Structured Low-Rank Matrix Factorization for Haplotype Assembly

Changxiao Cai, Sujay Sanghavi and Haris Vikalo, *Senior Member, IEEE*

***Abstract*—In matrix factorization problems, one seeks to decompose a data matrix into a product of two matrices – frequently, one captures meaningful information contained in the data and the other specifies how this information is combined to generate the data matrix. In this paper, matrix factorization that arises in haplotype assembly, an important NP-hard problem in genomics, is studied. Haplotypes are sequences of chromosomal variations in an individual's genome that are of critical importance for understudying the individual's susceptibility to various diseases. A novel formulation of haplotype assembly as the partially-observed low-rank matrix factorization problem is proposed and efficiently solved via a modified gradient descent method that exploits salient structural properties of sequencing data. In particular, the observed matrix in the problem at hand contains noisy samples of the product of an informative matrix with rows having entries from a finite alphabet and a matrix with rows that are standard unit basis. Convergence of the proposed algorithm is analyzed and its performance tested on both synthetic and experimental data. The results demonstrate superior accuracy and speed of the proposed method as compared to state-of-the-art haplotype assembly techniques.**

***Index Terms*—haplotype assembly, matrix factorization, low rank, gradient descent**

I. Introduction

In recent years, finding a low-rank approximation to a partially observed matrix has gained a lot of attention (e.g., see [1], [2], [3], [4], [5], [6] and the references therein). These studies have been motivated by a variety of applications, including collaborative filtering problems where the goal is to infer preference of users for unrated items based on a limited number of rankings (as in, e.g., the Netflix problem [7]). In many scenarios, it is of interest to represent the rank-k matrix $\mathbf{M} \in \mathcal{R}^{n \times m}$ as $\mathbf{M} = \mathbf{U}\mathbf{V}^T$ where $\mathbf{U} \in \mathcal{R}^{n \times k}$ and $\mathbf{V} \in \mathcal{R}^{m \times k}$. Examples include applications to clustering [8] and sparse PCA [9]. The bi-linear parametrization of the unknown matrix $\mathbf{M}$ leads to the problem of finding $\mathbf{U}$ and $\mathbf{V}$ such that a chosen performance metric (e.g., the Frobenius norm of the difference between the partially observed $\mathbf{M}$ and $\mathbf{U}\mathbf{V}^T$) is optimized. Typically, the bi-linearity of the representation renders the problem non-convex and, therefore, challenging. Among the often used heuristic, alternating minimization where one keeps either $\mathbf{U}$ or $\mathbf{V}$ fixed and optimizes over the other, has gained popularity [10], [6]. This is due to the fact that each of the two subproblems is typically convex and hence can be solved in a computationally efficient manner, which is of essential importance in large-scale settings.

A field that has been fundamentally transformed by the increasing availability of large-scale datasets is that of biological and biomedical research. In particular, due to rapid technological advancements that have dramatically brought down its cost and improved its accuracy, high-throughput DNA sequencing has become a ubiquitous tool and a key enabler for personalized medicine [11]. Sequencing tasks typically generate very large data sets whose processing presents difficult computational challenges. In this paper, we focus on one such task – namely, the problem of reconstructing single individual haplotypes from high-throughput sequencing data, typically referred to as *haplotype assembly* – and cast it as a structured low-rank matrix factorization problem.

Haplotypes provide information about genetic variations in a single individual genome. Note that humans are *diploid* organisms, i.e., have DNA organized in *pairs* of chromosomes (each chromosome in a pair is inherited from one of the parents). All but one of the chromosome pairs are homologous, i.e., the chromosomes in each pair differ from each other in a small fraction of positions. The most common variations between two chromosomes in a homologous pair are single nucleotide polymorphisms (SNPs), i.e., isolated variants along the chromosome sequences. The pair of SNPs at a given location, one from each chromosome in a pair, is referred to as *genotype*. A *haplotype* is the sequences of SNPs that are located on one chromosome in a homologous pair; therefore, ordered variations between two chromosomes in a pair are represented by a corresponding pair of haplotypes. Haplotype information is of critical importance for personalized medicine applications, including the discovery of an individual's susceptibility to various diseases [12], whole genome association studies [13], as well as the detection of genes under positive selection and discovery of recombination patterns [14].

To infer a target genomic sequence, high-throughput sequencing platforms employ so-called shotgun sequencing strategy where multiple copies of the target are generated and randomly broken into relatively short fragments, and the order of nucleotides in each fragment is determined. This procedure effectively creates a library of overlapping reads where each read in the library provides partial information about the chromosome from which the corresponding fragment is generated. In the haplotype assembly applications, the reference genome is typically known and therefore a read can be mapped to the reference (i.e., relative positions of the reads with respect to the reference can be established). A read that stretches across multiple SNPs of a chromosome may be used to assemble

C. Cai is with the Tsinghua University, Beijing, China. S. Sanghavi and H. Vikalo are with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, 78712, USA (e-mail: hvikalo.ece.utexas.edu).

This work was in part supported by the National Science Foundation grant CCF 1320273.

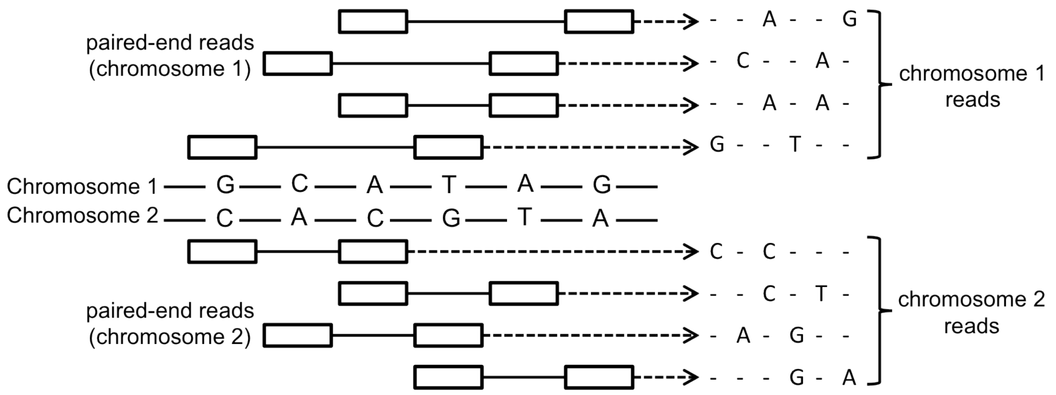


Fig. 1: *An illustration how paired-end reads generated by next-generation sequencing platforms provide information needed for haplotype assembly. The reads sample chromosomes/haplotypes but their origin (i.e., the chromosome from which they originate) is not known and needs to be inferred along with the corresponding haplotypes.*

the haplotype associated with that chromosome. However, since SNPs are relatively rare – the SNP rate between two chromosomes in a homologous pair of a human individual is roughly 10^{-3} [15] – and the reads are relatively short comprising only hundreds of bases, information they provide may be insufficient to assemble haplotypes [16]. Fortunately, recent sequencing technologies are capable of generating pairs of reads that are separated by inserts of unknown content but known length. These so-called paired-end reads help connect information across large distances of a chromosome. Fig. 1 illustrates how paired-end reads generated by high-throughput sequencing platforms provide information needed for the reconstruction of haplotypes.

While the chromosomes of diploid organisms (e.g., humans) are organized in pairs, chromosomes of polyploid species are organized in K-tuples (e.g., triplets, quadruplets, etc.). If the reads were free of sequencing errors, haplotype assembly would be straightforward and would require partitioning the reads in K clusters ($K = 2$ for diploids, $K = 3$ for triploids, etc.), each collecting the reads corresponding to one of the chromosomes in a K-tuple. However, sequencing is erroneous – the latest systems have error rates on the order of $10^{-3} - 10^{-2}$. This leads to ambiguities regarding the origin of a read and therefore renders the haplotype assembly challenging. For this reason, the vast majority of haplotype assembly techniques attempts to remove the aforementioned ambiguities by either discarding or altering sequencing data, which has led to the minimum fragment removal, minimum SNP removal and minimum error correction formulations of the assembly problem [17]. Most of the recent haplotype assembly methods focus on the minimum error correction (MEC) formulation where the goal is to find the smallest number of nucleotides in reads that need to be changed so that any read partitioning ambiguities would be resolved. It has been shown that finding optimal solution to the MEC formulation of the haplotype assembly problem is NP-hard [17], [18]. In [19], the authors used a branch-and-bound scheme to minimize the MEC objective over the space of reads, imposing a bound obtained by a random partition of the reads. Unfortunately, exponential growth of the complexity of this scheme makes it computationally infeasible even for moderate haplotype lengths. This has motivated several suboptimal heuristics. In a pioneering work [20], a greedy algorithm seeking the most likely haplotypes was used to assemble haplotypes of the first complete diploid individual genome obtained via high-throughput sequencing. To compute posterior joint probabilities of consecutive SNPs, Bayesian methods relying on MCMC and Gibbs sampling schemes were proposed in [21] and [22], respectively; unfortunately, slow convergence of Markov chains that these schemes rely on limits their practical feasibility. Following interpretation of the problem as that of grouping reads into clusters where each cluster collects reads originating from one of the chromosomes, a max-cut formulation of the haplotype assembly problem was proposed in [23]; an efficient algorithm (HapCut) that solves it and significantly outperforms the method in [20] was developed and widely used subsequently. A flow-graph based approach in [24], HapCompass, re-examined fragment removal strategy and demonstrated superior performance over HapCut. Most recent haplotype assembly methods include a greedy max-cut approach in [25], convex optimization program for minimizing the MEC score in [26], and a communication-theoretic interpretation of the problem solved via belief propagation in [27]. Note that intended personalized medicine applications require near-perfect accuracy of haplotype assembly, which is yet to be attained. Moreover, emergence of very long insert sizes in recent technologies such as fosmid [25] enables assembly of extremely long haplotype blocks but also imposes significant computational burden on the existing methods listed here. Therefore, highly accurate yet computationally efficient and scalable haplotype assembly methods are desired.

In this paper, we formulate haplotype assembly as the partially observed low-rank matrix factorization problem and propose a variant of the gradient descent algorithm to solve it at low computational cost. The algorithm explicitly imposes constraints on the special structure of the matrix $\mathbf{U}$ in factorization $\mathbf{M} = \mathbf{U}\mathbf{V}^T$ that is inherent to the haplotype assembly problem. Our proposed framework is related to the setting of the interesting work in [28] which provides a method for the completion of a matrix that consists of columns originating from one of k spaces, each of dimension r. Such a composition makes the data matrix high rank but imposes a special structure. While the problem setting we consider also involves a matrix with rows drawn from a union of 1-dimensional spaces, our problem has additional structure – namely, matrix elements are all from a discrete set. We exploit this structure explicitly, which we find empirically to improve the performance, and analyze the resulting algorithms. In addition, the algorithms in our paper are quite different from those in [28]. The paper is organized as follows. Section II introduces notation and presents formulation of haplotype assembly as a partially observed low-rank matrix factorization problem. Section III presents the structurally-constrained gradient descent algorithm while Section IV provides analysis of its convergence. Benchmarking of the algorithm on both simulated and experimental data is presented in Section V while Section VI concludes the paper. Matlab code implementation of the algorithms described in this paper is available for download from https://sourceforge.net/projects/scgdhap/.

II. Mathematical Model and Problem Statement

Data processing steps that precede the actual haplotype assembly include inference of the order of nucleotides in reads (so-called base calling [29], [30]), aligning reads to a reference [31], [32], and SNP and genotype calling [33]. Recall that SNPs occur at a relatively low frequency, e.g., one polymorphism in 1000 nucleotides. Therefore, long segments of each read will not cover any SNP locations, i.e., those segments provide no information about haplotypes and are hence discarded. Furthermore, a read covering only a single SNP position does not help in the process of infering a haplotype and is hence discarded as well. The remaining n reads (more precisely, the segments of n reads bearing information relevant for haplotype assembly) are organized into an $n \times m$ SNP fragment matrix $\mathbf{R}$, where m denotes the haplotype length. The i^{th} row of $\mathbf{R}$, $\mathbf{r}_i$, essentially collects the haplotype-relevant information provided by the i^{th} read. In humans and other diploid organisms SNP sites are bi-allelic, meaning that at each haplotype position there can be only two out of four possible nucleotides A, C, G or T; as in [24], [34], [27], we also consider polyploid haplotypes that have bi-allelic SNP positions. Therefore, we can label the nucleotides in SNP positions using binary symbols $\{1, -1\}$ where the mapping between letters and binary symbols at any position follows arbitrary convention. For convenience, entries in $\mathbf{r}_i$ that do not provide any SNP information are labeled by 0. After such a labeling, the resulting matrix $\mathbf{R}$ consists of ternary $\{-1, 0, 1\}$ entries. Specifically, the (i, j) entry of $\mathbf{R}$ is the information about the j^{th} SNP site provided by the i^{th} read; if the i^{th} read does not cover the j^{th} SNP site, the (i, j) entry of $\mathbf{R}$ is $R_{ij} = 0$. Below is an illustration of the SNP fragment matrix. Note the typical structure of the rows of R where two "islands" of non-zero entries (corresponding to SNP information provided by paired-end reads) are separated by all-zeros strings (corresponding to the non-informative inserts between the reads).

$$\mathbf{R}=\begin{bmatrix} 0 & 1 & 0 & 0 & -1 & 0 \\ -1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 \\ 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

Let $\mathcal{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_k\}$ denote the set of haplotype sequences of a k-ploid organism. It is convenient to introduce a projector operator $P_\Omega(\cdot)$ defined as

$$P_\Omega(\mathbf{M}) = \begin{cases} M_{ij}, & \text{if } (i,j) \in \Omega, \\ 0, & \text{otherwise,} \end{cases}$$

where Ω denotes the set of indices (i, j) such that $R_{ij} \neq 0$ (i.e., R_{ij} is an informative entry of $\mathbf{R}$). Therefore, $P_\Omega(\mathbf{M})$ is an operator that describes how sequencing reads, each read corresponding to a row of $\mathbf{R}$, sample the haplotypes. For instance, $R_{ij} = -1$, $(i, j) \in \Omega$ implies that the i^{th} read covers the j^{th} SNP positions and provides information encoded as "-1"; it is unknown, however, which of the k haplotypes is sampled by the i^{th} read. In general, matrix $\mathbf{R}$ can be thought of as being obtained by sampling, with errors, a low-rank $n \times m$ matrix $\mathbf{M}$,

$$\mathbf{M} = \mathbf{U}\mathbf{V}^T,$$

where $\mathbf{U}$ and $\mathbf{V}$ are $n \times k$ and $m \times k$ matrices, respectively, and where k denotes the ploidy (the number of haplotypes) of an organism.[1] The j^{th} column of $\mathbf{V}$, $\mathbf{v}_j$, is the sequence of the j^{th} haplotype, i.e., $\mathbf{v}_j = \mathbf{h}_j \in \mathcal{H}$. The i^{th} row of $\mathbf{U}$, $\mathbf{u}_i$, is the indicator of the origin of the i^{th} read. More specifically, the rows of $\mathbf{U}$ are the k-dimensional standard unit vectors consisting of all 0's except for one entry which is equal to 1. For instance, $\mathbf{u}_i = \mathbf{e}_l$ indicates that the i^{th} read is obtained by sampling the l^{th} chromosome/haplotype. Note that each row of the (unobservable) matrix $\mathbf{M}$, $\mathbf{m}_i$, is a full haplotype sequence (i.e., $\mathbf{m}_i \in \mathcal{H}$).

DNA sequencing is erroneous and thus $P_\Omega(\mathbf{R}) \neq P_\Omega(\mathbf{M})$. We assume the model where the entries in $\mathbf{R}$ are perturbed versions of the corresponding entries in $\mathbf{M}$, i.e., the $(i, j) \in \Omega$ entry in $\mathbf{R}$, R_{ij}, is obtained as

$$R_{ij} = \begin{cases} M_{ij}, & \text{w.p.} \quad 1-p, \\ -M_{ij}, & \text{w.p.} \quad p, \end{cases}$$

where p denotes the sequencing/genotyping error rate.

The goal of haplotype assembly can now be formulated as follows: *Given the SNP fragment matrix* $\mathbf{R}$*, find the matrix of haplotype sequences* $\mathbf{V}$. This is the problem of factorizing a noise-perturbed low-rank matrix with missing entries that we solve with a modified gradient search algorithm described in the following section.

[1]In the diploid ($k = 2$) case, $\mathbf{V}$ drops the rank since $\mathbf{v}_1 = -\mathbf{v}_2$ and thus $\mathbf{M}$ is rank-1. In the k-ploid case ($k > 2$), the rank of $\mathbf{V}$ (and $\mathbf{M}$) is k.

III. Gradient Descent Algorithm for SNP Fragment Matrix Factorization

As stated in Section II, given the SNP fragment matrix $\mathbf{R}$, the haplotype assembly problem can be solved by performing the low-rank matrix factorization

$$\mathbf{M} = \mathbf{U}\mathbf{V}^T \tag{1}$$

of the unobservable matrix $\mathbf{M}$ from its noisy sample with missing entries, $\mathbf{R}$. This can be done in a computationally efficient manner by relying on, e.g., gradient descent. We first describe the conventional gradient descent algorithm and then show how imposing the special structure of $\mathbf{U}$ in (1) can lead to significant improvements in the accuracy of the factorization.

A. Basic gradient descent algorithm

Define the objective function

$$f(\mathbf{U}, \mathbf{V}) = \|P_\Omega(\mathbf{R} - \mathbf{U}\mathbf{V}^T)\|_F^2, \tag{2}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of its argument. We would like to find $\mathbf{U}$ and $\mathbf{V}$ that minimize $f(\mathbf{U}, \mathbf{V})$ in (2).[2] Let $\mathbf{U}_0$ and $\mathbf{V}_0$ denote the initial guesses of $\mathbf{U}$ and $\mathbf{V}$, respectively. Gradient descent search iteratively updates estimates of $\mathbf{U}_0$ and $\mathbf{V}_0$, computing in each iteration $t = 0, 1, 2, \ldots$

$$\mathbf{V}_{t+1} = \mathbf{V}_t - \alpha \nabla f(\mathbf{U}_t, \mathbf{V}_t) \tag{3}$$

and

$$\mathbf{U}_{t+1} = \mathbf{U}_t - \beta \nabla f(\mathbf{U}_t, \mathbf{V}_{t+1}), \tag{4}$$

where $\nabla f(\mathbf{U}_t, \mathbf{V}_t)$ and $\nabla f(\mathbf{U}_t, \mathbf{V}_{t+1})$ denote the partial derivatives of $f(\mathbf{U}, \mathbf{V})$ with respect to $\mathbf{U}$ and $\mathbf{V}$ evaluated at $(\mathbf{U}_t, \mathbf{V}_t)$ and $(\mathbf{U}_t, \mathbf{V}_{t+1})$, respectively, and where α and β denote judiciously chosen step sizes of the iterations. Once a stopping criterion is met, the iterative procedure is terminated and the entries of $\mathbf{V}_{t_{max}}$ (where t_{max} denotes the last iteration) are rounded to ± 1 to yield an estimate $\hat{\mathbf{V}}$ of the matrix of haplotypes (i.e., the (i, j) entry of $\hat{\mathbf{V}}$ is formed as $\hat{V}_{ij} = \text{sign}(V_{ij,t_{max}})$). The described gradient descent procedure is formalized below as Algorithm 1.

Algorithm 1 Gradient Descent

```
Input: The SNP matrix $\mathbf{R}$
1: Initialization: Use power iteration method to generate $k$ left-singular vectors $\mathbf{U}_0$ and right-singular vectors $\mathbf{V}_0$.
2: repeat
3:     $\nabla f(\mathbf{V}_t) = -2(P_\Omega(\mathbf{R} - \mathbf{U}_t\mathbf{V}_t^T))^T\mathbf{U}_t$
4:     $\mathbf{V}_{t+1} = \mathbf{V}_t - \alpha \nabla f(\mathbf{V}_t)$
5:     $\nabla f(\mathbf{U}_t) = -2P_\Omega(\mathbf{R} - \mathbf{U}_t\mathbf{V}_{t+1}^T)\mathbf{V}_{t+1}$
6:     $\mathbf{U}_{t+1} = \mathbf{U}_t - \beta \nabla f(\mathbf{U}_t)$
7: until termination criterion is met.
Output: An estimate of the haplotype matrix $\mathbf{V}$ generated by quantizing entries of the most recent iteration $\mathbf{V}_{t_{max}}$ to $\pm 1$.
```

[2]We are primarily interested in finding $\mathbf{V}$, the matrix that consists of unknown haplotypes. For the purpose of haplotype assembly, $\mathbf{U}$ is a byproduct of the optimization.

B. Modified gradient descent algorithm that exploits the special structure of $\mathbf{U}$

The conventional gradient descent algorithm described in Section III-A does not exploit the special structure of matrix $\mathbf{U}$ – in particular, it ignores the fact that the rows of $\mathbf{U}$ are standard unit vectors which may have detrimental effects on the accuracy of the method. To enforce the structure of $\mathbf{U}$, we replace the iterations (3)-(4) by

$$\mathbf{V}_{t+1} = \mathbf{V}_t - \alpha \nabla f(\mathbf{U}_t, \mathbf{V}_t) \tag{5}$$

and

$$\mathbf{U}_{t+1} = \arg \min_{\mathbf{u}_i \in \Phi} f(\mathbf{U}, \mathbf{V}_{t+1}), \tag{6}$$

where the optimization in (6) is done by exhaustively searching over k vectors in $\Phi = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k\} = \{(1, 0, \ldots, 0), (0, 1, \ldots, 0), \ldots, (0, 0, \ldots, 1)\}$ to find the most likely $\mathbf{U}_{t+1}$. Since the number of haplotypes k is relatively small (typically, $k \leq 6$), the complexity of the exhaustive search (6) is very low. This procedure is formalized below as Algorithm 2 and referred to as structurally-constrained gradient descent. As we shall see in the results section, the structurally-constrained gradient descent algorithm enables significant performance improvements over the unconstrained gradient search.

Algorithm 2 Structurally-Constrained Gradient Descent

```
Input: The SNP matrix $\mathbf{R}$
1: Initialization: Use power iteration method to generate $k$ left-singular vectors $\mathbf{U}_0$ and right-singular vectors $\mathbf{V}_0$.
2: $\Phi = \{(1, 0, \cdots, 0), (0, 1, 0, \cdots, 0), \cdots, (0, 0, \cdots, 1)\}$.
3: repeat
4:     $\nabla f(\mathbf{V}_t) = -2(P_\Omega(\mathbf{R} - \mathbf{U}_t\mathbf{V}_t^T))^T\mathbf{U}_t$
5:     $\mathbf{V}_{t+1} = \mathbf{V}_t - \alpha \nabla f(\mathbf{V}_t)$
6:     $\mathbf{U}_{t+1} = \arg \min_{u_i \in \Phi} \sum_{(i,j) \in \Omega} \|P_\Omega(\mathbf{R} - \mathbf{U}_t\mathbf{V}_{t+1}^T)\|_F^2$
7: until termination criterion is met.
Output: An estimate of the haplotype matrix $\mathbf{V}$ generated by quantizing entries of the most recent iteration $\mathbf{V}_{t_{max}}$ to $\pm 1$.
```

Remark 1: For fast convergence, it is important to start the iterations with a reasonably accurate initial guess. For instance, in the diploid ($k = 2$) case we initialize the algorithm with the singular vector corresponding to the leading singular value of the matrix $\mathbf{R}$. Performing actual singular value decomposition is roughly cubic in the dimension of $\mathbf{R}$ and, for large problem dimensions typical of haplotype assembly, not feasible in practice. However, we only need to find $\mathbf{s}$, the leading singular vector of $\mathbf{R}$, and then estimate the haplotype sequence as $\text{sign}(\mathbf{s})$. This can be done in a computationally efficient way using the power iteration technique. In particular, the power iteration procedure requires computing in the j^{th} iteration vectors $\mathbf{x}^{(j)}$ and $\mathbf{y}^{(j)}$ defined as

$$\mathbf{x}^{(j)} = \mathbf{R}\mathbf{y}^{(j-1)}, \quad \mathbf{y}^{(j)} = \mathbf{R}^T\mathbf{x}^{(j)}. \tag{7}$$

To demonstrate convergence of the power iteration scheme (7), let us denote the singular values of $\mathbf{R}$ as $\sigma_i(\mathbf{R})$, where $\sigma_1(\mathbf{R}) \geq \sigma_2(\mathbf{R}) \geq \ldots \geq 0$. With a random initialization

$\mathbf{y}^{(0)}$, power iterations will converge to the singular vector $\mathbf{s}$ if the inequality $\sigma_1(\mathbf{R}) > \sigma_2(\mathbf{R})$ holds strictly. The speed of the convergence of power iterations depends on the ratio $\sigma_2(\mathbf{R})/\sigma_1(\mathbf{R})$. This can be easily shown by an analysis of the consecutive projections of the iteratively updated vectors $\mathbf{x}^{(j)}$ onto the singular vector $\mathbf{s}$. In particular, the projection of $\mathbf{x}^{(j)}$ onto $\mathbf{s}$ is $(\mathbf{s}^T\mathbf{x}^{(j)})\mathbf{s}$. A closer look into the singular value decomposition shows that $\mathbf{s}^T\mathbf{x}^{(j)}\mathbf{s} = (\sigma_1(\mathbf{R}))^2\mathbf{s}^T\mathbf{x}^{(j-1)}\mathbf{s}$ and $(\mathbf{x}^{(j)} - \mathbf{s}^T\mathbf{x}^{(j)}\mathbf{s}) \le (\sigma_2(\mathbf{R}))^2(\mathbf{x}^{(j-1)} - \mathbf{s}^T\mathbf{x}^{(j-1)}\mathbf{s})$. Therefore,

$$\begin{aligned}\frac{||\mathbf{x}^{(j)} - \mathbf{s}^T\mathbf{x}^{(j)}\mathbf{s}||}{||\mathbf{s}^T\mathbf{x}^{(j)}\mathbf{s}||} &\le \left(\frac{\sigma_2(\mathbf{R})}{\sigma_1(\mathbf{R})}\right)^2 \frac{||\mathbf{x}^{(j-1)} - \mathbf{s}^T\mathbf{x}^{(j-1)}\mathbf{s}||}{||\mathbf{s}^T\mathbf{x}^{(j-1)}\mathbf{s}||} \\ &\le \left(\frac{\sigma_2(\mathbf{R})}{\sigma_1(\mathbf{R})}\right)^{2j} \frac{||\mathbf{x}^{(0)} - \mathbf{s}^T\mathbf{x}^{(0)}\mathbf{s}||}{||\mathbf{s}^T\mathbf{x}^{(0)}\mathbf{s}||}.\end{aligned}$$

Clearly, power iterations will converge with any initialization if $\sigma_1(\mathbf{R}) > \sigma_2(\mathbf{R})$, and the speed of convergence depends on the ratio of $\sigma_1(\mathbf{R})$ and $\sigma_2(\mathbf{R})$ – the larger the ratio, the faster the convergence.

Remark 2: Another computationally efficient approach to minimizing (2) is by means of alternating minimization (see, e.g., [6] and the references therein for a recent treatment of the subject). In alternating minimization, (2) is optimized by alternatively finding via least-squares

$$\mathbf{V}_{t+1} = \underset{\mathbf{V}}{\operatorname{argmin}} \sum_{(i,j)\in\Omega} \left\|P_\Omega(\mathbf{R} - \mathbf{U}_t\mathbf{V}^T)\right\|_F^2$$

and

$$\mathbf{U}_{t+1} = \underset{\mathbf{U}}{\operatorname{argmin}} \sum_{(i,j)\in\Omega} \left\|P_\Omega(\mathbf{R} - \mathbf{U}\mathbf{V}_{t+1}^T)\right\|_F^2 \qquad (8)$$

until a termination criterion is met and then rounding the entries in $\mathbf{V}_{t_{max}}$ to ±1, where t_{max} denotes the last iteration. Similar to the discussion of gradient descent in Section III.A-B, the above described alternating minimization procedure does not exploit the special structure of $\mathbf{U}$. To enforce that in iteration $t+1$ the rows of $\mathbf{U}_{t+1}$ are standard unit vectors, we can replace the least squares solution in (8) by the exhaustive search over k vectors in $\Phi = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k\} = \{(1,0,\ldots,0),(0,1,\ldots,0),\ldots,(0,0,\ldots,1)\}$. The resulting structurally-constrained alternating minimization for haplotype assembly is formalized as Algorithm 3 below.

Algorithm 3 Structurally-Constrained Alt. Minimization

```
Input: The SNP matrix $\mathbf{R}$
1: Initialization: Use power iteration method to generate $k$ left-singular vectors $\mathbf{U}_0$ and right-singular vectors $\mathbf{V}_0$.
2: $\Phi = \{(1,0,\cdots,0),(0,1,0,\cdots,0),\cdots,(0,0,\cdots,1)\}$.
3: repeat
4:     $\mathbf{V}_{t+1} = \arg\min_{\mathbf{V}} \sum_{(i,j)\in\Omega} \|P_\Omega(\mathbf{R} - \mathbf{U}_t\mathbf{V}^T)\|_F^2$
5:     $\mathbf{U}_{t+1} = \arg\min_{u_i\in\Phi} \sum_{(i,j)\in\Omega} \|P_\Omega(\mathbf{R} - \mathbf{U}\mathbf{V}_{t+1}^T)\|_F^2$
6: until termination criterion is met.
Output: An estimate of the haplotype matrix $\mathbf{V}$ generated by quantizing entries of the most recent iteration $\mathbf{V}_{t_{max}}$ to $\pm1$.
```

Note that step 4 of Algorithm 3 involves computing the j^{th} row vector of $\mathbf{V}_{t+1}$, $\mathbf{v}_{j,t+1}$, according to $\mathbf{v}_{j,t+1} = \bar{\mathbf{U}}_t^{-1}\bar{\mathbf{r}}_t$ where the (q,l) entry of the $k\times k$ matrix $\bar{\mathbf{U}}_t$ is given by $\bar{U}_{ql,t} = \sum_{(i,j)\in\Omega} U_{iq,t}U_{il,t}$, the q^{th} element of the $k\times 1$ vector $\bar{\mathbf{r}}_t$ is $\bar{r}_{q,t} = \sum_{(i,j)\in\Omega} R_{ij}U_{iq,t}$, and where $U_{iq,t}$ denotes the (i,q) entry of $\mathbf{U}_t$.

IV. Convergence of the Structurally-Constrained Gradient Descent

In this section we analyze convergence of the structurally-constrained gradient descent algorithm (Algorithm 2 in Section III.B) for finding factorization $\mathbf{M} = \mathbf{U}\mathbf{V}^T$ from $\mathbf{R}$, the noisy observations of $P_\Omega(\mathbf{M})$, where the matrix $\mathbf{U}$ is constrained to have rows that belong to the set of unit vectors $\mathbf{e}_i \in \Phi$ and the entries in $\mathbf{V}$ are ±1.[3] As part of this analysis, we provide a recommendation for choosing coefficient α in (5). We begin with the following lemma.

Lemma 1: The Hessian of the scalar function $f(\mathbf{U},\mathbf{V})$ in (2) computed with respect to $\mathbf{V}$ where the rows of $\mathbf{U}$ are constrained to be from $\Phi = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k\} = \{(1,0,\ldots,0),(0,1,\ldots,0),\ldots,(0,0,\ldots,1)\}$ is a positive semi-definite matrix.

Proof. See Appendix A. □

The results of the convergence analysis are summarized in the following theorem.

Theorem 1: Let the step size α in (5) be selected as

$$\alpha = C\frac{\|\nabla f(V_t)^T\|_F^2}{\|P_\Omega(U_t\nabla f(V_t)^T)\|_F^2},$$

where $C \in (0,1)$ is a constant. Then the solution $(\mathbf{U}^*, \mathbf{V}^*)$ found by the structurally constrained gradient search algorithm described in Section III is a stationary point of the objective function $f(\mathbf{U},\mathbf{V})$ in (2).

Furthermore, if a fresh set Γ of uniformly distributed test samples is available, then executing one iteration of the algorithm from $(\mathbf{U}^*, \mathbf{V}^*)$ will reveal whether or not $f(\mathbf{U}^*, \mathbf{V}^*)$ is a global minimum.

Proof. Recall that the objective function of the optimization problem under consideration is

$$f(\mathbf{U},\mathbf{V}) = \|P_\Omega(\mathbf{R} - \mathbf{U}\mathbf{V}^T)\|_F^2.$$

Keeping $\mathbf{U}$ constant, the gradient of $f(\mathbf{U},\mathbf{V})$ with respect to $\mathbf{V}$ is readily computed as

$$\nabla f(\mathbf{V}) = -2(P_\Omega(\mathbf{R} - \mathbf{U}\mathbf{V}^T))^T\mathbf{U}.$$

In each iteration step, the algorithm computes a new value of the matrix $\mathbf{V}$ as

$$\mathbf{V}_{t+1} = \mathbf{V}_t - \alpha\nabla f(\mathbf{V}_t).$$

[3] As shown in the results section, structurally-constrained gradient descent (Algorithm 2) appears to fairly consistently outperform alternating minimization (Algorithm 3) as well as Algorithm 1 on data sets of interest. For that reason, Algorithm 2 will be our method of choice and thus we here focus on analyzing that method.

The corresponding change in the value of the objective function $f(\mathbf{U},\mathbf{V})$ is found as

$$\begin{aligned}
&f(\mathbf{U}_t,\mathbf{V}_{t+1})-f(\mathbf{U}_t,\mathbf{V}_t)\\
&=\|P_\Omega(\mathbf{R}-\mathbf{U}_t\mathbf{V}_{t+1}^T)\|_F^2-\|P_\Omega(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T)\|_F^2\\
&=\big\|P_\Omega\big(\mathbf{R}-\mathbf{U}_t\big(\mathbf{V}_t-\alpha\nabla f(\mathbf{V}_t)\big)^T\big)\big\|_F^2\\
&-\big\|P_\Omega(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T)\big\|_F^2\\
&=\big\|P_\Omega(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T+\alpha\mathbf{U}_t\nabla f(\mathbf{V}_t)^T)\big\|_F^2\\
&-\big\|P_\Omega(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T)\big\|_F^2\\
&=2\alpha\sum_{(i,j)\in\Omega}\big(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T\big)_{ij}\cdot\big(\mathbf{U}_t\nabla f(\mathbf{V}_t)^T\big)_{ij}\\
&+\alpha^2\sum_{(i,j)\in\Omega}\big((\mathbf{U}_t\nabla f(\mathbf{V}_t)^T)_{ij}\big)^2\\
&=-4\alpha\sum_{(i,j)\in\Omega}\big(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T\big)_{ij}\cdot\big(\mathbf{U}_t\mathbf{U}_t^TP_\Omega(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T)\big)_{ij}\\
&+4\alpha^2\sum_{(i,j)\in\Omega}\big(\mathbf{U}_t\mathbf{U}_t^TP_\Omega(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T)_{ij}\big)^2\\
&=-4\alpha\sum_{(i,j)}P_\Omega\big(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T\big)_{ij}\cdot\big(\mathbf{U}_t\mathbf{U}_t^TP_\Omega(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T)\big)_{ij}\\
&+4\alpha^2\sum_{(i,j)\in\Omega}\big(\mathbf{U}_t\mathbf{U}_t^TP_\Omega(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T)_{ij}\big)^2.
\end{aligned}\tag{9}$$

For any two matrices $\mathbf{A}\in\mathcal{R}^{n\times m}$ and $\mathbf{B}\in\mathcal{R}^{n\times k}$, it is straightforward to show that

$$\sum_{(i,j)}(\mathbf{B}^T\mathbf{A})_{ij}^2=\sum_{(i,j)}\mathbf{A}_{ij}\cdot\mathbf{B}\mathbf{B}^T\mathbf{A}_{ij}.\tag{10}$$

Using identity (10), the first term in the last line of equation (9) can be written as

$$\begin{aligned}
&-4\alpha\sum_{(i,j)}P_\Omega\big(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T\big)_{ij}\cdot\big(\mathbf{U}_t\mathbf{U}_t^TP_\Omega(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T)\big)_{ij}\\
&=-\alpha\sum_{(i,j)}\big(2\mathbf{U}_t^TP_\Omega(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T)\big)_{ij}^2\\
&=-\alpha\big\|-2\mathbf{U}_t^TP_\Omega(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T)\big\|_F^2\\
&=-\alpha\big\|\nabla f(\mathbf{V}_t)^T\big\|_F^2
\end{aligned}\tag{11}$$

Moreover, the second term in the last line of equation (9) can be written as

$$\begin{aligned}
&4\alpha^2\sum_{(i,j)\in\Omega}\big(\mathbf{U}_t\mathbf{U}_t^TP_\Omega(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T)_{ij}\big)^2\\
&=\alpha^2\big\|P_\Omega\big(-2\mathbf{U}_t\mathbf{U}_t^TP_\Omega(\mathbf{R}-\mathbf{U}_t\mathbf{V}_t^T)\big)\big\|_F^2\\
&=\alpha^2\big\|P_\Omega\big(\mathbf{U}_t\nabla f(\mathbf{V}_t)^T\big)\big\|_F^2.
\end{aligned}\tag{12}$$

By choosing the step size

$$\alpha=C\frac{\big\|\nabla f(\mathbf{V}_t)^T\big\|_F^2}{\big\|P_\Omega(\mathbf{U}_t\nabla f(\mathbf{V}_t)^T)\big\|_F^2},\tag{13}$$

where C is a constant, and substituting (11) and (12) into (9), we readily obtain

$$f(\mathbf{U}_t,\mathbf{V}_{t+1})-f(\mathbf{U}_t,\mathbf{V}_t)=(C^2-C)\frac{\|\nabla f(\mathbf{V}_t)^T\|_F^4}{\|P_\Omega(\mathbf{U}_t\nabla f(\mathbf{V}_t)^T)\|_F^2}.$$

Clearly, if $C\in(0,1)$, it must hold that

$$f(\mathbf{U}_t,\mathbf{V}_{t+1})-f(\mathbf{U}_t,\mathbf{V}_t)\le 0$$

for every t and therefore the value of the objective function monotonically improves with each iteration of the algorithm. Moreover, since

$$\mathbf{U}_{t+1}=\operatorname*{argmin}_{\mathbf{u}_i\in\Phi}\sum_{(i,j)\in\Omega}\big\|P_\Omega(\mathbf{R}-\mathbf{U}_t\mathbf{V}_{t+1}^T)\big\|_F^2,$$

it follows that

$$f(\mathbf{U}_{t+1},\mathbf{V}_t)-f(\mathbf{U}_t,\mathbf{V}_t)\le 0$$

at each iteration t of the algorithm. Therefore,

$$f(\mathbf{U}_{t+1},\mathbf{V}_{t+1})-f(\mathbf{U}_t,\mathbf{V}_t)\le 0$$

and the proposed structurally constrained gradient search algorithm converges.

Furthermore, it is easy to see that at convergence $\mathbf{V}^*$ is a local minimum of $f(\mathbf{U}^*,\mathbf{V})$ and hence the derivative with respect to $\mathbf{V}$ is 0. Thus $(\mathbf{U}^*,\mathbf{V}^*)$ is a stationary (i.e., zero-derivative) point.

Finally, suppose we had fresh samples Γ that were uniformly distributed and independent of Ω, $\mathbf{U}^*$ and $\mathbf{V}^*$. Note that, if we take an expectation over Γ, the loss function becomes $\|\mathbf{R}-\mathbf{U}\mathbf{V}^T\|$, where $\mathbf{R}$ is the true rank-one matrix. This particular function, while non-convex, has the special property that the global minimum is the only stationary point. Thus, if the size of Γ is large enough, concentration results will guarantee that if one iteration of the algorithm with these new samples significantly changes the $(\mathbf{U}^*,\mathbf{V}^*)$ pair, then this pair is not the global optimum; and conversely if $(\mathbf{U}^*,\mathbf{V}^*)$ do not change significantly then they are the global optimum pair. □

V. RESULTS

To obtain the numerical results in this section, we implemented our algorithms in Matlab and ran the codes on a single core processor laptop (2.7 GHz Intel Core i5, 8GB RAM).

A. Performance metrics

When the ground truth is known, as in simulation studies, the ability of an algorithm to reconstruct a haplotype may be measured by the reconstruction rate. In the case of diploids, the reconstruction rate is conveniently defined as [35]

$$R_r=1-\frac{\min(D(\mathbf{h}^1,\hat{\mathbf{h}}^1)+D(\mathbf{h}^2,\hat{\mathbf{h}}^2),D(\mathbf{h}^1,\hat{\mathbf{h}}^2)+D(\mathbf{h}^2,\hat{\mathbf{h}}^1))}{2m}$$

where $D(\mathbf{h}^i,\hat{\mathbf{h}}^j)=\sum_{l=1}^{m}d(h_l^i,\hat{h}_l^j)$ denotes the generalized Hamming distance between $\mathbf{h}^i$ and $\hat{\mathbf{h}}^j$, $(\mathbf{h}^1,\mathbf{h}^2)$ is the pair of true haplotypes, and $(\hat{\mathbf{h}}^1,\hat{\mathbf{h}}^2)$ is the pair of reconstructed haplotypes. A related measure of performance is the switch error rate (SWER), i.e., the rate at which switches occur in

TABLE I: *Comparison of the reconstruction rates of Algorithm 1, Algorithm 2, and Algorithm 3 on bi-allelic diploid haplotypes of length* 700 *using data from [35].*

Data error rate	Sequencing coverage	Algo. 1	Algo. 2	Algo. 3
0.1	5	0.6701	0.9513	0.9458
0.1	8	0.6758	0.9965	0.9942
0.1	10	0.6770	0.9986	0.9980
0.2	5	0.6233	0.7850	0.7250
0.2	8	0.6460	0.8992	0.8925
0.2	10	0.6577	0.9340	0.9291
0.3	5	0.5301	0.6070	0.5561
0.3	8	0.5547	0.6430	0.6115
0.3	10	0.5726	0.7021	0.6735

the reconstructed haplotype sequences. The switch between positions l and $l+1$ in the i^{th} haplotype sequence is defined as the event $\hat{h}_l^i = h_l^i$ and $\hat{h}_{l+1}^i \neq h_{l+1}^i$, or $\hat{h}_l^i \neq h_l^i$ and $\hat{h}_{l+1}^i = h_{l+1}^i$.

Another frequently used metric for quantification of the accuracy of haplotype assembly is the minimum error correction (MEC) score. For an arbitrary $k \geq 2$,

$$Z = \sum_{i=1}^{n} \min(D(\mathbf{r}_i, \mathbf{h}^1), \cdots, D(\mathbf{r}_i, \mathbf{h}^k)), \tag{14}$$

where $D(\mathbf{r}_i, \mathbf{h}) = \sum_{j:(i,j)\in\Omega} d(R(i,j), h_j)$ is the generalized Hamming distance, and where $R(i,j)$ is the (i,j) entry in $\mathbf{R}$. Note that we define the distance d between two symbols in the ternary alphabet $\{-1, 1, 0\}$ as

$$d(a,b) = \begin{cases} 1 \text{ if } a \neq 0 \text{ and } b \neq 0 \text{ and } a \neq b, \\ 0 \text{ otherwise.} \end{cases}$$

The MEC score quantifies the smallest number of informative entries (non-zeros) in $\mathbf{R}$ that should be changed (flipped) so that $\mathbf{R}$ could be interpreted as a noise-free sample of $\mathbf{M}$ – i.e., the MEC score is the most likely number of errors in $P_\Omega(\mathbf{R})$. Most of the existing haplotype assembly methods have been designed with the goal of minimizing the MEC score [36]. The algorithm that we propose in Section III essentially greedily minimizes the MEC score. The MEC score metric is particularly useful when dealing with experimental data where ground truth is not known and thus the MEC score serves as a proxy for the reconstruction rate and SWER metrics.

B. Simulation results

To benchmark the proposed algorithms, we first perform tests on synthetic data. For the diploid case (i.e., rank-1 $\mathbf{M}$ that is characteristic of, e.g., human genome), we rely on the data sets from [35] often used for comparing methods for haplotype assembly of diploids (these data sets emulate haplotype assembly under varied coverage, sequencing error rates and haplotype block lengths; we omit the results for extremely low coverage). For the polyploid case ($k > 2$), we synthesize the data that emulates latest sequencing technologies capable of assembly of long haplotype blocks and use it for a comparison with competing methods.

For the sake of compactness of the presentation, before comparing proposed algorithms with existing haplotype assembly methods we first directly compare Algorithms 1, 2 and 3 from Section III. Their accuracy when applied to the assembly of haplotype blocks of length 700 in [35], expressed in terms of the reconstruction rates, is shown in Table I. Note that the sequencing coverage in Table I is the average number of times each position in a haplotype is sampled by the sequencing reads (i..e, the average number of reads that "cover" a SNP position). As can be seen from Table I, structurally-constrained gradient descent (Algorithm 2) consistently outperforms the other two methods; we observed the same behavior in other experiments. Therefore, for the remaining benchmarking studies, we show only the comparison of Algorithm 2 with the existing haplotype assembly methods.

*Rank-*1 *case: haplotype assembly for diploids.* Using the data sets from [35], we compare the performance of our Algorithm 2 with several existing haplotype assembly methods and report the results in Table II and Table III. Specifically, we show the comparison of the achieved reconstruction rates with those of HapCut [23], SpeedHap [37], FastHare [38], 2d-med [39], MLF [40] and SHR-tree [41]. As evident from the table, structurally-constrained gradient descent (SCGD) outperforms all the competing methods in most of the situations (with the exception of few scenarios characterized by extremely low sequencing coverage). Note that the percentage of observed entries in $\mathbf{R}$ in Tables II and III is 2% and 1%, respectively.

Rank-k, $k > 2$ case: haplotype assembly for polyploids. For higher rank cases, we synthesize haplotype sequences by setting the distance between each pair of adjacent SNPs at random. In particular, the space between adjacent SNPs is generated as an instance of a geometric random variable with parameter p_{SNP} (the SNP rate). We simulate sequencing process suited for the reconstruction of long haplotype blocks by randomly generating paired-end reads having length 500bp and inserts having average length 10^4bp and standard deviation 10^3. We insert errors into the generated reads at the rates typical of state-of-the-art sequencing platforms [29], [30]. Note that the genotype calling (i.e., determining most likely pair of SNPs in each position of the haplotype sequences) is performed using a Bayesian approach [33] (details omitted for brevity).

Unlike the diploid case where a slew of haplotype assembly methods exist, few techniques for the assembly of polyploid haplotypes are available. We make a comparison with a recently developed HapCompass [24] and the belief propagation algorithm in [27], and show the results in Table IV (tetraploid data) and Table V (hexaploid data). As seen there, our method significantly outperforms the competing techniques in terms of the MEC scores and runtime in all scenarios; it also achieves the best SWERs except for scenarios with very low sequencing coverage.

TABLE II: *Reconstruction rates for several haplotype assembly algorithms on diploid data from [35], block length* $m = 350$. *The best results are in boldface fonts.*

Data error rate	Coverage	Structurally-constrained grad. descent	SPH	FAST	2d	MLF	SHR	Hap-Cut	BP
0.1	5	0.9587	**0.9592**	0.9453	0.9130	0.8582	0.7244	0.9130	0.8670
0.1	8	**0.9957**	0.9843	0.9852	0.9641	0.9327	0.7416	0.9641	0.8722
0.1	10	**0.9978**	0.9836	0.9948	0.9781	0.9616	0.7285	0.9781	0.8727
0.2	5	0.7872	0.7287	0.7456	0.7284	0.7278	0.6318	**0.8306**	0.8194
0.2	8	**0.8802**	0.8247	0.8529	0.7912	0.7985	0.6699	0.8616	0.8607
0.2	10	**0.9482**	0.8555	0.8774	0.8169	0.8314	0.6682	0.8672	0.8672
0.3	5	**0.6262**	0.5784	0.6021	0.6061	0.6063	0.5575	0.5817	0.5386
0.3	8	**0.6656**	0.6294	0.6259	0.6230	0.6339	0.6043	0.6206	0.5715
0.3	10	**0.6967**	0.6381	0.6437	0.6340	0.6408	0.6189	0.6641	0.5956

TABLE III: *Reconstruction rates for several haplotype assembly algorithms on diploid data from [35], block length* $m = 700$. *The best results are in boldface fonts.*

Data error rate	Coverage	Structurally-constrained grad. descent	SPH	FAST	2d	MLF	SHR	Hap-Cut	BP
0.1	5	**0.9513**	0.9471	0.9408	0.8805	0.8094	0.7158	0.9158	0.8704
0.1	8	**0.9965**	0.9848	0.9859	0.9483	0.8632	0.7429	0.8957	0.8716
0.1	10	**0.9986**	0.9861	0.9955	0.9649	0.8839	0.7260	0.8892	0.8741
0.2	5	0.7850	0.6810	0.7118	0.6969	0.6820	0.6171	**0.8250**	0.8119
0.2	8	**0.8992**	0.8006	0.8078	0.7512	0.7475	0.6529	0.8562	0.8545
0.2	10	**0.9340**	0.8127	0.8719	0.7780	0.7650	0.6748	0.8610	0.8610
0.3	5	**0.6070**	0.5232	0.5915	0.5961	0.5944	0.5622	0.5553	0.5224
0.3	8	**0.6430**	0.6158	0.6147	0.6126	0.6139	0.6113	0.5966	0.5500
0.3	10	**0.7021**	0.6271	0.6165	0.6219	0.6248	0.6251	0.6455	0.5881

TABLE IV: *SWER rates and MEC scores for the structurally constrained gradient descent (Algorithm 2 from Section III), HapCompass [24] and belief propagation [27] when applied to the assembly of biallelic tetraploid* ($k = 4$) *haplotypes with block length* $m = 1000$. *The best results are in boldface fonts.*

Error Rate	Coverage	Structurally-constrained GD			HapCompass			Belief Propagation		
		SWER	MEC	Time(s)	SWER	MEC	Time(s)	SWER	MEC	Time(s)
0.002	5	0.0927	**1558**	**64.1**	0.1105	3510	1913.3	**0.0680**	3641	171.3
0.002	10	**0.0516**	**2422**	**142.3**	0.0837	7298	1329.5	0.0645	7550	172.7
0.002	15	**0.0373**	**3123**	**209.1**	0.0738	12239	2482.2	0.0639	11436	386.4
0.01	5	0.0831	**1729**	**98.4**	0.1113	3461	11160.3	**0.0700**	3727	362.7
0.01	10	**0.0584**	**2807**	**164.8**	0.0807	7772	856.1	0.0691	7715	719.7
0.01	15	**0.0345**	**3382**	**199.8**	0.0853	12195	1666.4	0.0687	11752	810.6
0.05	5	0.0962	**2374**	**93.7**	0.0983	4873	2749.0	**0.0788**	4117	162.2
0.05	10	**0.0632**	**4341**	**175.1**	0.0807	10323	1556.4	0.0808	8534	412.7
0.05	15	**0.0415**	**5722**	**285.3**	0.0853	16030	1201.0	0.0855	13018	703.4

TABLE V: *SWER rates and MEC scores for the structurally constrained gradient descent (Algorithm 2 from Section III), HapCompass [24] and belief propagation [27] when applied to the assembly of biallelic hexaploid ($k = 6$) haplotypes with block length $m = 1000$. The best results are in boldface fonts.*

Error Rate	Coverage	Structurally-constrained GD			HapCompass			Belief Propagation		
		SWER	MEC	Time(s)	SWER	MEC	Time(s)	SWER	MEC	Time(s)
0.002	5	0.0814	**1823**	**81.6**	0.0862	4548	8075.3	**0.0762**	4958	160.6
0.002	10	**0.0452**	**3033**	**127.3**	0.0595	9283	7277.2	0.0806	10339	161.3
0.002	15	**0.0362**	**3864**	**168.3**	0.0567	15007	11841.0	0.0764	15713	193.8
0.01	5	0.1035	**2211**	**120.3**	0.1564	4561	6744.2	**0.0602**	5009	440.8
0.01	10	**0.0606**	**3615**	**193.7**	0.0677	9560	3595.0	0.0676	10479	442.8
0.01	15	**0.0413**	**5525**	**305.9**	0.0571	14072	5041.3	0.0590	15851	432.4
0.05	5	0.0765	**3021**	**102.3**	0.0908	5285	2736.2	**0.0568**	5445	1063.3
0.05	10	**0.0504**	**5724**	**181.7**	0.0591	11107	1423.8	0.0572	11319	587.8
0.05	15	**0.0452**	**7229**	**386.3**	0.0595	16871	1668.1	0.0596	17115	1176.5

C. Experimental results

We further tested the performance of our proposed algorithm on the experimental data generated as part of the *1000 Genomes Project*, an international study meant to provide a detailed map of human genetic variation [42]. In particular, we used the data for an individual NA12878 (often used to benchmark performance of different haplotype assembly algorithms) consisting of reads generated by the 454 sequencing platform (the data is available for download from ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/). The MEC scores and the runtimes of the structurally constrained gradient search are compared to those of HapCompass [24] and belief propagation [27] in Table VI (for all 22 homologous chromosomes). As can be seen there, our proposed algorithm achieves better accuracy at significant improvement in speed as compared to the competing schemes.

TABLE VI: *The MEC scores and runtimes for the structurally constrained gradient descent (Algorithm 2 from Section III), HapCompass [24] and belief propagation [27] when applied to the experimental data generated by the 1000 Genomes Project [42].*

chr	SCGD (Alg. 2)		HapCompass		BP	
	MEC	time(s)	MEC	time(s)	MEC	time(s)
1	1300	3.35	1496	9468.7	1488	29.2
2	1763	4.84	1938	10971.0	1921	28.7
3	1434	4.27	1627	8878.1	1615	29.0
4	1663	6.74	1863	9859.5	1849	31.4
5	1330	4.37	1505	8623.8	1488	26.2
6	2326	19.21	2771	8969.2	2719	27.5
7	1262	5.60	1423	8076.2	1417	22.4
8	1177	4.01	1261	8613.7	1255	23.6
9	895	2.92	1007	6145.0	1004	17.2
10	1093	3.73	1266	6576.4	1257	22.9
11	967	3.02	1121	6042.9	1115	19.7
12	915	2.60	1052	5610.8	1042	16.9
13	883	3.11	969	4942.0	960	14.6
14	561	1.84	658	4011.5	656	11.4
15	612	1.49	690	3679.4	686	10.1
16	760	2.73	889	4431.6	885	12.2
17	795	2.83	902	2745.5	895	7.7
18	594	1.63	651	3986.7	650	11.1
19	495	1.12	523	2303.9	517	6.4
20	459	1.39	496	2897.8	491	7.9
21	331	1.05	363	2046.7	354	5.4
22	232	0.73	269	1688.6	269	4.6

VI. Conclusion

We proposed and analyzed the structurally-constrained gradient descent algorithm for factorizing partially observed low-rank matrices that arise in haplotype assembly, i.e., in the problem of reconstructing haplotypes from high-throughput DNA sequencing reads. The rows of data matrices encountered in this application correspond to sequencing reads; each read is aligned to a reference and spans only those columns associated with single nucleotide polymorphisms covered by the read. Since the reads are much shorter than the haplotype blocks, most of the entries in each row of the data matrix are missing – hence the matrix is only partially observed. Moreover, each row can be thought of as being sampled from one among few haplotype sequences – therefore, the matrix is low rank. Finally, since the sequencing is erroneous, the observed matrix contains incorrect entries.

The structurally-constrained gradient descent algorithm imposes the special structure of the matrices in the sought after decomposition. Specifically, one of the matrices in the factorization consists of rows that are standard basis. We impose that constraint in the gradient descent to achieve significant improvement over the naive structure-unaware gradient descent algorithm. We analyzed the convergence of the proposed algorithm and extensively tested its performance on both synthetic and experimental data, comparing it with several state-of-the-art methods for haplotype assembly. The results demonstrate superior performance of the developed technique in terms of both accuracy and speed over competing schemes.

The present work is application driven and focused on developing a practical solution to the haplotype assembly problem and providing fundamental convergence guarantees of the proposed algorithm. As part of the future theoretical work, it is of interest to establish performance guarantees by determining bounds on the fraction of matrix entries that need to be observed in order to provide a desired level of accuracy. Moreover, it is of interest to explore related applications of high-throughput DNA sequencing including tumor haplotype assembly and viral quasispecies reconstruction.

APPENDIX

A. Proof of Lemma 1

Let $\mathbf{E}_{ij}$ and $\mathbf{E}_{st}$ be the $m \times k$ matrices having entry 1 in the (i, j) and (s, t) position, respectively, and zeros elsewhere. The Hessian of a function $f(\mathbf{U}, \mathbf{V})$ with respect to the $m \times k$ matrix $\mathbf{V}$ is given by (see, e.g., [43])

$$\mathbf{H}_V = \sum_{i=1}^{m}\sum_{j=1}^{k}\sum_{s=1}^{m}\sum_{t=1}^{k} \Big(vec\frac{\partial^2 f(\mathbf{U},\mathbf{V})}{\partial V_{st}\partial V_{ij}}\Big)\otimes(vec\mathbf{E}_{ij})(vec\mathbf{E}_{st})^T, \tag{15}$$

where $\otimes$ denotes the Kronecker product and $vec\mathbf{A}$ denotes a vectorized version of the matrix $\mathbf{A}$, i.e., if $\mathbf{A}$ consists of columns $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_l$ then $vec\mathbf{A} = [\mathbf{a}_1^T \quad a_2^T \quad \ldots \quad a_l^T]^T$. Note that in our problem $f(\mathbf{U}, \mathbf{V})$ is a scalar function and, therefore, so is the term $\big(vec\frac{\partial^2 f(\mathbf{U},\mathbf{V})}{\partial V_{st}\partial V_{ij}}\big)$ in (15). Moreover, note that $vec\mathbf{E}_{ij}$ is an $mk \times 1$ vector having entry 1 in the $((j-1)m+i)^{th}$ position and zeros elsewhere and $vec\mathbf{E}_{st}$ is an $mk \times 1$ vector having entry 1 in the $((t-1) \times m + s)^{th}$ position and zeros elsewhere. Therefore, $(vec\mathbf{E}_{ij})(vec\mathbf{E}_{st})^T$ is an $mk \times mk$ matrix having 1 in the $((j-1)m+i, (t-1)m+s)$ position and zeros elsewhere. When $i = s$ and $t = j$, this is a diagonal matrix.

It is straightforward to show that

$$\frac{\partial f(\mathbf{U},\mathbf{V})}{\partial V_{st}} = -2\sum_{(l,s)\in\Omega}\big(R_{ls} - \sum_{p=1}^{k} U_{lp}V_{sp}\big)U_{lt}. \tag{16}$$

Furthermore,

$$\frac{\partial^2 f(\mathbf{U},\mathbf{V})}{\partial V_{st}\partial V_{ij}} = \begin{cases} 0 & \text{if } s \neq i, \\ 2\sum\limits_{(l,i)\in\Omega} U_{lt}U_{lj} & \text{if } s = i. \end{cases} \tag{17}$$

Due to the constraint that $\mathbf{u}_i$ is a standard basis, $U_{lt}U_{lj} = 0$ for $t \neq j$. Therefore, $\frac{\partial^2 f(\mathbf{U},\mathbf{V})}{\partial V_{st}\partial V_{ij}}$ is non-zero only if $s = i$ and $t = j$, i.e.,

$$\frac{\partial^2 f(\mathbf{U},\mathbf{V})}{\partial V_{st}\partial V_{ij}} = 2\sum_{(l,i)\in\Omega} U_{lj}^2\delta(s-i, t-j), \tag{18}$$

where

$$\delta(s-i, t-j) = \begin{cases} 1, & \text{if } s = i \text{ and } t = j, \\ 0, & \text{otherwise.} \end{cases}$$

Note that this second derivative is always non-negative.

Since $\frac{\partial^2 f(\mathbf{U},\mathbf{V})}{\partial V_{st}\partial V_{ij}}$ is non-zero only when $i = s$ and $t = j$, $(vec\frac{\partial^2 f(\mathbf{U},\mathbf{V})}{\partial V_{st}\partial V_{ij}})\otimes(vec\mathbf{E}_{ij})(vec\mathbf{E}_{st})^T$ is not an all-zeros matrix only when $i = s$ and $t = j$; this implies that all the non-zero terms in the (15) are diagonal matrices. Moreover, $\mathbf{H}_V$ is a diagonal matrix since it is obtained as a summation of diagonal matrices. Its diagonal element $\frac{\partial^2 f(\mathbf{U},\mathbf{V})}{\partial V_{ij}^2}$ in the $((j-1) \times m + i, (j-1) \times m + i)$ position is given by (18) and is clearly non-negative.

As a result, the Hessian matrix $\mathbf{H}_V$ is positive semi-definite, as stated by the lemma.

□

REFERENCES

[1] H. Kim and H. Park, "Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 2, pp. 713–730, July 2008.

[2] E. Candes and Y. Plan, "Matrix completion with noise," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 925–936, 2010.

[3] R. H. Keshavan, A. Montanari, and S. Oh, "Matrix completion from a few entries," *IEEE Transactions on Information Theory*, vol. 56, no. 6, pp. 2980–2998, 2010.

[4] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010.

[5] C. Chen, B. He, and X. Yuan, "Matrix completion via an alternating direction method," *Journal of Numerical Analysis*, vol. 32, no. 1, pp. 227–245, 2012.

[6] P. Netrapalli, P. Jain, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," in *Symposium on Theory of Computing (STOC)*, 2013.

[7] "Netflix prize." [Online]. Available: http://www.netflixprize.com/.

[8] J. Kim and H. Park, "Sparse nonnegative matrix factorization for clustering," Technical Report GT-CSE-08-01, Georgia Institute of Technology, Tech. Rep., 2008.

[9] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *Journal of Computational and Graphical Statistics*, vol. 15, no. 2, pp. 262–286, 2006.

[10] Y. Koren, R. M. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[11] N. J. Loman, R. V. Misra, T. J. Dallman, C. Constantinidou, S. E. Gharbia, J. Wain, and M. J. Pallen, "Performance comparison of benchtop high-throughput sequencing platforms," *Nat Biotech*, vol. 30, no. 5, pp. 434–439, 05 2012. [Online]. Available: http://dx.doi.org/10.1038/nbt.2198

[12] A. G. Clark, "The role of haplotypes in candidate gene studies," *Genetic epidemiology*, vol. 27, no. 4, pp. 321–333, 2004.

[13] R. A. Gibbs, J. W. Belmont, P. Hardenbol, T. D. Willis, F. Yu, H. Yang, L.-Y. Ch'ang, W. Huang, B. Liu, Y. Shen *et al.*, "The international hapmap project," *Nature*, vol. 426, no. 6968, pp. 789–796, 2003.

[14] P. C. Sabeti, D. E. Reich, J. M. Higgins, H. Z. Levine, D. J. Richter, S. F. Schaffner, S. B. Gabriel, J. V. Platko, N. J. Patterson, G. J. McDonald *et al.*, "Detecting recent positive selection in the human genome from haplotype structure," *Nature*, vol. 419, no. 6909, pp. 832–837, 2002.

[15] R. Sachidanandam, D. Weissman, S. C. Schmidt, J. M. Kakol, L. D. Stein, G. Marth, S. Sherry, J. C. Mullikin, B. J. Mortimore, D. L. Willey *et al.*, "A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms," *Nature*, vol. 409, no. 6822, pp. 928–933, 2001.

[16] H. Si, H. Vikalo, and S. Vishwanath, "Haplotype assembly: An information-theoretic view," in *Proceedings of the IEEE Information Theory Workshop*, 2014, pp. 182–186.

[17] G. Lancia, V. Bafna, S. Istrail, R. Lippert, and R. Schwartz, "Snps problems, complexity, and algorithms," in *Algorithms—ESA 2001*. Springer, 2001, pp. 182–193.

[18] R. Cilibrasi, L. Van Iersel, S. Kelk, and J. Tromp, "On the complexity of several haplotyping problems," in *Algorithms in Bioinformatics*. Springer, 2005, pp. 128–139.

[19] R.-S. Wang, L.-Y. Wu, Z.-P. Li, and X.-S. Zhang, "Haplotype reconstruction from snp fragments by minimum error correction," *Bioinformatics*, vol. 21, no. 10, pp. 2456–2462, 2005.

[20] S. Levy, G. Sutton, P. C. Ng, L. Feuk, A. L. Halpern, B. P. Walenz, N. Axelrod, J. Huang, E. F. Kirkness, G. Denisov *et al.*, "The diploid genome sequence of an individual human," *PLoS biology*, vol. 5, no. 10, p. e254, 2007.

[21] V. Bansal, A. L. Halpern, N. Axelrod, and V. Bafna, "An MCMC algorithm for haplotype assembly from whole-genome sequence data," *Genome research*, vol. 18, no. 8, pp. 1336–1346, 2008.

[22] J. H. Kim, M. S. Waterman, and L. M. Li, "Diploid genome reconstruction of ciona intestinalis and comparative analysis with ciona savignyi," *Genome research*, vol. 17, no. 7, pp. 1101–1110, 2007.

[23] V. Bansal and V. Bafna, "Hapcut: an efficient and accurate algorithm for the haplotype assembly problem," *Bioinformatics*, vol. 24, no. 16, pp. i153–i159, 2008.

[24] D. Aguiar and S. Istrail, "Hapcompass: a fast cycle basis algorithm for accurate haplotype assembly of sequence data," *Journal of Computational Biology*, vol. 19, no. 6, pp. 577–590, 2012.

[25] J. Duitama, G. K. McEwen, T. Huebsch, S. Palczewski, S. Schulz, K. Verstrepen, E.-K. Suk, and M. R. Hoehe, "Fosmid-based whole genome haplotyping of a hapmap trio child: evaluation of single individual haplotyping techniques," *Nucleic acids research*, p. gkr1042, 2011.

[26] S. Das and H. Vikalo, "SDhaP: Haplotype assembly for diploids and polyploids via semi-definite programming," *BMC Genomics*, vol. 16:260, April 2015.

[27] Z. Puljiz and H. Vikalo, "Decoding genetic variations: Communications-inspired haplotype assembly," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2016.

[28] B. Eriksson, L. Balzano, and R. Nowak, "High-rank matrix completion," in *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, La Palma, Canary Islands, 2012, pp. 373–381.

[29] S. Das and H. Vikalo, "Onlinecall: fast online parameter estimation and base calling for illumina's next-generation sequencing," *Bioinformatics*, vol. 28, no. 13, pp. 1677–83, 2012.

[30] ——, "Base calling for high-throughput short-read sequencing: Dynamic programming solutions," *BMC Bioinformatics*, vol. 14, no. 129, 2013.

[31] H. Li and R. Durbin, "Fast and accurate short-read alignment with burrows-wheeler transform," *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, 2009.

[32] ——, "Fast and accurate long-read alignment with burrows-wheeler transform," *Bioinformatics*, vol. 26, no. 5, pp. 589–595, 2010.

[33] M. A. DePristo, E. Banks, R. Poplin, K. V. Garimella, J. R. Maguire, C. Hartl *et al.*, "A framework for variation discovery and genotyping using next-generation dna sequencing data," *Nature Genetics*, vol. 43, no. 5, pp. 491–8, 2011.

[34] E. Berger, D. Yorukoglu, J. Peng, and B. Berger, "Haptree: A novel bayesian framework for single individual polyplotyping using ngs data," *PLoS Computational Biology*, vol. 10, no. 3, 2014.

[35] F. Geraci, "A comparison of several algorithms for the single individual snp haplotyping reconstruction problem," *Bioinformatics*, vol. 26, no. 18, pp. 2217–2225, 2010.

[36] R. Schwartz *et al.*, "Theory and algorithms for the haplotype assembly problem," *Communications in Information & Systems*, vol. 10, no. 1, pp. 23–38, 2010.

[37] L. Genovese *et al.*, "Speedhap: an accurate heuristic for the single individual snp haplotyping problem with many gaps, high reading error rate and low coverage," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 5, pp. 492–502, 2008.

[38] A. Panconesi and M. Sozio, "Fast hare: A fast heuristic for single individual snp haplotype reconstruction," *Algorithms in Bioinformatics*, no. 3240, pp. 266–277, 2004.

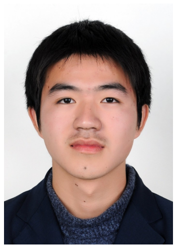
[39] Y. Wang *et al.*, "A clustering algorithm based on two distance functions for mec model," *Computational Biology and Chemistry*, no. 31, pp. 148–150, 2007.

[40] Y. Zhao *et al.*, "Haplotype assembly from aligned weighted snp fragments," *Computational Biology and Chemistry*, no. 29, pp. 281–287, 2005.

[41] Z. Chen *et al.*, "Linear time probabilistic algorithms for the singular haplotype reconstruction problem from snp fragments," *Journal of Computational Biology*, no. 15, pp. 535–546, 2008.

[42] The 1000 Genomes Project Consortium, "A map of human genome variation from population-scale sequencing," *Nature*, vol. 467, no. 7319, pp. 1061–1073, 2010.

[43] J. R. Magnus and H. Neudecker, *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Wiley, 1999.



Changxiao Cai is a fourth year undergraduate student in the Department of Electronic Engineering at Tsinghua University, Beijing, China. From January 2015 to June 2015, he was an exchange student at UT Austin.

Sujay Sanghavi is an Associate Professor in ECE at UT Austin. He has an MS each in ECE and Mathematics, and a PhD in ECE, all from the University of Illinois; and prior to that a B. Tech from IIT Bombay. Sujay's research interests are in Machine Learning, Optimization, Algorithms and Networks. He is a recipient of the NSF Career award and the DTRA Young Investigator award.

Haris Vikalo received the B.S. degree from the University of Zagreb, Croatia, in 1995, the M.S. degree from Lehigh University in 1997, and the Ph.D. degree from Stanford University in 2003, all in electrical engineering. He held a short-term appointment at Bell Laboratories, Murray Hill, NJ, in the summer of 1999. From January 2003 to July 2003 he was a Postdoctoral Researcher, and from July 2003 to August 2007 he was an Associate Scientist at the California Institute of Technology. Since September 2007, he has been with the Department of Electrical and Computer Engineering, the University of Texas at Austin, where he is currently an Associate Professor. He is a recipient of the 2009 National Science Foundation Career Award. His research interests include signal processing, bioinformatics, machine learning, and communications.