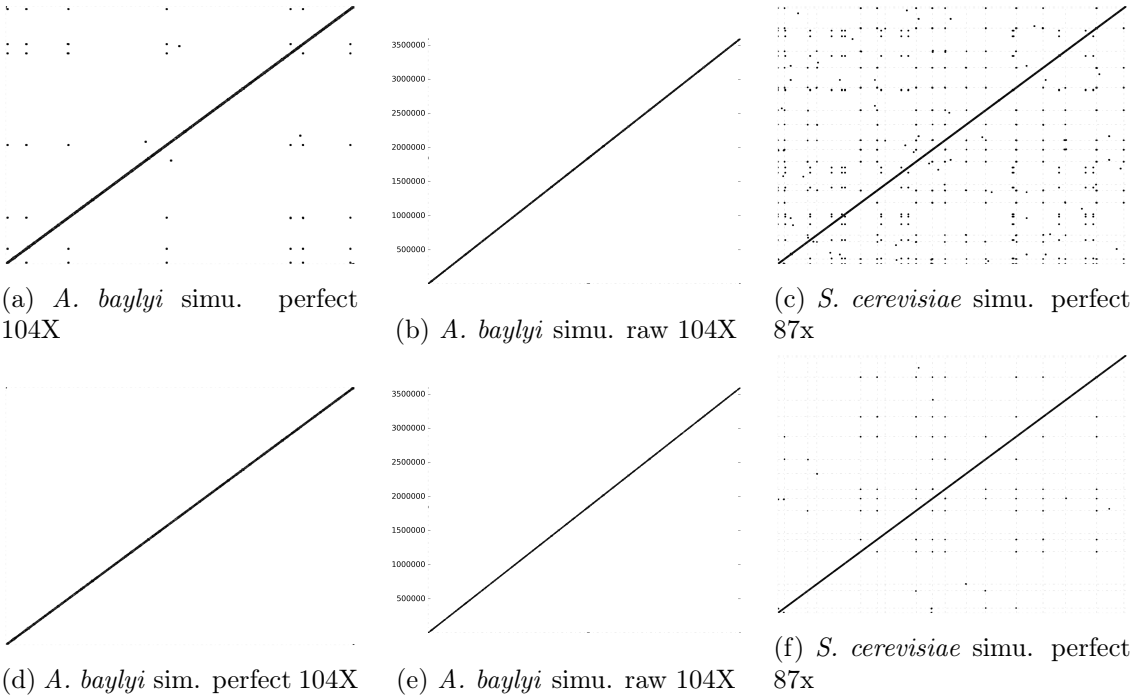
Figure A.3 shows the locations of non-zero values for simulated similarity matrices thresholded at several values. It illustrates that setting a threshold on the overlap score removes outliers.

Subfigures A.3a and A.3d (respectively A.3c and A.3f) represent the similarity for reads generated with NanoSim from the *A. baylyi* ONT R7.3 (respectively *S. cerevisiae* ONT R9) dataset with option –perfect, which means these synthetic reads follow the same length distribution than the original dataset, but have no errors, and have the coverage specified above. The matrices A.3b and A.3e were generated from the *A. baylyi* ONT R7.3 dataset without the –perfect option, which means they have the same length and error distribution than the original data, but with higher coverage.

For perfect and noisy synthetic *A. baylyi* reads and with sufficient coverage, all outliers could be removed by thresholding while keeping a connected similarity graph (all matrices in

the Figure are connected). On the other hand, the similarity matrix generated with *S. cerevisiae* perfect reads still harbors a few outliers after removing 90% of the overlaps (with lowest score).

When increasing the threshold value, the connectivity within some individual chromosomes will be broken before all outliers have been removed. Additional structural information (as used in Canu or Miniasm) will be required to resolve repeats in such situations.



(a) *A. baylyi* simu. perfect 104X

(b) *A. baylyi* simu. raw 104X

(c) *S. cerevisiae* simu. perfect 87x

(d) *A. baylyi* sim. perfect 104X

(e) *A. baylyi* simu. raw 104X

(f) *S. cerevisiae* simu. perfect 87x

Figure A.3: Ordered similarity matrices for simulated datasets after removing 50% of the overlaps (a-c) or 90% (d-f). The reads were simulated with NanoSim [Yang et al., 2017a], from the *A. baylyi* ONT R7.3 and *S. cerevisiae* ONT R9 datasets.

Finally, Figure A.4 illustrates the bandwidth heuristic. It shows that an input similarity matrix containing outliers imparts a reordered matrix with a large bandwidth (and an incorrect reordering). In this experiment, the bandwidth is about 50 times as large as in the absence of outliers. This significant gap (an order of magnitude difference) between the bandwidth of the matrix reordered with the spectral algorithm depending on whether the original matrix (ordered by increasing position of the reads) contained outliers (*i.e.*, is band-diagonal) or not motivated the development of the heuristic for assessing the ordering found by the spectral algorithm, as explained in 2.2.3 (Chapter 2). However, this heuristic is not applicable when the size of the similarity matrix is small. For instance, if the matrix is of size 100, the bandwidth cannot exceed 100 and the use of the heuristic is precluded.

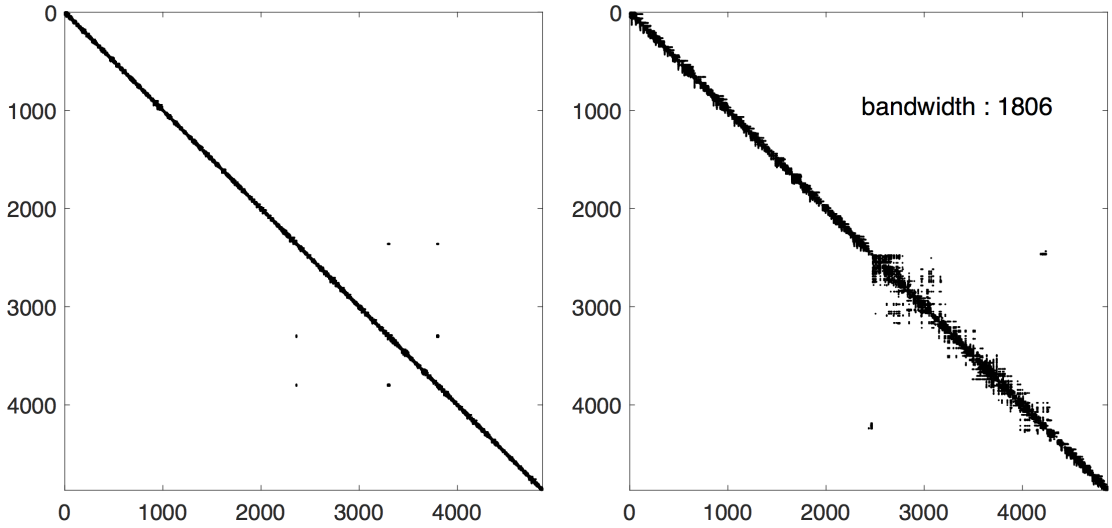
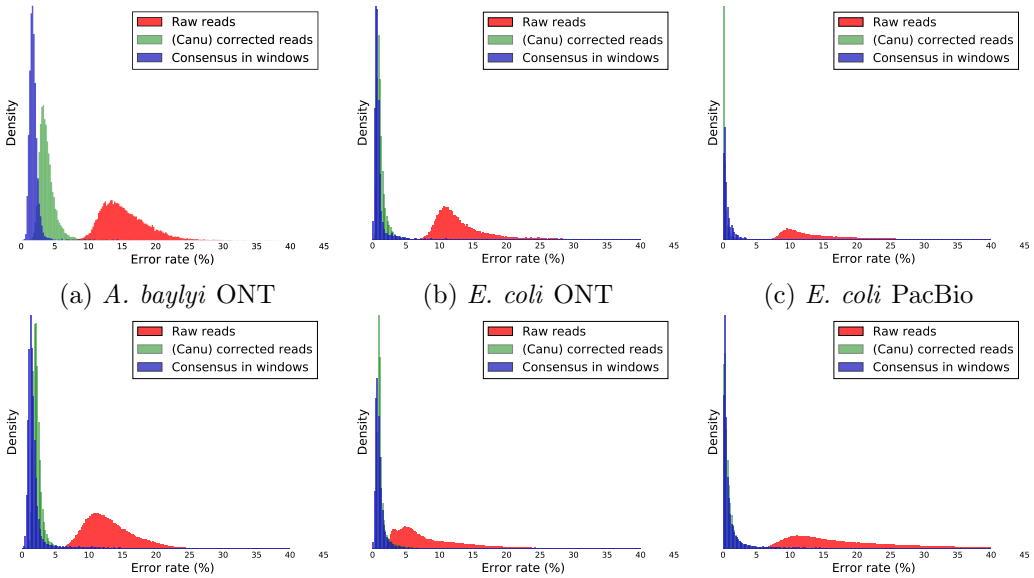


Figure A.4: Similarity matrices containing outliers, displayed with true ordering (obtained by mapping the reads to the reference genome with GraphMap) and generated with a subset of *A. baylyi* ONT NanoSim perfect reads A.4a, and the same matrix incorrectly reordered with the spectral algorithm A.4b.

A.3 Consensus accuracy



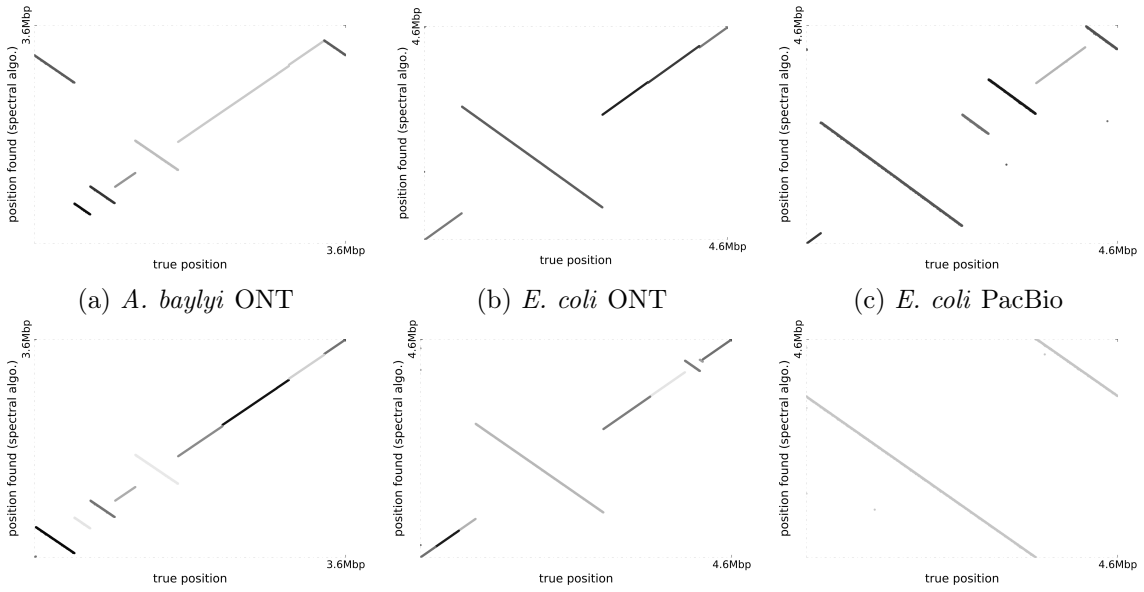
(a) *A. baylyi* ONT (b) *E. coli* ONT (c) *E. coli* PacBio

(d) *S. cerevisiae* ONT R7.3 (e) *S. cerevisiae* ONT R9 (f) *S. cerevisiae* PacBio

Figure A.5: Error-rates in consensus windows, raw reads and corrected reads for the six real datasets.

Figure A.5 is the analog of Figure 2.8 in Chapter 2. It shows the error rate in the raw reads, in the reads corrected with Canu, and in the consensus windows. With ONT R7.3 data, the consensus produced by our pipeline appears more accurate than via the correction module of Canu, while the contrary is true for PacBio data.

A.4 Additional Assembly Results

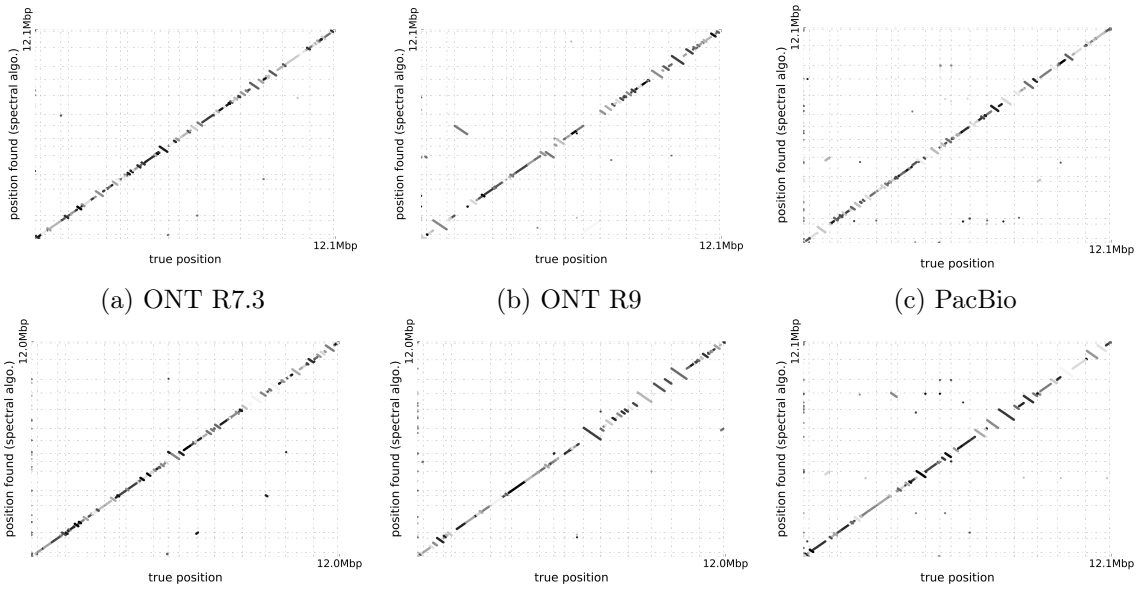


(a) *A. baylyi* ONT (b) *E. coli* ONT (c) *E. coli* PacBio

(d) *A. baylyi* ONT corr. (e) *E. coli* ONT corr. (f) *E. coli* PacBio corr.

Figure A.6: Ordering of the reads computed with the spectral algorithm vs true ordering (obtained by mapping the reads to the reference genome with GraphMap) for the original (a-c) and corrected (d-f) bacterial datasets. All contigs are artificially displayed on the same plot for compactness.

Figures A.6 and A.7 show the layout obtained with the spectral method for all datasets (only two of them are displayed in Chapter 2). It includes the corrected datasets (obtained by using the correction module of canu on the raw reads). The correction slightly improves the layout for the yeast genomes.



(a) ONT R7.3 (b) ONT R9 (c) PacBio

(d) ONT R7.3 corr. (e) ONT R9 corr. (f) PacBio corr.

Figure A.7: Ordering of the reads computed with the spectral algorithm vs true ordering (obtained by mapping the reads to the reference genome with GraphMap) for the original (a-c) and corrected (d-f) yeast (*S. cerevisiae*) datasets. All contigs are artificially displayed on the same plot for compactness. The dashed lines represent the boundaries between chromosomes.

Table A.2: Assembly results of several assemblers across the datasets **corrected** with Canu

		Miniasm	Spectral	Canu	Miniasm+Racon	Miniasm+Racon (2 iter.)	Spectral+Racon
A. baylyi ONT R7.3 28x (26x)	Ref. size [bp]	3598621	3598621	3598621	3598621	3598621	3598621
	Total bases [bp]	3493724	3523055	3516777	3540178	3540766	3522315
	Ref. chr. [#]	1	1	1	1	1	1
	Contigs [#]	5	2 (9)	2	5	5	2 (9)
	Aln. bp ref [bp]	3594663 (99.89%)	3596069 (99.93%)	3595264 (99.91%)	3595193 (99.90%)	3595193 (99.90%)	3596269 (99.93%)
	Aln. bp query [bp]	3492976 (99.98%)	3522804 (99.99%)	3516440 (99.99%)	3539856 (99.99%)	3540444 (99.99%)	3522311 (100.00%)
	Misassemblies [#]	2	1	2	2	2	1
	Avg. identity	96.40	**97.87**	97.61	97.79	97.85	97.86
E. coli ONT R7.3 30x (27x)	Ref. size [bp]	4641652	4641652	4641652	4641652	4641652	4641652
	Total bases [bp]	4597538	4613973	4627578	4617120	4617100	4613521
	Ref. chr. [#]	1	1	1	1	1	1
	Contigs [#]	3	1 (8)	2	3	3	1 (8)
	Aln. bp ref [bp]	4639179 (99.95%)	4639815 (99.96%)	4639396 (99.95%)	4639355 (99.95%)	4639355 (99.95%)	4639420 (99.95%)
	Aln. bp query [bp]	4597389 (100.00%)	4613972 (100.00%)	4627577 (100.00%)	4617119 (100.00%)	4617099 (100.00%)	4613520 (100.00%)
	Misassemblies [#]	2	2	4	2	2	2
	Avg. identity	98.89	**99.43**	99.41	99.42	**99.43**	**99.43**
S. cerevisiae ONT R7.3 68x (38x)	Ref. size [bp]	12157105	12157105	12157105	12157105	12157105	12157105
	Total bases [bp]	11814836	11959669	12112186	11877015	11876882	11949674
	Ref. chr. [#]	17	17	17	17	17	17
	Contigs [#]	29	67 (126)	37	28	28	67 (126)
	Aln. bp ref [bp]	12061456 (99.21%)	11963869 (98.41%)	12068379 (99.27%)	12062161 (99.22%)	12061809 (99.22%)	11969742 (98.46%)
	Aln. bp query [bp]	11814252 (100.00%)	11930637 (99.76%)	12069253 (99.65%)	11876268 (99.99%)	11876225 (99.99%)	11925068 (99.79%)
	Misassemblies [#]	19	22	26	20	20	24
	Avg. identity	97.81	98.32	98.36	**98.39**	**98.39**	98.38
S. cerevisiae ONT R9 86x (40x)	Ref. size [bp]	12157105	12157105	12157105	12157105	12157105	12157105
	Total bases [bp]	11946760	12081487	12184545	11970672	11970529	12061759
	Ref. chr. [#]	17	17	17	17	17	17
	Contigs [#]	21	65 (108)	30	20	20	65 (108)
	Aln. bp ref [bp]	12055448 (99.16%)	11851023 (97.48%)	12110461 (99.62%)	12056562 (99.17%)	12056734 (99.17%)	11879607 (97.72%)
	Aln. bp query [bp]	11944969 (99.99%)	12043650 (99.69%)	12184122 (100.00%)	11970041 (99.99%)	11969729 (99.99%)	12040521 (99.82%)
	Misassemblies [#]	21	32	26	22	22	38
	Avg. identity	98.83	98.90	**99.06**	**99.06**	99.05	99.04
E. coli PacBio 161x (38x)	Ref. size [bp]	4641652	4641652	4641652	4641652	4641652	4641652
	Total bases [bp]	4642736	4663427	4670125	4642423	4642443	4662179
	Ref. chr. [#]	1	1	1	1	1	1
	Contigs [#]	1	1 (1)	1	1	1	1 (1)
	Aln. bp ref [bp]	4639048 (99.94%)	4640514 (99.98%)	4641652 (100.00%)	4641623 (100.00%)	4641616 (100.00%)	4641652 (100.00%)
	Aln. bp query [bp]	4639955 (99.94%)	4662891 (99.99%)	4670125 (100.00%)	4642423 (100.00%)	4642443 (100.00%)	4662172 (100.00%)
	Misassemblies [#]	2	4	4	4	4	4
	Avg. identity	99.59	99.97	**99.99**	**99.99**	**99.99**	**99.99**
S. cerevisiae PacBio 127x (37x)	Ref. size [bp]	12157105	12157105	12157105	12157105	12157105	12157105
	Total bases [bp]	12174558	12232964	12346261	12194786	12193481	12217702
	Ref. chr. [#]	17	17	17	17	17	17
	Contigs [#]	26	55 (86)	29	26	26	55 (86)
	Aln. bp ref [bp]	12036689 (99.01%)	12008560 (98.78%)	12091871 (99.46%)	12042104 (99.05%)	12041381 (99.05%)	12018488 (98.86%)
	Aln. bp query [bp]	12151704 (99.81%)	12179852 (99.57%)	12304982 (99.67%)	12177020 (99.85%)	12175701 (99.85%)	12172316 (99.63%)
	Misassemblies [#]	74	75	76	76	76	80
	Avg. identity	99.22	99.78	99.87	**99.88**	**99.88**	99.86

Table A.2 provides the assembly results obtained by using the reads corrected by Canu's correction module. These corrected datasets were obtained by running Canu with the `saveReadCorrections=True` option on the datasets presented in 2.3.1. Canu includes correction and trimming, resulting in a removal of short reads and a lower coverage than in the original raw data. However, it is the coverage of the raw dataset which is relevant since higher coverage in the latter will result in longer reads in the corrected data, even though the coverage in all corrected datasets are roughly below 40x. We indicate the coverage of the corrected datasets in parentheses next to the coverage of the original dataset. For the spectral method, we give the results after the contig merging step (see 2.3.3). The number of contigs before this post-processing is given between parentheses. Unlike with raw data, the polishing effect of adding Racon to our pipeline is not significant. All methods have comparable results on the corrected datasets. The best result in terms of average identity only is indicated in bold (but other metrics should also be used to compare the assemblies).

Table A.3 is a mis-assembly report obtained with QUAST [Gurevich et al., 2013] (only a subset of the report is shown). Given the accuracy of the Miniasm assembly, it is likely that the zeros in the Miniasm column are due to the fact that the algorithm failed to correctly match the sequences, rather than the absence of misassemblies. On all ONT datasets, the Spectral and Spectral+Racon methods are among those yielding the least global misassemblies (relocation, translocation or inversions).

Table A.4 is the analog of Table A.3 for the corrected datasets. We observe that the number of local misassemblies is smaller than with the uncorrected data, but the number of global ones is not. None of the assemblers has a significantly smaller or larger number of misassemblies compared to the others.

A.5 Implementation and reproducibility

Spectrassembler is implemented in python and available on https://github.com/antrec/spectrassembler with a usage example of how to reproduce the results obtained with *E. coli* ONT data. We used the following software :

- SPOA - https://github.com/rvaser/spoa
- Minimap - https://github.com/lh3/minimap
- Miniasm - https://github.com/lh3/miniasm
- Canu v1.4 - https://github.com/marbl/canu
- Racon - https://github.com/isovic/racon
- MUMmer's DNAdiff version 1.2, NUCmer version 3.07 - http://mummer.sourceforge.net/

- QUAST - https://sourceforge.net/projects/quast/files/
- GraphMap - https://github.com/isovic/GraphMap
- `errorrates.py` from samscripts - https://github.com/isovic/samscripts
- NanoSim - https://github.com/bcgsc/NanoSim

SPOA is used in our pipeline for performing multiple sequence alignment. For generating the consensus in windows, it was run with the options : `-l 2 -r 0 -x -3 -o -5 -e -2` (semi-global alignment with custom gap and mismatch penalties). minimap was run with options `-Sw5 -L100 -m0 -t12` (long reads specific values and multithreading with 12 threads). miniasm was run with default parameters when used as a comparative method. Canu was run with saveReadCorrections=True option and data specifications (e.g., `genomeSize=3.6m -nanopore-raw`). Racon was run with the alignment generated with minimap (to map the draft assembly, either from miniasm or from our pipeline) with default parameters. GraphMap [Sović et al., 2016] was used to generate alignment between the reads and the reference genome in order to have the position of the reads and their error rate (which was computed with the script errorrates.py). DNAdiff and QUAST were used to evaluate the assemblies. To concatenate the contigs obtained with our method, we extracted their ends (end length used : 35kbp) and used minimap with options `-Sw5 -L500` to compute overlaps between them, and ran miniasm with options `-1 -2 -e 0 -c 0 -r 1,0` (no pre-selection, no cutting small unitigs, no overlap drop). The related script is available in the tools folder of our GitHub code. We also publish the other scripts we used (although they may be poorly written and undocumented), including our implementation of the optical mapping algorithm of Nagarajan et al. [2008], in the tools folder.

Table A.3: Misassemblies report of the different assemblers across the various datasets

		Miniasm	Spectral	Canu	Miniasm+ Racon	Miniasm+ Racon (x2)	Spectral+ Racon
A. baylyi ONT R7.3 28x	Relocations [#]	0	0	2	2	2	0
	Translocations [#]	0	0	0	0	0	0
	Inversions [#]	0	0	0	0	0	0
	Missmbld. contigs [#]	0	0	1	1	1	0
	Missmbld. contigs length [bp]	0	0	3513432	1993457	1994286	0
	Local misassemblies [#]	0	7	5	0	0	0
	Mismatches [#]	0	0	0	0	0	0
	Indels [#]	0	0	0	0	0	0
	Indels length [bp]	0	0	0	0	0	0
E. coli ONT R7.3 30x	Relocations [#]	0	2	6	3	3	2
	Translocations [#]	0	0	0	0	0	0
	Inversions [#]	0	0	2	0	0	0
	Missmbld. contigs [#]	0	1	2	2	2	1
	Missmbld. contigs length [bp]	0	2160837	4625543	3743081	3740186	2148788
	Local misassemblies [#]	0	50	2	2	2	3
	Mismatches [#]	0	55	0	0	0	0
	Indels [#]	0	1	1	0	0	0
	Indels length [bp]	0	30	1	0	0	0
S. cerevisiae ONT R7.3 68x	Relocations [#]	0	0	17	6	7	1
	Translocations [#]	0	7	17	12	12	10
	Inversions [#]	0	0	0	0	0	0
	Missmbld. contigs [#]	0	7	16	11	11	10
	Missmbld. contigs length [bp]	0	1223452	4852688	4638491	4638515	909031
	Local misassemblies [#]	0	57	17	9	10	12
	Mismatches [#]	0	63	0	0	0	0
	Indels [#]	0	3	2	3	2	1
	Indels length [bp]	0	90	124	167	132	54
S. cerevisiae ONT R9 86x	Relocations [#]	0	5	22	9	9	4
	Translocations [#]	0	18	17	9	10	32
	Inversions [#]	0	0	0	0	0	0
	Missmbld. contigs [#]	0	11	11	10	11	10
	Missmbld. contigs length [bp]	0	3149392	5957900	4545988	4563372	2661541
	Local misassemblies [#]	0	41	88	11	11	30
	Mismatches [#]	0	0	0	0	0	0
	Indels [#]	0	2	4	3	3	2
	Indels length [bp]	0	161	250	208	207	157
E. coli PacBio 161x	Relocations [#]	0	3	2	2	2	2
	Translocations [#]	0	0	0	0	0	0
	Inversions [#]	0	2	2	2	2	2
	Missmbld. contigs [#]	0	1	1	1	1	1
	Missmbld. contigs length [bp]	0	2848876	4670125	4653228	4645420	2818134
	Local misassemblies [#]	0	66	2	3	2	2
	Mismatches [#]	0	0	0	0	0	0
	Indels [#]	0	0	0	1	0	0
	Indels length [bp]	0	0	0	66	0	0
S. cerevisiae PacBio 127x	Relocations [#]	0	17	31	21	20	18
	Translocations [#]	0	40	44	39	38	50
	Inversions [#]	0	0	1	1	1	0
	Missmbld. contigs [#]	0	28	24	22	21	31
	Missmbld. contigs length [bp]	0	6470761	10214689	9569247	9421896	6683508
	Local misassemblies [#]	0	157	26	42	30	33
	Mismatches [#]	0	0	0	5	0	0
	Indels [#]	0	3	8	9	6	2
	Indels length [bp]	0	132	260	416	245	78

Table A.4: Misassemblies report of the different assemblers across the datasets corrected with Canu

		Miniasm	Spectral	Canu	Miniasm+ Racon	Miniasm+ Racon (x2)	Spectral+ Racon
A. baylyi ONT R7.3 28x (26x)	Relocations [#]	2	1	2	2	2	1
	Translocations [#]	0	0	0	0	0	0
	Inversions [#]	0	0	0	0	0	0
	Missmbld. contigs [#]	1	1	1	1	1	1
	Missmbld. contigs length [bp]	1949981	3245660	2802152	1976843	1977319	3244955
	Local misassemblies [#]	4	1	3	2	1	0
	Mismatches [#]	0	0	0	0	0	0
	Indels [#]	0	0	0	0	0	0
	Indels length [bp]	0	0	0	0	0	0
E. coli ONT R7.3 30x (27x)	Relocations [#]	2	2	2	2	2	2
	Translocations [#]	0	0	0	0	0	0
	Inversions [#]	0	0	2	0	0	0
	Missmbld. contigs [#]	1	1	2	1	1	1
	Missmbld. contigs length [bp]	3945897	4613973	4627578	3962753	3962721	4613521
	Local misassemblies [#]	5	2	2	2	2	2
	Mismatches [#]	58	0	0	77	77	77
	Indels [#]	3	1	1	2	2	2
	Indels length [bp]	13	1	1	2	2	2
S. cerevisiae ONT R7.3 68x (38x)	Relocations [#]	6	7	14	7	6	9
	Translocations [#]	13	15	12	13	14	15
	Inversions [#]	0	0	0	0	0	0
	Missmbld. contigs [#]	11	15	14	11	11	15
	Missmbld. contigs length [bp]	5025689	2643657	2808407	5053047	5052895	2634865
	Local misassemblies [#]	12	26	10	6	7	10
	Mismatches [#]	21	0	0	0	0	0
	Indels [#]	3	1	1	3	1	1
	Indels length [bp]	122	78	78	235	78	78
S. cerevisiae ONT R9 86x (40x)	Relocations [#]	11	7	13	11	11	8
	Translocations [#]	10	25	13	11	11	30
	Inversions [#]	0	0	0	0	0	0
	Missmbld. contigs [#]	10	12	12	9	9	13
	Missmbld. contigs length [bp]	4954988	3199985	3534917	4573865	4573600	3361506
	Local misassemblies [#]	12	58	8	9	10	16
	Mismatches [#]	55	0	0	0	0	0
	Indels [#]	1	0	1	1	1	0
	Indels length [bp]	7	0	54	54	54	0
E. coli PacBio 161x (38x)	Relocations [#]	2	2	2	2	2	2
	Translocations [#]	0	0	0	0	0	0
	Inversions [#]	0	2	2	2	2	2
	Missmbld. contigs [#]	1	1	1	1	1	1
	Missmbld. contigs length [bp]	4642736	4663427	4670125	4642423	4642443	4662179
	Local misassemblies [#]	13	5	2	2	2	3
	Mismatches [#]	0	0	0	0	0	0
	Indels [#]	0	0	0	0	0	0
	Indels length [bp]	0	0	0	0	0	0
S. cerevisiae PacBio 127x (37x)	Relocations [#]	29	22	31	33	33	24
	Translocations [#]	44	52	44	42	42	56
	Inversions [#]	1	1	1	1	1	0
	Missmbld. contigs [#]	22	33	24	22	22	34
	Missmbld. contigs length [bp]	10163939	9816851	10214692	10180811	10178266	9840033
	Local misassemblies [#]	49	59	26	24	25	28
	Mismatches [#]	28	0	0	0	0	0
	Indels [#]	8	6	8	5	6	7
	Indels length [bp]	462	216	260	147	153	222

Appendix B

Supplementary Material for Chapter 3, Multi-dimensional Spectral Ordering : Reconstructing Linear Orderings via Spectral Embedding

Notation: We will commonly denote σ a permutation of $\{1, \ldots, n\}$ and $\mathfrak{S}$ the set of all such permutations. When represented matricially, σ will often be noted Π while cyclic permutation of $\{1, \ldots, n\}$ will be noted as τ. A will usually denote the matrix of raw pair-wise similarities. S will denote the similarity matrix resulting from Algorithm 3.3, and k a neighboring parameter. Finally we use indexed version ν (resp., λ) to denote eigenvalues of a similarity matrix (resp. a graph Laplacian).

B.1 Additional Algorithms

B.1.1 Merging connected components

The new similarity matrix S computed in Algorithm 3.3 is not necessarily the adjacency matrix of a connected graph, even when the input matrix A is. For instance, when the number of nearest neighbors k is low and the points in the embedding are non uniformly sampled along a curve, S may have several, disjoint connected components (let us say there are C of them in the following). Still, the baseline Algorithm 3.1 requires a connected similarity matrix as input. When S is disconnected, we run 3.1 separately in each of the C components, yielding C sub-orderings instead of a global ordering.

However, since A is connected, we can use the edges of A between the connected components to merge the sub-orderings together. Specifically, given the C ordered subsequences, we build a meta similarity matrix between them as follows. For each pair of ordered subsequences (c_i, c_j),

we check whether the elements in one of the two ends of c_i have edges with those in one of the two ends of c_j in the graph defined by A. According to that measure of similarity and to the direction of these meta-edges (*i.e.*, whether it is the beginning or the end of c_i and c_j that are similar), we merge together the two subsequences that are the closest to each other. We repeat this operation with the rest of the subsequences and the sequence formed by the latter merge step, until there is only one final sequence, or until the meta similarity between subsequences is zero everywhere. We formalize this procedure in the greedy Algorithm B.1, which is implemented in the package at https://github.com/antrec/mdso.

Given C reordered subsequences (one per connected component of S) $(c_i)_{i=1,\dots,C}$, that form a partition of $\{1,\dots,n\}$, and a window size h that define the length of the ends we consider (h must be smaller than half the smallest subsequence), we denote by c_i^- (resp. c_i^+) the first (resp. the last) h elements of c_i, and $a(c_i^\epsilon, c_j^{\epsilon'}) = \sum_{u\in c_i^\epsilon, v\in c_j^{\epsilon'}} A_{uv}$ is the similarity between the ends c_i^ϵ and $c_j^{\epsilon'}$, for any pair c_i, c_j, $i \neq j \in \{1,\dots,C\}$, and any combination of ends $\epsilon, \epsilon' \in \{+,-\}$. Also, we define the meta-similarity between c_i and c_j by,

$$s(c_i, c_j) \triangleq \max(a(c_i^+, c_j^+), a(c_i^+, c_j^-), a(c_i^-, c_j^+), a(c_i^-, c_j^-)) \ , \tag{B.1}$$

and $(\epsilon_i, \epsilon_j) \in \{+,-\}^2$ the combination of signs where the argmax is realized, *i.e.*, such that $s(c_i, c_j) = a(c_i^{\epsilon_i}, c_j^{\epsilon_j})$. Finally, we will use $\bar{c}_i$ to denote the ordered subsequence c_i read from the end to the beginning, for instance if $c = (1,\dots,n)$, then $\bar{c} = (n,\dots,1)$.

Algorithm B.1 Merging connected components

Input: C ordered subsequences forming a partition $P = (c_1, \dots, c_C)$ of $\{1, \dots, n\}$, an initial similarity matrix A, a neighborhood parameter h.

1: **while** $C > 1$ **do**
2: Compute meta-similarity $\tilde{S}$ such that $\tilde{S}_{ij} = s(c_i, c_j)$, and meta-orientation (ϵ_i, ϵ_j), for all pairs of subsequences with equation B.1.
3: **if** $\tilde{S} = 0$ **then**
4: break
5: **end if**
6: find $(i, j) \in \operatorname{argmax} \tilde{S}$, and (ϵ_i, ϵ_j) the corresponding orientations.
7: **if** $(\epsilon_i, \epsilon_j) = (+, -)$ **then**
8: $c^{\text{new}} \leftarrow (c_i, c_j)$
9: **else if** $(\epsilon_i, \epsilon_j) = (+, +)$ **then**
10: $c^{\text{new}} \leftarrow (c_i, \bar{c}_j)$
11: **else if** $(\epsilon_i, \epsilon_j) = (-, -))$ **then**
12: $c^{\text{new}} \leftarrow (\bar{c}_i, c_j)$
13: **else if** $(\epsilon_i, \epsilon_j) = (-, +))$ **then**
14: $c^{\text{new}} \leftarrow (\bar{c}_i, \bar{c}_j)$
15: **end if**
16: Remove c_i and c_j from P.
17: Add c^{new} to P.
18: $C \leftarrow C - 1$
19: **end while**

Output: Total reordered sequence c^{final}, which is a permutation if $C = 1$ or a set of reordered subsequences if the loop broke at line 5.

B.1.2 Computing Kendall-Tau score between two permutations describing a circular ordering

Suppose we have data having a circular structure, *i.e.*, we have n items that can be laid on a circle such that the higher the similarity between two elements is, the closer they are on the circle. Then, given an ordering of the points that respects this circular structure (*i.e.*, a solution to Circular Seriation), we can shift this ordering without affecting the circular structure. For instance, in Figure B.1, the graph has a $\mathcal{C}_R$ affinity matrix whether we use the indexing printed in black (outside the circle), or a shifted version printed in purple (inside the circle). Therefore, we transpose the Kendall-Tau score between two permutations to the case where we want to compare the two permutations up to a shift with Algorithm B.2

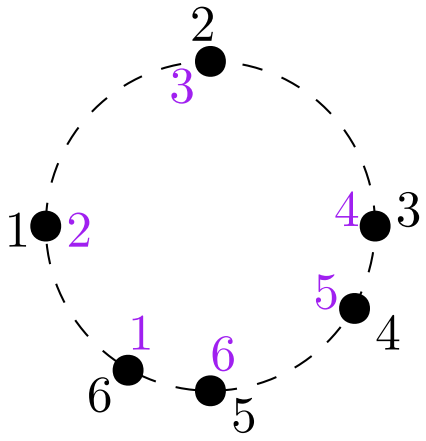


Figure B.1: Illustration of the shift-invariance of permutations solution to a Circular Seriation problem.

Algorithm B.2 Comparing two permutation defining a circular ordering

Input: Two permutations vectors of size n, $\sigma = \big(\sigma(1), \dots, \sigma(n)\big)$ and $\pi = \big(\pi(1), \dots, \pi(n)\big)$

1: **for** $i = 1$ **to** n **do**

2: $\quad KT(i) \leftarrow$ Kendall-Tau$\big(\sigma, \big(\pi(i), \pi(i+1), \dots, \pi(n), \pi(1), \dots, \pi(i-1)\big)\big)$

3: **end for**

4: best score $\leftarrow \max_{i=1,\dots,n} KT(i)$

Output: best score

B.2 Additional Numerical Results

B.2.1 Genome assembly experiment (detailed)

Here we provide details about the application of seriation methods for genome assembly and details about our experiment. We used the *E. coli* reads from Loman et al. [2015]. They were sequenced with Oxford Nanopore Technology (ONT) MinION device. The sequencing experiment is detailed in `http://lab.loman.net/2015/09/24/first-sqk-map-006-experiment` where the data is available. We also used the *A. baylyi* dataset sequenced at the Genoscope, introduced in Chapter 2, Section 2.3.1. The overlaps between raw reads were computed with minimap2 [Li, 2018] with the ONT preset (`minimap2 -x ava ont`). The similarity matrix was constructed directly from the output of minimap2. For each pair (i, j) of reads where an overlap was found, we let the number of matching bases be the similarity value associated (and zero where no overlap are found). The only preprocessing on the matrix is that we set a threshold to remove short overlaps. In practice we set the threshold to the median of the similarity values, *i.e.*, we discard the lower half of the overlaps. We then apply our method to the similarity matrix. The laplacian embedding is shown in Figure B.2a. We used no scaling of the Laplacian

as it corrupted the filamentary structure of the embedding, but we normalized the similarity matrix beforehand with $W \leftarrow D^{-1}WD^{-1}$ as in Coifman and Lafon [2006]. The resulting similarity matrix S computed from the embedding in Algorithm 3.3 is disconnected. Then, Algorithm 3.1 is applied in each connected component, yielding a fragmented assembly with correctly ordered contigs, as shown in Figure B.2b. However, if the new similarity matrix S is disconnected, the input matrix A is connected. The fragmentation happened while "scanning" the nearest-neighbors from the embedding. One can therefore merge the ordered contigs using the input matrix A as follows. For each contig, we check from A if there are non-zero overlaps between reads at the edges of that contig and some reads at the edges of another contig. If so, we merge the two contigs, and repeat the procedure until there is only one contig left (or until there is no more overlaps between edges from any two contigs). This procedure is detailed in Algorithm B.1. Note that the *E. coli* genome is circular, therefore computing the layout should be casted as a Circular Seriation problem, as illustrated in Figure 3.1. Yet, since the genome is fragmented in subsequences since S is disconnected, we end up using Algorithm 3.1 in each connected component, *i.e.*, solving an instance of Linear Seriation in each contig.

The experiment can be reproduced with the material on `https://github.com/antrec/mdso`, and the parameters easily varied. Overall, the final ordering found is correct when the threshold on the overlap-based similarity is sufficient (in practice, above $\sim$ 50% of the non-zero values for *E. coli*, and $\sim$ 70% for *A. baylyi*). When the threshold increases or when the number of nearest neighbors k from Algorithm 3.3 decreases, the new similarity matrix S gets more fragmented, but the final ordering remains the same after the merging procedure, except for very large values where many reads end up with no overlap with any other read. A good heuristic to choose the threshold value is to take the highest value that leaves the resulting merged component contiguous (and of size comparable to the number of input reads, meaning that few reads lost all their overlaps/edges in the thresholding procedure).

B.2.2 Gain over baseline

In Figure 3.5, each curve is the mean of the Kendall-tau (a score directly interpretable by practitioners) over many different Gaussian random realizations of the noise. The shaded confidence interval represents the area in which the true expectation is to be with high probability but not the area in which the score of an experiment with a given noisy similarity would be. As mentioned in the main text, the shaded interval is the standard deviation divided by $\sqrt{n_{\text{exps}}}$, since otherwise the plot was hard to read, as the intervals crossed each others.

Practitioners may use this method in one-shot (e.g. for one particular data-set). In that case, it would be more relevant to show directly the standard deviation on the plots, which is the same as what is displayed, but multiplied by 10. Then, the confidence intervals between the baseline and our method would cross each other. However, the standard deviation on all experiments is due to the fact that some instances are more difficult to solve than some others.

On the difficult instances, the baseline and our method perform more poorly than on easy instances. However, we also computed the gain over the baseline, *i.e.*, the difference of score between our method and the baseline, for each experiment, and it is always, or almost always positive, *i.e.*, our method almost always beats the baseline although the confidence intervals cross each other.

B.2.3 Numerical results with KMS matrices

In Figure B.3 we show the same plots as in Section 3.6 but with matrices A such that $A_{ij} = e^{\alpha|i-j|}$, with $\alpha = 0.1$ and $n = 500$.

B.2.4 Sensitivity to parameter k (number of neighbors)

Here we show how our method performs when we vary the parameter k (number of neighbors at step 4 of Algorithm 3.3), for both linearly decrasing, banded matrices, $A_{ij} = \max\left(c - |i-j|, 0,\right)$ (as in Section 3.6), in Figure B.4 and with matrices A such that $A_{ij} = e^{\alpha|i-j|}$, with $\alpha = 0.1$ (Figure B.5.

We observe that the method performs roughly equally well with k in a range from 5 to 20, and that the performances drop when k gets too large, around $k = 30$. This can be interpreted as follows. When k is too large, the assumption that the points in the embedding are locally fitted by a line no longer holds. Note also that in practice, for small values of k, *e.g.*, $k = 5$, the new similarity matrix S can be disconnected, and we have to resort to the merging procedure described in Algorithm B.1.

B.2.5 Sensitivity to the normalization of the Laplacian

We performed experiments to compare the performances of the method with the default Laplacian embedding (d-**LE**) (red curve in Figure B.6 and B.7) and with two possible normalized embeddings ((α, d)-**LE**) (blue and black curve). We observed that with the default d-**LE**, the performance first increases with d, and then collapses when d gets too large. The CTD scaling (blue) has the same issue, as the first d eigenvalues are roughly of the same magnitude in our settings. The heuristic scaling (α, d)-**LE** with $\alpha_k = 1/\sqrt{k}$ that damps the higher dimensions yields better results when d increases, with a plateau rather than a collapse when d gets large. We interpret these results as follows. With the (d-**LE**), Algorithm 3.3, line 5 treats equally all dimensions of the embedding. However, the curvature of the embedding tends to increase with the dimension (for $\mathcal{C}_R$ matrix, the period of the cosines increases linearly with the dimension). The filamentary structure is less smooth and hence more sensitive to noise in high dimensions, which is why the results are improved by damping the high dimensions (or using a reasonably small value for d).

B.2.6 Supplementary Figures for Hi-C data experiments

B.3 Proof of Theorem 3.3.2

In this Section, we prove Theorem 3.3.2. There are many technical details, notably the distinction between the cases n even and odd. The key idea is to compare the sums involved in the eigenvalues of the circulant matrices $A \in \mathcal{C}_R^*$. It is the sum of the b_k times values of cosines. For λ_1, we roughly have a reordering inequality where the ordering of the b_k matches those of the cosines. For the following eigenvalues, the set of values taken by the cosines is roughly the same, but it does not match the ordering of the b_k. Finally, the eigenvectors of the Laplacian of A are the same than those of A for circulant matrices A, as observed in §3.3.3.

We now introduce a few lemmas that will be useful in the proof.

Notation. In the following we denote $z_k^{(m)} \triangleq \cos(2\pi km/n)$ and $S_p^{(m)} \triangleq \sum_{k=1}^{p} z_k^{(m)}$. Let's define $\mathcal{Z}_n = \{\cos(2\pi k/n) \mid k \in \mathbb{N}\} \setminus \{-1; 1\}$. Depending on the parity of n, we will write $n = 2p$ or $n = 2p+1$. Hence we always have $p = \lfloor \frac{n}{2} \rfloor$. Also when m and n are not coprime we will note $m = dm'$ as well as $n = dn'$ with n' and m' coprime.

B.3.1 Properties of sum of cosines.

The following lemma gives us how the partial sum sequence $(S_q^{(m)})$ behave for $q = p$ or $q = p-1$ as well as it proves its symmetric behavior in (B.3).

Lemma B.3.1. *For $z_k^{(m)} = \cos(\frac{2\pi km}{n})$, $n = 2p+1$ and any $m = 1, \dots, p$*

$$S_p^{(m)} \triangleq \sum_{k=1}^{p} z_k^{(m)} = -\frac{1}{2} \ . \tag{B.2}$$

Also, for $1 \leq q \leq p/2$,

$$S_{p-q}^{(1)} \geq S_q^{(1)} \ . \tag{B.3}$$

For n and $m \geq 2$ even ($n = 2p$), we have

$$S_{p-1-q}^{(1)} = S_q^{(1)} \quad \text{for} \quad 1 \leq q \leq (p-1)/2 \tag{B.4}$$

$$S_{p-1}^{(1)} = 0 \quad \text{and} \quad S_{p-1}^{(m)} = -1 \ . \tag{B.5}$$

Finally for n even and m odd we have

$$S_p^{(m)} = S_p^{(1)} = -1 \ . \tag{B.6}$$

Proof.

Let us derive a closed form expression for the cumulative sum $S_q^{(m)}$, for any $m, q \in \{1, \ldots, p\}$

$$\begin{aligned} S_q^{(m)} = \sum_{k=1}^q z_k^{(m)} &= \mathrm{Re}\left(\sum_{k=1}^q e^{\frac{2i\pi km}{n}} \right) \\ &= \mathrm{Re}\left(e^{2i\pi m/n} \frac{1-e^{2i\pi qm/n}}{1-e^{2i\pi m/n}} \right) \\ &= \cos\left(\pi(q+1)m/n\right) \frac{\sin(\pi qm/n)}{\sin(\pi m/n)} \ . \end{aligned} \tag{B.7}$$

Let us prove equation (B.2) with the latter expression for $q = p$. Given that $n = 2p + 1 = 2(p + 1/2)$, we have,

$$\begin{aligned} \frac{\pi(p+1)m}{n} = \frac{\pi(p+1/2+1/2)m}{2(p+1/2)} &= \frac{\pi m}{2} + \frac{\pi m}{2n}, \\ \frac{\pi pm}{n} = \frac{\pi(p+1/2-1/2)m}{2(p+1/2)} &= \frac{\pi m}{2} - \frac{\pi m}{2n} \ . \end{aligned}$$

Now, by trigonometric formulas, we have,

$$\cos\left(\frac{\pi m}{2} + x\right) = \begin{cases} (-1)^{m/2} \cos(x), & \text{if } m \text{ is even} \\ (-1)^{(m+1)/2} \sin(x), & \text{if } m \text{ is odd} \end{cases}$$

$$\sin\left(\frac{\pi m}{2} - x\right) = \begin{cases} (-1)^{(1+m/2)} \sin(x), & \text{if } m \text{ is even} \\ (-1)^{(m-1)/2} \cos(x), & \text{if } m \text{ is odd} \end{cases}$$

It follows that, for any m,

$$\cos\left(\frac{\pi m}{2} + x\right) \sin\left(\frac{\pi m}{2} - x\right) = -\cos(x)\sin(x) = -\frac{1}{2}\sin(2x)$$

Finally, with $x = \pi m/(2n)$, this formula simplifies the numerator appearing in equation (B.7) and yields the result in equation (B.2).

Let us now prove equation (B.3) with a similar derivation. Let $f(q) \triangleq \cos\left(\pi(q+1)/n\right)\sin(\pi q/n)$, defined for any real $q \in [1, p/2]$. We wish to prove $f(p-q) \geq f(q)$ for any integer $q \in \{1, \ldots, \lfloor p/2 \rfloor\}$. Using $n = 2(p + 1/2)$, we have,

$$\begin{aligned} \frac{\pi(p-q+1)}{n} = \frac{\pi(p+1/2-(q-1/2))}{2(p+1/2)} &= \frac{\pi}{2} - \frac{\pi(q-1/2)}{n}, \\ \frac{\pi(p-q)}{n} = \frac{\pi(p+1/2-(q+1/2))}{2(p+1/2)} &= \frac{\pi}{2} - \frac{\pi(q+1/2)}{n} \ . \end{aligned}$$

Using $\cos(\pi/2 - x) = \sin(x)$ and $\sin(\pi/2 - x) = \cos(x)$, we thus have,

$$f(p-q) = \cos\big(\pi(q+1/2)/n\big)\sin(\pi(q-1/2)/n) = f(q-1/2) \tag{B.8}$$

To conclude, let us observe that $f(q)$ is non-increasing on $[1, p/2]$. Informally, the terms $\{z_k^1\}_{1\leq k\leq q}$ appearing in the partial sums $S_q^{(1)}$ are all non-negative for $q \leq p/2$. Formally, remark that the derivative of f, $df/dq(q) = (\pi/n)\cos\big(\pi(2q+1)/n\big)$ is non-negative for $q \in [1, p/2]$. Hence, for $q \leq p/2$, $f(q-1/2) \geq f(q)$, which ends the proof of equation (B.3).

To get the first equality of (B.5), from the exact form in (B.7), we have ($n = 2p$)

$$S_{p-1}^{(1)} = \cos(\pi p/(2p))\frac{\sin(\pi(p-1)/n)}{\sin(\pi/n)} = 0 \ .$$

For the second equality in (B.5), we have ($m = 2q$):

$$S_{p-1}^{m} = \cos(\pi q)\frac{\sin(\pi q - \pi m/n)}{\sin(\pi m/n)} = (-1)^q\frac{-(-1)^q\sin(\pi m/n)}{\sin(\pi m/n)} = -1 \ .$$

Finally to get (B.6), let us write ($n = 2p$ and m odd):

$$\begin{aligned} S_p^{(m)} &= (-1)^{m+1}\frac{\cos(\pi(p+1)m/n)}{\sin(\pi m/n)} = (-1)^{m+1}\frac{\cos(\pi m/2 + \pi m/n)}{\sin(\pi m/n)} \\ &= (-1)^m\sin(\pi m/2) = -1 \ . \end{aligned}$$

■

The following lemma gives an important property of the partial sum of the $z_k^{(m)}$ that is useful when combined with proposition B.3.3.

Lemma B.3.2. *Denote by $z_k^{(m)} = \cos(2\pi km/n)$. Consider first $n = 2p$ and m even. For $m = 1, \ldots, p$ and $q = 1, \ldots, p-2$*

$$S_q^{(1)} = \sum_{k=1}^{q} z_k^{(1)} \geq \sum_{k=1}^{q} z_k^{(m)} = S_q^{(m)} \ . \tag{B.9}$$

Otherwise we have for every $(m, q) \in \{1, \ldots, p\}^2$

$$S_q^{(1)} > S_q^{(m)} \ , \tag{B.10}$$

with equality when $q = p$.

Proof. Case m and n coprime. Values of $\big(z_k^{(m)}\big)_{k=1,\ldots,p}$ are all distinct. Indeed $z_k^{(m)} = z_{k'}^{(m)}$ implies that n divides $k + k'$ or $k - k'$. It is impossible (the range of $k + k'$ is $[2, 2p]$) unless $k = k'$.

Case m and n not coprime. $m = dm'$ and $n = dn'$, with $d \geq 3$. In that situation we need to distinguish according to the parity of n.

Case $n = 2p + 1$. Let's first remark that $\left(z_k^{(1)}\right)_{k=1,\dots,p}$ takes all values but two (-1 and 1) of the cosinus of multiple of the angle $\frac{2\pi}{n}$, e.g. $\left(z_k^{(1)}\right)_{k=1,\dots,p} \subset \mathcal{Z}_n$. Also $(z_k^{(1)})_{k=1,\dots,p}$ is non-increasing.

Let's prove (B.10) by distinguishing between the various values of q.

- Consider $q = p - (n' - 1), \dots, p$. From (B.2) in lemma (B.3.2), we have $S_p^{(1)} = S_p^{(m)}$. The $\left(z_k^{(1)}\right)_k$ are ordered in non-increasing order and the $\left(z_k^{(m)}\right)_{k=p-n'+1,\dots,p}$ take value in $\mathcal{Z}_n \cup \{1\}$ without repetition (it would requires $k \pm k' \sim 0\ [n']$). Also the partial sum of $z_k^{(1)}$ starting from the ending point p are lower than any other sequence taking the same or greater value without repetition. Because 1 is largest than any possible value in $\mathcal{Z}_n$, we hence have

$$\sum_{k=q}^{p} z_k^{(1)} \leq \sum_{k=q}^{p} z_k^{(m)} \text{ for any } q = p - (n' - 1), \dots, p\ . \tag{B.11}$$

  Since $S_q^{(m)} = S_p^{(m)} - \sum_{k=q+1}^{p} z_k^{(m)}$, (B.11) implies (B.10) for that particular set of q.

- For $q = 1, \dots, n' - 1$ it is the same type of argument. Indeed the $(z_k^{(1)})_k$ takes the highest values in $\mathcal{Z}_n$ in decreasing order, while $(z_k^{(m)})_k$ takes also its value in $\mathcal{Z}_n$ (because $z_q^{(m)} \neq 1$). This concludes (B.10).

  Note that when $n' \geq \frac{p+1}{2}$, (B.10) is then true for all q. In the sequel, let's then assume that this is not the case, e.g. $n' < \frac{p+1}{2}$.

- For $q = n' - 1, \dots, \left\lfloor \frac{p}{2} \right\rfloor$, the $z_q^{(1)}$ are non-negative. Hence $S_q^{(1)}$ is non-decreasing and lower bounded by $S_{n'-1}^{(1)}$. Also because $S_{n'}^{(m)} = 0$ and $S_{n'-1}^{(1)} \geq S_k^{(m)}$ for $k = 1, \dots, n'$, it is true that for all q in the considered set, $S_q^{(m)}$ is upper-bounded by $S_{n'-1}^{(1)}$. All in all it shows (B.10) for these values of q.

- For $q = \left\lfloor \frac{p}{2} \right\rfloor + 1, \dots, p - n'$, we apply (B.3) with $q = n'$ (and indeed $n' \leq \frac{p}{2}$) to get $S_{p-n'}^{(1)} \geq S_{n'}^{(1)}$. Because $S_q^{(m)}$ is upper-bounded by $S_{n'-1}^{(1)}$, it follows that $S_{p-n'}^{(1)} \geq S_q^{(m)}$. Finally since $(S_q^{(1)})$ is non-increasing for the considered sub-sequence of q, (B.10) is true.

Case $n = 2p$. Here $\left(z_k^{(1)}\right)_{k=1,\dots,p}$ takes unique values in $\mathcal{Z}_n \cup \{-1\}$. We also need to distinguish according to the parity of m.

- $\left(z_k^{(m)}\right)_{k=1,\dots,n'-1}$ takes also unique value in $\mathcal{Z}_n$. We similarly get (B.10) for $q = 1, \dots, n' - 1$, and for $q = n'$ because $S_{n'}^{(m)} = 0$.

- Consider m odd, from (B.6), $S_p^{(m)} = S_p^{(1)} = -1$ so that we can do the same reasoning as with n odd to prove (B.10) for $q = p - n' + 1, \ldots, p$ and $q = 1, \ldots, n'$. The remaining follows from the symmetry property (B.4) of the sequence $(S_q^{(1)})_q$ in Lemma B.3.1.
- m and n even, we have that $S_{p-1}^{(1)} = 0$ and $S_{p-1}^{(m)} = -1$ so that

$$S_{p-1}^{(1)} \geq S_{p-1}^{(m)} + 1 \ .$$

  $S_q^{(1)} \geq S_q^{(m)}$ for $q < p - 1$ follows with same techniques as before.

■

B.3.2 Properties on R-Toeplitz circular matrix.

This proposition is a technical method that will be helpful at proving that the eigenvalues of a R-circular Toeplitz matrix are such that $\nu_1 > \nu_m$.

Proposition B.3.3. *Suppose than for any $k = 1, \ldots, q$:*

$$W_k \triangleq \sum_{i=1}^{k} w_i \geq \sum_{i=1}^{k} \tilde{w}_i \triangleq \tilde{W}_k \ ,$$

with (w_i) and $(\tilde{w}_i)$ two sequences of reals. Then, if $(b_k)_k$ is non increasing and non negative, we have

$$\sum_{k=1}^{q} b_k w_k \geq \sum_{k=1}^{q} b_k \tilde{w}_k \ . \tag{B.12}$$

Proof. We have

$$\begin{aligned}
\sum_{k=1}^{q} b_k w_k &= \sum_{k=1}^{q} b_k (W_k - W_{k-1}) \\
&= \underbrace{b_q}_{\geq 0} W_q + \sum_{k=1}^{q-1} \underbrace{(b_k - b_{k+1})}_{\geq 0} W_k \\
&\geq b_q \tilde{W}_q + \sum_{k=1}^{q-1} (b_k - b_{k+1}) \tilde{W}_k = \sum_{k=1}^{q} b_k \tilde{W}_k \ .
\end{aligned}$$

■

As soon as there exists $k_0 \in \{1, \ldots, q\}$ such that

$$\sum_{i=1}^{k_0} w_i > \sum_{i=1}^{k_0} \tilde{w}_i \ ,$$

then (B.12) holds strictly.

The following proposition gives the usual derivations of eigenvalues in the R-circular Toeplitz case.

Proposition B.3.4. *Consider A, a circular-R Toeplitz matrix of size n.*

For $n = 2p + 1$

$$\nu_m \triangleq b_0 + 2\sum_{k=1}^{p} b_k \cos\left(\frac{2\pi km}{n}\right) . \tag{B.13}$$

For $m = 1, \ldots, p$ each ν_m are eigenvalues of A with multiplicity 2 and associated eigenvectors

$$\begin{aligned} y^{m,cos} &= \tfrac{1}{\sqrt{n}}\left(1, \cos\left(2\pi m/n\right), \ldots, \cos\left(2\pi m(n-1)/n\right)\right) \\ y^{m,sin} &= \tfrac{1}{\sqrt{n}}\left(1, \sin\left(2\pi m/n\right), \ldots, \sin\left(2\pi m(n-1)/n\right)\right) . \end{aligned} \tag{B.14}$$

For $n = 2p$

$$\nu_m \triangleq b_0 + 2\textstyle\sum_{k=1}^{p-1} b_k \cos\left(\frac{2\pi km}{n}\right) + b_p \cos\left(\pi m\right) , \tag{B.15}$$

where ν_0 is still singular, with $y^{(0)} = \frac{1}{\sqrt{n}}(1, \ldots, 1)$. ν_p also is, with $y^{(p)} = \frac{1}{\sqrt{n}}(+1, -1, \ldots, +1, -1)$, and there are $p - 1$ double eigenvalues, for $m = 1, \ldots, p - 1$, each associated to the two eigenvectors given in equation (B.14).

Proof. Let us compute the spectrum of a circular-R, symmetric, circulant Toeplitz matrix. From Gray et al. [2006], the eigenvalues are

$$\nu_m = \sum_{k=0}^{n-1} b_k \rho_m^k , \tag{B.16}$$

with $\rho_m = \exp(\frac{2i\pi m}{n})$, and the corresponding eigenvectors are,

$$y^{(m)} = \frac{1}{\sqrt{n}}\left(1, e^{-2i\pi m/n}, \ldots, e^{-2i\pi m(n-1)/n}\right) , \tag{B.17}$$

for $m = 0, \ldots, n - 1$.

Case n is odd, with $n = 2p + 1$. Using the symmetry assumption $b_k = b_{n-k}$, and the fact

that $\rho_m^{n-k} = \rho_m^n \rho_m^{-k} = \rho_m^{-k}$, it results in real eigenvalues,

$$\begin{aligned} \nu_m &= b_0 + \sum_{k=1}^{p} b_k \rho_m^k + \sum_{k=p+1}^{n-1} b_k \rho_m^k \\ &= b_0 + \sum_{k=1}^{p} b_k \rho_m^k + \sum_{k=1}^{p} b_{n-k} \rho_m^{n-k} \\ &= b_0 + \sum_{k=1}^{p} b_k (\rho_m^k + \rho_m^{-k}) \\ &= b_0 + 2 \sum_{k=1}^{p} b_k \cos\left(\tfrac{2\pi k m}{n}\right) . \end{aligned} \tag{B.18}$$

Observe also that $\nu_{n-m} = \nu_m$, for $m = 1, \ldots, n-1$, resulting in $p+1$ real distinct eigenvalues. ν_0 is singular, whereas for $m = 1, \ldots, p$, ν_m has multiplicity 2, with eigenvectors y^m and y^{n-m}. This leads to the two following real eigenvectors, $y^{m,\cos} = 1/2(y^m + y^{n-m})$ and $y^{m,\sin} = 1/(2i)(y^m - y^{n-m})$

$$\begin{aligned} y^{m,\cos} &= \tfrac{1}{\sqrt{n}} \left(1, \cos\left(2\pi m/n\right), \ldots, \cos\left(2\pi m(n-1)/n\right)\right) \\ y^{m,\sin} &= \tfrac{1}{\sqrt{n}} \left(1, \sin\left(2\pi m/n\right), \ldots, \sin\left(2\pi m(n-1)/n\right)\right) \end{aligned} \tag{B.19}$$

Case n is even, with $n = 2p$. A derivation similar to (B.18) yields,

$$\nu_m = b_0 + 2 \sum_{k=1}^{p-1} b_k \cos\left(\tfrac{2\pi k m}{n}\right) + b_p \cos\left(\pi m\right) \tag{B.20}$$

ν_0 is still singular, with $y^{(0)} = \frac{1}{\sqrt{n}} (1, \ldots, 1)$, ν_p also is, with $y^{(p)} = \frac{1}{\sqrt{n}} (+1, -1, \ldots, +1, -1)$, and there are $p-1$ double eigenvalues, for $m = 1, \ldots, p-1$, each associated to the two eigenvectors given in equation (B.14).

■

The following proposition is a crucial property of the eigenvalues of a circular Toeplitz matrix. It later ensures that when choosing the second eigenvalues of the laplacian, it will corresponds to the eigenvectors with the lowest period. It is paramount to prove that the latent ordering of the data can be recovered from the curve-like embedding.

Proposition B.3.5. *A circular-R, circulant Toeplitz matrix has eigenvalues $(\nu_m)_{m=0,\ldots,p}$ such that $\nu_1 \geq \nu_m$ for all $m = 2, \ldots, p$ with $n = 2p$ or $n = 2p+1$.*

Proof. Since the shape of the eigenvalues changes with the parity of n, let's again distinguish the cases.

For n odd, $\nu_1 \geq \nu_m$ is equivalent to showing

$$\sum_{k=1}^{p} b_k \cos(2\pi k/n) \geq \sum_{k=1}^{p} b_k \cos(2\pi k m/n) . \tag{B.21}$$

It is true by combining proposition B.3.3 with lemma B.3.2. The same follows for n even and m odd.

Consider n and m even. We now need to prove that

$$2\sum_{k=1}^{p-1} b_k \cos\left(\frac{2\pi k}{n}\right) - b_p \geq 2\sum_{k=1}^{p-1} b_k \cos\left(\frac{2\pi km}{n}\right) + b_p \ . \tag{B.22}$$

From lemma B.3.2, we have that

$$\sum_{k=1}^{q} z_k^{(1)} \geq \sum_{k=1}^{q} z_k^{(m)} \text{ for } q = 1, \ldots, p-2 \tag{B.23}$$

$$\sum_{k=1}^{p-1} z_k^{(1)} \geq \sum_{k=1}^{p-1} z_k^{(m)} + 1 \ . \tag{B.24}$$

Applying proposition B.3.3 with $w_k = z_k^{(1)}$ and $\tilde{w}_k = z_k^{(m)}$ for $k \leq p-2$ and $\tilde{w}_{p-1} = z_{p-1}^{(m)} + 1$, we get

$$\sum_{k=1}^{p-1} z_k^{(1)} b_k \geq \sum_{k=1}^{p-1} b_k z_k^{(m)} + b_{p-1} \tag{B.25}$$

$$2\sum_{k=1}^{p-1} z_k^{(1)} b_k \geq 2\sum_{k=1}^{p-1} b_k z_k^{(m)} + 2b_p \ . \tag{B.26}$$

The last inequality results from the monotonicity of (b_k) and is equivalent to (B.22). It concludes the proof. ■

B.3.3 Recovering exactly the order.

Here we provide the proof for Theorem 3.3.2.

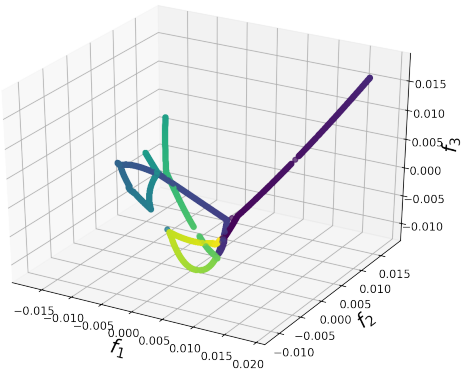
Theorem B.3.6. *Consider the seriation problem from an observed matrix $\Pi S \Pi^T$, where S is a R-circular Toeplitz matrix. Denote by L the associated graph Laplacian. Then the two dimensional laplacian spectral embedding ((d-2SUM) with d=2) of the items lies ordered and equally spaced on a circle.*

Proof. Denote $A = \Pi S \Pi^T$. The unnormalized Laplacian of A is $L \triangleq \text{diag}(A1) - A$. The eigenspace associated to its second smallest eigenvalue corresponds to that of μ_1 in A. A and S share the same spectrum. Hence the eigenspace of μ_1 in A is composed of the two vectors $\Pi y^{1,sin}$ and $\Pi y^{1,cos}$.

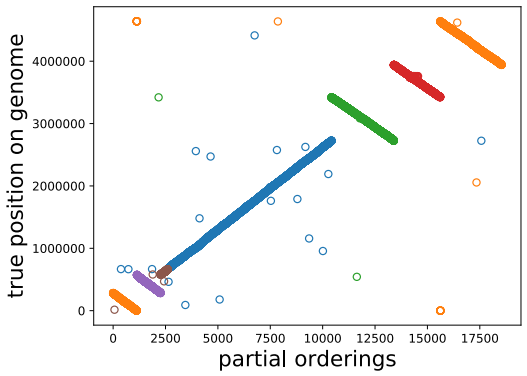
Denote by $(p_i)_{i=1,\ldots,n} \in \mathbb{R}^2$ the 2-**LE**. Each point is parametrized by

$$p_i = (\cos(2\pi\sigma(i)/n), \sin(2\pi\sigma(i)/n)) \ , \tag{B.27}$$

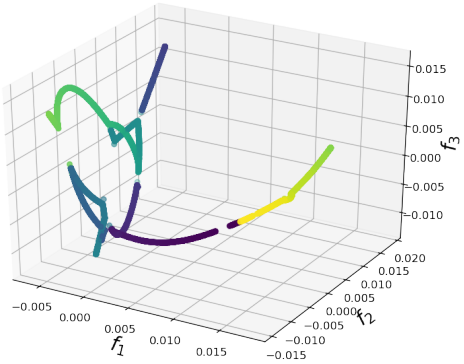
where σ is the permutation represented matricially by Π. ■



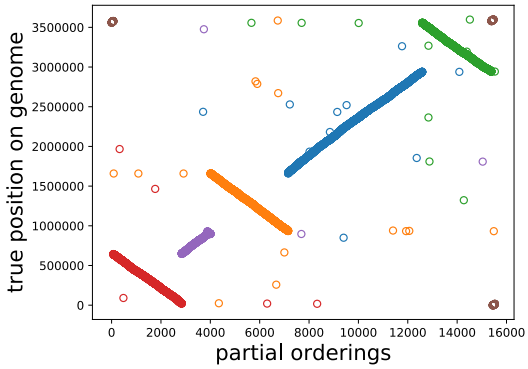
(a) *E. coli* 3-**LE**



(b) partial orderings

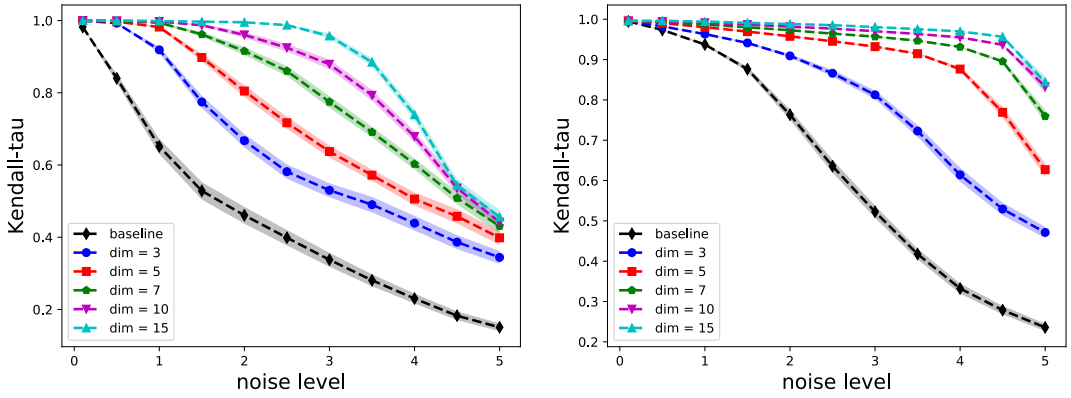


(c) *A. baylyi* 3-**LE**



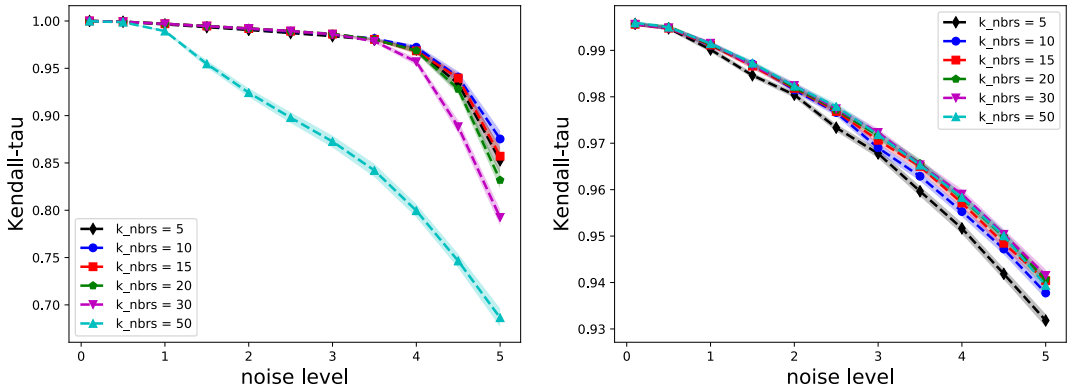
(d) partial orderings

Figure B.2: 3d Laplacian embedding from overlap-based similarity matrix of *E. coli* (B.2a) and *A. baylyi* (B.2c) reads, and the orderings found in each connected component of the new similarity matrix created in Algorithm 3.3 (B.2b and B.2d) versus the position of the reads within a reference genome obtained by mapping tge reads to the reference with minimap2 (all plotted on the same plot for compactness). The orderings have no absolute direction, *i.e.*, $(1, 2, \ldots, n)$ and $(n, n-1, \ldots, 1)$ are equivalent, which is why the lines in Figures B.2b and B.2d can be either diagonal or anti-diagonal.



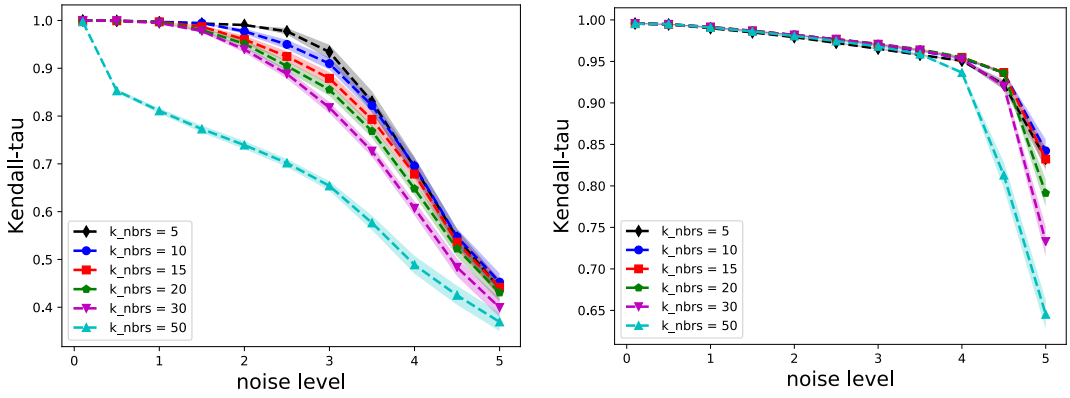
(a) Linear KMS (b) Circular KMS

Figure B.3: K-T scores for Linear (B.3a) and Circular (B.3b) Seriation for noisy observations of KMS, Toeplitz, matrices, displayed for several values of the dimension parameter of the d-**LE**.



(a) Linear Banded (b) Circular Banded

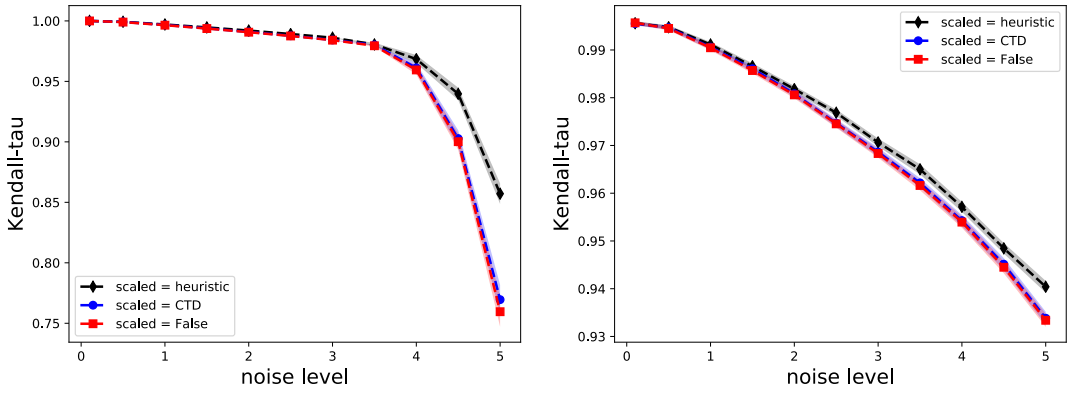
Figure B.4: K-T scores for Linear (B.4a) and Circular (B.4b) Seriation for noisy observations of banded, Toeplitz, matrices, displayed for several values of the number of nearest neighbors k, with a fixed value of the dimension of the d-**LE**, $d = 10$.



(a) Linear KMS

(b) Circular KMS

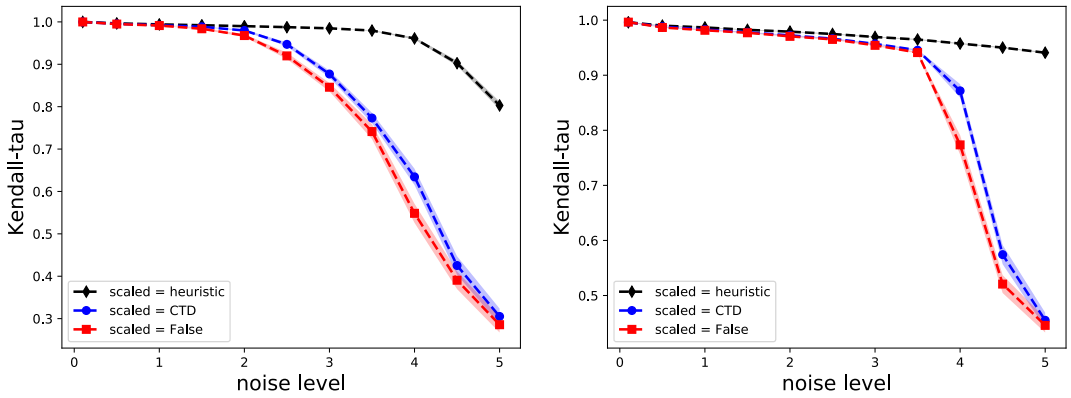
Figure B.5: K-T scores for Linear (B.5a) and Circular (B.5b) Seriation for noisy observations of KMS, Toeplitz, matrices, displayed for several values of the number of nearest neighbors k, with a fixed value of the dimension of the d-**LE**, $d = 10$.



(a) Linear Banded

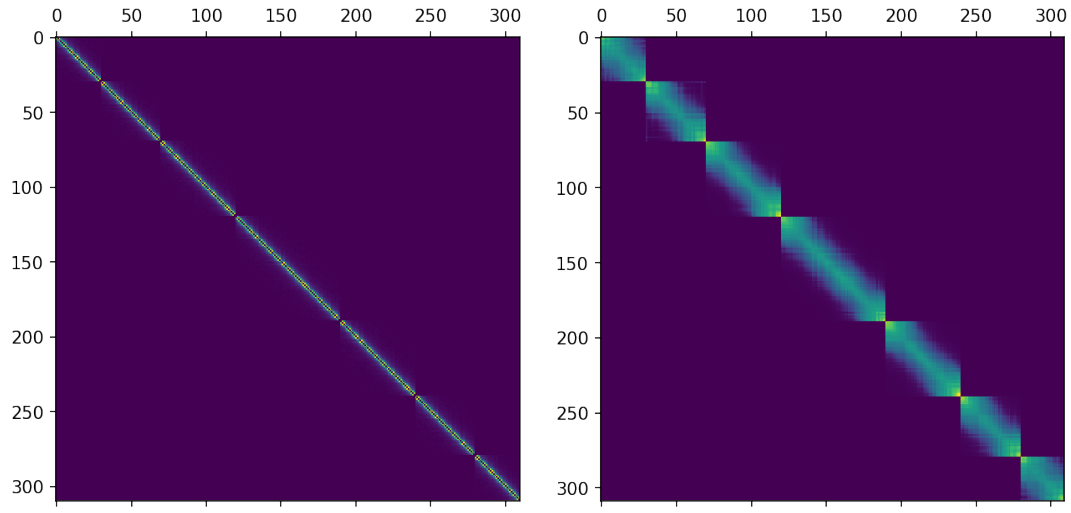
(b) Circular Banded

Figure B.6: Mean of Kendall-Tau for Linear (B.6a) and Circular (B.6b) Seriation for noisy observations of banded, Toeplitz, matrices, displayed for several scalings of the Laplacian embedding, with a fixed number of neighbors $k = 15$ and number of dimensions $d = 10$ in the d-**LE**.



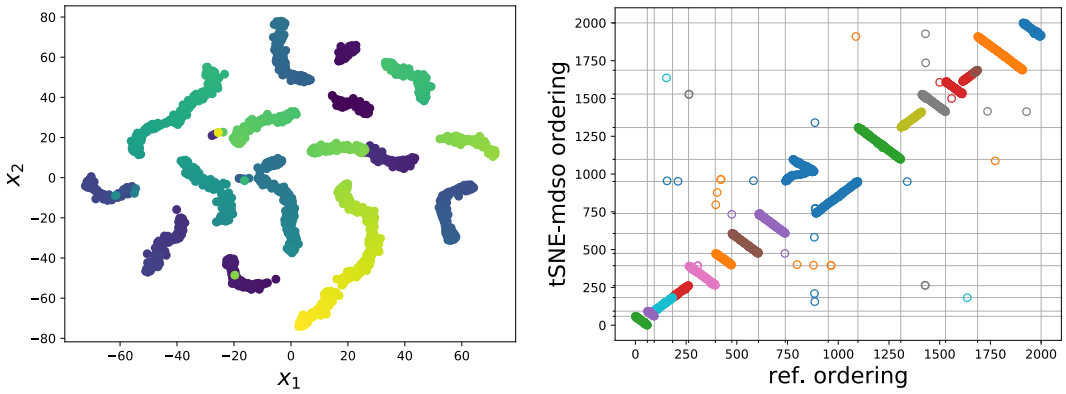
(a) Linear Banded (b) Circular Banded

Figure B.7: Mean of Kendall-Tau for Linear (B.7a) and Circular (B.7b) Seriation for noisy observations of banded, Toeplitz, matrices, displayed for several scalings of the Laplacian embedding, with a fixed number of neighbors $k = 15$ and number of dimensions $d = 20$ in the d-**LE**.



(a) raw (b) pre-processed

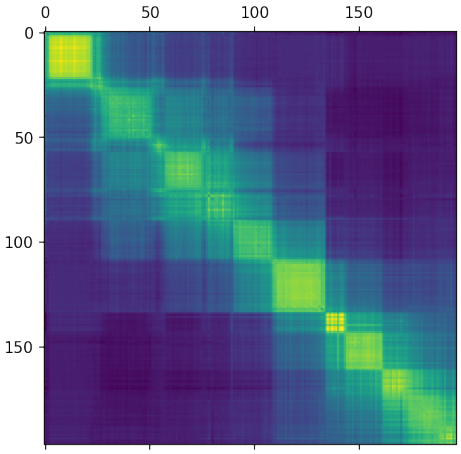
Figure B.8: Similarity matrix (with main diagonal removed) from synthetic, multiple chromosomes Hi-C data (DL1) without (B.8a) and with (B.8b) preprocessing as defined in Section 3.6.4.



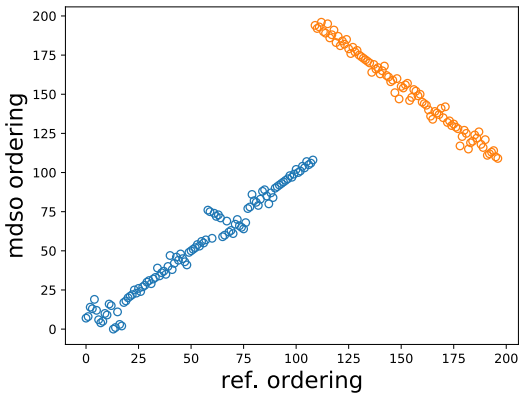
(a) tSNE embedding

(b) sub-orderings

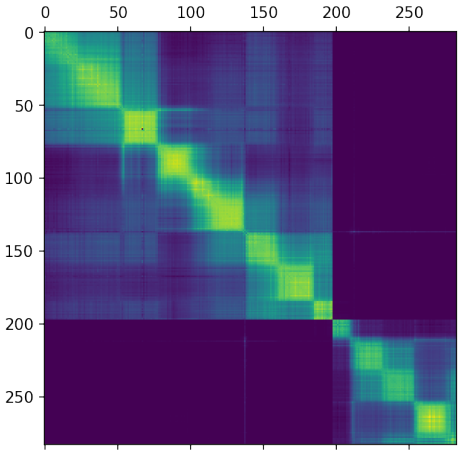
Figure B.9: t-SNE embedding (B.9a), and resulting sub-orderings found with mdso (B.9b) on the *Plasmodium knowlesi* Hi-C data, in an experiment leading to a weighted Kendall-Tau score of 61.6%.



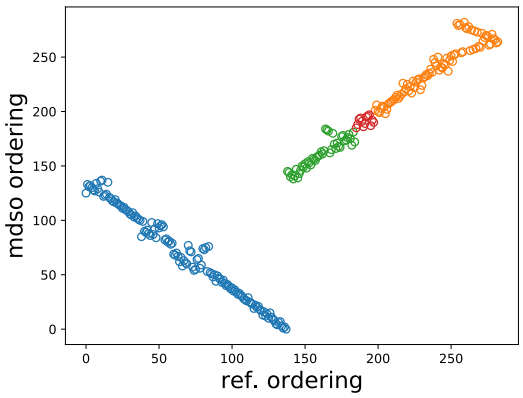
(a) matrix



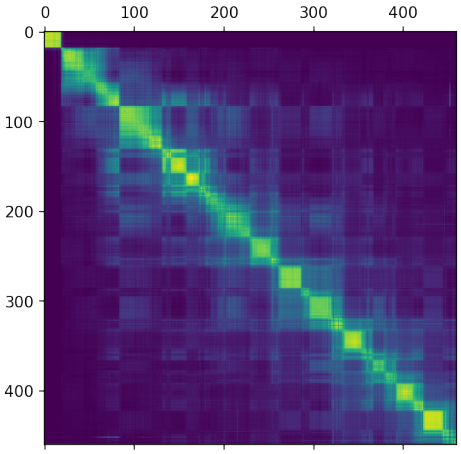
(b) sub-orderings



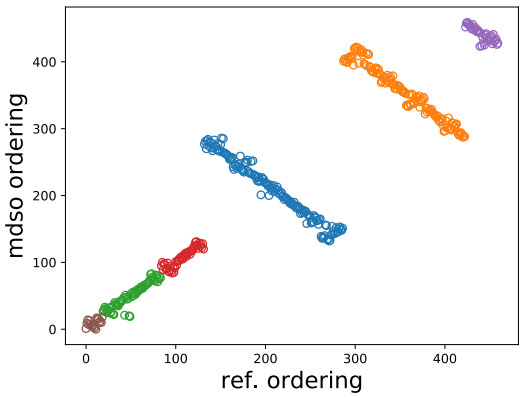
(c) matrix



(d) sub-orderings



(e) matrix



(f) sub-orderings

Figure B.10: Similarity matrices for the *Spodoptera frugiperda* data, Sf200 (B.10a), Sf669 (B.10c) and Sf846 (B.10e), and the corresponding orderings found with mdso (B.10b), (B.10d), (B.10f).

Appendix C

Supplementary Material for Chapter 4, Robust Seriation

C.1 Seriation and Robust Seriation Algorithms

C.2 Supplementary Tables

Tables C.1 and C.2 display the Kendall-τ correlation between the ordering found and the ground truth for different values of s/s_{lim} and of n, with $\delta = n/10$ and $\delta = n/20$ respectively. For given values of δ/n and s/s_{lim}, the problem is easier (*i.e.*, the methods perform better) when n increases.

Table C.1: Kendall-τ score for different values of s/s_{lim}, for the same methods as in Table 4.1, for different values of n (namely, 100 , 200 , 500), and $\delta = n/10$ (namely, 10 , 20 , 50).

		$s/s_{\text{LIM}} = 0.5$	$s/s_{\text{LIM}} = 1$	$s/s_{\text{LIM}} = 2.5$	$s/s_{\text{LIM}} = 5$	$s/s_{\text{LIM}} = 7.5$	$s/s_{\text{LIM}} = 10$
$n = 100$	SPECTRAL	0.91 ±0.08	0.83 ±0.13	0.72 ±0.19	0.62 ±0.21	0.55 ±0.20	0.48 ±0.21
	GNCR	0.92 ±0.13	0.82 ±0.23	0.70 ±0.26	0.62 ±0.26	0.55 ±0.25	0.48 ±0.24
	FAQ	0.93 ±0.09	0.85 ±0.17	0.72 ±0.24	0.61 ±0.25	0.55 ±0.25	0.48 ±0.23
	LWCD	0.93 ±0.10	0.85 ±0.17	0.72 ±0.24	0.61 ±0.25	0.55 ±0.25	0.48 ±0.23
	UBI	0.92 ±0.09	0.85 ±0.16	0.73 ±0.24	0.62 ±0.24	0.56 ±0.24	0.49 ±0.23
	MANOPT	0.92 ±0.08	0.84 ±0.13	0.72 ±0.19	0.62 ±0.21	0.55 ±0.20	0.48 ±0.21
$n = 100$	η-SPECTR.	0.99 ±0.00	0.98 ±0.00	0.89 ±0.17	0.74 ±0.25	0.65 ±0.26	0.56 ±0.26
	HGNCR	0.98 ±0.06	0.96 ±0.14	0.80 ±0.25	0.65 ±0.30	0.54 ±0.29	0.49 ±0.29
	H-FAQ	0.97 ±0.09	0.90 ±0.16	0.80 ±0.25	0.70 ±0.29	0.64 ±0.28	0.55 ±0.26
	H-LWCD	0.97 ±0.09	0.90 ±0.16	0.80 ±0.25	0.70 ±0.29	0.65 ±0.28	0.55 ±0.28
	H-UBI	0.99 ±0.00	0.98 ±0.04	0.88 ±0.20	0.75 ±0.25	0.62 ±0.26	0.54 ±0.25
	H-MANOPT	0.98 ±0.05	0.91 ±0.14	0.78 ±0.23	0.65 ±0.24	0.56 ±0.21	0.48 ±0.21
$n = 100$	R-FAQ	0.96 ±0.09	0.91 ±0.16	0.80 ±0.25	0.70 ±0.28	0.65 ±0.27	0.54 ±0.28
	R-LWCD	0.95 ±0.09	0.89 ±0.17	0.78 ±0.24	0.69 ±0.28	0.62 ±0.28	0.53 ±0.28
$n = 200$	SPECTRAL	0.96 ±0.01	0.95 ±0.01	0.91 ±0.03	0.86 ±0.06	0.84 ±0.06	0.80 ±0.09
	GNCR	0.98 ±0.00	0.96 ±0.04	0.93 ±0.07	0.87 ±0.15	0.81 ±0.20	0.80 ±0.18
	FAQ	0.98 ±0.00	0.97 ±0.00	0.94 ±0.02	0.89 ±0.08	0.87 ±0.08	0.82 ±0.13
	LWCD	0.98 ±0.00	0.97 ±0.00	0.94 ±0.02	0.89 ±0.08	0.87 ±0.08	0.82 ±0.13
	UBI	0.97 ±0.00	0.96 ±0.01	0.92 ±0.03	0.89 ±0.06	0.86 ±0.07	0.82 ±0.12
	MANOPT	0.97 ±0.00	0.95 ±0.01	0.91 ±0.03	0.86 ±0.06	0.84 ±0.06	0.80 ±0.09
$n = 200$	η-SPECTR.	0.99 ±0.00	0.99 ±0.00	0.98 ±0.00	0.97 ±0.00	0.96 ±0.00	0.94 ±0.06
	HGNCR	1.00 ±0.00	0.99 ±0.00	0.99 ±0.00	0.89 ±0.22	0.85 ±0.23	0.83 ±0.25
	H-FAQ	1.00 ±0.00	1.00 ±0.00	0.99 ±0.01	0.95 ±0.08	0.94 ±0.09	0.91 ±0.13
	H-LWCD	1.00 ±0.00	1.00 ±0.00	0.99 ±0.02	0.94 ±0.09	0.94 ±0.09	0.90 ±0.14
	H-UBI	0.99 ±0.00	0.99 ±0.00	0.98 ±0.00	0.97 ±0.00	0.96 ±0.01	0.94 ±0.03
	H-MANOPT	1.00 ±0.00	0.99 ±0.00	0.97 ±0.02	0.92 ±0.06	0.89 ±0.07	0.84 ±0.10
$n = 200$	R-FAQ	1.00 ±0.00	1.00 ±0.00	0.99 ±0.04	0.95 ±0.10	0.94 ±0.10	0.90 ±0.15
	R-LWCD	0.99 ±0.00	1.00 ±0.00	0.99 ±0.04	0.94 ±0.09	0.94 ±0.10	0.90 ±0.16
$n = 500$	SPECTRAL	0.98 ±0.00	0.98 ±0.00	0.96 ±0.00	0.95 ±0.01	0.94 ±0.01	0.93 ±0.01
	GNCR	0.99 ±0.00	0.99 ±0.00	0.98 ±0.00	0.97 ±0.00	0.96 ±0.00	0.95 ±0.05
	FAQ	0.99 ±0.00	0.99 ±0.00	0.98 ±0.00	0.97 ±0.00	0.96 ±0.00	0.95 ±0.00
	LWCD	0.99 ±0.00	0.99 ±0.00	0.98 ±0.00	0.97 ±0.00	0.96 ±0.00	0.95 ±0.00
	UBI	0.99 ±0.00	0.98 ±0.00	0.97 ±0.00	0.96 ±0.01	0.95 ±0.00	0.94 ±0.00
	MANOPT	0.99 ±0.00	0.98 ±0.00	0.97 ±0.00	0.95 ±0.00	0.94 ±0.01	0.93 ±0.01
$n = 500$	η-SPECTR.	1.00 ±0.00	1.00 ±0.00	0.99 ±0.00	0.99 ±0.00	0.99 ±0.00	0.98 ±0.00
	HGNCR	1.00 ±0.00	1.00 ±0.00	0.99 ±0.00	0.99 ±0.00	0.99 ±0.00	0.99 ±0.00
	H-FAQ	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00	0.99 ±0.00	0.99 ±0.00	0.99 ±0.00
	H-LWCD	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00	0.99 ±0.00	0.99 ±0.00	0.99 ±0.00
	H-UBI	1.00 ±0.00	1.00 ±0.00	0.99 ±0.00	0.99 ±0.00	0.99 ±0.00	0.98 ±0.00
	H-MANOPT	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00	0.99 ±0.00	0.98 ±0.01	0.97 ±0.01
$n = 500$	R-FAQ	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00
	R-LWCD	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00	1.00 ±0.01

Table C.2: Kendall-τ score for different values of s/s_{lim}, for the same methods as in Table 4.1, for different values of n (namely, 100 , 200 , 500), and $\delta = n/20$ (namely, 5 , 10 , 25).

		$s/s_{\text{LIM}} = 0.5$	$s/s_{\text{LIM}} = 1$	$s/s_{\text{LIM}} = 2.5$	$s/s_{\text{LIM}} = 5$	$s/s_{\text{LIM}} = 7.5$	$s/s_{\text{LIM}} = 10$
$n = 100$	SPECTRAL	0.46 ±0.24	0.39 ±0.21	0.31 ±0.20	0.25 ±0.16	0.22 ±0.15	0.20 ±0.14
	GNCR	0.43 ±0.28	0.37 ±0.21	0.32 ±0.21	0.25 ±0.16	0.25 ±0.14	0.20 ±0.13
	FAQ	0.45 ±0.25	0.39 ±0.22	0.31 ±0.21	0.25 ±0.17	0.23 ±0.15	0.22 ±0.14
	LWCD	0.45 ±0.26	0.39 ±0.22	0.31 ±0.21	0.25 ±0.17	0.23 ±0.15	0.22 ±0.14
	UBI	0.45 ±0.26	0.40 ±0.22	0.32 ±0.21	0.26 ±0.17	0.23 ±0.15	0.23 ±0.14
	MANOPT	0.46 ±0.25	0.40 ±0.21	0.31 ±0.20	0.25 ±0.16	0.22 ±0.15	0.21 ±0.14
$n = 100$	η-SPECTR.	0.65 ±0.33	0.50 ±0.28	0.37 ±0.24	0.28 ±0.19	0.25 ±0.16	0.23 ±0.16
	HGNCR	0.53 ±0.31	0.43 ±0.26	0.36 ±0.22	0.25 ±0.17	0.22 ±0.15	0.18 ±0.14
	H-FAQ	0.48 ±0.26	0.41 ±0.23	0.33 ±0.23	0.28 ±0.17	0.24 ±0.16	0.23 ±0.15
	H-LWCD	0.49 ±0.27	0.42 ±0.23	0.34 ±0.23	0.28 ±0.18	0.24 ±0.16	0.23 ±0.16
	H-UBI	0.60 ±0.35	0.52 ±0.29	0.40 ±0.26	0.28 ±0.19	0.25 ±0.16	0.23 ±0.15
	H-MANOPT	0.54 ±0.30	0.44 ±0.25	0.33 ±0.22	0.25 ±0.16	0.22 ±0.15	0.21 ±0.14
$n = 100$	R-FAQ	0.48 ±0.25	0.41 ±0.22	0.33 ±0.21	0.26 ±0.18	0.23 ±0.15	0.23 ±0.16
	R-LWCD	0.47 ±0.24	0.41 ±0.22	0.32 ±0.21	0.25 ±0.16	0.22 ±0.15	0.22 ±0.15
$n = 200$	SPECTRAL	0.72 ±0.21	0.59 ±0.24	0.49 ±0.26	0.42 ±0.23	0.35 ±0.20	0.31 ±0.18
	GNCR	0.69 ±0.29	0.56 ±0.31	0.45 ±0.26	0.37 ±0.27	0.34 ±0.22	0.32 ±0.23
	FAQ	0.72 ±0.24	0.60 ±0.26	0.49 ±0.26	0.41 ±0.24	0.35 ±0.21	0.33 ±0.20
	LWCD	0.72 ±0.24	0.60 ±0.26	0.49 ±0.27	0.42 ±0.25	0.36 ±0.21	0.33 ±0.20
	UBI	0.73 ±0.26	0.59 ±0.28	0.50 ±0.28	0.42 ±0.25	0.35 ±0.21	0.33 ±0.21
	MANOPT	0.72 ±0.22	0.59 ±0.24	0.49 ±0.26	0.42 ±0.24	0.35 ±0.20	0.31 ±0.18
$n = 200$	η-SPECTR.	0.99 ±0.00	0.91 ±0.21	0.65 ±0.33	0.52 ±0.30	0.41 ±0.25	0.37 ±0.23
	HGNCR	0.73 ±0.33	0.61 ±0.32	0.50 ±0.31	0.44 ±0.29	0.39 ±0.25	0.35 ±0.22
	H-FAQ	0.75 ±0.24	0.63 ±0.27	0.53 ±0.29	0.46 ±0.27	0.38 ±0.23	0.35 ±0.23
	H-LWCD	0.75 ±0.23	0.62 ±0.27	0.53 ±0.29	0.46 ±0.27	0.38 ±0.23	0.35 ±0.22
	H-UBI	0.94 ±0.19	0.82 ±0.30	0.69 ±0.34	0.57 ±0.32	0.46 ±0.28	0.40 ±0.23
	H-MANOPT	0.84 ±0.23	0.67 ±0.29	0.54 ±0.29	0.45 ±0.26	0.36 ±0.21	0.31 ±0.19
$n = 200$	R-FAQ	0.75 ±0.23	0.62 ±0.26	0.53 ±0.28	0.45 ±0.27	0.38 ±0.23	0.33 ±0.23
	R-LWCD	0.74 ±0.22	0.62 ±0.25	0.51 ±0.27	0.44 ±0.25	0.37 ±0.23	0.33 ±0.21
$n = 500$	SPECTRAL	0.96 ±0.03	0.93 ±0.05	0.86 ±0.11	0.76 ±0.18	0.71 ±0.19	0.67 ±0.21
	GNCR	0.90 ±0.21	0.80 ±0.28	0.71 ±0.31	0.60 ±0.31	0.62 ±0.29	0.55 ±0.31
	FAQ	0.98 ±0.03	0.95 ±0.06	0.87 ±0.13	0.76 ±0.21	0.72 ±0.22	0.67 ±0.24
	LWCD	0.98 ±0.03	0.95 ±0.06	0.87 ±0.13	0.76 ±0.21	0.72 ±0.22	0.67 ±0.24
	UBI	0.97 ±0.02	0.95 ±0.04	0.88 ±0.14	0.76 ±0.24	0.71 ±0.25	0.67 ±0.25
	MANOPT	0.97 ±0.03	0.94 ±0.06	0.86 ±0.12	0.76 ±0.18	0.72 ±0.19	0.67 ±0.22
$n = 500$	η-SPECTR.	1.00 ±0.00	1.00 ±0.00	0.99 ±0.00	0.96 ±0.12	0.88 ±0.18	0.81 ±0.24
	HGNCR	1.00 ±0.00	0.96 ±0.18	0.87 ±0.28	0.80 ±0.32	0.70 ±0.36	0.75 ±0.33
	H-FAQ	0.99 ±0.02	0.98 ±0.06	0.91 ±0.13	0.82 ±0.21	0.78 ±0.23	0.74 ±0.26
	H-LWCD	0.99 ±0.03	0.97 ±0.07	0.90 ±0.13	0.80 ±0.21	0.77 ±0.23	0.72 ±0.25
	H-UBI	1.00 ±0.00	1.00 ±0.00	0.99 ±0.00	0.98 ±0.07	0.95 ±0.13	0.92 ±0.17
	H-MANOPT	1.00 ±0.00	0.99 ±0.01	0.93 ±0.12	0.81 ±0.21	0.76 ±0.22	0.72 ±0.25
$n = 500$	R-FAQ	0.99 ±0.03	0.97 ±0.07	0.90 ±0.13	0.80 ±0.21	0.76 ±0.23	0.72 ±0.25
	R-LWCD	0.98 ±0.03	0.96 ±0.06	0.89 ±0.13	0.80 ±0.21	0.76 ±0.23	0.71 ±0.25

Appendix D

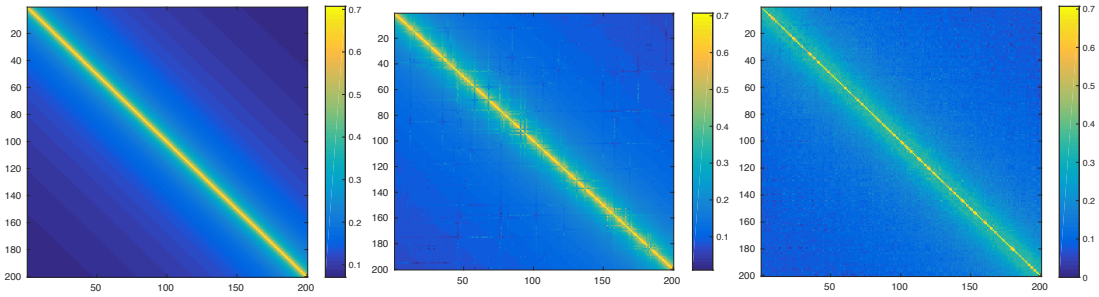
Supplementary Material for Chapter 5, Seriation with Duplications

D.1 Supplementary Figures

We present qualitative illustrations about the behavior of Algorithm 5.1, and about the evaluation of the output assignment matrix Z, given a ground truth assignment. Figures D.1 and D.2 show qualitative results on the output of the algorithm for dense and sparse similarity matrices S, respectively. Figure D.4 illustrates the different steps of Algorithm 5.1. Finally, Figures D.5 and D.6 illustate the meanDist metric used to compare assignment matrices Z.

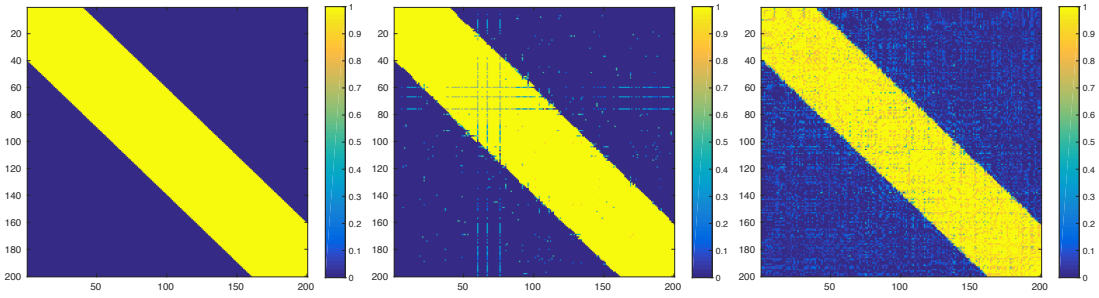
D.2 Supplementary Tables

Tables D.1, D.2 and D.3 display additional results of Seriation with Duplication (with the same scores as in Table 5.2) on dense matrices expanding the results from Section 5.4. Tables D.4 and D.5 expand these results to matrices in $\mathcal{M}_N(\delta, s)$.



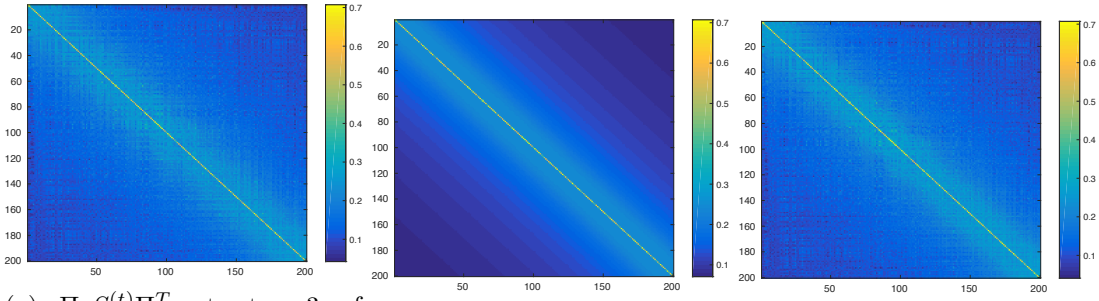
(a) Ground truth S (b) S^{out} for $N/n = 1.33$ (c) S^{out} for $N/n = 4$

Figure D.1: Original matrix S (with parameter $\gamma = 0.5$) from which the data (A, c) is generated (A), output S^{out} recovered from (A, c) by Algorithm 5.1 (used with η-spectral) with $N/n = 1.33$ (B) and with $N/n = 4$ (C). The meanDist metric is 0.98 for $N/n = 1.33$ (B) and 10.40 for $N/n = 4$ (C)



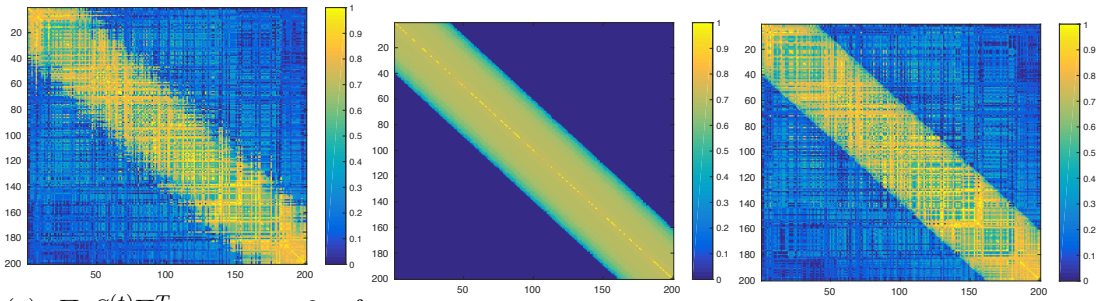
(a) Ground truth S (b) S^{out} for $N/n = 1.33$ (c) S^{out} for $N/n = 4$

Figure D.2: Original matrix S (with parameters $\delta = n/5$, $s = 0$) from which the data (A, c) is generated (A), output S^{out} recovered from (A, c) by Algorithm 5.1 (used with η-spectral) with $N/n = 1.33$ (B) and with $N/n = 4$ (C). The meanDist metric is 1.03 for $N/n = 1.33$ (B) and 7.26 for $N/n = 4$ (C)



(a) $\Pi_* S^{(t)} \Pi_*^T$ at step 3 of Alg 5.1 (b) $S^{(t+\frac{1}{2})}$ at step 3 of Alg 5.1 (c) S_A at step 4 of Alg 5.1

Figure D.3: Three steps of Algorithm 5.1 for a dense matrix S (with parameter $\gamma = 0.5$, $n = 200$, $N/n = 4$).



(a) $\Pi_* S^{(t)} \Pi_*^T$ at step 3 of Alg 5.1 (b) $S^{(t+\frac{1}{2})}$ at step 3 of Alg 5.1 (c) S_A at step 4 of Alg 5.1

Figure D.4: Three steps of Algorithm 5.1 for a sparse matrix S with parameters $n = 200$, $\delta = 40$, $s = 0$, $N/n = 4$.

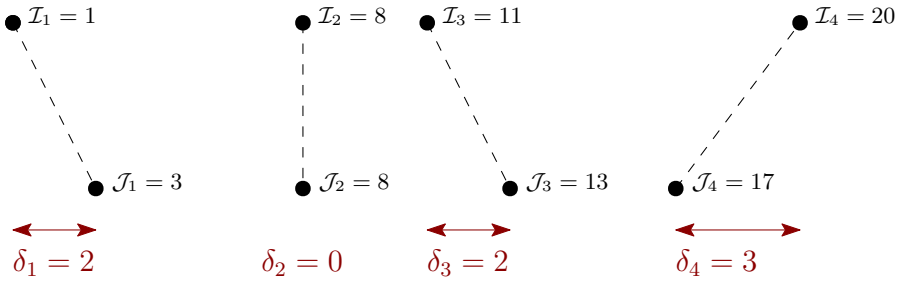


Figure D.5: Mean distance computation between two assignments $\mathcal{I} = \{1, 8, 11, 20\}$ and $\mathcal{J} = \{3, 8, 13, 17\}$ corresponding to the non-zeros in a given row i of two assignment matrices Z_1 and Z_2. Before computing the δ_i, a matching between $\mathcal{I}$ and $\mathcal{J}$ is performed.

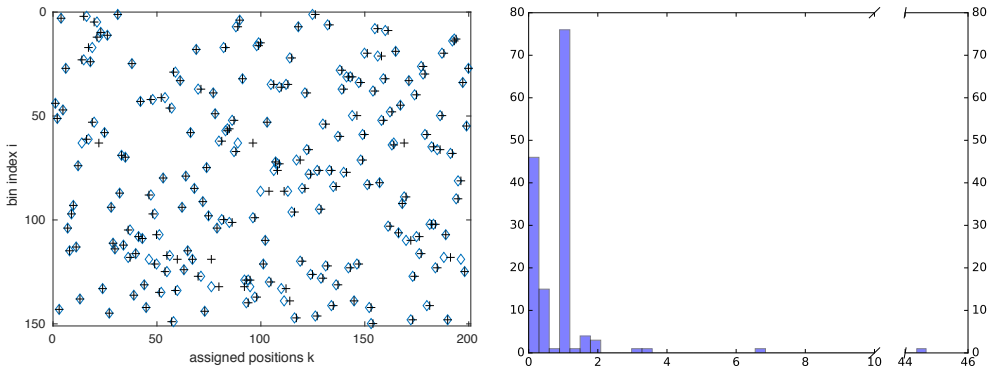


Figure D.6: Plot of the true assignment matrix Z (blue diamonds) vs the one obtained with Algorithm 5.1 (black crosses) for an experiment with a sparse matrix S with $n = 200$, $\delta = n/5$. For each row, we compute the mean distance between the non-zero represented by the blue diamonds and the black crosses, as illustrated in Figure D.5. The average over all rows of this mean distance is of 1.03 here. *Left:* assignment matrices. *Right:* Histogram of the mean distance between the matched non-zero locations (distance between black crosses and associated blue diamond), among the rows i of the two assignment matrices plotted on the left.

Table D.1: Results for Seriation with Duplications on dense, strong-R matrices (with several values of the parameter γ and N/n), and no noise added.

γ	N/n	METHOD	D2S	HUBER (X1E-7)	MEANDIST	STDDIST	TIME (X1E-2S)
0.1	1.33	SPECTRAL	0.03 ±0.00	8.33 ±0.01	3.0 ±0.7	5.5 ±1.0	5.14 ±1.36
		η-SPECTR.	0.03 ±0.00	8.33 ±0.01	3.0 ±0.7	5.5 ±1.0	5.75 ±1.41
		H-UBI	0.02 ±0.00	8.33 ±0.01	2.8 ±0.7	5.2 ±1.0	6.37 ±1.60
	2	SPECTRAL	0.03 ±0.00	8.37 ±0.01	7.1 ±1.0	7.5 ±1.0	5.05 ±1.05
		η-SPECTR.	0.03 ±0.00	8.37 ±0.01	7.1 ±1.0	7.6 ±1.0	5.41 ±1.07
		H-UBI	0.03 ±0.00	8.37 ±0.01	7.0 ±1.0	7.5 ±0.9	6.14 ±1.17
	4	SPECTRAL	0.02 ±0.00	8.35 ±0.01	12.8 ±2.2	7.8 ±1.8	5.41 ±2.03
		η-SPECTR.	0.02 ±0.00	8.35 ±0.01	12.9 ±2.3	7.9 ±2.1	5.86 ±2.16
		H-UBI	0.03 ±0.00	8.35 ±0.01	13.1 ±1.4	7.9 ±1.4	7.05 ±2.36
0.5	1.33	SPECTRAL	0.25 ±0.04	1.36 ±0.03	6.1 ±1.8	7.9 ±1.6	8.74 ±4.85
		η-SPECTR.	0.15 ±0.02	1.30 ±0.01	2.2 ±0.7	3.7 ±1.1	6.12 ±4.84
		H-UBI	0.24 ±0.04	1.35 ±0.03	5.5 ±1.6	7.3 ±1.4	11.06 ±7.56
	2	SPECTRAL	0.27 ±0.02	1.41 ±0.02	9.5 ±1.6	8.4 ±1.3	7.47 ±3.20
		η-SPECTR.	0.22 ±0.02	1.37 ±0.02	6.6 ±1.5	6.7 ±1.9	7.89 ±3.89
		H-UBI	0.26 ±0.02	1.40 ±0.02	9.0 ±1.5	8.1 ±1.2	10.09 ±4.90
	4	SPECTRAL	0.18 ±0.01	1.35 ±0.01	14.4 ±2.8	8.7 ±2.7	6.53 ±1.90
		η-SPECTR.	0.18 ±0.01	1.35 ±0.01	14.3 ±2.9	8.9 ±2.9	7.59 ±2.28
		H-UBI	0.19 ±0.01	1.35 ±0.01	14.8 ±2.5	8.8 ±2.1	8.62 ±2.46
1	1.33	SPECTRAL	0.61 ±0.02	2.10 ±0.13	15.2 ±2.4	15.2 ±1.4	9.04 ±8.61
		η-SPECTR.	0.30 ±0.06	1.48 ±0.08	2.2 ±1.4	3.1 ±1.5	15.35 ±7.54
		H-UBI	0.30 ±0.12	1.50 ±0.15	2.4 ±2.1	3.1 ±2.1	26.12 ±2.96
	2	SPECTRAL	0.60 ±0.03	2.46 ±0.11	19.3 ±6.6	12.6 ±4.9	1.78 ±0.31
		η-SPECTR.	0.42 ±0.04	1.91 ±0.13	10.3 ±8.6	9.8 ±6.4	1.20 ±0.51
		H-UBI	0.49 ±0.05	2.06 ±0.14	10.4 ±7.9	8.5 ±6.0	2.57 ±0.21
	4	SPECTRAL	0.37 ±0.02	1.81 ±0.05	19.3 ±4.7	11.6 ±4.4	1.96 ±0.50
		η-SPECTR.	0.34 ±0.01	1.78 ±0.04	20.0 ±6.9	13.2 ±6.1	1.00 ±0.34
		H-UBI	0.36 ±0.01	1.80 ±0.04	18.9 ±5.2	11.5 ±4.8	2.25 ±0.65

Table D.2: Results for Seriation with Duplications on dense, strong-R matrices (with several values of the parameter γ and N/n), and noiseProp=5%.

γ	N/n	METHOD	D2S	HUBER (X1E-7)	MEANDIST	STDDIST	TIME (X1E-2S)
0.1	1.33	SPECTRAL	0.07 ±0.00	8.36 ±0.01	5.7 ±0.9	7.2 ±1.3	1.27 ±0.78
		η-SPECTR.	0.07 ±0.00	8.36 ±0.01	5.7 ±0.9	7.2 ±1.2	1.39 ±0.80
		H-UBI	0.07 ±0.00	8.35 ±0.02	5.2 ±0.9	6.4 ±1.4	1.48 ±0.93
	2	SPECTRAL	0.07 ±0.00	8.38 ±0.01	8.5 ±0.8	7.7 ±0.8	6.62 ±4.69
		η-SPECTR.	0.07 ±0.00	8.38 ±0.01	8.5 ±0.8	7.7 ±0.9	7.62 ±5.05
		H-UBI	0.07 ±0.00	8.37 ±0.01	8.4 ±0.8	7.5 ±0.9	8.75 ±6.02
	4	SPECTRAL	0.06 ±0.00	8.35 ±0.01	13.7 ±2.4	7.9 ±2.7	5.15 ±1.49
		η-SPECTR.	0.06 ±0.00	8.35 ±0.01	13.8 ±2.3	8.0 ±2.7	5.47 ±1.58
		H-UBI	0.06 ±0.00	8.35 ±0.01	13.8 ±2.2	7.9 ±2.7	6.17 ±1.58
0.5	1.33	SPECTRAL	0.27 ±0.04	1.37 ±0.03	6.7 ±1.8	8.4 ±1.6	1.60 ±0.58
		η-SPECTR.	0.17 ±0.02	1.31 ±0.01	2.6 ±0.7	4.1 ±1.0	1.61 ±0.78
		H-UBI	0.25 ±0.03	1.36 ±0.02	5.6 ±1.5	7.3 ±1.4	2.01 ±0.74
	2	SPECTRAL	0.28 ±0.02	1.41 ±0.02	9.7 ±1.5	8.5 ±1.2	1.07 ±0.58
		η-SPECTR.	0.23 ±0.02	1.37 ±0.02	6.7 ±1.4	6.6 ±1.9	1.08 ±0.64
		H-UBI	0.26 ±0.02	1.40 ±0.02	9.0 ±1.5	8.1 ±1.3	1.46 ±0.91
	4	SPECTRAL	0.19 ±0.01	1.35 ±0.01	14.4 ±2.4	8.4 ±2.1	6.21 ±1.85
		η-SPECTR.	0.19 ±0.01	1.35 ±0.01	14.2 ±2.8	8.7 ±2.6	6.72 ±1.72
		H-UBI	0.19 ±0.01	1.35 ±0.01	14.8 ±2.6	8.8 ±2.2	7.86 ±1.89
1	1.33	SPECTRAL	0.62 ±0.02	2.10 ±0.13	15.3 ±2.4	15.3 ±1.3	9.20 ±8.34
		η-SPECTR.	0.32 ±0.06	1.49 ±0.08	2.3 ±1.1	3.2 ±1.3	18.45 ±6.40
		H-UBI	0.32 ±0.11	1.50 ±0.16	2.6 ±2.3	3.2 ±2.3	26.30 ±3.52
	2	SPECTRAL	0.61 ±0.03	2.46 ±0.12	19.4 ±6.7	12.6 ±4.9	2.13 ±0.66
		η-SPECTR.	0.42 ±0.04	1.92 ±0.13	10.6 ±8.8	10.1 ±6.6	1.52 ±0.75
		H-UBI	0.49 ±0.05	2.06 ±0.15	10.3 ±7.9	8.4 ±6.1	3.43 ±0.80
	4	SPECTRAL	0.37 ±0.02	1.80 ±0.05	19.0 ±4.9	11.2 ±4.6	1.44 ±0.30
		η-SPECTR.	0.35 ±0.01	1.79 ±0.04	20.0 ±6.8	13.2 ±6.0	0.77 ±0.17
		H-UBI	0.37 ±0.02	1.80 ±0.05	18.9 ±4.9	11.3 ±4.4	1.83 ±0.41

Table D.3: Results for Seriation with Duplications on dense, strong-R matrices (with several values of the parameter γ and N/n), and noiseProp=10%.

γ	N/n	METHOD	D2S	HUBER (X1E-7)	MEANDIST	STDDIST	TIME (X1E-2S)
0.1	1.33	SPECTRAL	0.13 ±0.00	8.36 ±0.01	8.0 ±0.7	8.0 ±1.0	1.26 ±0.74
		η-SPECTR.	0.13 ±0.00	8.36 ±0.01	7.9 ±0.7	7.9 ±1.1	1.23 ±0.83
		H-UBI	0.13 ±0.00	8.35 ±0.02	7.1 ±0.7	6.5 ±1.0	1.43 ±0.87
	2	SPECTRAL	0.12 ±0.00	8.38 ±0.01	11.1 ±0.9	8.4 ±0.9	6.44 ±4.31
		η-SPECTR.	0.12 ±0.00	8.38 ±0.01	11.0 ±0.8	8.4 ±0.9	7.08 ±4.86
		H-UBI	0.12 ±0.00	8.38 ±0.01	10.8 ±0.9	8.2 ±1.0	8.49 ±5.50
	4	SPECTRAL	0.11 ±0.00	8.35 ±0.01	15.6 ±2.8	8.2 ±2.9	5.54 ±1.55
		η-SPECTR.	0.11 ±0.00	8.35 ±0.01	15.5 ±2.5	8.3 ±3.0	6.19 ±2.23
		H-UBI	0.11 ±0.00	8.35 ±0.01	15.5 ±2.0	8.2 ±1.8	6.98 ±2.38
0.5	1.33	SPECTRAL	0.31 ±0.03	1.38 ±0.02	7.5 ±1.6	9.0 ±1.4	1.73 ±0.50
		η-SPECTR.	0.21 ±0.02	1.31 ±0.01	3.0 ±0.6	4.3 ±1.1	1.67 ±0.79
		H-UBI	0.29 ±0.03	1.36 ±0.02	6.1 ±1.6	7.5 ±1.5	2.06 ±0.81
	2	SPECTRAL	0.29 ±0.02	1.41 ±0.02	9.8 ±1.4	8.5 ±1.2	1.08 ±0.59
		η-SPECTR.	0.25 ±0.02	1.38 ±0.02	7.0 ±1.3	6.9 ±1.9	0.90 ±0.57
		H-UBI	0.28 ±0.02	1.40 ±0.02	9.3 ±1.5	8.1 ±1.3	1.25 ±0.69
	4	SPECTRAL	0.21 ±0.01	1.35 ±0.01	14.6 ±2.6	8.5 ±2.2	6.65 ±2.23
		η-SPECTR.	0.21 ±0.01	1.35 ±0.01	14.4 ±3.3	8.8 ±3.1	7.54 ±2.72
		H-UBI	0.21 ±0.01	1.35 ±0.01	15.1 ±2.5	8.8 ±2.2	8.95 ±3.53
1	1.33	SPECTRAL	0.64 ±0.02	2.10 ±0.13	15.4 ±2.3	15.4 ±1.3	8.93 ±8.70
		η-SPECTR.	0.35 ±0.05	1.52 ±0.07	2.5 ±1.1	3.4 ±1.3	20.46 ±7.24
		H-UBI	0.36 ±0.10	1.54 ±0.16	2.9 ±2.4	3.4 ±2.4	29.20 ±3.68
	2	SPECTRAL	0.61 ±0.03	2.46 ±0.11	19.6 ±6.6	12.9 ±4.8	1.70 ±0.36
		η-SPECTR.	0.43 ±0.04	1.92 ±0.13	10.4 ±8.6	9.9 ±6.4	1.18 ±0.54
		H-UBI	0.50 ±0.04	2.07 ±0.14	10.6 ±7.8	8.7 ±6.2	2.49 ±0.24
	4	SPECTRAL	0.38 ±0.02	1.81 ±0.05	19.7 ±5.2	11.7 ±5.0	1.59 ±0.42
		η-SPECTR.	0.36 ±0.01	1.79 ±0.04	20.0 ±6.9	13.1 ±6.0	0.87 ±0.25
		H-UBI	0.38 ±0.02	1.80 ±0.05	19.5 ±5.8	11.9 ±5.4	1.85 ±0.43

Table D.4: Results for Seriation with Duplications on sparse, strong-R matrices (with several values of the parameter s/s_{lim} and N/n), and $\delta = n/5$.

s/s_{LIM}	N/n	METHOD	D2S	HUBER (X1E-7)	MEANDIST	STDDIST	TIME (X1E-2S)
0	1.33	SPECTRAL	0.53 ±0.08	1.67 ±0.33	11.8 ±3.5	13.2 ±1.7	7.45 ±4.08
		η-SPECTR.	0.12 ±0.06	0.76 ±0.06	0.8 ±0.8	2.4 ±2.2	2.85 ±1.78
		H-UBI	0.09 ±0.06	0.74 ±0.05	0.6 ±0.6	1.8 ±1.9	3.99 ±2.76
	2	SPECTRAL	0.38 ±0.05	1.48 ±0.26	10.3 ±4.2	10.5 ±2.8	1.30 ±0.25
		η-SPECTR.	0.21 ±0.04	0.99 ±0.12	4.1 ±4.1	6.9 ±3.9	0.50 ±0.19
		H-UBI	0.19 ±0.05	0.96 ±0.14	4.0 ±5.8	6.2 ±4.6	0.79 ±0.31
	4	SPECTRAL	0.29 ±0.02	1.45 ±0.09	18.4 ±4.5	11.8 ±3.1	1.34 ±0.23
		η-SPECTR.	0.22 ±0.02	1.29 ±0.06	16.3 ±6.8	12.2 ±5.1	0.61 ±0.14
		H-UBI	0.22 ±0.02	1.26 ±0.06	15.9 ±7.2	12.0 ±5.6	0.91 ±0.25
0.5	1.33	SPECTRAL	0.52 ±0.08	1.68 ±0.33	11.1 ±3.5	12.9 ±1.8	8.79 ±3.83
		η-SPECTR.	0.21 ±0.03	0.87 ±0.06	1.3 ±0.7	2.6 ±2.0	4.15 ±3.10
		H-UBI	0.19 ±0.02	0.85 ±0.04	0.9 ±0.5	1.8 ±1.5	5.95 ±4.06
	2	SPECTRAL	0.40 ±0.04	1.55 ±0.23	10.3 ±3.8	10.5 ±2.6	1.33 ±0.24
		η-SPECTR.	0.24 ±0.03	1.07 ±0.11	4.3 ±4.0	7.0 ±3.9	0.55 ±0.22
		H-UBI	0.23 ±0.05	1.06 ±0.16	4.6 ±7.0	6.4 ±5.1	0.76 ±0.33
	4	SPECTRAL	0.30 ±0.03	1.50 ±0.09	19.0 ±5.1	12.1 ±3.5	1.35 ±0.19
		η-SPECTR.	0.24 ±0.02	1.34 ±0.06	16.3 ±7.1	12.0 ±5.1	0.65 ±0.17
		H-UBI	0.24 ±0.02	1.31 ±0.06	15.8 ±7.1	11.8 ±5.6	0.97 ±0.26
1	1.33	SPECTRAL	0.51 ±0.07	1.65 ±0.30	9.9 ±3.1	12.4 ±1.8	1.03 ±0.28
		η-SPECTR.	0.26 ±0.02	0.95 ±0.05	1.5 ±0.6	2.7 ±1.9	0.41 ±0.33
		H-UBI	0.25 ±0.02	0.94 ±0.04	1.2 ±0.5	2.1 ±1.7	0.59 ±0.72
	2	SPECTRAL	0.39 ±0.04	1.51 ±0.18	9.2 ±3.9	10.0 ±2.7	1.24 ±0.25
		η-SPECTR.	0.27 ±0.04	1.13 ±0.13	4.5 ±5.2	6.9 ±4.4	0.55 ±0.23
		H-UBI	0.26 ±0.04	1.11 ±0.14	4.3 ±6.3	6.3 ±4.7	0.80 ±0.36
	4	SPECTRAL	0.30 ±0.02	1.50 ±0.09	18.7 ±5.0	12.1 ±3.3	1.29 ±0.18
		η-SPECTR.	0.25 ±0.02	1.37 ±0.06	16.5 ±7.2	12.1 ±5.2	0.64 ±0.20
		H-UBI	0.25 ±0.01	1.34 ±0.06	15.4 ±6.6	11.4 ±4.9	0.91 ±0.27
2.5	1.33	SPECTRAL	0.51 ±0.05	1.71 ±0.23	8.1 ±2.4	11.1 ±2.0	1.79 ±1.50
		η-SPECTR.	0.35 ±0.01	1.25 ±0.04	1.9 ±0.4	2.7 ±1.4	0.90 ±1.39
		H-UBI	0.35 ±0.01	1.24 ±0.04	1.8 ±0.4	2.4 ±1.3	1.20 ±1.48
	2	SPECTRAL	0.43 ±0.03	1.69 ±0.13	9.3 ±4.5	10.2 ±3.4	1.24 ±0.22
		η-SPECTR.	0.34 ±0.03	1.39 ±0.13	5.1 ±6.3	7.0 ±4.8	0.49 ±0.18
		H-UBI	0.34 ±0.04	1.38 ±0.15	5.1 ±7.0	6.3 ±5.1	0.75 ±0.30
	4	SPECTRAL	0.36 ±0.02	1.64 ±0.07	19.1 ±5.3	12.1 ±3.7	1.30 ±0.20
		η-SPECTR.	0.32 ±0.01	1.52 ±0.06	16.6 ±7.2	12.1 ±5.3	0.64 ±0.15
		H-UBI	0.32 ±0.01	1.49 ±0.05	15.6 ±6.3	11.3 ±4.6	0.97 ±0.29
5	1.33	SPECTRAL	0.54 ±0.02	2.01 ±0.09	6.7 ±1.0	9.0 ±1.8	1.08 ±0.14
		η-SPECTR.	0.45 ±0.01	1.77 ±0.03	2.7 ±0.3	3.0 ±1.1	0.43 ±0.34
		H-UBI	0.45 ±0.01	1.77 ±0.03	2.8 ±0.3	3.1 ±1.0	0.98 ±0.54
	2	SPECTRAL	0.49 ±0.02	2.00 ±0.10	9.1 ±5.0	9.5 ±3.5	1.21 ±0.20
		η-SPECTR.	0.43 ±0.03	1.83 ±0.11	5.5 ±6.4	6.4 ±4.7	0.45 ±0.14
		H-UBI	0.43 ±0.03	1.83 ±0.11	5.5 ±6.2	6.3 ±4.5	0.89 ±0.42
	4	SPECTRAL	0.45 ±0.01	1.83 ±0.07	19.7 ±5.3	12.3 ±3.9	1.25 ±0.22
		η-SPECTR.	0.43 ±0.01	1.76 ±0.06	17.5 ±7.1	11.8 ±4.9	0.61 ±0.16
		H-UBI	0.43 ±0.01	1.74 ±0.05	16.5 ±5.9	11.2 ±4.5	0.87 ±0.29

Table D.5: Results for Seriation with Duplications on sparse, strong-R matrices (with several values of the parameter s/s_{lim} and N/n), and $\delta = n/10$.

s/s_{LIM}	N/n	METHOD	D2S	HUBER (X1E-7)	MEANDIST	STDDIST	TIME (X1E-2S)
0	1.33	SPECTRAL	0.85 ±0.04	6.42 ±0.63	29.1 ±14.3	23.4 ±8.1	2.13 ±3.72
		η-SPECTR.	0.28 ±0.17	1.86 ±1.13	5.4 ±12.2	6.6 ±8.5	5.67 ±3.65
		H-UBI	0.29 ±0.22	2.09 ±1.66	8.5 ±17.2	8.4 ±11.7	10.01 ±5.36
	2	SPECTRAL	0.87 ±0.02	1.01 ±0.06	44.7 ±11.9	26.7 ±7.4	9.01 ±13.31
		η-SPECTR.	0.49 ±0.10	0.43 ±0.11	26.3 ±17.2	21.1 ±10.8	85.15 ±27.35
		H-UBI	0.53 ±0.14	0.52 ±0.21	28.9 ±17.9	22.3 ±11.8	176.26 ±39.43
	4	SPECTRAL	0.78 ±0.05	1.19 ±0.10	47.5 ±7.7	21.3 ±5.1	1.04 ±0.62
		η-SPECTR.	0.39 ±0.02	0.44 ±0.02	29.6 ±7.2	18.0 ±5.4	0.60 ±0.13
		H-UBI	0.50 ±0.17	0.65 ±0.33	33.1 ±10.6	18.6 ±6.0	1.76 ±0.40
0.5	1.33	SPECTRAL	0.86 ±0.04	6.90 ±0.63	29.6 ±14.6	23.7 ±8.3	2.07 ±3.74
		η-SPECTR.	0.37 ±0.13	2.38 ±1.09	6.0 ±12.8	7.2 ±9.1	6.62 ±3.44
		H-UBI	0.37 ±0.17	2.46 ±1.51	7.6 ±16.0	7.4 ±11.0	10.88 ±4.48
	2	SPECTRAL	0.87 ±0.01	1.05 ±0.06	45.1 ±12.4	27.0 ±7.4	8.42 ±3.70
		η-SPECTR.	0.51 ±0.09	0.47 ±0.10	27.1 ±17.4	21.8 ±11.4	89.50 ±28.54
		H-UBI	0.56 ±0.13	0.58 ±0.21	29.5 ±18.4	22.5 ±12.0	175.28 ±48.61
	4	SPECTRAL	0.78 ±0.05	1.23 ±0.11	47.1 ±7.8	20.7 ±5.2	1.08 ±0.60
		η-SPECTR.	0.40 ±0.02	0.46 ±0.02	29.9 ±7.1	18.6 ±5.5	0.62 ±0.15
		H-UBI	0.49 ±0.16	0.64 ±0.32	31.8 ±9.8	18.2 ±6.1	1.78 ±0.42
1	1.33	SPECTRAL	0.88 ±0.03	7.67 ±0.69	29.4 ±14.3	23.5 ±8.2	1.57 ±3.20
		η-SPECTR.	0.42 ±0.11	2.79 ±1.26	5.7 ±12.5	6.6 ±8.9	6.34 ±3.79
		H-UBI	0.41 ±0.14	2.81 ±1.45	6.4 ±14.6	6.5 ±10.0	10.22 ±4.59
	2	SPECTRAL	0.87 ±0.01	1.14 ±0.06	44.7 ±12.2	26.7 ±7.1	1.53 ±2.76
		η-SPECTR.	0.51 ±0.08	0.51 ±0.12	26.1 ±17.7	21.1 ±11.7	8.06 ±2.78
		H-UBI	0.58 ±0.13	0.64 ±0.23	29.0 ±18.4	21.6 ±11.9	17.74 ±4.40
	4	SPECTRAL	0.75 ±0.06	1.26 ±0.14	44.6 ±7.7	20.5 ±5.2	1.21 ±0.55
		η-SPECTR.	0.40 ±0.01	0.48 ±0.02	29.4 ±7.0	18.3 ±6.2	0.63 ±0.16
		H-UBI	0.42 ±0.08	0.51 ±0.18	28.8 ±8.6	18.0 ±6.4	1.59 ±0.38
2.5	1.33	SPECTRAL	0.90 ±0.03	9.46 ±0.74	30.2 ±14.5	23.8 ±8.6	1.76 ±3.33
		η-SPECTR.	0.51 ±0.05	4.19 ±0.66	3.9 ±8.3	5.1 ±6.2	6.31 ±3.76
		H-UBI	0.54 ±0.11	4.58 ±1.53	9.4 ±17.5	8.5 ±12.5	11.94 ±4.72
	2	SPECTRAL	0.88 ±0.01	1.33 ±0.06	44.8 ±12.2	26.9 ±7.1	2.28 ±3.51
		η-SPECTR.	0.55 ±0.05	0.63 ±0.10	26.0 ±17.3	21.1 ±11.5	7.16 ±2.69
		H-UBI	0.61 ±0.11	0.75 ±0.25	28.0 ±18.9	21.2 ±12.5	18.28 ±4.10
	4	SPECTRAL	0.72 ±0.06	1.31 ±0.19	41.8 ±8.7	20.6 ±5.4	1.35 ±0.41
		η-SPECTR.	0.45 ±0.01	0.55 ±0.03	31.5 ±7.1	19.1 ±5.3	0.62 ±0.16
		H-UBI	0.44 ±0.01	0.53 ±0.03	28.2 ±7.9	17.8 ±6.2	1.49 ±0.38
5	1.33	SPECTRAL	0.93 ±0.02	1.33 ±0.09	31.4 ±13.8	24.9 ±8.8	2.60 ±4.09
		η-SPECTR.	0.64 ±0.04	0.75 ±0.07	6.5 ±11.3	7.0 ±8.9	6.73 ±4.22
		H-UBI	0.68 ±0.10	0.83 ±0.19	12.4 ±18.4	10.6 ±13.2	16.54 ±3.74
	2	SPECTRAL	0.88 ±0.01	1.73 ±0.08	44.4 ±11.6	27.0 ±6.9	4.87 ±5.68
		η-SPECTR.	0.63 ±0.03	0.87 ±0.08	26.5 ±16.0	21.3 ±10.4	7.03 ±2.69
		H-UBI	0.65 ±0.06	0.92 ±0.16	26.0 ±17.9	20.0 ±12.1	19.58 ±2.81
	4	SPECTRAL	0.66 ±0.04	1.23 ±0.22	35.4 ±7.3	18.5 ±5.3	1.42 ±0.19
		η-SPECTR.	0.57 ±0.01	0.68 ±0.03	30.9 ±5.6	19.0 ±4.9	0.58 ±0.14
		H-UBI	0.56 ±0.01	0.66 ±0.03	28.6 ±7.1	17.2 ±5.8	0.94 ±0.29

Table D.6: Results of synthetic experiments for Seriation+Clustering with Duplications from dense, strong-R matrices of size $n = 200$, with an additive block matrix with 10 clusters, with Algorithm 5.1 (that do not take the cluster structure into account), denoted SerDupli and Algorithm 5.3, denoted SerDuClus. Both are used with the η-Spectral method at step 3 of the alternate projections Algorithm. The results are averaged over 20 experiments and the standard deviation is given after the $\pm$ sign.

N/n	method	Huber	meanDist	stdDist
1.33	SerDupli	8.086e+06 $\pm$2.199e+06	30.6 $\pm$15.1	26.6 $\pm$13.0
	SerDuClus	6.618e+06 $\pm$1.317e+06	13.2 $\pm$13.5	12.7 $\pm$11.3
2	SerDupli	1.111e+07 $\pm$2.599e+06	40.0 $\pm$8.3	27.4 $\pm$6.8
	SerDuClus	9.271e+06 $\pm$2.341e+06	28.4 $\pm$9.7	20.2 $\pm$7.4
4	SerDupli	2.125e+07 $\pm$2.843e+06	42.8 $\pm$10.9	19.6 $\pm$5.1
	SerDuClus	1.504e+07 $\pm$2.835e+06	35.0 $\pm$8.7	17.0 $\pm$4.2

Bibliography

James A Albano and David W Messinger. Euclidean commute time distance embedding and its application to spectral anomaly detection. In *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XVIII*, volume 8390, page 83902G. International Society for Optics and Photonics, 2012.

Erling D Andersen and Knud D Andersen. The mosek interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High performance optimization*, pages 197–232. Springer, 2000.

Christopher Aston, Bud Mishra, and David C Schwartz. Optical mapping and its potential for large-scale sequencing projects. *Trends in biotechnology*, 17(7):297–302, 1999.

Jonathan E Atkins and Martin Middendorf. On physical mapping and the consecutive ones property for sparse matrices. *Discrete Applied Mathematics*, 71(1-3):23–40, 1996.

Jonathan E Atkins, Erik G Boman, and Bruce Hendrickson. A spectral algorithm for seriation and the consecutive ones problem. *SIAM Journal on Computing*, 28(1):297–310, 1998.

Francis R Bach and Zaïd Harchaoui. Diffrac: a discriminative and flexible framework for clustering. In *Advances in Neural Information Processing Systems*, pages 49–56, 2008.

Stephen T Barnard, Alex Pothen, and Horst Simon. A spectral algorithm for envelope reduction of sparse matrices. *Numerical linear algebra with applications*, 2(4):317–334, 1995.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Mikhail Belkin and Partha Niyogi. Semi-supervised learning on riemannian manifolds. *Machine learning*, 56(1-3):209–239, 2004.

Konstantin Berlin, Sergey Koren, Chen-Shan Chin, James P Drake, Jane M Landolin, and Adam M Phillippy. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature biotechnology*, 33(6):623, 2015.

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

Albrecht BoeÓttcher and Sergei M Grudsky. *Spectral properties of banded Toeplitz matrices*, volume 96. Siam, 2005.

Albrecht Böttcher, Sergei M Grudsky, and Egor A Maksimenko. On the structure of the eigenvectors of large hermitian toeplitz band matrices. In *Recent Trends in Toeplitz and Pseudodifferential Operators*, pages 15–36. Springer, 2010.

Albrecht Böttcher, Johan Manuel Bogoya, SM Grudsky, and Egor Anatol'evich Maximenko. Asymptotics of eigenvalues and eigenvectors of toeplitz matrices. *Sbornik: Mathematics*, 208 (11):1578, 2017.

Nicolas Boumal, Bamdev Mishra, P-A Absil, and Rodolphe Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15(1):1455–1459, 2014. URL http://www.manopt.org.

Michael J Brusco. A branch-and-bound algorithm for fitting anti-robinson structures to symmetric dissimilarity matrices. *Psychometrika*, 67(3):459–471, 2002.

Michael J Brusco, Hans-Friedrich Köhn, and Stephanie Stahl. Heuristic implementation of dynamic programming for matrix permutation problems in combinatorial data analysis. *Psychometrika*, 73(3):503, 2008.

F Bünger. Inverses, determinants, eigenvalues, and eigenvectors of real symmetric toeplitz matrices with linearly increasing entries. *Linear Algebra and its Applications*, 459:595–619, 2014.

Jitender Cheema, TH Noel Ellis, and Jo Dicks. Thread mapper studio: a novel, visual web server for the estimation of genetic linkage maps. *Nucleic acids research*, 38(suppl_2):W188–W193, 2010.

Chen-Shan Chin, David H Alexander, Patrick Marks, Aaron A Klammer, James Drake, Cheryl Heiner, Alicia Clum, Alex Copeland, John Huddleston, Evan E Eichler, et al. Nonhybrid, finished microbial genome assemblies from long-read smrt sequencing data. *Nature methods*, 10(6):563, 2013.

Fan Chung and S-T Yau. Discrete green's functions. *Journal of Combinatorial Theory, Series A*, 91(1-2):191–214, 2000.

Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

Ronald R Coifman, Yoel Shkolnisky, Fred J Sigworth, and Amit Singer. Graph laplacian tomography from unknown random projections. *IEEE Transactions on Image Processing*, 17 (10):1891–1899, 2008.

Phillip EC Compeau, Pavel A Pevzner, and Glenn Tesler. How to apply de bruijn graphs to genome assembly. *Nature biotechnology*, 29(11):987, 2011.

Chandler Davis and William Morton Kahan. The rotation of eigenvectors by a perturbation. iii. *SIAM Journal on Numerical Analysis*, 7(1):1–46, 1970.

Tom Davot, Annie Château, Rodolphe Giroudeau, and Mathias Weller. On the hardness of approximating linearization of scaffolds sharing repeated contigs. In *RECOMB International conference on Comparative Genomics*, pages 91–107. Springer, 2018.

Job Dekker, Marc A Marti-Renom, and Leonid A Mirny. Exploring the three-dimensional organization of genomes: interpreting chromatin interaction data. *Nature Reviews Genetics*, 14(6):390–403, 2013.

Persi Diaconis, Sharad Goel, and Susan Holmes. Horseshoes in multidimensional scaling and local kernel methods. *The Annals of Applied Statistics*, pages 777–807, 2008.

Chris Ding and Xiaofeng He. Linearized cluster assignment via spectral ordering. In *Proceedings of the twenty-first international conference on Machine learning*, page 30. ACM, 2004.

Olga Dudchenko, Sanjit S Batra, Arina D Omer, Sarah K Nyquist, Marie Hoeger, Neva C Durand, Muhammad S Shamim, Ido Machol, Eric S Lander, Aviva Presser Aiden, et al. De novo assembly of the aedes aegypti genome using hi-c yields chromosome-length scaffolds. *Science*, 356(6333):92–95, 2017.

Sven-Erik Ekström, Carlo Garoni, and Stefano Serra-Capizzano. Are the eigenvalues of banded symmetric toeplitz matrices known in almost closed form? *Experimental Mathematics*, pages 1–10, 2017.

Xenophon Evangelopoulos, Austin J Brockmeier, Tingting Mu, and John Y Goulermas. A graduated non-convexity relaxation for large scale seriation. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 462–470. SIAM, 2017a.

Xenophon Evangelopoulos, Austin J Brockmeier, Tingting Mu, and John Y Goulermas. Approximation methods for large scale object sequencing. *Submitted to Machine Learning*, 2017b.

Fajwel Fogel, Rodolphe Jenatton, Francis Bach, and Alexandre d'Aspremont. Convex relaxations for permutation problems. In *Advances in Neural Information Processing Systems*, pages 1016–1024, 2013.

Sebastien François, Rumen Andonov, Hristo Djidjev, and Dominique Lavenier. Global optimization methods for genome scaffolding. In *12th International Workshop on Constraint-Based Methods for Bioinformatics*, 2016.

Sébastien Francois, Rumen Andonov, Dominique Lavenier, and Hristo Djidjev. Global optimization approach for circular and chloroplast genome assembly. *bioRxiv*, page 231324, 2017.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Michael Friendly. Corrgrams: Exploratory displays for correlation matrices. *The American Statistician*, 56(4):316–324, 2002.

Gemma C Garriga, Esa Junttila, and Heikki Mannila. Banded structure in binary matrices. *Knowledge and information systems*, 28(1):197–226, 2011.

Alan George and Alex Pothen. An analysis of spectral envelope reduction via quadratic assignment problems. *SIAM Journal on Matrix Analysis and Applications*, 18(3):706–732, 1997.

Michel X. Goemans. Smallest compact formulation for the permutahedron. *Mathematical Programming*, pages 1–7, 2014.

Sara Goodwin, James Gurtowski, Scott Ethe-Sayers, Panchajanya Deshpande, Michael C Schatz, and W Richard McCombie. Oxford nanopore sequencing, hybrid error correction, and de novo assembly of a eukaryotic genome. *Genome research*, 25(11):1750–1756, 2015.

Robert M Gray et al. Toeplitz and circulant matrices: A review. *Foundations and Trends® in Communications and Information Theory*, 2(3):155–239, 2006.

Alexey Gurevich, Vladislav Saveliev, Nikolay Vyahhi, and Glenn Tesler. Quast: quality assessment tool for genome assemblies. *Bioinformatics*, 29(8):1072–1075, 2013.

Michael Hahsler. An experimental comparison of seriation methods for one-mode two-way data. *European Journal of Operational Research*, 257(1):133–143, 2017.

Michael Hahsler, Kurt Hornik, and Christian Buchta. Getting things in order: an introduction to the r package seriation. *Journal of Statistical Software*, 25(3):1–34, 2008.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.

Brandon W Higgs, Jennifer Weller, and Jeffrey L Solka. Spectral embedding finds meaningful (relevant) structure in image and microarray data. *Bmc Bioinformatics*, 7(1):74, 2006.

Daniel H Huson, Knut Reinert, and Eugene W Myers. The greedy path-merging algorithm for contig scaffolding. *Journal of the ACM (JACM)*, 49(5):603–615, 2002.

Bradley R Jones, Ashok Rajaraman, Eric Tannier, and Cedric Chauve. Anges: reconstructing ancestral genomes maps. *Bioinformatics*, 28(18):2388–2390, 2012.

Gerald Karp, Janet Iwasa, and Wallace Marshall. *Cell and Molecular Biology: Concepts and Experiments*. Wiley, 2015.

Tjalling C Koopmans and Martin Beckmann. Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, pages 53–76

Jan O Korbel and Charles Lee. Genome assembly and haplotyping with Hi-C. *Nature biotechnology*, 31:1099–1101, December 2013. ISSN 1546-1696. doi: 10.1038/nbt.2764.

Sergey Koren and Adam M Phillippy. One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly. *Current opinion in microbiology*, 23:110–120, 2015.

Sergey Koren, Michael C Schatz, Brian P Walenz, Jeffrey Martin, Jason T Howard, Ganeshkumar Ganapathy, Zhong Wang, David A Rasko, W Richard McCombie, Erich D Jarvis, et al. Hybrid error correction and de novo assembly of single-molecule sequencing reads. *Nature biotechnology*, 30(7):693, 2012.

Sergey Koren, Brian P Walenz, Konstantin Berlin, Jason R Miller, Nicholas H Bergman, and Adam M Phillippy. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome research*, 27(5):722–736, 2017.

Joseph B Kruskal and Myron Wish. *Multidimensional scaling*, volume 11. Sage, 1978.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97, 1955.

Stefan Kurtz, Adam Phillippy, Arthur L Delcher, Michael Smoot, Martin Shumway, Corina Antonescu, and Steven L Salzberg. Versatile and open software for comparing large genomes. *Genome biology*, 5(2):R12, 2004.

Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. In *Advances in Neural Information Processing Systems*, pages 496–504, 2015.

Stéphane S Lafon. *Diffusion maps and geometric harmonics*. PhD thesis, Yale University PhD dissertation, 2004.

Monique Laurent and Matteo Seminaroti. The quadratic assignment problem is easy for robinsonian matrices with toeplitz structure. *Operations Research Letters*, 43(1):103–109, 2015.

Christopher Lee, Catherine Grasso, and Mark F Sharlow. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–464, 2002.

Heng Li. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, page btw152, 2016.

Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 1:7, 2018.

E. Lieberman-Aiden, N. L. van Berkum, L. Williams, M. Imakaev, T. Ragoczy, A. Telling, I. Amit, B. R. Lajoie, P. J. Sabo, M. O. Dorschner, R. Sandstrom, B. Bernstein, M. A. Bender, M. Groudine, A. Gnirke, J. Stamatoyannopoulos, L. A. Mirny, E. S. Lander, and J. Dekker. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *Science*, 326(5950):289–293, Oct 2009a.

Erez Lieberman-Aiden, Nynke L Van Berkum, Louise Williams, Maxim Imakaev, Tobias Ragoczy, Agnes Telling, Ido Amit, Bryan R Lajoie, Peter J Sabo, and Michael O Dorschner. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *science*, 326(5950):289–293, 2009b.

Innar Liiv. Seriation and matrix reordering methods: An historical overview. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 3(2):70–91, 2010.

Cong Han Lim and Stephen Wright. Beyond the birkhoff polytope: Convex relaxations for vector permutation problems. In *Advances in Neural Information Processing Systems*, pages 2168–2176, 2014.

Cong Han Lim and Steve Wright. A box-constrained approach for hard permutation problems. In *International Conference on Machine Learning*, pages 2454–2463, 2016.

Jie Liu, Dejun Lin, Gurkan Yardimci, and William Noble. Unsupervised embedding of single-cell hi-c data. *bioRxiv*, page 257048, 2018.

Zehua Liu, Huazhe Lou, Kaikun Xie, Hao Wang, Ning Chen, Oscar M Aparicio, Michael Q Zhang, Rui Jiang, and Ting Chen. Reconstructing cell cycle pseudo time-series via single-cell transcriptome data. *Nature communications*, 8(1):22, 2017.

Nicholas J Loman, Joshua Quick, and Jared T Simpson. A complete bacterial genome assembled de novo using only nanopore sequencing data. *Nature methods*, 12(8):733, 2015.

Vince Lyzinski, Donniell E Fishkind, Marcelo Fiori, Joshua T Vogelstein, Carey E Priebe, and Guillermo Sapiro. Graph matching: Relax at your own risk. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):60–73, 2016.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

Mohammed-Amin Madoui, Stefan Engelen, Corinne Cruaud, Caroline Belser, Laurie Bertrand, Adriana Alberti, Arnaud Lemainque, Patrick Wincker, and Jean-Marc Aury. Genome assembly using nanopore-guided long and error-free dna reads. *BMC genomics*, 16(1):327, 2015.

Hervé Marie-Nelly, Martial Marbouty, Axel Cournac, Jean-François Flot, Gianni Liti, Dante Poggi Parodi, Sylvie Syan, Nancy Guillén, Antoine Margeot, Christophe Zimmer, et al. High-quality genome (re) assembly using chromosomal contact data. *Nature communications*, 5:5695, 2014.

Adam Massey, Steven J Miller, and John Sinsheimer. Distribution of eigenvalues of real symmetric pavndromic toeplitz matrices and circulant matrices. *Journal of Theoretical Probability*, 20(3):637–662, 2007.

João Meidanis, Oscar Porto, and Guilherme P Telles. On the consecutive ones property. *Discrete Applied Mathematics*, 88(1):325–354, 1998.

Eugene W Myers, Granger G Sutton, Art L Delcher, Ian M Dew, Dan P Fasulo, Michael J Flanigan, Saul A Kravitz, Clark M Mobarry, Knut HJ Reinert, Karin A Remington, et al. A whole-genome assembly of drosophila. *Science*, 287(5461):2196–2204, 2000.

Gene Myers. Efficient local alignment discovery amongst noisy long reads. In *International Workshop on Algorithms in Bioinformatics*, pages 52–67. Springer, 2014.

Niranjan Nagarajan and Mihai Pop. Sequence assembly demystified. *Nature Reviews Genetics*, 14(3):157, 2013.

Niranjan Nagarajan, Timothy D Read, and Mihai Pop. Scaffolding and validation of bacterial genome assemblies using optical restriction maps. *Bioinformatics*, 24(10):1229–1235, 2008.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

Mihai Pop. Shotgun sequence assembly. *Advances in computers*, 60(1):193–248, 2004.

Huaijun Qiu and Edwin R Hancock. Clustering and embedding using commute times. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11), 2007.

Antoine Recanati, Thomas Brüls, and Alexandre d'Aspremont. A spectral algorithm for fast de novo layout of uncorrected long nanopore reads. *Bioinformatics*, 2016.

Antoine Recanati, Thomas Kerdreux, and Alexandre d'Aspremont. Reconstructing latent orderings by spectral clustering. *arXiv preprint arXiv:1807.07122*, 2018a.

Antoine Recanati, Nicolas Servant, Jean-Philippe Vert, and Alexandre d'Aspremont. Robust seriation and applications to cancer genomics. *arXiv preprint arXiv:1806.00664*, 2018b.

Lothar Reichel and Lloyd N Trefethen. Eigenvalues and pseudo-eigenvalues of toeplitz matrices. *Linear algebra and its applications*, 162:153–185, 1992.

William S Robinson. A method for chronologically ordering archaeological deposits. *American antiquity*, 16(4):293–301, 1951.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *Journal of the ACM (JACM)*, 23(3):555–565, 1976.

Mark Schmidt. minfunc: unconstrained differentiable multivariate optimization in matlab. *Software available at http://www. cs. ubc. ca/˜ schmidtm/Software/minFunc. htm*, 2005.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, pages 583–588. Springer, 1997.

Amit Singer. From graph to manifold laplacian: The convergence rate. *Applied and Computational Harmonic Analysis*, 21(1):128–134, 2006.

Temple F Smith and Michael S Waterman. Comparison of biosequences. *Advances in applied mathematics*, 2(4):482–489, 1981.

Ivan Sović, Mile Šikić, Andreas Wilm, Shannon Nicole Fenlon, Swaine Chen, and Niranjan Nagarajan. Fast and sensitive mapping of nanopore sequencing reads with graphmap. *Nature communications*, 7, 2016.

Dorine Tabary, Tom Davot, Mathias Weller, Annie Château, and Rodolphe Giroudeau. New results about the linearization of scaffolds sharing repeated contigs. In *International Conference on Combinatorial Optimization and Applications*, pages 94–107. Springer, 2018.

William F Trench. On the eigenvalue problem for toeplitz band matrices. *Linear Algebra and its Applications*, 64:199–214, 1985.

Robert Vaser, Ivan Sović, Niranjan Nagarajan, and Mile Šikić. Racon-rapid consensus module for raw de novo genome assembly of long uncorrected reads. In *London calling conference 2016*, 2016.

Joshua T Vogelstein, John M Conroy, Vince Lyzinski, Louis J Podrazik, Steven G Kratzer, Eric T Harley, Donniell E Fishkind, R Jacob Vogelstein, and Carey E Priebe. Fast approximate quadratic programming for large (brain) graph matching. *arXiv preprint arXiv:1112.5507*, 2011.

U. Von Luxburg, O. Bousquet, and M. Belkin. Limits of spectral clustering. In *Advances in Neural Information Processing Systems 17*, pages 857–864, Cambridge, MA, USA, July 2005. Max-Planck-Gesellschaft, MIT Press.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

Kilian Q Weinberger and Lawrence K Saul. Unsupervised learning of image manifolds by semidefinite programming. *International journal of computer vision*, 70(1):77–90, 2006.

Neil I Weisenfeld, Vijay Kumar, Preyas Shah, Deanna M Church, and David B Jaffe. Direct determination of diploid genome sequences. *Genome research*, 2017.

Mathias Weller, Annie Chateau, and Rodolphe Giroudeau. Exact approaches for scaffolding. *BMC bioinformatics*, 16(14):S2, 2015.

Chen Yang, Justin Chu, René L Warren, and Inanç Birol. Nanosim: nanopore sequence read simulator based on statistical characterization. *GigaScience*, 6(4):1–6, 2017a.

Tao Yang, Feipeng Zhang, Galip Gurkan Yardimci, Fan Song, Ross C Hardison, William Stafford Noble, Feng Yue, and Qunhua Li. Hicrep: assessing the reproducibility of hi-c data using a stratum-adjusted correlation coefficient. *Genome research*, pages gr–220640, 2017b.

Yi Yu, Tengyao Wang, and Richard J Samworth. A useful variant of the davis–kahan theorem for statisticians. *Biometrika*, 102(2):315–323, 2014.

Denny Zhou, Jason Weston, Arthur Gretton, Olivier Bousquet, and Bernhard Schölkopf. Ranking on data manifolds. In *Advances in neural information processing systems*, pages 169–176, 2004.

Résumé

Les technologies de séquençage d'ADN ne permettent de lire que de courts fragments, dont on ignore la position sur le génome. L'assemblage *de novo* vise à reconstituer une séquence d'ADN entière en mettant ces fragments bout-à-bout, tel un puzzle. Dans l'approche OLC (overlap-layout-consensus), on calcule le *chevauchement* entre fragments afin de les disposer en ordre (*réarrangement*), puis extraire une séquence *consensus*. Le *réarrangement* peut s'écrire comme un problème combinatoire de *sériation*, où l'on réordonne des éléments comparable entre eux, de sorte que deux éléments adjacents sont similaires. Ce problème est résolu efficacement par un algorithme spectral en l'absence de bruit, mais il en va autrement des données génomiques réelles. En particulier, des régions du génome sont similaires bien qu'éloignées (séquences répétées), rendant l'assemblage problématique.

Les méthodes d'assemblage emploient des algorithmes hiérarchiques et gloutons pour désambiguïser les séquences répétées. Nous proposons ici une approche épurée où l'on réarrange tous les fragments « d'un coup »via la résolution de *sériation*.

Notre première contribution montre que l'emploi de la méthode spectrale pour le réarrangement s'intègre parfaitement dans le schéma OLC, produisant des résultats de qualité semblable aux méthodes standard. Cependant, du fait des séquences répétées, cette méthode produit des assemblages fragmentés (typiquement en quelques sous-séquences au lieu d'une).

La deuxième contribution est un prolongement de la méthode spectrale lié à la réduction de dimension sous conservation de distances, englobant les problèmes de sériation et de sériation circulaire (une variante où les éléments peuvent être ordonnés selon un cycle) dans un cadre unifié. Ce prolongement rend l'algorithme robuste au bruit et résout le problème de fragmentation de l'assemblage précédent.

Notre troisième contribution formalise la sériation robuste, où l'on souhaite réordonner des données bruitées. Nous décrivons des liens avec d'autres problèmes combinatoires, en particulier pour des matrices modélisant les données réelles d'ADN. Nous proposons des algorithmes adaptés, améliorant expérimentalement la robustesse sur données synthétiques et réelles, bien que moins clairement que la deuxième contribution.

La quatrième contribution présente le problème de sériation avec duplication, motivé par l'assemblage de génomes cancéreux via des données de conformation spatiale, que nous tentons de résoudre avec un algorithme de projections alternées fondé en partie sur les méthodes de sériation robuste, sur données synthétiques.

Mots-clés

sériation, méthodes spectrales, optimisation combinatoire, relaxations convexes, permutations, permutaèdre, optimisation robuste, assemblage *de novo* , séquençage de troisième génération, Oxford Nanopore Technology, Overlap-Layout-Consensus, classement.

Abstract

In a sequencing experiment, we can only "read" small fragments (*reads*) of DNA due to physical limitations, whose location on the genome is unknown. *De novo* assembly aims to put them together to retrieve the full DNA sequence, like a jigsaw puzzle. The OLC approach computes pairwise Overlaps between reads to find their Layout, and then derive a Consensus sequence.

The layout can be cast as an instance of the Seriation combinatorial problem, seeking to reorder a set of elements based on their pairwise similarity, such that similar elements are nearby. In a noiseless setting, a spectral method can solve Seriation efficiently. Still, it often fails on noisy, real DNA data. Notably, assembly is challenged by repeated genomic regions (*repeats*) causing distant fragments to be similar.

Most assembly engines follow hierarchical, greedy schemes, including modules dedicated to detect and disambiguate repeats while constructing the output sequence. We explore a simpler approach using Seriation to lay out all reads at once.

Our first contribution is to show that the spectral method can be seamlessly integrated in an OLC framework, yielding competitive results compared to standard methods on real data. However, due to repeats, the method can only find fragmented assemblies (with a few large assembled fragments), $i.e.$, it does not succeed to layout all the reads together at once.

In our second contribution, we extend the spectral method using a multidimensional spectral embedding. It provides a unifying framework for seriation and circular seriation, a variant searching for a cyclic ordering of the data. This method significantly improves the robustness of the original algorithm on noisy data, and yields single-contig assembly of bacterial genomes.

As a third contribution, we introduce the Robust Seriation framework, formalizing the task of seriation on corrupted data. We outline the relation between (robust) seriation and other combinatorial problems, particularly for stylized matrices modeling DNA sequencing data. We propose dedicated algorithms that experimentally improve robustness on synthetic and real data, although they turn out to be more sensitive than the method constituting our second contribution.

In a fourth contribution, we introduce the problem of Seriation with Duplications, which is motivated by the application of assembling cancer genome from spatial conformation (Hi-C) data. We propose an alternated minimization algorithm that can utilize methods designed to solve Robust Seriation, and evaluate it on toy data.

Keywords

seriation, spectral methods, combinatorial optimization, convex relaxations, permutations, permutahedron, robust optimization, *de novo* genome assembly, third generation sequencing, Oxford Nanopore Technology, overlap-layout-consensus, layout problems, ordering.