The Dissertation Committee for Zrinka Puljiz
certifies that this is the approved version of the following dissertation:

State Reconstruction from Partial Observations: Theory and Applications

Committee:

Constantine Caramanis, Supervisor

Sanjay Shakkottai, Supervisor

John Hasenbein

Sujay Sanghavi

Haris Vikalo

Sriram Vishwanath

State Reconstruction from Partial Observations: Theory and Applications

by

Zrinka Puljiz, B.S.; B.S.; M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2015

Dedicated to my family.

State Reconstruction from Partial Observations: Theory and Applications

Publication No. _____

Zrinka Puljiz, Ph.D.
The University of Texas at Austin, 2015

Supervisors: Constantine Caramanis
Sanjay Shakkottai

This thesis considers the problem of signal reconstruction in the setting of partial observations. We consider this in three different contexts. First, we consider the abstract problem of determining the state of a time-varying vector at discrete time instances, in a setting where the changes are sparse (i.e., only a few components of the vector change at each time). Given a collection of observers, each of which is able to provide linear mixtures of a subset of the components, the goal is to determine the vector values at certain desired time instants. We derive conditions on the number of measurements, observers, and their corresponding subsets that lead to exact reconstruction (with high probability) of the signal at the desired time instants. We then propose an iterative algorithm that can achieve such a reconstruction, and present simulation results to demonstrate its performance. Furthermore, we propose an ℓ_1 relaxed heuristic and simulate its performance.

Second, we study an application of this problem in structured wireless networks, where the overlap between pilot signals impacts channel estimation. Channel estimation is a critical aspect of wireless communications; thus, pilot contamination can significantly reduce the system capacity. We apply the multi-observer signal recovery framework in this setting, and develop algorithms for distributed channel estimation.

Third, we study an application to haplotype assembly, where the haplotype is sequence of (location, nucleotide) pairs in a chromosome that vary over the human population. Here, the underlying signal (halpotype) can be modeled to have binary support, and does not change in time. However, the information about signal – multiple partially overlapping fragments (each fragment associated with a different virtual observer) – is noisy and incomplete. We design fast sequence reconstruction algorithms.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this thesis we explore the topic of signal reconstruction from partial information. Detection or estimation using partial measurements is a classical problem, and it is important due to its prevalence in a wide array of applications. On the analytical side, recovery from partial information requires developing new tools and analysis techniques.

In sensor networks, as well as a variety of other application domains, partial and noisy measurements are a natural phenomenon. Partial measurements may be the result of geographically separated sensors taking only local measurements of a global phenomenon. Partial measurements may also be the consequence of the measurement approach itself, as is the case in network tomography, where different measurements are influenced by only parts of the network. We consider two different measurement or sensing regimes, with partial information: time varying system with sparse changes and partial observations; and structured sparse measurements of a still signal. In this Chapter, we first provide an overview of the two systems and outline the objectives of the signal reconstruction for each of the two systems. We also provide a roadmap for the rest of this document.

Tracking and State Reconstruction with Multiple Observers: Here we consider the recovery of a time varying signal that undergoes sparse changes from partial observations. Tracking a signal that changes over time is also a classical and important problem in control, signal processing and networking. In a comparatively new development under the moniker of compressed sensing, new results have emerged for estimating sparse signals, from very few measurements. These results have proved important in statistics, signal processing and machine learning. However, these results do not extend to time varying signals with sparse changes. Our problem differs from compressed sensing in three major aspects. First, in compressed sensing each measurement is a linear combination of each element of the signal, while our measurements each measure (different) subsets of the components. Second, in compressed sensing, the signal is static, and hence each measurement is a linear combination of the same signal. Here, in contrast, subsequent measurements may measure a different signal, since the signal itself is dynamic. If the changes in the signal from one time step to the next were completely arbitrary, the problem would essentially reduce to the single-shot problem, since at each time we would be estimating from scratch. Instead, we consider a setting of a gradually changing vector, where by *gradual change*, we mean that our signal changes sparsely over time. Finally, the goal of compressed sensing is to recover the complete sparse signal – the change vector; we are interested in resynchronization at some time step.

A few more comments about the sparse change assumption, and our

multiple-observer regime are in order. As mentioned, the basic structural assumption of our problem is that the signal changes in only a few coordinates (i.e., *sparsely*) from one time step to the next. Except from sparsity, there are no additional restrictions on the change process, i.e. it can affect any of the components of the signal and it can change the values of the components in an arbitrary way (each component of the signal is an element in $\mathbb{R}$). By a *sparse change*, we mean that the number of changes per time slot is much smaller than the number of components in the signal. The signal is measured by a collection of observers who make partial measurements. That is, a given observer makes a *scalar* measurement that represents the linear combination of only a subset of the coefficients of the signal; this subset of coefficients may be different from the subset corresponding to the coefficients measured by another observer.

Another way to think of these measurements, is that each observation corresponds to a random projection of the vector corresponding to the subset of observed components. When dealing with partial observations, the regime of most interest is precisely the setting where it is not possible for each individual observer to decode its part of the signal at each point in time. That is, collaboration is required.

We characterize the conditions under which resynchronization is possible and develop low complexity algorithms that achieve this.

Next we look into a specific application of the partial and time varying observations: the pilot contamination problem. This is an application of mul-

tidimensional signal recovery from limited/partial observations in structured wireless networks, where the overlap between pilot signals impacts channel estimation. Channel estimation is a critical aspect of wireless communications; thus, pilot contamination [41] can significantly reduce the system capacity. In massive MIMO systems with time division duplex, due to the channel reciprocity the signal can efficiently be estimated only at the base station and this estimation can be used for both uplink and downlink communication.

We look into this system from a point of view where the coherence time of the channel varies for different users with additional assumption that for most of the users the coherence time is long, i.e. the users channel coefficients change slowly in time. The sparsely changing signal in our setting is the signal of user's channel coefficients. The observations consists of linear mixtures of channel impaired pilot sequences from all the users within the range of the base station.

The formulation of estimating channel coefficients in massive MIMO TDD systems as a recovery of time varying sparsely changing signal is novel. Furthermore, we propose the use of non-orthogonal pilot sequences that are assigned in a random manner, as such sequences have provenly good signal recovery properties in the compressed sensing setting. In addition, there is no need to manage the assignment of the pilot sequence to users, as well as no need for base station collaboration.

We develop algorithms for pilot sequence recovery and channel estimation, and simulate the performance of the massive MIMO TDD system

with respect to system capacity, channel estimation error and the number of admissible users.

Still signal reconstruction from structured sparse measurement: The second system of interest consists of a static (non-dynamic) system with partial observations. The support of the signal and measurements is binary. Each observation consists of inverted or direct set of observed components, and each element is potentially erroneous. This setup corresponds to the problem of haplotype assembly [72]. A haplotype is a sequence of (location,nucleotide) pairs in a chromosome – these locations are the ones where the nucleotide type differs across the population (in most locations, the nucleotide is the same for all members of the population). We are interested in single individual haplotyping. In this case the focus is on finding a genetic variation between chromosomes in a single individual. The partial observations here correspond to fragments (sub-sequences) of haplotypes which are both error prone as well as varying in length/coverage. The goal is to reassemble the entire haplotype from these fragments.

The goodness of the reassembled haplotype is a relative term, since for real signals the underlying true haplotype is not available for comparison. Therefore, in haplotype assembly usually measuring the accuracy of the solution is usually done through proximity measurements. One such measurement is MEC (minimize error correction) score where the goal is to minimize the number of corrections for the observations that would lead to consistent haplotype.

Our key observation in haplotype assembly problem is that haplotype assembly can be viewed as a decoding a noise impaired signal. This is in a strong contrast to the usual formulation of haplotype assembly as max cut problem. Under this formulation, the MEC score is strongly connected to the problem of finding sparsest error vector consistent with a linear transformation of the observations, and corresponds to the ℓ_0 norm of the error vector.

Further, we derive a bit flipping algorithm for haplotype assembly and a belief propagation algorithm. Both of these algorithms give competitive results when compared to max cut based algorithms. In addition, we extend out belief propagation algorithm to polyploid case, where the number of chromosomes is greater than 2. Both algorithms have low time complexity, and belief propagation algorithm can be very easily parallelized, thus leading to potentially O(1) execution time.

1.1 Organization

We give the summary of our work on the tracking and state reconstruction problem in the next Chapter. We first discuss the system model and then describe our results, proposed algorithms and in the end the simulations. In Chapter 3, we present the pilot contamination problem and our contributions. We first show the system and communication model, and build towards benefits of using the non-orthogonal pilots and exploiting the time correlation, we end the Chapter with the simulation results. Chapter 4 introduces in detail the problem of haplotype assembly and gives the novel approach on the hap-

lotype assembly through the perspective of signal decoding. We introduce the algorithms for both diploid as polyploid case, and simulate their performance on both simulated and real datasets. Finally, in Chapter 5 we summarize the presented work and give possible directions for the future work.

Chapter 2

Tracking and State Reconstruction with Multiple Observers

2.1 Introduction

We consider a (multi-dimensional) signal that changes sparsely over time. A collection of sensors make partial measurements – partial in that a given measurement represents the combination of only a subset of the coefficients of the signal. One sensor can get multiple measurements, while each measurements itself is a scalar value. Note that this is markedly different from the compressed sensing setup, where each measurement is a linear combination of *each* element of the signal, and moreover, because the signal is static, each measurement is a linear combination of *the same signal*. Here, subsequent measurements may be measuring a different signal; but not arbitrarily different, rather, one that has undergone some sparse changes.

This setting presents several specific challenges not encountered in the standard literature. For one thing, the regime of most interest is precisely when it is not possible for each individual sensor to decode its part of the signal at each point in time. That is, collaboration is required. Moreover, we are interested in understanding when a signal might be fully recoverable by

some time T, even if it is not recoverable at intermediate points.

2.1.1 Related work

Linear dynamic system are characterized with the transition matrix, input, observation matrix and the noise in the observation system. For tracking, state reconstruction and estimation of such systems the classical approach relies on the Kalman filter [44]. In the case of Kalman filtering the input to the system needs to have known statistics, while the optimality is derived specifically for gaussian inputs. This makes Kalman filter inappropriate for our setting, because our dynamics vary considerably from the standard model. When the state of the system is known to be sparse there has been interesting work based on adaptation of Kalman filtering [80, 81, 75, 46], leading to better estimation with fewer observations. In our setting, the state of the system *changes in sparse but otherwise arbitrary manner*. Moreover, the issue at hand is not filtering out additive noise, but rather coping with only few measurements that may not be enough to recover the trajectory itself.

Recovery of a sparse vector has recently been extensively studied via compressed sensing [23, 22, 14, 86, 29, 15, 10, 19, 16]. Very nice recent tutorial on the subject is available in [29]. Due to the fast expansion of the field we apologize for not including all significant work that has been done so far. For sparse noiseless observations of a vector the fundamental limits of the recovery via ℓ_1 minimization are given in [23, 22]. The compressed sensing work, however, does not typically incorporate the main elements of our model:

random observations of a signal that changes over time, and observations that are partial.

Partial observations for still signals have been explored in [11, 65, 88, 8, 1]. In [11] authors introduce the concept of jointly sparse signals, and show that partially observable signals are recoverable via compressed sensing. The partial random observations for a still signal with noisy setup has been explored in [65, 88] with interesting results that state that even sparse sensing matrices can be used for successful signal recovery. This results however are applicable in our scenario since the time-varying component is missing.

There has been some work on recovery of time-varying signals, and development of new algorithms for this (e.g., [92, 7, 91, 74, 4]) although again, none of these consider sparse changes over time. The work in [83, 82, 46] does consider the sparse changes in time-varying signal, however, it also considers an overall sparse signal which is crucial for the results to follow. The work in [32] considers sparse signal changes, but instead of looking at the overall dynamics, it introduces the bound that is equivalent to doing sparse recovery at each time instance.

While the idea of using partial measurements to recover a signal that incurs sparse changes over time seems natural, we are not aware of results that address this problem, either on the side of fundamental system requirements for recovery, or on the algorithmic development side. This is the topic of this chapter.

Contributions

We consider and formally describe the problem of recovering the final state at time T, of a dynamic multi-dimensional signal that evolves via sparse changes from some known initial state. A collection of observers make measurements of the signal; each observer collects scalar measurements that are a linear combination of a *fixed subset* of the components of the unknown vector process. This is the case, for instance, if each observer represents a node able to collect measurements of only those objects that are within radio range. We make the setting precise in Section 2.2. Our contributions, then, are as follows.

- We give necessary and sufficient conditions for the single-observer setting, that characterize precisely when it is possible to recover the final state of the trajectory of the vector. That is, we characterize what is the minimal sequence of measurements needed that is able to recover a signal's final state, given the sequence of changes. We note, again, that the measurements made are not of the same vector, and hence one cannot simply appeal to results from sparse recovery.

- We then turn to the main setting of interest: multiple observers. Again we give a necessary and sufficient condition for when multiple observers are able to recover the vector's final state. That is, we characterize when a sequence of observations is able to recover the final state of a vector, given a sequence of changes of the vector. As one would expect, this

depends not only on the number of measurements made, but also on the structure of the observers.

- The results mentioned above are algorithm-independent, and characterize the fundamental limits of what is possible. We next turn to algorithms and give an algorithm that is guaranteed to recover the signal's final state, if any algorithm can. Our algorithm is a natural version of a greedy local algorithm. Our simulations show that with high probability, our algorithm is efficient.

- We cast the problem as an ℓ_0 optimization problem, and give an ℓ_1 approximation that is computationally efficient.

2.2 System Model

The mathematical setting of our problem is as follows. We assume that there is a signal $\mathbf{x}(t)$ that is known at some initial time, t_0. For simplicity, and without loss of generality, we assume we have: $\mathbf{x}(0) = \mathbf{0}$. Our goal is to estimate $\mathbf{x}(T)$ at some final time T. The signal changes at discrete time instances $t \in \{0, 1, 2, \ldots\}$ in a sparse manner, and a collection of observers indexed by $i \in \{1, 2, \ldots, p\}$, makes measurements of the signal.

We now explain in detail the change model and measurement model.

2.2.1 Change model

Components of the signal $\mathbf{x}$ change in time in sparse but otherwise arbitrary fashion. Let $k_1, \ldots, k_t$ be the number of changes in the signal at time t. Then, we can write:

$$k_t = ||\mathbf{x}(t) - \mathbf{x}(t-1)||_0,$$

where $||\cdot||_0$ denotes the cardinality of the support. In this setup we do not assume any particular distribution of changes or location of changes, nor do we impose any further restriction on the nature of the changes. Thus, the recoverability question has to do only with the vector $(k_1, \ldots, k_{T-1})$.

2.2.2 Measurement model

We measure the signal $\mathbf{x}(t)$ through a set of observers $\{1, 2, \ldots, p\}$. Observer i is associated with a subset $O_i \subseteq \{1, \ldots, n\}$ of the coefficients of the signal, and receives scalar measurements that are a linear combination of the components in its respective subset. Note that in our current setup, this association of a user to a subset does not change. The subsets $\{O_i\}$ may be overlapping or not. The only (natural) condition that must always be imposed, is that their union contains all coefficients (without this, recovery would be impossible by any means). Just as there is a sequence of changes, there is also a sequence of measurements, that specify the number of measurements each receiver obtains at a given time t. The setting of practical interest, which we focus on below, is when the measurement schedule is fixed, e.g., periodic with

some period for each observer, but the change schedule is stochastic. Then we can ask when are we guaranteed recovery with high probability.

This setup is not about high dimensional geometric conditions such as restricted isometry, null space properties, etc. We do need to assume, however, that the linear measurements are appropriately in general position, thus avoiding pathological situations. To avoid technical issues that are not at the core of the contributions here, we simply assume that measurements are random projections of the subset of components of $\mathbf{x}$.

2.2.3 Notation:

t	discrete time index, $t \in \{0, 1, \ldots\}$,
T	discrete time index of final time step, $T < \infty$,
$\mathbf{x}(t)$	the value of signal $\mathbf{x}$ at time t, $\mathbf{x}(t) \in \mathbb{R}^n$,
$\mathbf{x}_j(t)$	component j of the signal $\mathbf{x}$ at time t, $\mathbf{x}_j(t) \in \mathbb{R}$,
$\delta_{j,t}$	the change in the component j of signal $\mathbf{x}(t)$, $\delta_{j,t} \in \mathbb{R}$, $\delta_{j,t} = \mathbf{x}_j(t) - \mathbf{x}_j(t-1)$,
k_t	the number of changes at time t, $k_t \in \{0, 1, \ldots\}$,
p	the number of different observers,
O_i	the subset of components of $\mathbf{x}(t)$ for observer i, $O_i \subset \{1, 2, \ldots n\}$,
$\mathbf{x}_{O_i}(t)$	signal $\mathbf{x}(t)$ restricted to the components in O_i,
$c_{i,t}$	number of measurements of O_i at t, $c_{i,t} \in \{0, 1, \ldots\}$,
$\Phi^{(i)}(t)$	measurement matrix of O_i, at t, size $c_{i,t} \times \lvert O_i \rvert$,
$\mathbf{y}^{(i)}(t)$	measured values by O_i at time t, length $c_{i,t}$, $\mathbf{y}^{(i)}(t) = \Phi^{(i)}(t)\mathbf{x}_{O_i}(t)$.

For component j of $\mathbf{x}$ we will also use the notation coordinate j of $\mathbf{x}$.

2.2.4 Illustrating Example

To illustrate the system model we use the following example.

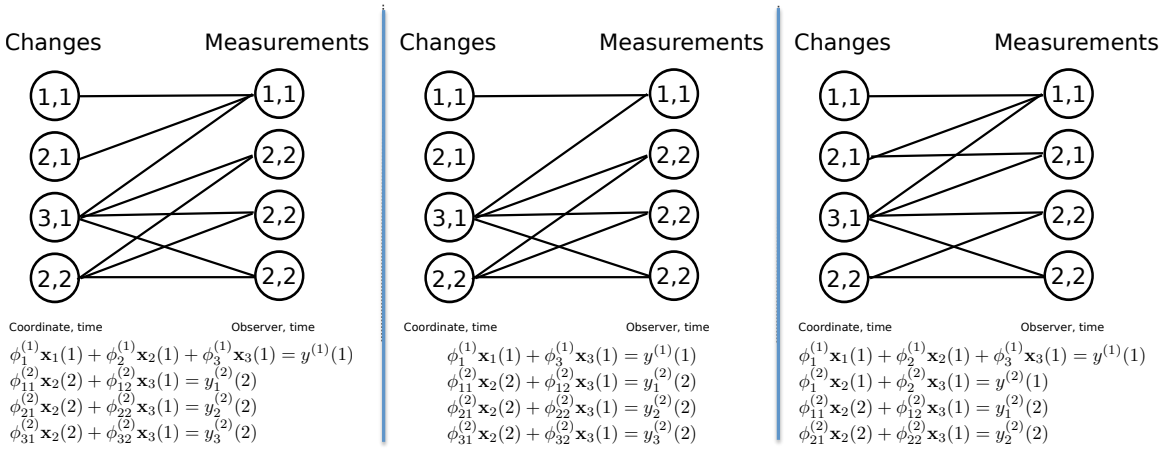

Figure 2.1: The three panels illustrate three different change and measurement patterns using a bipartite graph model. Nodes on the left labeled (i, t) indicate a change in coordinate i at time t. A node (j, t) on the right indicates a measurement at time t by observer j. The edges indicate which measurements were captured by which observer. The edges reflect both the topology of the measurement graph, but also causality (a measurement at time 1 cannot reflect a change at time 2). While the number of measurements is the same in all three panels, the measurement topology is different. In the first panel, the final state is not recoverable. In the middle panel, the final state is recoverable, but not the entire trajectory. In the third panel, both are recoverable.

Example 1. Consider a simple system:

$$1.1\mathbf{x}_1(1) + 2.7\mathbf{x}_2(1) + 0.8\mathbf{x}_3(1) = 5.4 \tag{2.1}$$

$$-0.3\mathbf{x}_2(2) + 2.4\mathbf{x}_3(2) = 1.2 \tag{2.2}$$

$$-0.9\mathbf{x}_1(2) + 0.2\mathbf{x}_2(2) + 3.2\mathbf{x}_3(2) = 2.8 \tag{2.3}$$

Now, this system correspond to a 3-dimensional time-varying system with two observers with a final time being $T = 2$. The first observer has access to all the components of the signal (eq. (2.1) and (2.3)), while second observer has access only to second and third component of the signal (eq. (2.2)). The first observer gets one observation at time instant $t = 1$ and one more at $t = 2$, while second observer gets just one observation at time $t = 2$. Now we can fill the notation with the exact values of all of the parameters:

$n = 3,$	the dimensionality of the signal,
$T = 2,$	the last time step
$p = 2,$	since we have two types of equations,
$O_1 = \{1, 2, 3\},$	first observer observes all components, of $\mathbf{x}(t)$, (2.1), (2.3)
$O_2 = \{2, 3\},$	second observer observes just, components $\mathbf{x}_2(t)$ and $\mathbf{x}_3(t)$,
$c_{1,1} = 1,$	number of equations for O_1 at $t = 1$,
$c_{2,1} = 0,$	number of equations for O_2 at $t = 1$,
$c_{1,2} = 1,$	number of equations for O_1 at $t = 2$,
$c_{2,2} = 1,$	number of equations for O_2 at $t = 2$,
$\Phi^{(1)}(1) =$	$[1.1 \quad 2.7 \quad 0.8]$,
$\mathbf{y}^{(1)}(1) = 5.1,$	
$\Phi^{(2)}(2) =$	$[0.3 \quad 2.1]$,
$\mathbf{y}^{(2)}(2) = 1.2$	
$\Phi^{(1)}(2) =$	$[0.9 \quad 0.2 \quad 3.1]$,
$\mathbf{y}^{(1)}(2) = 4.3.$	

Although this gives us some information about the setup, the system is clearly underdetermined, with 3 equations and 6 unknowns. Nevertheless, if we know that the total number of changes in the system is low, we still might get a solution. This is different from compressed sensing in three major ways: first, the number of measurements is comparable to the number of changes, second, the measurements are obtained from multiple observers whose structure is specified by the problem, and third the measurements are causal, i.e. measurement at time t reflects just the current value of the signal. In case of the system in this example, if we know there were at most two changes the system is *uniquely* solvable. To see this consider values of $\mathbf{x}_1(1) = 0, \mathbf{x}_2(1) = 2, \mathbf{x}_3(1) = 0, \mathbf{x}_1(2) = 0, \mathbf{x}_2(2) = 2, \mathbf{x}_3(2) = 0.75$ □

2.3 Main Results and Discussion

We are interested in two categories of results: fundamental recoverability conditions, that characterize when any algorithm (of potentially exponential complexity) is able to recover the final state of the system and then on the other side, algorithms that accomplish this recovery.

For a total number of changes that happened in the signal we can form an enumerable series of hypothesis each of which considers a particular pattern of changes, i.e. which component of the signal changes and at what time step. Given a particular hypothesis we use a bipartite graph representation of the changes and measurements in our algorithm and in the proofs of the main results. The bipartite graph $G(U, V, E)$ is defined in the following manner. If

there the component j changes its value at time t, then $(j,t) \in U$. For each measurements that observer i makes at time t, there will be an additional node (i,t) in V. The edges of the graph are formed in causal way; there exists an edge if the changed component is observed by observer i at time t and no edge otherwise. We illustrate this representation, as well as the problem and the associated challenges, in the following example and accompanying Figure 2.1.

Example 2. Consider three different systems of changes and observations. In Figure 2.1, the nodes on the left of the bipartite graph represent the changes and the times when they occur, while the nodes on the right represent the measurements made by each observer, and the time of each measurement. The edges encode topology and causality: they indicate which changes are reflected in each measurement of a particular observer.

The three panels in the figure have the same number of changes and measurements, yet different recoverability. Thus they illustrate the role of the topology of the observers and their subsets $\{O_i\}$. In the scenario in the first panel on the left, we have a system where even the final state is not recoverable. This is due to the fact that for equation 1 given by the observer 1 after elimination of the third unknown value we still have two unknowns one of which is the last value of the signal in the component 1. So, although we can find the end values of the components 2 and 3, the end value of the signal is left unknown.

For each change, the value of the signal at this component represents a new unknown introduced to the system. Measurements now correspond to

a set of linear equations in these unknowns. Recovering the entire trajectory requires finding the values of each of these unknowns. Our main goal is to recover only a subset of these unknowns: those that correspond to the final value of $\mathbf{x}(t)$.

The middle panel illustrates the scenario in which the trajectory of the signal is not recoverable, but the end value is. In this case we can recover all the end values of the components, but the first change in the second component remains unknown. The system in the right figure shows a system that is solvable for the trajectory and end value as well. □

We organize our results as follows. First, we consider the signal recovery problem with only a single observer. Here, all the signal components are observed. This illustrates the challenges introduced due to the changes over time. We provide a result that characterizes precisely when the final value is recoverable, given a schedule of changes and measurements. Next, we turn to the multi-observer setting, and provide an analogous result. Finally, we provide an algorithm that is guaranteed to recover the final value of the trajectory if any algorithm is able to do so. Our results in Section 2.4 illustrate the algorithm performance.

We turn first to the single observer setting. Intuitively all the measurements gathered at some time instant are interchangeable.

Definition 1 (Correct recoverability). A time varying signal $\mathbf{x}(T)$ is considered to be correctly recoverable at time T if its components are uniquely determined.

Proposition 1 (Recoverability of the signal). Suppose $\mathbf{x}(0)$ is known and

$$\sum_{i=1}^{T} c_i = \sum_{i=1}^{T} k_i + 1 \tag{2.4}$$

with $\{c_i, k_i\}$ being non-negative integers.

Then $\mathbf{x}(T)$ is correctly recoverable if and only if

$$\sum_{i=1}^{t} c_i \leq \sum_{i=1}^{t} k_i \qquad \forall t \in \{1, 2, \ldots, T-1\}. \tag{2.5}$$

□

Notice that this characterization amounts essentially to dimension counting. However, as the example given above illustrates, the setting with multiple observers is not quite as simple. We need the bipartite graph representation to properly generalize this counting. In the general setting, recovery amounts to finding a *matching* in a subset of the bipartite graph that corresponds to recovery of the final state $\mathbf{x}(T)$.

Theorem 2.3.1. Let $V = \{(i,t) | \delta_{i,t} \neq 0\}$ denote the set of changes, in (coordinate,time). Let $Z = \{(i,t) | (i,t) \in V, t = \arg\max_{s \leq T} \{(i,s) | (i,s) \in V\}\}$ denote the set of last changes for each of the changing components in the signal $\mathbf{x}(t)$.

Let $G(V, U, E)$ be the bipartite graph representing changes and measurements, and let $\Gamma(Y)$ be the image of a set $Y \subset U$. Then $\mathbf{x}(T)$ can be recovered if and only if:

(i) $\exists X \subset V$ such that $Z \subset X$

(ii) $\exists Y \subset U$ such that $\Gamma(Y) = X$

(iii) For bipartite subgraph $G(X, Y, E')$ the Hall's conditions [37] are satisfied

□

Alone this theorem is not very useful, as it requires us to know the pattern of changes in advance. However we use it, and the importance of the matching, to characterize conditions for recoverability that do not depend on the specific pattern of changes, but rather only depend on the characteristics of the set of observers.

Theorem 2.3.2. Let

$$\hat{c}_t = \min_{i \in \{1,2,\ldots,n\}} \sum_{j=1}^{p} c_{j,t} \mathbf{1}_{\{i \in O_j\}} \tag{2.6}$$

for any $t < T$. For the last time step T we define $\hat{c}_T$ differently. Consider the collection of measurements received at time T $\{c_{j,T}\}$ from all the Observers $j = 1, \ldots, p$ and measurement partitions $(c^{(1)}_{j,T}, c_{j,T} - c^{(1)}_{j,T})$. Let

$$\hat{c}^{(1)}_T(c^{(1)}_{1,T}, \ldots, c^{(1)}_{p,T}) = \min_{i \in \{1,\ldots,n\}} \sum_{j=1}^{p} c^{(1)}_{j,T} \mathbf{1}_{i \in O_j} \tag{2.7}$$

be a function of measurement partitions. Now $\hat{c}_T$ is given by:

$$\hat{c}_T = \max_{\left\{c^{(1)}_{1,T},\ldots,c^{(1)}_{p,T} \,:\, \hat{c}^{(1)}_T \geq 1\right\}} \min_{i \in \{1,\ldots,n\}} \sum_{j=1}^{p} \left(c_{j,T} - c^{(1)}_{j,T}\right) \mathbf{1}_{i \in O_j}. \tag{2.8}$$

Suppose

$$\sum_{s=1}^{T} k_s = \sum_{s=1}^{T} \hat{c}_s. \tag{2.9}$$

Then at time T signal $\mathbf{x}(T)$ is correctly recoverable if and only if

$$\sum_{s=1}^{t} k_s \geq \sum_{s=1}^{t} \hat{c}_s \qquad \forall t < T. \tag{2.10}$$

□

The key concept that drives the proofs is merging the relaxed recovery requirement of finding only the final state, with the need for a perfect matching. This leads to what we call a *coarsening* procedure which results in a smaller bipartite graph where finding a matching is easier. Finally in Section 2.4.2, we propose a low-complexity approximation (based on an ℓ_1 approximation) and empirically study its performance. In particular, we show that it compares favorably to a *genie-aided* version of the traditional compressed sensing algorithm that has additional side-information via explicit knowledge about changes – how many and where these changes occur (however, our proposed ℓ_1 approximation does not have this extra information).

2.3.1 The Breadth-Search Algorithm

Having characterized the necessary and sufficient conditions for recoverability of the signal at the last time step, we now provide a greedy algorithm that is guaranteed to recover the final state if any algorithm can.

Suppose that at some time T, the signal is recoverable (by some algorithm, of potentially arbitrary complexity). Our algorithm is iterative, and each iteration has two phases: Computation and Collection (communication). The maximum number of iterations is bounded by p, the number of observers.

The algorithm is seeded with the local measurements that each observer has made.

A few definitions are required. We form a graph where each observer is a node. Observers i and j are connected by an edge in this graph if $O_i \cap O_j \neq \emptyset$. That is, an observer's one-hop neighbors are all those observers whose measurements sense common components. For each $l = \{1, 2, \ldots, p\}$ and each Observer i, this graph induced an $l-$hop neighborhood, denoted by $N_l(i)$.

Algorithm 1. While some component of the signal $\mathbf{x}(T)$ is not yet decoded:

- **Computation in iteration** l. Each observer attempts to recover the signal using measurements obtained in the Collection Phase of iteration $(l-1)$ (i.e., measurements from observers in $N_{l-1}(i)$).
- **Collection in iteration** l. If computation phase for observer j was unsuccessful, observer j collects all measurements from observers in $N_l(i)$.

□

In words, this algorithm operates as follows. Each observer tries to decode locally, using a brute force approach. This means that the algorithm sequentially searches all possible change patterns i.e. all possible sets of (coordinate, time) tuples, for a set which has a solution that satisfies all local measurements collected up to time slot T. If such a set does not exist, we conclude that it is not possible to recover the signal with just local measurements. The number of candidate subsets is bounded by $(nT)^k$, where n is

the number of coordinates being considered by the local measurements, and k is the total number of changes. As we do not at this stage discuss concepts like RIP ([13]), etc., this local decoding algorithm is not efficient and its inefficiency is addressed by ℓ_1 approximation described in 2.3.3. Moreover, as long as the footprint of observer (i.e., the number of coordinates measured by each observer) is small, as well as the number of changes that happened, the complexity of this algorithm can be manageable. Any observer that succeeds passes its information to all observers that have any overlap in their observation subsets, i.e., its one-hop neighbors. This process terminates either with successful recovery of the final state, or with some observers having failed to do so. In this case, these observers receive all the measurements of their one-hop neighbors, thus effectively obtaining a larger observation set $\mathcal{O} = \bigcup_{j \in N_i} O_j$. The process then repeats, and if it terminates, again the sets are enlarged.

Evidently, we have the following.

Proposition 2. If the system satisfies the conditions of Theorem 2.3.1 then the algorithm recovers the signal with probability 1.

The proof of this proposition is immediate, since in the worst case scenario we are able to recover the signal once we include the measurements from all the observers.

Remark 1. The interesting question is understanding when successful recovery does not require expanding the neighborhoods too much. We now consider a specific model for the changes, and bound the probability that a decoding locally (i.e., without expanding neighborhoods) is possible.

Let $X_{i,t}$ be the indicator random variable that the component i has changed at time t. We assume that collection of $X_{i,t}$ is i.i.d. over components and time and that $P(X_{i,t} = 1) = q$, and the magnitude of change is arbitrary. With this specific model for changes in the signal, consider any network of observers attempting to decode the signal at some fixed time T using Algorithm 1. Recall that in iteration l of the algorithm's operation, an observer i has access to the measurements from it's $l-$hop neighborhood. It follows immediately from Theorem 2.3.2 (with the graph, i.e. observers and components, restricted to the $l-$hop neighborhood, $N_l(i)$, of Observer i and the corresponding components that influence these observers' measurements) that a sufficient condition for Observer i to successfully recover the signal in a fixed iteration $l \in \{1, 2, \ldots, p\}$ is the following:

(i) For each $t = 1, 2, \ldots, T$,

$$\sum_{s=t}^{T} \sum_{r \in N_l(i)} X_{r,s} \leq \sum_{s=t}^{T} \hat{c}_s, \tag{2.11}$$

where $\hat{c}_s$ is defined analogous to that in Theorem 2.3.2, but with the graph restricted to the observers in $N_l(i)$ and the corresponding signal components that influence any of their measurements,

(ii) $X_{r,s} = 0$ is satisfied $\forall s \in \{1, 2, \ldots, T\}$ and $\forall r \in (\tilde{\Gamma}(\bigcup_{m \in N_l(i)} O_m) / (\tilde{\Gamma}(\bigcup_{m \in N_{l-1}(i)} O_m)$, where $\tilde{\Gamma}(\cdot)$ corresponds to the set of signal components that influence any of the measurements available to the collection of observers in its domain $(\cdot)$.

The first condition essentially means that the conditions of Theorem 2 are satisfied for the network of observers *restricted* to the $l-$hop neighborhood of Observer i. The second condition is one of "usefulness" of all the measurements: The measurements from the $l^{th}-$hop neighbor of Observer i is useful if there are no changes in any of the signal components from *outside* the restricted network and *that affects* the measurements within the restricted network we consider.

Now coming back to the analysis of "local" $l-$hop recovery, the sparse change (Bernoulli) model along with i.i.d. changes, and given any specific network structure, we can easily compute the probability that conditions *(i)* and *(ii)* above are satisfied (for each of the observers), thus resulting in the probability of correct recoverability.

Interestingly, we show through simulations, that for many natural settings of interest, this algorithm does not require enlarging the observation sets significantly, and thus presents potentially significant gains over the fully centralized algorithm.

2.3.2 Time evolution of single observer

In the case of a single observer we look at the time evolution of the system. In particular we are interested not just when is it possible to recover the signal but also given a constant budget of measurements that are available at each time step will the resulting system be able to infinitely often resynchronize. So for the following analysis we assume there is a single observer

with access to all the components of the signal, and that the number of measurements available at each time step is fixed at $\bar{c}$. Now, we look at a specific change process that obeys the following assumptions.

Assumption 1. Let $\{K_i\}$ be a stationary discrete time process that denotes the number of changes at time t such that:

$$\mathbb{P}\left(\frac{\sum_{i=s}^{t} K_i}{t-s} - \bar{k} > \epsilon\right) \leq \frac{C(\epsilon)}{(t-s)^2}, \qquad \text{for all } s \text{ and } t$$

where $C(\epsilon)$ is a constant that is dependent only on the value of ϵ and is not a function of t. Notice that the $\bar{k}$ does not need to be the mean value of K, just a value around which the process concentrates.

Given a single observer system with the constant number of measurements for each time step, i.e. $c(1) = c(2) = \ldots = \bar{c}$ and a change process that satisfies the assumptions 1 we can make the following claim.

Theorem 2.3.3. Let the process $\{K_t\}$ satisfy the assumptions 1. Let the number of measurements $c(t)$ be $\bar{c}$ for every t. Let τ be the inter-recovery time. Then the expected time $E[\tau]$ that the system will be recoverable has the following property:

- $E[\tau] < C$, if $\bar{c} > \bar{k}$,
- $E[\tau] = \infty$, if $\bar{c} < \bar{k}$.

where C is a bound dependent on the change process.

What this theorem says is that large burst of changes in one time step can be compensated with small number of changes in the following time steps.

In particular, at each time step we are not bounded by providing enough measurements for the maximal number of changes, but with the number of changes around which the change process concentrates. This is sharply in contrast with the prerequisites of a stepwise compressed sensing, that at each point needs to adjust the the number of measurements such that it would accommodate for the maximal number of changes, i.e. $\bar{c} = f(\max\{K_t\})$, while in our case we have $\bar{c} = f(\bar{k})$.

Furthermore, if the change process concentrates around $\bar{k}$ but at every time step we only observe $\bar{c} < \bar{k}$ measurements then eventually the number of changes will build up too much for recovery to be possible.

Only requirement for recovery with $\bar{c}$ number of measurements is that the change process concentrates around $\bar{k}$, and this is significantly milder than asking for independence or even identical distribution of random variables K_t.

Now the probability of being in the recoverable state of the system is given by $P(recovery) \geq \frac{1}{E[\tau]}$.

2.3.3 ℓ_1 Approximation

The breadth-search algorithm although having good performance still leads to exponential computational complexity. To deal with this problem, we look into a simple approximation with low complexity.

As we are not interested in recovering the trajectory, but just the end value of the signal, minimization of ℓ_0 norm of the signal subject to having consistent measurements gives unique solution in the sense of getting unique

value of $\mathbf{x}(T)$ at the time T when the recovery is possible. Therefore, we can obtain the solution to the equation by writing the following:

$$\begin{aligned} \min_{\mathbf{x}(1),\mathbf{x}(2),..,\mathbf{x}(T)} \quad & \sum_{i=1}^{T} \|\mathbf{x}(i) - \mathbf{x}(i-1)\|_0 \\ \text{subject to} \quad & \Phi^{(i)}(t)\mathbf{x}_{O_i}(t) = \mathbf{y}^{(i)}(t) \\ & \mathbf{x}(0) = [00\ldots0] \end{aligned} \tag{2.12}$$

As this is a ℓ_0 norm minimization, it is non-convex. So, we look into a relaxation of ℓ_0 norm by ℓ_1 norm. This leads to a different objective of the optimization function:

$$\begin{aligned} \min_{\mathbf{x}(1),\mathbf{x}(2),..,\mathbf{x}(T)} \quad & \sum_{i=1}^{T} \|\mathbf{x}(i) - \mathbf{x}(i-1)\|_1 \\ \text{subject to} \quad & \Phi^{(i)}(t)\mathbf{x}_{O_i}(t) = \mathbf{y}^{(i)}(t) \\ & \mathbf{x}(0) = [00\ldots0] \end{aligned} \tag{2.13}$$

Notice that although this looks like a standard compressed sensing problem, and can in fact be rewritten as one there are major differences between this formulation and compressed sensing formulation. The most important difference is that our goal is *not recovery of the whole trajectory, but the resynchronization with the signal*. Furthermore, we do not know in advance at what time T will the system become solvable, and, unlike compressed sensing each component of the signal is not roughly observed the same amount of time. In particular the earlier changes have more chances to be observed than

later changes, but at the same time the later changes can completely mask the earlier change if they happened at the same component.

In a single time step system, this corresponds to a compressed sensing noiseless setup. It immediately follows that when provided with enough measurements at each time step our formulation would reduce to the sequential compressed sensing problem. However, this reduction only happens when the maximum number of changes per time step is bounded by some constant K, and if at each time step we had enough measurements to recover these K changes.

In the existing literature for ℓ_1 norm minimization in noiseless scenario there is no closed form for the ℓ_0/ℓ_1 equivalence threshold when considering Gaussian measurements (unlike for the case of noisy measurements with LASSO). There is, however, a curve that follows from the combinatorial geometry and can be numerically evaluated as stated in [25]. In the case of the sub-exponential growth of the dimensionality with respect to the number of measurements the weak and strong asymptotic have been derived. The weak asymptotic hold for "almost all signals", while strong asymptotic holds for "all signals". These results are summarized in [24]:

Let $\delta = m/n$ and let $\rho = k/m$. Then, strong asymptotic is given by:

$$\rho(\delta) \approx |2e \log(\delta\sqrt{\pi})|^{-1}$$

and weak asymptotic is:

$$\rho(\delta) \approx |2 \log(\delta)|^{-1}$$

In the limit as $n \to \infty$, if we let $m = \beta k \log(n)$, then β for the strong asymptotic approaches $2e$ from below, while for weak asymptotic β approaches 2 from below. The week asymptotic result therefore means that in the limit the noiseless and noisy setup lead to the recovery with the same constant.

2.4 Simulation Results

In this section we consider the performance of the combinatorial distributed algorithm and the ℓ_1 approximation.

2.4.1 Breadth-Search Algorithm Simulation

In particular we are interested in comparing the computational complexity of our algorithm compared to the worst-case complexity. We consider a natural setup motivated by sensor networks, where the partial observation subsets are given by a sensing range. Hence we can think of the observers as information aggregators. Given a fixed area, suppose there are $n = 1,000$ signal components that must be measured, and these represent measurements of phenomena distributed throughout the fixed area. The changes occur in the signal uniformly at random over all the components. The probability of change per each component is 0.0105, i.e. expected 10.5 changes in one time step over 1000 components. The p observers are also uniformly distributed, and they collect linear measurements of those components that fall within a given radius r of their location. We choose the radius so that each observer on average has $\log(n)$ components within range, and choose the number of ob-

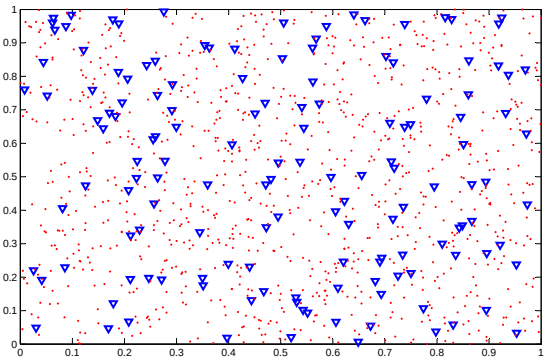

Figure 2.2: Example of a simulation setup; blue triangles represent the observers while red dots represent the components of the signal.

servers so that each component is within the range of at least three observers, on average. We depict a typical realization of this process in Figure 2.2.

It is clear that our breadth-search algorithm (in Section 2.3.1) can recover the final signal if any algorithm can. What is of interest here, is to investigate how much the local neighborhoods must be expanded in order to allow recovery, i.e., what is the complexity of our algorithm for this practically motivated setting. Figure 2.3 illustrates our results. On randomly generated instances, for each observer we compute the neighborhood $N_{l_i}(i)$ that is required (given the necessary and sufficient conditions of Theorem 2.3.1) for success of local decoding. Recall that the breadth-search algorithm in Section 2.3.1 operates iteratively: For a fixed observer j, and in iteration k, the algorithm *jointly* considers the measurements available at its k hop neighbors, i.e., the observations available at observers $\{r \in N_k(j)\}$ and attempts to decode the

signal $\mathbf{x}(T)$ using the combinatorial procedure described in Section 2.3.1. If it fails, it then expands the observer neighborhood by an additional hop, and proceeds to iteration $(k+1)$. Thus (in the context of this sensor example), the observer j at iteration k has measurements containing linear mixtures from the sensors in $\cup_{r\in N_k(j)} O_r$.

Our numerical experiment is conducted as follows: For each run and for each observer j, we record $R_j = |\cup_{r\in N_{k^*(j)}(j)} O_r|$, i.e., the number of sensors that participate in decoding at observer j at the iteration $k^*(j)$ that corresponds to the smallest iteration when observer j in succeeds in decoding. We define support size $S = \max_j R_j$ (for each run), and plot the associated empirical distribution in Figure 2.3. Note that the support size is a measure of the sample complexity of the breadth-search algorithm. The maximum support size (equal to $1,000$ here) corresponds to performing the full computation having aggregated all the information, and not attempting any local computations.

The three curves drawn show the performance of our algorithm on 1000 random instances of our problem, for different numbers of observers. Since our algorithm always computes the correct solution, the figure of merit here is the support size — as mentioned, a proxy for complexity. The simulations demonstrate that the algorithm terminates far before we approach the full support size, and thus, as one would hope, the breadth-search algorithm 2.3.1 performs well.

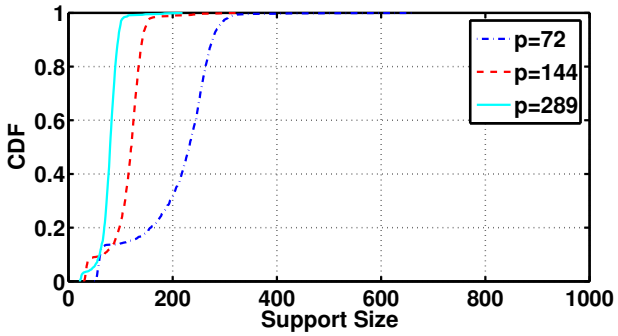

Figure 2.3: Performance of breadth-search algorithm: For various numbers of observers, we report the empirical CDF with regard to the support size (proxy for complexity of the recovery) based on 1,000 runs.

2.4.2 ℓ_1 Approximation

The approximation that we use is simple ℓ_0/ℓ_1 norm relaxation given by the equations:

$$\begin{aligned} \min_{\mathbf{x}(1),\mathbf{x}(2),..,\mathbf{x}(T)} \quad & \sum_{i=1}^{T} \|\mathbf{x}(i) - \mathbf{x}(i-1)\|_1 \\ \text{subject to} \quad & \Phi^{(i)}(t)\mathbf{x}_{O_i}(t) = \mathbf{y}^{(i)}(t) \\ & \mathbf{x}(0) = [00\ldots0] \end{aligned} \tag{2.14}$$

As we mentioned before, when the number of measurements at each time step is enough to recover all the changes that happened in the system, the ℓ_1 approximation will result in single step recovery which is equivalent to standard compressed sensing. This implies that the improvement over compressed sensing is possible only when the average number of changes is much smaller than the maximal number of changes.

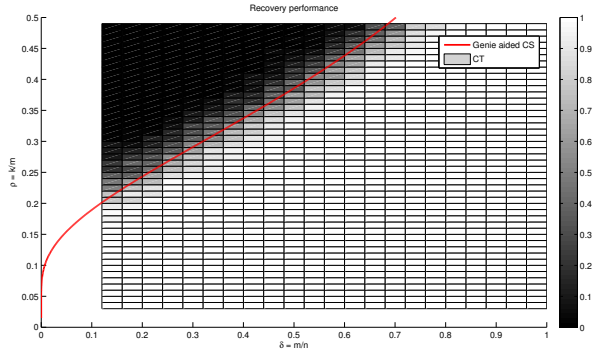

Figure 2.4: Comparison of algorithms based on change trajectory $k_1 = k, k_2 = 0$

We look at the performance of this approximation in specific path of changes $K_1 = k$ and $K_2 = 0$. This is an interesting scenario since the optimization does not know that all the changes are happening in the first time step. Also this particular setup can be compared to "genie aided" compressed sensing. In genie-aided compressed sensing, the algorithm is given side-information via explicit knowledge about how many changes happen, and in which time step these occur; the genie-aided algorithm can use this information to form a compact form of measurements that all measure the same exact changes. In our algorithm we do not use any such thing, making it blind to the fact that there are no changes in the second time step. Then we can compare how well we are doing with the reference point of the standard compressed sensing. In the plots we refer to our algorithm as CT.

First we look at the single observer scenario where the measurements are coming from a normal distribution. The location of changes of the compo-

nents is arbitrary, and the magnitude of the change in component i at time t is given by $\text{sign}(z) + z$ where z is a random variable following gaussian distribution. This is done in order to avoid having component values very close to zero. The number of measurements reported m is equal to the total number of measurements obtained. In particular, on the figure 2.4 this means that the number of measurements is given by $m = 2c$ where c is the number of measurements obtained in each time step. In the figure 2.4 we see that the l_1 approximation performs statistically equivalently to the compressed sensing with the same number of measurements. This is quite interesting since the compressed sensing version of the problem has more information about the underlying process, in particular, in the genie-aided compressed sensing problem *the algorithm is given side-information there are no changes in the second time-step*. Also, notice that if we wanted to do stepwise compressed sensing decoding of this problem without using the information about which time-steps have many changes and which have close to zero changes, then we would always need to have enough measurements to recover from the maximal number of changes.

The figure 2.4 is obtained by looking at specific value of $n = 500$. Each rectangle corresponds to the probability of recovery given 100 random runs of the problem instances. While we kept the dimensionality fixed by varying sparsity and number of measurements we looked at the recovery probability. In addition to this small instance of the problem, we also looked at a slightly bigger problem with $n = 4000$ repeated 200 times to get the average perfor-

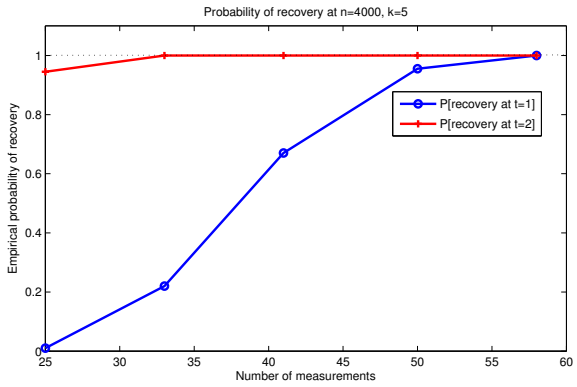

Figure 2.5: Path $k_1 = 5, k_2 = 0, n = 4000$

mance, but here we kept the sparsity fixed as well, and set to $k = 5$ which is represented in the figure 2.5. From this plot we can explicitly see that while using just the measurements from the first time step most of the time the probability of recovery is close to zero almost always, while the probability of recovery when using our algorithm with measurements from both time instances is close to 1.

Chapter 3

Mitigating Pilot Contamination with Non-orthogonal Training and by Exploiting Time Correlation

3.1 Introduction

We focus on massive MIMO systems where the challenge of estimating the channel coefficients poses a key bottleneck for maximizing system throughput. In particular, in multi-cell time division duplex (TDD) systems, the ability to accurately estimate the channel coefficients is reduced due to the interference on the training signals, otherwise known as the pilot contamination problem. We show how to mitigate the pilot contamination problem by using non-orthogonal pilots and by exploiting time correlation.

While in SISO systems (or standard MIMO systems with small number of antennas) channel estimation is governed by noise encountered by the signal in the system, in massive MIMO systems, due to the large number of antennas, the noise averages out. Ideally, noise averaging out leads to the system capacity increasing with the increase in the number of antennas used. To estimate the channel coefficients, special training sequences called pilots are transmitted between the user and the base station. When the pilots transmitted by different users at the same time are perfectly orthogonal their effect on

each other cancels out, leading to the expected increase in the system capacity. However, number of perfectly orthogonal sequences is limited by the length of the training signal. If the channel is shared among adjacent cells, orthogonal training sequences get reused over adjacent cells. Reuse of the training sequence results in interference; the channel estimation is governed not by noise that can be averaged out, but by the legitimate signal of users in adjacent cells. If users use the same training sequence, a.k.a. pilots, then their signals are perfectly aligned. These aligned signals are thus persistent interference that grows linearly with the signal strength leading to a plateau in system capacity. This problem is well known as the problem of pilot contamination. Figure 3.1. depicts this problem.

A related issue in massive MIMO systems is the upper bound on the number of users that the base station can serve at any instant of time. In the massive MIMO system, ideally we would like to serve as many users as the number of antennas at the base station. However, for successful communication within the interval of time in which the channel stays constant (coherence time) we need to both estimate the channel and transmit the data. This means that the maximal number of users that can be served is a function of the coherence time in addition to the number of antennas at the base station. In order to maximize the number of users in a cell the coherence time is split equally between the training and data transfer. This means that *a significant fraction of the system capacity is used for training.*

There have been several approaches suggested to mitigate/bypass both

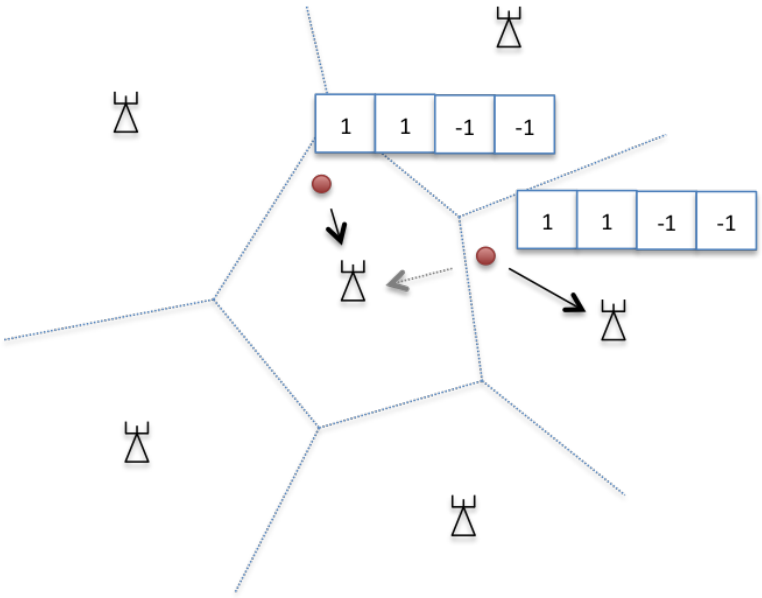

Figure 3.1: Pilot contamination occurs when users in the neighboring cells use the same pilot sequence. Channel estimate is inaccurate due the interference from the same pilot signal transmission by a user in an adjacent cell. Inaccurate channel estimate, in addition to amplifying the signal with increase in the number of antennas, amplifies the interference as well. This results in plateau effect in the system capacity.

of these issues; however, in most cases, the overall performance has increased by only a small linear factor.

We mitigate pilot contamination by expanding the number of training sequences (by allowing non-orthogonal sequences) and by exploiting the time correlation. In high dimensional systems, it has recently been observed that random sequence have a quasi-orthogonal behavior, meaning that their inner product is small. Such sequences are good candidates for training signals, as they expand the set of possible pilot signals beyond standard orthogonal ones while limiting interference between the users. In addition, if there is time correlation between the two training periods, using sparse signal recovery we can learn the training sequence in addition to the channel coefficients and recover the signal. These key observations allows us to pack more users over time and enjoy the benefits of the increased capacity of massive MIMO systems.

3.2 Related work

The term *pilot contamination* was coined in [41], where the authors explored a time division duplex (TDD) operated multi-cell system with multiple antennas at the base station (M) and K users equipped with single antennas. When users in different cells use the same pilot sequence and transmit the information in the same time frame, the channel estimate at the base station is corrupted. The analysis of the system has been done under channel reciprocity (pilot contamination in uplink dictates performance for downlink communication) and assuming that the number of antennas at the base station is larger

than the number of users. Furthermore, work in a *massive MIMO* context ($M >> K$) has shown that pilot contamination effect becomes the bottleneck in the system performance under both infinite dimensional and finite dimensional channel [60, 64, 35]. In this setup the effect of precoding matrix design [42, 6] and effect of shifting the pilots [5] have been investigated. In addition to aforementioned work, there has been some work when the user terminals are equipped with larger number of antennas [39, 40]. On a different note, non orthogonal sequences have been used in literature in other contexts, for example in CDMA code selection [49, 85].

In addition, recently there has been some work on pilot decontamination for massive MIMO networks where the coordination between the base stations is allowed [63, 90], however, in addition to the coordination required at the base station, these works rely on nonlinear channel estimation methods that could significantly reduce performance in real systems.

3.3 System model

We consider a time slotted, multi-cell cellular system with B base stations numbered $1, 2, \ldots, B$. Each base station has M antennas. Every base station b can serve up to U users. The number of users per base station is assumed to be always less than the number of antennas at the base station ($U < M$; this is a standard massive MIMO assumption).

The propagation model between users and base station includes path loss, shadowing and fading effects. Path loss and shadowing effects change

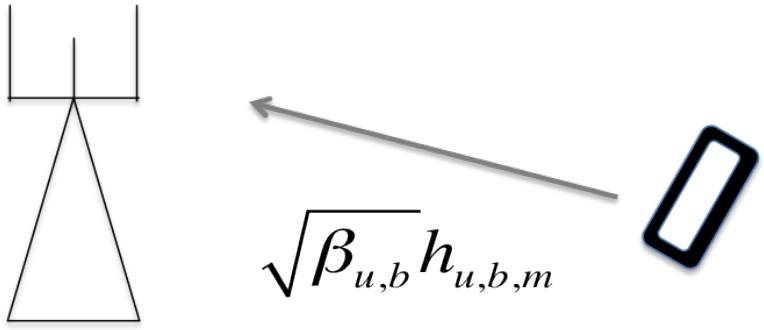

Figure 3.2: Propagation between the user u and the antenna m at the base station b consists of antenna agnostic path loss coefficient $\beta_{u,b}$ and the antenna specific fading coefficient $h_{u,b,m}$.

slowly with time and are invariant of the specific antenna at the base station; we model them with a factor that is a function of the user and the base station. The fading effects change with the coherence time. The propagation model between user u and antenna m of base station b is given by $\sqrt{\beta_{u,b}}h_{u,b,m}$. Coefficients $\beta_{u,b}$ are a function of the distance between the base station and user, while $h_{u,b,m}$ follow complex normal distribution $\mathcal{CN}(0,1)$ and represent fading effects. In our model we estimate both β and h coefficients. The propagation model is given in Figure 3.2.

In time division duplex (TDD), channel reciprocity holds, meaning the channel coefficients $\sqrt{\beta_{u,b}}h_{u,b,m}$ are the same for downlink and uplink communication. This assumption is important since channel estimation for massive MIMO is feasible only at the base station.

Multi-cell massive MIMO TDD systems are known to lead to pilot contamination [42]. The transmission in TDD systems needs to be synchronized,

which in commonly deployed systems means that all users are transmitting their pilots in the same time. Due to the multi-cell system assumption, frequency band will get reused (with appropriate reuse factor), leading to the interference on the pilot signals.

The system we consider is a time slotted system, and all the mentioned parameters are a function of time. The time component of our system is important, since in this work we exploit the time correlation between two time instants. In addition, the number of users at each base station is also time varying. We assume orthogonal frequency division multiplex (OFDM) operation, and the granularity of time is set to the duration of one OFDM packet.

Typically, the length of a packet is predetermined by the coherence time of the system. The coherence time is the interval of time during which the channel can be assume to be fixed. In our study, we assume that the coherence time is large with the respect to a time slot. Thus, we assume that channel coefficients h do not necessarily change from one time instant to another, creating an opportunity to learn them and exploit that knowledge over multiple time intervals.

Each packet contains a training signal. The training signal consists of a sequence of S symbols. We say that the s^{th} training symbol is at a location s. This is represented in Figure 3.3.A user's pilot sequence is simply a subset of these S locations, i.e. it sends energy in the chosen locations and nothing outside the selected subset. In order to keep the notation clean, unless it is

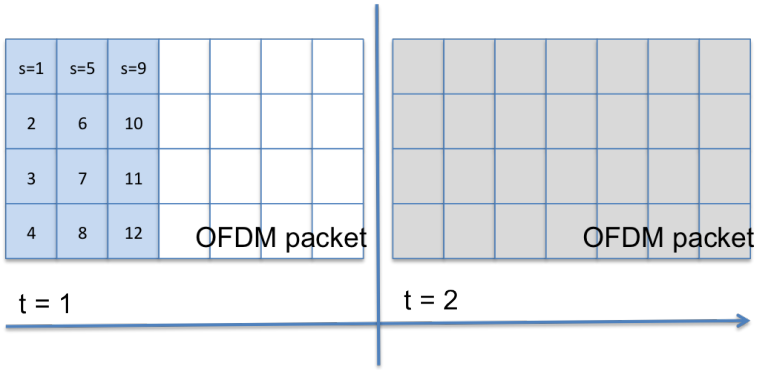

Figure 3.3: The OFDM packets are exchanged between the users and base station. Each packet has reserved space for the training, in the Figure we see 12 locations that serve as training. Note that the location of the training is arbitrary, thou we depicted it in the beginning of the packet. The time index shows that each time interval we are dealing with a separate OFDM packet.

necessary to the meaning of the equation we leave out the time index.

The noise is modeled as AWGN and we write $w_{b,m,s}$ to denote the noise received at antenna m of base station b at training location s. Statistically, the noise follows a complex normal distribution $\mathcal{CN}(0, \sigma)$.

We assume that the transmission is over a flat channel. This in general is not true in real systems, however, if we restrict our analysis to a single subcarrier, the assumption of the flat channel holds and the results can be reproduced for each subcarrier separately.

3.4 Communication model

For uplink communication, a user first sends the pilot sequence and then its data. The base station uses the pilot sequence to estimate the channel

coefficients and decode the data.

Downlink communication operates in two phases: uplink channel training and downlink data transmission. During the uplink channel training phase users transmit pilots that are subsequently used at the base station for obtaining channel estimates. From the channel estimates the base station forms a precoding matrix and sends the data back to the user. This way the need for channel estimation at the end user is removed. The same method can be used for uplink data transmission where the pilots are used for channel estimation so that interference cancelation can be performed at the base station. This is a standard setup for time division duplex massive MIMO systems [79].

3.4.1 Uplink training

At the beginning of each time interval t, all users transmit their training sequences. The length of the sequence is given by S, and we enumerate all the training locations s as $\{1, 2, \ldots, S\}$.

Each user u chooses in a random way a set $S(u)$, such that $S(u)$ is a subset of cardinality $k < S$ from the set $\{1, 2, \ldots, S\}$. Now, we can write the received signal at the training location s at the base station b, at the antenna m as:

$$y_{b,s,m} = \sum_{u \in U} \sqrt{\beta_{u,b}} h_{u,b,m} \mathbf{1}_{\{s \in S(u)\}} + w_{b,m,s}. \tag{3.1}$$

If we write this in vector form aggregating over the slots we get the

following equation:

$$\mathbf{y}_{b,m} = \mathbf{\Psi}\sqrt{(\mathbf{D}_b)}\mathbf{h}_{b,m} + \mathbf{w}_{b,s}, \tag{3.2}$$

where matrix Ψ aggregates all pilot sequences $\Psi_{s,u} = \mathbf{1}_{\{s \in S(u)\}}$, matrix $\mathbf{D}_b$ is a diagonal matrix of path loss, $\mathbf{D}_b = \text{diag}[\beta_{1,b} \quad \dots \quad \beta_{U,b}]$, and $\mathbf{h}_{b,m}$ is the channel coefficient vector ordered to correspond to the users.

We show in the next section how to find the sets $S(u)$ and parameters $\beta_{u,b}$. For now, we assume that those parameters are known at the base station.

The high level problem is to simultaneously determine three sets of unknowns at each base station: $S(u)$ for each user, $\{\beta_{u,b}\}$ and the channel coefficients h.

The goal is to estimate channel coefficients from a linear system corrupted with Gaussian noise. As the received vector is jointly Gaussian with the sent vector, the optimal MMSE estimator is a linear MMSE estimator ([43]):

$$\hat{\mathbf{h}}_{b,m} = \sqrt{\mathbf{D}_b}\mathbf{\Psi}^\dagger \left(\mathbf{\Psi}\mathbf{D_b}\mathbf{\Psi}^\dagger + \sigma\mathbf{I}\right)^{-1} \mathbf{y}_{b,m} \tag{3.3}$$

This follows from standard results from statistics and signal processing.

3.4.2 Downlink transmission

For downlink communication we use linear precoding. The vector $\mathbf{q}_b$ is a U_b long signal sent from the base station b that gets multiplied with the precoding matrix $\mathbf{A}_b = f(\hat{\mathbf{h}})$ with dimensions $M \times U_b$. The received signal at user u is given by:

$$x_u = \sum_{b=1}^{B} \sum_{m=1}^{M} \sqrt{\beta_{u,b}} h_{u,b,m} \mathbf{a}_{b,m} \mathbf{q_b} + z_u, \tag{3.4}$$

where $\mathbf{a}_{b,m}$ is row m of precoding matrix $\mathbf{A}_b$, and z_u is Gaussian noise.

3.4.3 Achievable rates with linear precoding

In this section we derive upper bound on the achievable rates with linear precoding under pilot contamination. We restrict our attention to linear precoding methods that satisfy the average power constraint at the base station. Average power constraints are defined as $\mathbb{E}[q_b] = 0$, $\mathbb{E}[q_b q_b^\dagger] = \mathbf{I}$ and $\mathrm{Tr}[\mathbf{A}_b^\dagger \mathbf{A}_b] = 1$, i.e., the precoding should be zero mean, uncorrelated across the antennas and the average power across all the antennas should be 1.

We derive an expression for the achievable rates, following the method suggested in [59] and used in [42]. First we change the received signal form into:

$$\begin{aligned} x_u &= \sum_{b=1}^{B} \sum_{j=1}^{U(t)} g_{u,b}^{j} q_{b,j} + z_u \\ &= \sum_{b=1}^{B} \mathbb{E}\left[g_{u,b}^{u}\right] q_{b,u} + \sum_{b=1}^{B} \left[g_{u,b}^{u} - \mathbb{E}\left[g_{u,b}^{u}\right]\right] q_{b,u} + \sum_{b=1}^{B} \sum_{j \neq u} g_{u,b}^{j} q_{b,j} + z_u \end{aligned} \tag{3.5}$$

where $g_{u,b}^{j} = \sqrt{\beta_{u,b}} \mathbf{h}_{u,b}^{T} \mathbf{a}_{b,j}$, and $\mathbf{a}_{b,j}$ is the j^{th} column of precoding matrix M. We split this signal into expected signal and effective noise:

$$z'_u = \sum_{b=1}^{B} \left[g_{u,b}^{u} - \mathbb{E}\left[g_{u,b}^{u}\right]\right] q_{b,u} + \sum_{b=1}^{B} \sum_{j \neq u} g_{u,b}^{j} q_{b,j} + z_u$$

Effective noise in this term aggregates the effects of interference and signal variance in addition to the received noise.

The effective noise term, z'_u, is uncorrelated with the signal. In addition $q_{b,u}$ is independent of all $q_{i,j}$ such that $(b,u) \neq (i,j)$ and the signals are independent of the precoding matrix. Independent Gaussian noise is the worst case noise distribution for uncorrelated noise [38], for which the achievable rates are:

$$R_u = C\left(\frac{\sum_{b=1}^{B}\left|\mathbb{E}\left[g_{u,b}^{u}\right]\right|^2}{\sigma + \sum_{b=1}^{B}\left(\mathrm{Var}\{g_{u,b}^{u}\} + \sum_{j\neq u}\mathbb{E}\left[\left|g_{u,b}^{j}\right|^2\right]\right)}\right),$$

where $C(x) = \log_2(1+x)$.

This expression allows for the signal transmission via multiple base stations to the user, allowing for network MIMO mode of communication.

For the setting of communication of a user with a dedicated base station, i.e., user u has a dedicated base station $b(u)$, we rewrite the achievable rates for user u as:

$$R_u = C\left(\frac{\left|\mathbb{E}\left[g_{u,b(u)}^{u}\right]\right|^2}{\sigma + \left(\mathrm{Var}\{g_{u,b(u)}^{u}\} + \sum_{j\neq u}\mathbb{E}\left[\left|g_{u,b(j)}^{j}\right|^2\right]\right)}\right)$$

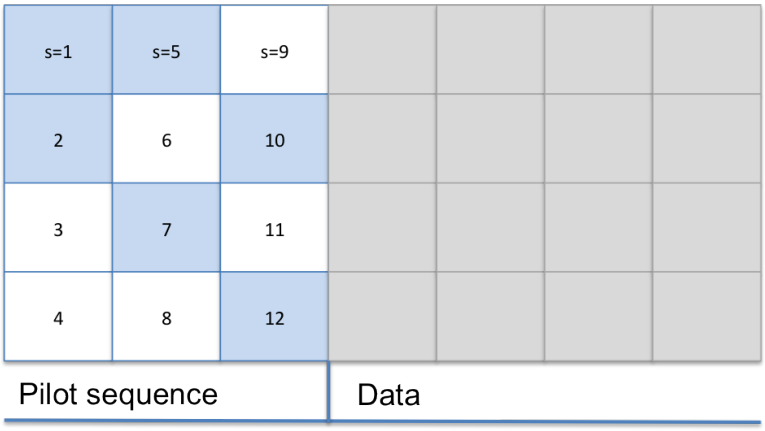

Figure 3.4: Example of pilot signal for a user. Each pilot signal consists of sending the signal on k locations out of S possible locations, and in this example $k = 6$ and $S = 12$. The pilot signal is fully defined by the locations $1, 2, 5, 7, 9, 12$ at which the signal is transmitted. The number of different pilot sequences for this example is 924 as compared to 12 orthogonal sequences.

3.5 Pilot Sequence Design

Traditionally pilot sequences are orthogonal; instead we look into non-orthogonal pilot sequence design. In particular, we look into pilot sequences such that exactly k out of possible S locations are transmitting unit power signal. A pilot sequence is defined with the subset of the locations on which signal is sent. Each user simply chooses k random slots out of the S possible ones; thus for large value of S their inner product will be small. This is presented in Figure 3.4.

Each user picks the pilot sequence without coordination. This is in contrast with the traditional pilot sequence selection where the user through a separate signaling channel through which a pilot sequence is assigned to the

user.

The signal received signal at the antenna m, at base station b, at pilot location s is:

$$y_{b,s,m} = \sum_{u} \sqrt{\beta_{u,b}} h_{u,b,m} \mathbf{1}_{\{s \in S(u)\}} + w_{b,s,m}.$$

Notice that we sum over all the users u in the system, including the users that are assigned to different base stations.

Statistically, $y_{b,s,m}$ is a zero mean random variable following a gaussian zero mean distribution with variance $\sum_{u=1} \beta_{u,b} \mathbf{1}_{\{s \in S(u)\}} + \sigma$.

3.6 Time varying behavior

We look at the time evolution of our system. In particular we want to see if we can benefit from the time correlation of the channel coefficients. Specifically, we are interested in a setting where the channel coefficients for any fixed user changes infrequently over consecutive time slots. his corresponds to a large coherence time with respect to an OFDM time slot.

3.6.1 Uplink training

When sending training signals, the received signal at base station b in time slot t at pilot location s is given by:

$$y_{b,s,m}(t) = \sum_{u \in U(t)} \sqrt{\beta_{u,b}} h_{u,b,m}(t) \mathbf{1}_{\{s \in S(u)\}}(t) + w_{b,m,s}(t). \tag{3.6}$$

For the next time step we can write:

$$\begin{aligned} y_{b,s,m}(t+1) &= \sum_{u\in U(t)} \sqrt{\beta_{u,b}} h_{u,b,m}(t) \mathbf{1}_{\{s\in S(u)\}} \\ &\quad - \sum_{u\in U^-(t+1)} \sqrt{\beta_{u,b}} h_{u,b,m} \mathbf{1}_{\{s\in S(u)\}} \\ &\quad + \sum_{u\in U^+(t+1)} \sqrt{\beta_{u,b}} h_{u,b,m} \mathbf{1}_{\{s\in S(u)\}} + w_{b,m,s}(t+1) \\ &= \sum_{u\in U(t)} \sqrt{\beta_{u,b}} h_{u,b,m} \mathbf{1}_{\{s\in S(u)\}} \\ &\quad + \sum_{u\in \Delta U(t+1)} \sqrt{\beta_{u,b}} h_{u,b,m} \mathbf{1}_{\{s\in S(u)\}} + w_{b,m,s}(t+1). \end{aligned}$$

In the equation above, $U^-(t+1)$ is the set of all users who were present in time instant t but have left the system by time $t+1$, as well as those users whose channel coefficients changed between time t and $t+1$. Set $U^+(t+1)$ is the set of new users, i.e., the users that were not present at time t, but are in the system at time $t+1$. Finally, the set $\Delta U(t+1)$ is the set of all users that changed in the system.

Now, expanding the system up to the final time T:

$$y_{b,s,m}(T) = \sum_{u\in U(0)} \sqrt{\beta_{u,b}} h_{u,b} \mathbf{1}_{\{s\in S(u)\}} + \sum_{t=1}^{T} \sum_{u\in \Delta U(t)} \sqrt{\beta_{u,b}} h_{u,b} \mathbf{1}_{\{s\in S(u)\}} + w_{b,m,s}(T).$$

Notice that out of this representation we can write the recursive equation for the system:

$$y_{b,s,m}(t+1) = y_{b,s,m}(t) + \sum_{u\in \Delta U(t+1)} \sqrt{\beta_{u,b}} h_{u,b} \mathbf{1}_{\{s\in S(u)\}} + w_{b,m,s}(t+1) - w_{b,m,s}(t),$$

or, more clearly:

$$\Delta y_{b,s,m}(t+1) \quad = \quad \sum_{u \in \Delta U(t+1)} \sqrt{\beta_{u,b}} h_{u,b} \mathbf{1}_{\{s \in S(u)\}} + w'_{b,m,s}(t+1),$$

where w' has variance 2σ.

The uplink training signal, written in the matrix form is:

$$\Delta \mathbf{y}_{b,m}(t) = \mathbf{\Psi}(t) \sqrt{(}\mathbf{D}_b(t)) \Delta \mathbf{h}_{b,m}(t) + \Delta \mathbf{w}_{b,s}(t), \tag{3.7}$$

where matrix $\mathbf{D}_b$ is the matrix diag$[\beta_{1,b} \quad \ldots \quad \beta_{\Delta U(t),b}]$ for all the users, $\mathbf{\Psi}(t)$ is an $S \times \Delta U(t)$ matrix with elements $\Psi_{s,u(t)} = \mathbf{1}_{\{s \in S(u)\}}$, $\Delta \mathbf{y}_{b,m}(t)$ is the difference of the received signals at time t and time $t-1$ arranged by the training locations, and $\Delta \mathbf{h}_{b,m}(t)$ are channel coefficients arranged by the new users. Finally, $\Delta \mathbf{w}_{b,s}(t)$ is the noise vector in the system, with variance 2σ.

3.7 Algorithms

In this section, we describe three channel estimation related algorithms. All algorithms are executed at the base station. They focus on finding the unknowns in the system: the users u in the change set, the pilot sequence $S(u)$ and path loss coefficients $\beta_{u,b}$ and corresponding fading coefficients h. The first algorithm, pilot signal recovery algorithm, jointly recovers the pilot sequence $S(u)$ and the path loss coefficients $\beta_{u,b}$. The second one, channel estimation algorithm, uses that information to recover the fading coefficients h. The last algorithm combines the results of the first two algorithms to provide tracking of the changes in the system as users are entering the system, leaving

the system or their coefficients are changing. While the channel estimation algorithm is simply the standard MMSE algorithm which is well known and widely used for channel estimation [**?**], the pilot sequence recovery algorithm and the user tracking algorithms are novel in this context.

3.7.1 Pilot signal recovery

For pilot signal recovery we focus on the signal received at the pilot locations. The signal $y_{b,s,m}$ is the signal received at the antenna m of the base station b at the pilot location s. The received signal $y_{b,s,m}$ is zero mean, with the variance that is a function of the users transmitting at the pilot location s in the system. To simplify the notation we drop the time index. The average power of the received signal (averaged over the receive antennas) is:

$$\gamma_{b,s} = \frac{1}{M}\sum_{m=1}^{M}(y_{b,s,m})^2 = \frac{1}{M}\sum_{m=1}^{M}\left(\sum_{u\in U(t)}\sqrt{\beta_{u,b}}h_{u,b,m}\mathbf{1}_{\{s\in S(u)\}} + w_{b,s,m}\right)^2. \tag{3.8}$$

Signal $\gamma_{b,s}$ is statistically following a scaled χ^2-distribution with M degrees of freedom. The expected value of the signal $\gamma_{b,s}$ is:

$$E[\gamma_{b,s}] = \sum_{u=1}^{U(t)}\beta_{u,b}\mathbf{1}_{\{s\in S(u)\}} + \sigma.$$

Further, the variance of the signal $\gamma_{b,s}$ is reducing towards zero with the increase in the number of antennas (follows directly from SLLN).

For each base station b, signals $\gamma_{b,s}$, received at all the pilot locations s form the input to the pilot recovery algorithm.

We design a simple and fast algorithm inspired by orthogonal matching pursuit (OMP) algorithm [67]. The OMP algorithm is often used to recover a high dimensional sparse signal x from a series of measurements y when we know the matrix M such that $Mx = y$ and has provenly good sparse signal recovery properties [78, 77]. It is an iterative algorithm that at each iteration tries to find a column of M that is most aligned with the residual signal. Although theoretical results for this algorithm are not as tight as for the ℓ_1 norm minimization, in practice it has been observed to perform very well in terms of time and accuracy of the solution.

For pilot signal recovery, matrix M is a matrix with columns such that the exactly k of them are set to 1 and the rest are set to 0. In addition, values $\gamma_{b,s}$ are all positive numbers, due to the aggregation procedure. The algorithm 1 describes the signal recovery. Notice that choosing the indices of largest k values of the signal is equivalent to finding the max inner product of the residual signal with the columns of matrix M.

Although it seams different from the standard OMP algorithm, our algorithm is different only in one point. Instead of doing a projection of the signal to the selected pilots, at each iteration we define the value of the signal as the minimum value over all the components. We do this since the path loss coefficients are always positive and we want to maintain the positivity of all the residual signals, which projection does not guarantee to preserve.

Algorithm 1 Pilot signal recovery

procedure DETECTUSERS
  $\mathbf{Y} \leftarrow$ Received Signal
  $\gamma \leftarrow$ Avg Power Of $\mathbf{Y}$
  $\mathbf{\Psi}_0 \leftarrow []$
  $\beta_0 \leftarrow []$
  $\mathbf{r}_0 \leftarrow \gamma$ *Residual signal*
  $i \leftarrow 1$
  repeat *While there are new users*
    $\mathbf{v} \leftarrow$ Indices of the k max values of $\mathbf{r}_{i-1}$.
    $\mathbf{\Psi}_i \leftarrow [\mathbf{\Psi}_{i-1}; \mathbf{v}]$ *Update pilot sequences*
    $\beta_i \leftarrow [\beta_{i-1}, \min_{\mathbf{v}}(\mathbf{r}_{i-1})]$ *Update shadowing coefficients*
    $\mathbf{r}_i \leftarrow \mathbf{y} - < \mathbf{I}_i \beta_i$ *Update residual value*
    $i \leftarrow i + 1$
  until $(i >$ Max Users$)$ **or** $\|\mathbf{r}_{i-1}\|_1 \leq \epsilon$
end procedure

3.7.2 Channel estimation

Channel estimation is done at the base station, and channel coefficients are used for forming a precoding matrix. For a linear system corrupted with AWGN noise, it is well known that linear MMSE estimator is optimal [43]. The MMSE estimator uses the knowledge of exact pilot sequence as well as shadowing coefficients. However, we only have the estimated values of $\tilde{\mathbf{D}}_b$ and pilot sequence, $\tilde{\mathbf{\Psi}}$. The estimator that we use is:

$$\hat{\mathbf{h}}_m = \sqrt{\tilde{\mathbf{D}}}\tilde{\mathbf{\Psi}}^{\dagger}\left(\tilde{\mathbf{\Psi}}\tilde{\mathbf{D}}\tilde{\mathbf{\Psi}}^{\dagger} + \sigma\mathbf{I}\right)^{-1}\mathbf{y}_m \tag{3.9}$$

3.7.3 Tracking user change

In the time varying system there are three events that can happen with users: new user can come in the system, old user can leave the system and the

Algorithm 2 Channel estimation

procedure EstimateChannel
  $\mathbf{\Psi} \leftarrow$ Estimated pilot sequences
  $\mathbf{D} \leftarrow \text{diag}[\beta]$ Estimated shadowing coefficients
  $m \leftarrow 1$ For each antenna at the BS
  repeat
    $\mathbf{y}_m \leftarrow$ Received training vector at antenna m
    $\hat{\mathbf{h}}_m \leftarrow \sqrt{\mathbf{D}}\mathbf{\Psi}^\dagger \left(\mathbf{\Psi}\mathbf{D}\mathbf{\Psi}^\dagger + \sigma\mathbf{I}\right)^{-1} \mathbf{y}_m$
    $m \leftarrow m + 1$
  until $m >$ Number of antennas
end procedure

user channel can change. All this changes are reflected by the pilot sequence in the system. If a user enters a system a new pilot sequence is present, if the user leaves the system, the pilot sequence is absent from the training, and finally if the user's channel changes, the pilot sequence persists with different coefficients. We drop the index indicating the base station, as this calculation is local for each base station. At times we also drop the index indicating the user when the user is not changing.

In the section 3.6 we described the time varying system. To recap, we look at the signal $\Delta\mathbf{Y}(t+1)$ from which we estimate the pilot sequences for users $\mathbf{\Psi}(t)$, path loss coefficients $\hat{\beta}$ and the fading coefficients $\mathbf{h}$ for the users in the change set.

We identify the user through the pilot sequence. In order to do so we use algorithm 3 at each base station. The change signal $\Delta\mathbf{Y}(t)$ is the change between what we would expect the signal to look like if all users and channel coefficients stay the same ($\tilde{\mathbf{Y}}(t-1)$) and the received signal.

Algorithm 3 User recovery and estimation

procedure RECOVERUSERS
  $\mathbf{Y}(0) \leftarrow \mathbf{0}$
  $\mathbf{\Psi}(0) \leftarrow []$
  $\beta(0) \leftarrow []$
  $t \leftarrow 1$ *For each time step*
  repeat
    $\mathbf{Y}(t) \leftarrow$ Received signal at time t
    $\Delta\mathbf{Y}(t) \leftarrow \mathbf{Y}(t) - \tilde{\mathbf{Y}}(t-1)$
    $\Delta\mathbf{\Psi}(t), \Delta\beta(t) \leftarrow \text{DetectUsers}(\Delta\mathbf{Y}(t))$
    $\Delta\mathbf{h} \leftarrow \text{EstimateChannel}(\Delta\mathbf{\Psi}(t), \Delta\beta(t), \Delta\mathbf{Y}(t))$
    $u \leftarrow 1$ *For each pilot sequence in* $\mathbf{\Psi}(t-1)$
    repeat
      if (There is u_1 such that $\Delta\mathbf{\Psi}_{u_1}(t) = \mathbf{\Psi}_u(t-1)$) **then**
        if ($\|\sqrt{\Delta\beta_{u_1}(t)}\Delta\mathbf{h}_{u_1}(t) + \sqrt{\beta_u(t-1)}\mathbf{h}_u(t-1)\| \geq \epsilon$) **then**
          $\beta(t), \mathbf{h}(t) \leftarrow \text{UpdateCoefficients}(\beta, \mathbf{h}, \Delta\beta_{u_1}(t), \Delta\mathbf{h}_u(t))$
        else
          User is no longer present, do not update
        end if
      end if
      $u \leftarrow u + 1$
    until $u > \|\beta(t-1)\|_0$
    $u \leftarrow 1$ *For each new pilot sequence*
    repeat
      if (There is no u_1 such that $\in \mathbf{\Psi}_{u_1}(t-1) = \Delta\mathbf{\Psi}_u(t)$) **then**
        $\beta(t), \mathbf{h}(t) \leftarrow \text{AddCoefficients}(\beta, \mathbf{h}, \Delta\beta_u(t), \Delta\mathbf{h}_u(t))$
      end if
      $u \leftarrow u + 1$
    until $u > \|\Delta\beta(t-1)\|_0$
    $t \leftarrow t + 1$ *Next time interval*
  until
end procedure

Whenever we detect the existing pilot sequence in the change signal, we assume that some event has occurred for that user. There are two possible outcomes: the user left the system, or the user's channel coefficients changed. We detect the event that the user has left the system when the new estimate of the channel coefficients are canceling the known channel coefficients:

$$\|\sqrt{\beta(t-1)}\mathbf{h}(t-1) + \sqrt{\Delta\beta(t)}\Delta\mathbf{h}(t)\| < \epsilon$$

If the new channel coefficients are not canceling the existing coefficients, then we need to combine them into a new estimate of the channel coefficients for the user. As the parameter β corresponds to the average power of the received signal, we get the new estimate of coefficient β by combining the power of the expected signal with the change signal:

$$\beta(t) = \frac{1}{M}\sum_{m=1}^{M}\left\|\sqrt{\beta(t-1)}h_m(t-1) + \sqrt{\Delta\beta(t)}\Delta h_m(t)\right\|^2.$$

We combine the fading coefficients into a new estimate in the following way:

$$h_m(t) = \frac{1}{\sqrt{\beta(t)}}\left(\sqrt{\beta(t-1)}h_m(t-1) + \sqrt{\Delta\beta(t)}\Delta h_m(t)\right)$$

Notice that this just normalizes the resulting channel coefficients.

If we detect a new pilot sequence in the change signal, we assume a new user entered the system at this time step. For new users, there are no previously known channel coefficients. The channel estimates are then trivially set to the estimated channel coefficients corresponding to the new pilot sequence in the change signal:

$$\beta(t) = \Delta\beta(t), \quad h_m(t) = \Delta h_m(t).$$

3.8 Novelty of the approach

The two key novelties in the communication system that we introduce are: (i) reusing the estimated channel coefficients when there is a strong time correlation, and (ii) extending the number of available pilot sequences to non-orthogonal ones.

Now, as different users have different coherence times, at each time step we do the channel estimation on *the change signal*. When the user coefficients stay same, the change signal consists of noise. When users enter, leave or change the coefficients, the change signal contains that information in form of the mixture of appropriate pilot sequences received at the base station.

With exploiting the time correlation, the management of the orthogonal pilot sequence assignment to users becomes more difficult to solve. This is due to the pilot contamination, where interfering users in neighboring cells shouldn't be getting the same sequence in order to avoid pilot contamination.

Such an assignment is equivalent to graph coloring problem on the interference graph. Here the number of pilot sequences is the number of colors, making the pilot sequence management NP-complete problem. An advantage of our algorithm lies in the simplification of the management by resorting to the semi-orthogonal pilot sequences. The users pick the pilot sequence uniformly at random, and due to the size of the set of admissible pilot sequences, the probability of reusing the same sequence in neighboring cell is low.

3.9 Simulations results and Discussion

In this section we show experimental evidence for the communication model described. We focus on the following properties of the system: (i) Sum rate performance, (ii) Accuracy of the recovered channel coefficients, and (iii) Number of users that we can pack in the system. Each of the above measures of performance is examined in detail in separate subsections.

3.9.1 Sum rate performance

For the sum rate performance, we compare of our system with the following systems:

- **Genie aided system:** The channel coefficients for each user, base station pair are perfectly recovered at the base station.
- **Orthogonal pilots:** Each user picks random orthogonal pilots for transmission. The orthogonal pilot sequences are constructed from the Walsh-

Hadamard matrix [61]. The channel coefficients $h_{u,b,m}$ for each user, base station pair are estimated using MMSE with known values of coefficients $\beta_{u,b}$.

We look at the system with 16 pilot locations. The number of base stations is fixed to 7, the number of users in the system is in the set $\{4, 8, 12\}$. The number of antennas at the base stations is varied by steps of 5 from 5 to 50.

Our system is denoted as the non-orthogonal pilot system. Each user picks a pilot sequence uniformly at random over all the possible pilot sequences with signal on exactly 4 locations out of 16.

For each combination of simulation parameters we first identify the schedule, i.e. which users should transmit the data so that the sum rate is optimal under the genie aided system. Next, we compare the sum rate achievable by the genie aided system, the sum rate under the estimated channels with orthogonal pilots, as well as the sum rate with the estimated channels and non-orthogonal pilots.

We repeat the experiments 500 times for each combination of simulation parameters, with different geographical placement of the users (therefore different parameters $\beta_{u,b}$). The reported sum rate is the average sum rate. The results are given in Figure 3.5. As we can see under optimal scheduling, both orthogonal and non-orthogonal pilots achieve sum rate comparable to the genie aided system for the reported number of users. The gap is the largest

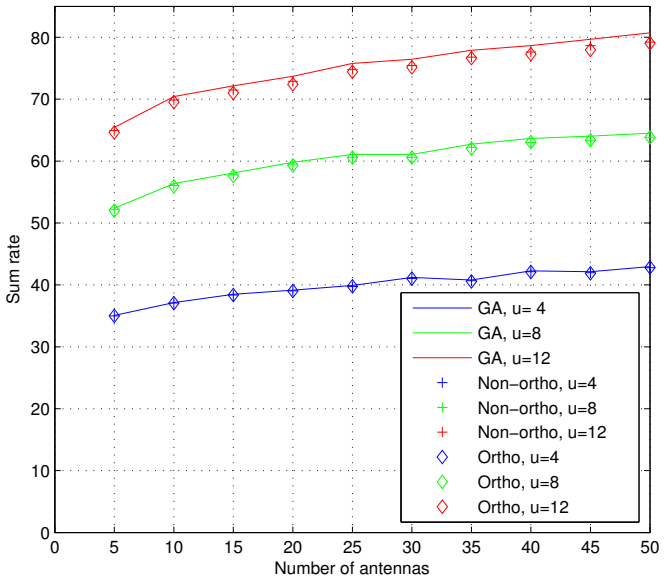

Figure 3.5: Comparisons of the sum rate performance for three different systems: the genie aided system, the system with orthogonal pilots and the system with non-orthogonal pilots.

when there are 12 users in the system.

3.9.2 Recovery of the non-orthogonal pilots in single time step

Both Figure 3.6 and Figure 3.7 show the probability of accurate recovery of channel coefficients of the users that are changing at a single time step. The setup of the experiment is as follows. Number of pilot locations is set to 20. Number of antennas at the base stations is fixed to 50. Number of base stations is growing from 1 till 104. Configuration of base stations is hexagonal. Number of users that are arriving into the system grows from 1 till 100. Users

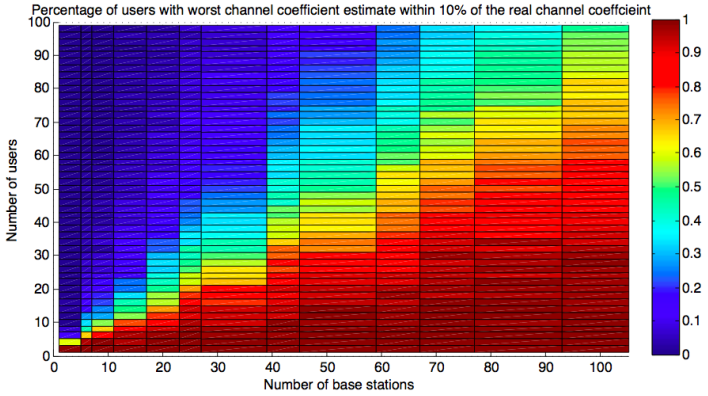

Figure 3.6: Percentage of users in the system, whose all channel coefficients are recovered with less then 10% of an error. We vary the number of users and the number of base stations. We can notice a linear trend, showing that base stations recover for high percentage of users their corresponding channel coefficients within the 10% error margin.

are placed uniformly at random on the unit square. The coefficients $\beta_{u,b}$ are determined by the distance of the user from the base station, following the path loss model. The path loss exponent is set to 3.8 which corresponds to the path loss of lossy environment. The noise floor is set to -90dB. In the setup users interfere with users in the immediately neighboring cells. Each new user picks 5 pilot locations uniformly at random, and sends the uplink training. The base station finds the coefficients γ:

$$\gamma_{b,s} = \frac{1}{M}\sum_{m=1}^{M}(y_{b,s,m})^2 = \sum_{m=1}^{M}\left(\sum_{u\in U(t)}\sqrt{\beta_{u,b}}h_{u,b,m}\mathbf{1}_{\{s\in S(u)\}} + w_{b,s,m}\right)^2$$

We use our OMP inspired algorithm, that at each step identifies 5 largest components of the residual signal as the pilot sequence and sets the

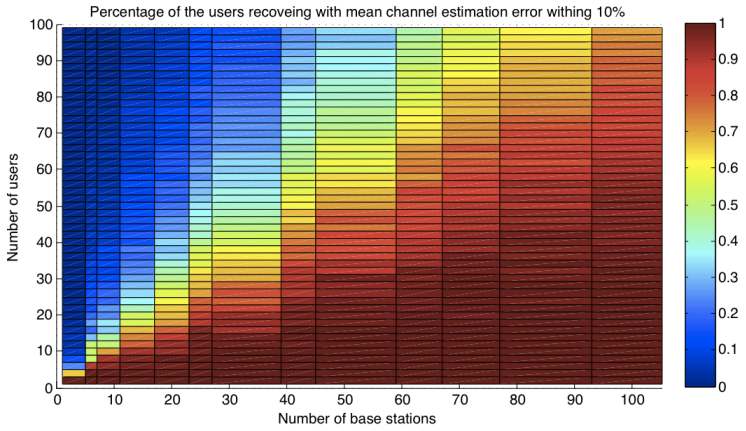

Figure 3.7: Percentage of users in the system, whose mean channel coefficient estimation error is less then 10%. We vary the number of users and the number of base stations. We can notice a linear trend, showing that base stations recover for high percentage of users their corresponding channel within mean error less than 10% .

corresponding path loss parameter β to the smallest value of those components.

Channel estimates $h_{u,b,m}$ are recovered afterwards with the MMSE estimator.

In Figure 3.6 we plot the percentage of users whose worst channel coefficient (over all antennas) is within 10% of the true channel coefficient. Similarly, in Figure 3.7 we see the percentage of users with average estimation error within 10% of the real channel coefficients.

3.9.3 Time varying behavior of admitted users

Lastly, we look into the system that is updating the channel coefficients for the users who are leaving the system, and new users who are entering the

system.

In this setup we look at four different systems:

- **Genie aided system:** The channel coefficients for each user, base station pair are perfectly known at the base station, therefore the arrival or departure of a user is always recoverable.

- **Random orthogonal pilots:** Each user picks random orthogonal pilots for transmission. If there is already a user at the same base station that is using the same pilot sequence the new user cannot be admitted into the system.

- **Orthogonal aligned pilots:** Each user checks if the base station has a free orthogonal pilot sequence. If there is such sequence, user claims it and uses it until it leaves. If there is no such sequence, the user is not admitted into the system.

- **Non-orthogonal pilots:** Each user picks a random non-orthogonal pilot sequence. The user is admitted into the system if it uses a distinct pilot sequence from all other users in the cell, and it recovers the channel coefficients within 10% of the real channel coefficients.

Notice, that the number of orthogonal sequences is always set to the number of pilot locations S, while the number of non-orthogonal sequences can be polynomial in S.

The comparison of the systems in terms of admissible users is given in Figure 3.8. The setup details are the following. There are 16 pilot location, $S = 16$. At the initial time step all base stations have 16 users associated with them, with users in the same cell having distinct pilot sequences. The average change in the number of users at each time step 0.4 users per base station (actual values are random, following Poisson distribution). On average, half of these users are leaving system while half are entering the system. Arriving users are places uniformly over the coverage of all base stations. There are $S^2 = 256$ non-orthogonal pilot sequences. We run the experiment 200 times for each of the 300 time steps and average the results over all realizations per each time step. What we can see is that in slowly changing system, usage of non-orthogonal pilots helps results in higher number of admitted users.

3.10 Conclusion

In Time Division Duplex massive MIMO systems, channel estimation is an important bottleneck for maximizing throughput of the system. In particular, the errors in the channel estimation that are a result of pilot sequence reuse have dominant effect on the SNR of the transmitted/received signal. In order to improve on the channel estimation we exploit the time correlation and present a new way of using non-orthogonal pilots in order to recover the channel coefficients.

We show empirical evidence in favor of feasibility of this communication model in terms of accurate channel estimation, increased sum rate perfor-

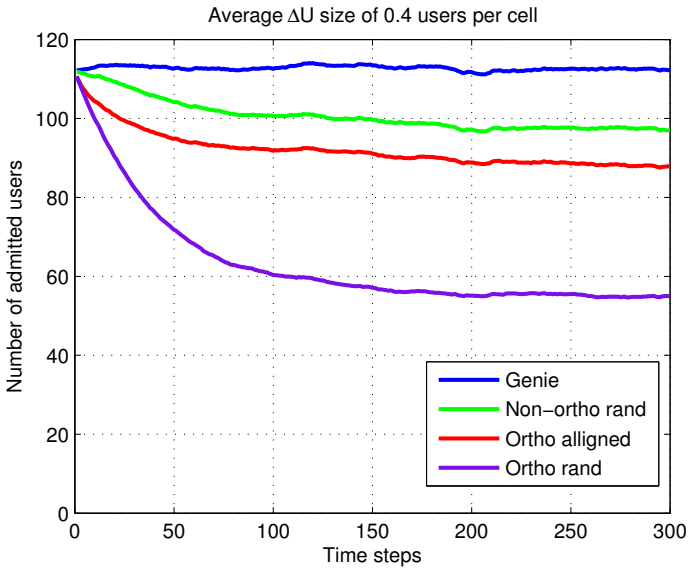

Figure 3.8: The number of admissible users in the system.

mance, and the increase in the number of users that can be admitted in slowly time varying system.

Chapter 4

Decoding Genetic Variations: Communications-Inspired Haplotype Assembly

4.1 Introduction

Recent advancements in high-throughput DNA sequencing [70, 36, 58, 71] have enabled fast and affordable re-sequencing of individual genomes and hence opened up the possibility of conducting routine tests of genetic variations. Identification and study of such variations helps reveal susceptibility to genetic and complex diseases, and may lead to the development of personalized treatment plans adjusted to individual genetic codes [53, 84, 20, 30]. Majority of chromosome pairs in diploid organisms, including humans, are homologous – they carry fundamentally the same type of information and are structurally similar but not identical. A common variation among chromosomes in a homologous pair is single nucleotide polymorphisms (SNPs), i.e., occurrence of different nucleotides in the corresponding locations on the chromosomes. Variations between chromosomes are fully specified by haplotypes, ordered sequences of SNPs associated with each of the chromosomes. [1]

[1] The work in this chapter recently appears in [68].

To assemble haplotypes of an individual organism, we may rely on high-throughput DNA sequencing platforms which oversample the DNA sequence to create a library of overlapping reads. The reads are relatively short – typically, on the order of hundreds of nucleotides. Paired-end reads link genome fragments that are large distances apart (hundreds to thousands of bases) by having inserts of approximately known lengths that separate two reads. To provide information that can be used to assemble haplotypes, a pair of linked reads must cover more than one SNP location. If sequencing were not affected by errors and the subsequent SNP and genotype calling steps were free of any uncertainties, haplotype assembly for diploid organisms would be straightforward and could be reduced to separating reads into two subgroups – one for each haplotype in a pair. In the realistic case of having erroneous data, such classification necessarily results in subgroups that contain reads with conflicting information. In literature, haplotype assembly has led to several optimization problems, most of them attempting to minimize the number of transformations of the data set needed to make the reads consistent with having originated from one of the chromosomes [73]. In particular, it motivated the use of the minimum fragment removal, minimum edge removal, minimum SNP removal, and minimum error correction (MEC) optimization criteria. The MEC criterion, which attracted the most attention in recent years and is known to be NP-hard [57], is considered in this chapter. Various methods for doing haplotype assembly by optimizing the MEC criterion have been developed. The optimal yet computationally intensive branch-and-bound

scheme was proposed in [87]. As an alternative, several heuristic algorithms that trade off accuracy for speed have been proposed [87, 17, 56, 55, 47, 48, 54]. More recent methods include HapCompass [3] and the widely used HapCut [9] algorithm.

Our key observation presented in this chapter has been that the MEC formulation of the haplotype assembly problem is identical to the task of deciphering a coded message transmitted over a noisy communication channel [21, 69]. Decoding of noisy messages has been extensively studied in communication theory over the last several decades [31, 50, 51, 52, 62]. Exploiting the aforementioned connection, we first propose a haplotype assembly method that relies on the bit-flipping algorithm originally developed in the context of decoding low-density parity check codes [31]. We then design a belief propagation algorithm that provides higher accuracy than the bit-flipping scheme at the cost of a slight increase in computational complexity. When tested on the *1000 Genomes Project* and Fosmid [28] data sets, the proposed methods compare favorably with HapCut and HapCompass while being significantly faster.

Beside diploids, high-throughput sequencing has enabled studies of genetic variations in polyploid organisms which have $k > 2$ chromosomes. Haplotype assembly for polyploids is considerably more challenging and requires larger coverage to enable separation of the reads. Existing prior work on haplotype assembly for polyploids includes HapCompass [3] and HapTree [12]. We extend our belief propagation algorithm to the assembly of polyploid haplo-

types and demonstrate in simulation studies that it significantly outperform HapCompass.

The remainder of the chapter is organized as follows. We start by considering haplotype assembly for diploids and introduce the system model and problem formulation in Section 4.2. In Sections 4.3 and 4, we present the main contributions: a representation of haplotype assembly as a decoding problem and two algorithms that rely on that representation, respectively. An analytical bound on the performance is given in Section 4.5. Section 6 extends the belief propagation algorithm to the polyploid case. Results and discussion are presented in Section 4.7 while the conclusion and future work are in Section 4.8.

4.2 Notation and problem statement

Following sequencing, aligning to a reference, and SNP and genotype calling, to facilitate haplotype assembly the data is typically organized in a SNP fragment matrix. Segments of the reads that cover homozygous sites provide no information about haplotypes and are hence discarded. Since the SNP sites in diploids are typically bi-allelic, we may denote them using binary symbols $\{0, 1\}$. The (i, j) entry of the SNP fragment matrix $\mathbf{R}$ indicates 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 denoted by the symbol $\times$. As an illustration, the resulting SNP fragment matrix may have the following

form,

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

4.2.1 Notation

Throughout this chapter, we use the following notation:

$(\mathbf{h}, \bar{\mathbf{h}})$: an unordered pair of a haplotype and its complement, with support in $\{0,1\}^n$.

$(\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{k-1})$: an unordered k-touple of all the haplotypes on all the chromosomes with support in $\{0,1\}^n$.

$\mathbf{r}_i$: read i, $1 \leq i \leq m$, with support in $\{0,1,\times\}^n$, where $\times$ denotes unobserved alleles.

$\mathbf{R}$: an $n \times m$ matrix whose i^{th} row corresponds to $\mathbf{r}_i$ and j^{th} column corresponds to the j^{th} SNP site.

$\mathbf{s}$: a vector indicating whether a read is associated with $\mathbf{h}_0$, $\mathbf{h}_1$,... or $\mathbf{h}_{k-1}$, with support in $\{0,1,\ldots,k-1\}^m$.

$\mathbf{c}$: a vector collecting numeric entries of the matrix $\mathbf{R}$.

$\mathbf{G}$: generator matrix of a linear block code.

H: parity-check matrix associated with the generator matrix **G**.

4.2.2 Problem statement

Define a measure of distance d between two symbols in the ternary alphabet $\{0, 1, \times\}$ as

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

With the adopted MEC criterion, the goal of haplotype assembly for diploid organisms is to minimize Z over a binary vector $\mathbf{h}$,

$$Z = \sum_{i=1}^{m} \min(\mathrm{hd}(\mathbf{r}_i, \mathbf{h}), \mathrm{hd}(\mathbf{r}_i, \bar{\mathbf{h}})), \tag{4.1}$$

where $\mathrm{hd}(\cdot,\cdot)$ denotes the generalized Hamming distance defined as

$$\mathrm{hd}(\mathbf{r}_i, \mathbf{h}) = \sum_{j=1}^{n} d(R(i,j), h_j),$$

where $R(i,j)$ denotes the (i,j) entry in $\mathbf{R}$. Intuitively, minimizing (4.1) leads to finding the smallest number of binary entries (alleles) in $\mathbf{R}$ that should be flipped so that the rows of $\mathbf{R}$ can be unambiguously associated with either $\mathbf{h}$ or $\bar{\mathbf{h}}$. For convenience, we will refer to $\mathbf{h}$ as the reference haplotype.

The MEC objective is often used as a proxy for the switch error rate (SWER). The switch between positions i and $i+1$ is defined as the event $\hat{\mathbf{h}}_i = \mathbf{h}_i$ and $\hat{\mathbf{h}}_{i+1} \neq \mathbf{h}_{i+1}$, or $\hat{\mathbf{h}}_i \neq \mathbf{h}_i$ and $\hat{\mathbf{h}}_{i+1} = \mathbf{h}_{i+1}$. Note that, unlike the MEC score that is a function of the fragment matrix and the recovered haplotype, SWER is calculated with respect to a (known) underlying haplotype. We will use SWER to test the performance of the proposed algorithms in Section 7.

4.3 Reformulating Haplotype assembly as the decoding problem

It is beneficial to briefly consider the scenario where the SNP fragment matrix $\mathbf{R}$ is error-free. In particular, let us assume that the reference haplotype

$$\mathbf{h}=[0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1]$$

and its complement $\bar{\mathbf{h}}$ (each comprising $m = 6$ alleles) were sampled with $n = 8$ reads, and that the origin of the reads is indicated in the following read select vector,

$$\mathbf{s}=[0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1]^T.$$

If the value of the i^{th} component of $\mathbf{s}$ is 0, that indicates the i^{th} row of $\mathbf{R}$ originated by sampling $\mathbf{h}$; otherwise, it originated by sampling $\bar{\mathbf{h}}$. Then the corresponding error-free fragment matrix is of the form

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

Clearly, all of the variables in the previous representation of the data ($\mathbf{h}$, $\mathbf{s}$, and $\mathbf{R}$) have binary numerical entries. A closer inspection reveals that if the (i, j) entry in $\mathbf{R}$, $R_{i,j}$, is numerical (i.e., 0 or 1), it is obtained as the result of an exclusive-OR (XOR) operation between the i^{th} and j^{th} entries of $\mathbf{s}$ and $\mathbf{h}$, respectively, as indicated in Table 4.1.

s_i	h_j	$R_{i,j}$
0	0	0
0	1	1
1	0	1
1	1	0

Table 4.1: Tabular representation of the known entries in the error-free fragment SNP matrix as a function of the reference haplotype and read select variables.

Interestingly, XOR functions are building blocks of error-correcting codes in communication systems, which we briefly summarize next.

4.3.1 Communication systems

In data communication systems, the goal of point-to-point communication is to reliably transmit and receive (decode) messages that are adversely affected by the transmission medium. The most simple communication system consisting of a source, encoder, channel, decoder and destination is illustrated in Fig. 4.1. The coder introduces redundancy into the message in order to combat unknown effects of the noisy channel by adding redundancy to the binary messages generated by the source. Output of the coder – a codeword that belongs to a pre-defined codebook – is transmitted across the channel that modifies it in a nondeterministic way. The task of the decoder is to reverse the effects of the channel and map the received signal back to the "closest" valid codeword, which is then converted back into the message and forwarded to the destination.

A special class of codes are those where each bit in the codeword is

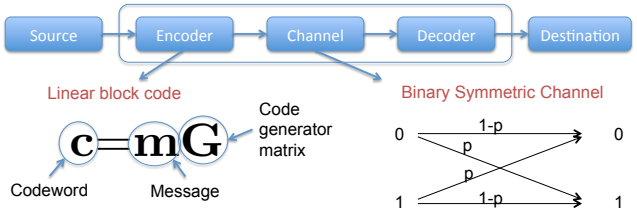

Figure 4.1: Components of a simple communication system operate as follows: 1) a message is sent by a source, 2) a coder maps messages to a codeword using a set of linear functions, 3) the codeword is corrupted by the binary symmetric channel, 4) a decoder maps back the corrupted codeword into a valid message, 5) the recovered message reaches the destination.

a linear function of the message bits. Such linear codes are fully described by a code generator matrix $\mathbf{G}$. Each column of $\mathbf{G}$ specifies the linear function used to obtain the corresponding codeword bit, and the entire codeword is formed by simply multiplying over $GF(2)$ the message with the code generator matrix; multiplications over $GF(2)$ are identical to the exclusive-OR operations illustrated in Table 4.1. We remark that a subclass of communication channels, so-called binary symmetric channels, invert each transmitted bit independently with the same probability.

4.3.2 Decoding haplotypes

Let us define a "message" as the vector formed by concatenating the haplotype vector $\mathbf{h}$ with the read select vector $\mathbf{s}$, $\mathbf{m} = [\mathbf{h} \quad \mathbf{s}]$. Let $\{f_k\}$ denote the collection of indices $\{(i_k, j_k)\}$ identifying positions where the matrix $\mathbf{R}$ has numeric entries, i.e., $R(i_k, j_k) \neq \times$, $1 \leq k \leq M$, where M denotes the total number of informative (binary) entries in $\mathbf{R}$. Define the "code generating"

matrix $\mathbf{G}$ with the entries

$$G(l,k) = \begin{cases} 1 \text{ if } l = j_k \text{ or } l = i_k + n,\ 1 \leq k \leq M, \\ 0,\ \text{otherwise,} \end{cases}$$

where n denotes the length of the haplotype. Therefore, each column of $\mathbf{G}$ by construction has only two non-zero elements, one at the location (j_k, k) and the other at the location $(n + i_k, k)$. To clarify the construction of $\mathbf{G}$, we give an example next.

Example 3. Consider the following SNP fragment matrix,

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

The set $\{f_k\}$ collects indices of the binary elements in $\mathbf{R}$ in a row-wise order, $\{f_k\} = \{(1,1),\ (1,2), (2,2), (2,3), (3,1), (3,3)\}$. The corresponding code generating matrix $\mathbf{G}$ has the following form,

$$\mathbf{G}=\left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right]$$

Each column of $\mathbf{G}$ is associated with one binary entry in $\mathbf{R}$, where the two non-zero entries in the k^{th} column of $\mathbf{G}$ correspond to indices (i_k, j_k) of the k^{th} (ordered row-wise) informative entry in $\mathbf{R}$, $R(i_k, j_k) \neq \times$. The horizontal line in $\mathbf{G}$ separates the rows of $\mathbf{G}$ associated with the columns i_k of $\mathbf{R}$ (i.e., with the SNP position) from the rows of $\mathbf{G}$ associated with the rows j_k of $\mathbf{R}$ (i.e., with the reads).

In the absence of SNP-calling errors, it holds that

$$\mathbf{c} = [\mathbf{h} \quad \mathbf{s}]\mathbf{G}, \tag{4.2}$$

where the k^{th} entry of $\mathbf{c}$ is equal to the k^{th} (ordered row-wise) numeric entry of $\mathbf{R}$, i.e., $c_k = R(i_k, j_k)$.

Remark: Note that any permutation of the rows or columns of $\mathbf{G}$ corresponds to the permutation of the components in the vectors $\mathbf{h}$ and $\mathbf{s}$, or the order in which numeric entries of matrix $\mathbf{R}$ occur in the codeword $\mathbf{c}$. All such generated code generator matrices are valid (and equivalent) representations of $\mathbf{R}$.

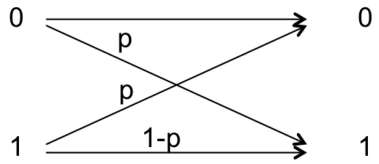

Figure 4.2: An illustration of a binary symmetric channel with crossover probability p.

Going back to the setting of Example 1 where $\mathbf{h} = [0\ 1\ 1]$, $\mathbf{s} = [0\ 0\ 1]$ and $\mathbf{c} = [0\ 1\ 1\ 1\ 1\ 0]$, it is easy to verify (4.2). However, sequencing errors adversely affect SNP and genotype calling and hence the SNP fragment matrix $\mathbf{R}$ typically has a fraction of entries that are incorrect (flipped). This can be modeled by thinking of the "codeword" $\mathbf{c}$ in (4.2) as being transmitted across the binary symmetric channel (BSC) illustrated in Fig. 4.2, where p denotes the probability of inverting a bit (i.e., p represents the error rate in $\mathbf{R}$). Therefore, the possibly erroneous entries in $\mathbf{R}$ can be represented as

$$\mathbf{y} = [\mathbf{h} \quad \mathbf{s}]\mathbf{G} + \mathbf{e}, \tag{4.3}$$

where the k^{th} entry of $\mathbf{y}$ is $y_k = R(i_k, j_k)$, $\mathbf{e}$ denotes the error vector, and all the operations are in $GF(2)$. The goal of haplotype assembly can be restated as follows: *Given the vector* $\mathbf{y}$ *and the matrix* $\mathbf{G}$*, both derived from the SNP-fragment matrix* $\mathbf{R}$*, find the most likely vector* $[\mathbf{h} \;\; \mathbf{s}]$.

To facilitate the decoding of $[\mathbf{h} \;\; \mathbf{s}]$ and hence perform haplotype assembly, we rely on the parity check matrix $\mathbf{H}$. For the linear codes defined by the encoding operation (4.2), $\mathbf{H}$ is orthogonal to $\mathbf{G}^T$, i.e., the range of $\mathbf{G}^T$ is the null space of $\mathbf{H}$. Given $\mathbf{G}$, we find $\mathbf{H}$ by means of the simple Gaussian elimination. Note that

$$\mathbf{Hy} = \mathbf{H}\left(\mathbf{G}^T[\mathbf{h} \quad \mathbf{r}]^T + \mathbf{e}\right) = \mathbf{He},$$

and that the vector $\mathbf{e}$ can be viewed as the distance between the observations $\mathbf{y}$ and the closest valid codeword. The minimum distance decoding is concerned with finding the codeword $\mathbf{c}$ (or, equivalently, $\mathbf{e}$) that solves the minimization problem

$$\begin{aligned} \min_{\mathbf{c}\in\mathcal{C}} d(\mathbf{c},\mathbf{y}) &= \min_{\mathbf{c}\in\mathcal{C}} \|\mathbf{y}-\mathbf{c}\|_0 = \min_{\mathbf{c}\in\mathcal{C},\mathbf{y}=\mathbf{c}+\mathbf{e}} \|\mathbf{e}\|_0 \\ &= \min_{\mathbf{e}:\mathbf{H}(\mathbf{y}+\mathbf{e})=0} \|\mathbf{e}\|_0, \end{aligned} \tag{4.4}$$

where $\|\cdot\|_0$ denotes the l_0-norm (the number of non-zero entries) of its argument. Recall that the goal of the minimum error correction (MEC) formulation of the haplotype assembly problem is to select $\mathbf{h}$ minimizing the number of entries in $\mathbf{R}$ that need to be flipped so that there is no conflicting information in the SNP-fragment matrix. Therefore, its objective (4.1) can be restated as

$$Z = \min_{\mathbf{e}:\mathbf{H}(\mathbf{y}+\mathbf{e})=0} \|\mathbf{e}\|_0.$$

Comparing the two optimizations above, we see that the minimum distance decoding also leads to minimization of the MEC score. Moreover, note that in the absence of any prior information the minimum distance decoding coincides with the maximum a posteriori decoding.

Based on the observed connection between communications and haplotype assembly, we design graphical models and utilize them for the design of algorithms solving the latter problem.

4.3.3 Graphical models

Fig. 4.3 shows the graphical representation of the measurement model (4.3), illustrating the interactions between elements of the read select vector $\mathbf{s}$, reference haplotype $\mathbf{h}$ and the observations collected in $\mathbf{y}$. Both the reference haplotype and read select vector are unobserved variables in this model, and the interactions between them are driven by the structure of the code generating matrix $\mathbf{G}$. These interactions are depicted by the square nodes in the graph, each connected to exactly one SNP variable (a component of $\mathbf{h}$), one read select variable (a component of $\mathbf{s}$) and one observation (a component of $\mathbf{y}$, connected via the codeword $\mathbf{c}$). This graphical model will be the basis for the derivation and implementation of the belief propagation algorithm for haplotype assembly presented in the next section.

An alternative graphical model, based on the parity check matrix $\mathbf{H}$, is illustrated in Fig. 4.4. The variables in this model are associated with the entries in the vector $\mathbf{y}$. The rows of the parity check matrix define parity check

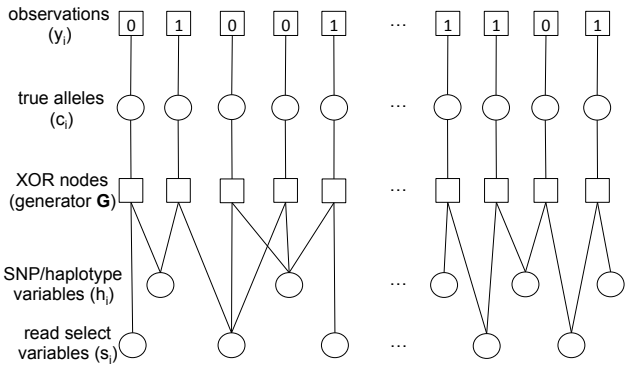

Figure 4.3: A graphical model illustrating how the data used for haplotype assembly is generated. The numeric entries in the SNP fragment matrix $\mathbf{R}$ are collected into a vector $\mathbf{y}$; due to sequencing and data processing errors, these may differ from the true alleles. Read select variables (components of $\mathbf{s}$), SNP variables (components of the haplotype vector $\mathbf{h}$), and true alleles (components of $\mathbf{c}$) are connected through check nodes (i.e., XOR functions defined by the structure of $\mathbf{G}$).

nodes illustrated in the figure. The edges in the graphical model emanating from a parity check node connect to the variables identified by the locations of the non-zero entries of the corresponding row of $\mathbf{H}$. When the variables connected to the parity check nodes are such that their linear combinations (over GF(2)) at each node are zero, consistent haplotype can be recovered. This will be exploited to design and implement the bit-flipping haplotype assembly algorithm in the next section.

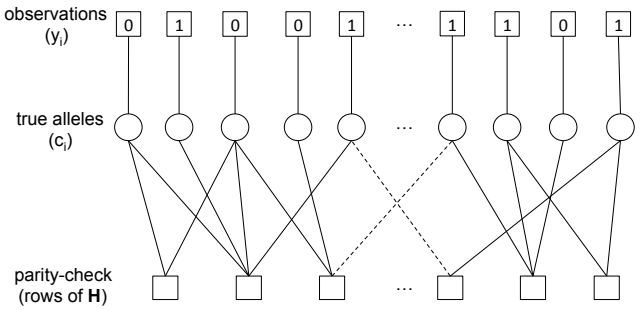

Figure 4.4: A graphical model facilitating haplotype assembly via satisfying conditions imposed by the parity check matrix $\mathbf{H}$. The i^{th} parity check node, defined by the i^{th} row of $\mathbf{H}$, is connected to the variable c_k if $H(l,k) \neq 0$.

4.4 Haplotype Assembly via Decoding of Linear Block Codes

Motivated by the decoding algorithms that correct noise-induced errors in communication systems, in this section we present two haplotype assembly methods: the bit flipping algorithm and the belief propagation algorithm. These are computationally efficient yet highly accurate heuristics for solving the NP-hard assembly problem.

4.4.1 The bit-flipping algorithm

The bit-flipping algorithm relies on the graphical model illustrated in Fig. 4.4. The basic idea of the algorithm is to examine each variable node of the graph and find the number of parity check equations $\mathbf{Hy} = 0$ it violates; there exists one such variable for each numerical entry in $\mathbf{R}$. The bit in $\mathbf{y}$ with the largest margin of unsatisfied parity check equations versus satisfied

Algorithm 4 Bit Flipping Haplotype Assembly

procedure BF
  $H \leftarrow$ *getParityCheckMatrix*
  $\mathbf{c}(0) \leftarrow 1 - 2\tilde{\mathbf{c}}$ set initial values of check nodes to 1 or -1
  $t \leftarrow 1$ iterations
  repeat
    $v_{i,j}^{(t)} \leftarrow c_i(t-1)$ messages from entries to checks
    $u_{j,i}^{(t)} \leftarrow \prod_{k \in N_j i} v_{k,j}^{(t)}$ messages from checks to entries
    $\delta_i(t) \leftarrow -c_i(t-1) \left(\sum_{j \in N_i} u_{j,i}^{(t)} \right)$ marginal values
    if $\min_i(\delta_i(t)) < 0$ **then**
      $k \leftarrow \arg\min(\delta_i(t))$ find the index of highest margin.
      $c_k(t) \leftarrow -c_k(t-1)$ flip the value of the entry.
      $\forall i \neq k : c_i(t) \leftarrow c_i(t-1)$ keep all other values same
    end if
    $t \leftarrow t + 1$
  until $(t >$ MAXITER$) \vee (\forall i, j : c_i(t) == u_{j,i}^{(t)}) \vee (\min_i(\delta_i(t)) \geq 0)$
end procedure

parity check equations is flipped (i.e., the corresponding entry in $\mathbf{y}$ is changed from 0 to 1 or vice versa). We proceed greedily, identifying and changing the component of $\mathbf{y}$ such that the number of unsatisfied parity check equations reduces in each step.

Intuitively, the algorithm in each step attempts to improve the objective function of the minimum-distance decoding, which coincides with the goal of the MEC haplotype assembly. The procedure is terminated when there are no more alleles with a negative drift (with more unsatisfied than satisfied check nodes) or when there are no parity check node violations. The procedure is formalized as Algorithm 4.

There are two main sources of randomness in the bit-flipping algorithm

– construction of the parity check matrix and breaking ties among flipping variables. For one codebook there are many different code generator matrices with many different orthogonal parity check matrices. We use breadth-first search on the bi-partite graph of the haplotype and read select variables to form a code generator matrix from a random starting point and rely on the Gaussian elimination to obtain the parity check matrix. While performing the bit-flipping algorithm, in case of several variables having the same number of unsatisfied versus satisfied check nodes, we break the ties uniformly at random.

The bit-flipping algorithm was originally proposed for decoding of low-density parity check (LDPC) codes in [31], with the difference that the algorithm there flips more than one bit in each iteration. We should also point out that the bit-flipping algorithm is related to the coordinate descent algorithm for l_1-norm minimization, where the goal is to reconstruct a sparse error vector.

4.4.2 The belief propagation algorithm

Belief propagation is a message-passing scheme for inference in graphical models. A node in the graph receives messages from the nodes its connected to and, based on the received messages, computes and broadcasts its belief about the associated variable. Our belief propagation algorithm works with the code generator matrix and does not require computation of the parity check matrix; for this reason, its computational complexity compares favorably with bit-flipping, as shown in Section 7.

The algorithm takes as input four variables: the starting point sp, the probability of error p_e, the maximal number of iterations MAXITER, and the precision ϵ used as a stopping criterion. In the graphical model representing the haplotype assembly problem, the edges connecting the read nodes with the SNP (haplotype) nodes are associated with the numeric entries in $\mathbf{R}$; since those entries are potentially erroneous, we define an auxiliary variable $y_{i,j}$ that accounts for the probability of read errors,

$$y_{i,j} = (1/2 - p_e)(2R(i,j) - 1) + 1/2. \tag{4.5}$$

In fact, the variables $y_{i,j}$ are the beliefs propagated from the haplotype variables to the read select variables via the XOR function nodes in Fig. 4.

We use bars $\bar{}$ to denote the belief complement of a variable; in particular, for any value x, $\bar{x} = 1 - x$. We define the set of active haplotype positions A_H and the set of active reads A_R to be the collection of haplotype and read nodes which received a message on any of their edges in a given iteration, respectively. The belief propagation algorithm for haplotype assembly is formalized as Algorithm 2.

4.4.3 Complexity

Haplotype assembly concerned with optimizing the MEC criterion is known to be NP-hard [73]. Both bit flipping and belief propagation algorithms are heuristics, and we are interested in characterizing their complexity. Here we provide the complexity analysis in terms of the length of the connected

Algorithm 5 Belief Propagation Haplotype Assembly

procedure BP
  $sp \leftarrow randomStartingPoint$
  if starting point is a read **then**
    $r_i \leftarrow sp$
    $A_R^{(0)} \leftarrow r_i$ (set active reads)
    $A_H^{(0)} \leftarrow \Gamma(A_R^{(0)})$ (set active haplotype)
    $b_{r_i}^{(0)} \leftarrow 1 - p_e$ (set initial beliefs)
    $m_{r_i \to h_j}^{(0)} \leftarrow 1 - p_e$ (set initial messages)
    $m_{h_j \to r_i}^{(0)} \leftarrow \frac{1}{Z_{h_j}} \prod_{r_k \in A_R^{(0)} \setminus r_i} \left(m_{r_k \to h_j}^{(0)} y_{k,j} + \bar{m}_{r_k \to h_j}^{(0)} \bar{y}_{k,j} \right)$
  else
    $h_j \leftarrow sp$
    $A_H^{(0)} \leftarrow h_j$ set active haplotype
    $b_{h_j}^{(0)} \leftarrow 1 - p_e$ (set initial belief)
    $m_{h_j \to r_i}^{(0)} \leftarrow 1 - p_e$ (set initial messages)
  end if
  $t \leftarrow 1$ iterations
  repeat
    $A_R^{(t)} \leftarrow \Gamma(A_H^{(t-1)})$ update the active reads
    $m_{r_i \to h_j}^{(t)} \leftarrow \frac{1}{Z r_{i,j}} \prod_{h_k \in A_H^{(t-1)} \setminus h_j} \left(m_{h_k \to r_i}^{(t-1)} y_{i,k} + \bar{m}_{h_k \to r_i}^{(t-1)} \bar{y}_{i,k} \right)$
    $b_{r_i}^{(t)} \leftarrow \frac{1}{Z r_{b_i}} \prod_{h_j \in A_H^{(t-1)}} \left(m_{h_j \to r_i}^{(t-1)} y_{i,j} + \bar{m}_{h_j \to r_i}^{(t-1)} \bar{y}_{i,j} \right)$
    $A_H^{(t)} \leftarrow \Gamma(A_R^{(t)})$ update active haplotype
    $m_{h_j \to r_i}^{(t)} \leftarrow \frac{1}{Z h_{i,j}} \prod_{r_k \in A_R^{(t)} \setminus r_i} \left(m_{r_k \to h_j}^{(t)} y_{k,j} + \bar{m}_{(r_k \to h_j)}^{(t)} \bar{y}_{k,j} \right)$
    $b_{h_j}^{(t)} \leftarrow \frac{1}{Z h_{b_j}} b_{h_j}^{(t-1)} \prod_{r_i \in A_R^{(t)}} \left(m_{r_i \to h_j}^{(t)} y_{i,j} + \bar{m}_{r_i \to h_j}^{(t)} \bar{y}_{i,j} \right)$
    $t \leftarrow t + 1$
  until $(t >$ MAXITER$) \vee (\|\mathbf{b_h}^{(t)} - \mathbf{b_h}^{(t-1)}\|_2 \leq \epsilon)$
end procedure

component, i.e., a haplotype block that is connected with the reads. For convenience, we will keep the same notation but with this slightly altered meaning: variable n denotes the length of the connected components of the haplotype (i.e., the length of a haplotype block) and $\mathbf{R}$ is the submatrix of the SNP-fragment matrix containing information relevant to the haplotype block under consideration. We assume that the total number of binary entries in $\mathbf{R}$ is $O(n)$, which is a reasonable assumption given the sparse structure of the SNP-fragment matrix.

We distinguish between time and space complexity. Time complexity characterizes the minimum time needed to obtain the output of the algorithm and is concerned with its longest sequential path. This measure is particularly interesting when running algorithms on distributed systems. On the other hand, space complexity characterizes the algorithm's memory requirements.

The bit-flipping algorithm. The time complexity for the bit-flipping algorithm is $O(n)$. To show this, we divide the procedure in two major steps: finding the parity check matrix and running the bit flipping algorithm. The time complexity of finding the parity check matrix from the code generator matrix is $O(n)$. This is lower than the standard Gaussian elimination since we arrange the code generator matrix to have lower triangular form by following an $O(n)$ breadth-first search that permutes the haploype and read select variables in an appropriate fashion. In addition, each pivot has only one non-zero value above it, and thus the time needed for the transformation to the fully reduced form is again $O(n)$. On the other hand, the maximal number of iter-

ations of the bit flipping algorithm is $O(n)$. The number of iterations is upper bounded by the number of unsatisfied parity check nodes. Since at each iteration this number is reduced by at least 1, the maximal number of iterations is upper bounded by the number of check nodes in the parity check matrix. Now, from the procedure used to generate parity check matrix one can observe that the number of check nodes in the parity check matrix is strictly less than the number of entries in $\mathbf{R}$, which is again given by $O(n)$. The space complexity of the BF algorithm does not exceed $O(n^2)$ since the number of connections between parity check nodes and observed alleles is at most quadratic in n.

The belief propagation algorithm. The time complexity of the belief propagation algorithm is bounded by the number of iterations. Unlike the bit-flipping algorithm, the belief propagation algorithm does not need the transformation from the code generator matrix to the parity check matrix. The number of iterations of the belief propagation algorithm is upper bounded by the MAXITER variable. Note that MAXITER variable needs to be larger than the depth of the graphical model since we want all the SNPs and read select variables to be in the active set. Also note that in each iteration all the values of the active nodes need to be updated, whereas in the bit flipping algorithm only a single variable is updated. The space complexity of the belief propagation algorithm is $O(n)$ since all that is required to store is the description of the original graphical model.

Initialization and reruns (for both algorithms). Performance of both algorithms depends on the starting point; hence to obtain better results we

repeat the execution of the algorithm from different starting points. We use a random restart and keep searching for the solution as long as the MEC score improves. The number of times we rerun the algorithms is adaptively determined based on the incremental change in the MEC score.

4.5 Limits of performance of haplotype assembly

We find analytical expressions for the achievable accuracy of the haplotype assembly problem concerned with optimization (4.4). In particular, we compute the lower bounds on the probability of haplotype assembly and switch errors, and the expected number of SNP and switch errors.

To start, in Fig. 4.5 we introduce a simple bipartite graph model of the SNP fragment matrix. The nodes on the left correspond to the haplotype/SNP values (i.e., components of $\mathbf{h}$) while the nodes on the right correspond to the read select variables (components of $\mathbf{s}$). The edge between a SNP and a read select node exists if the read covers the SNP and provides information about its allele. Here the actual values of the observed alleles are not important since the error probability characterizing the performance of a linear block code is the property of a codebook, not a specific codeword. In Fig. 4.5 there are three examples of cuts that will be of interest in our analysis. Cut C1 isolates a single SNP node, and we define the cardinality of that cut to be equal to the coverage of the isolated SNP. Cuts C2 and C3 partition the SNP nodes in two non-empty sets.

The probability of error, defined as the probability that the assembled

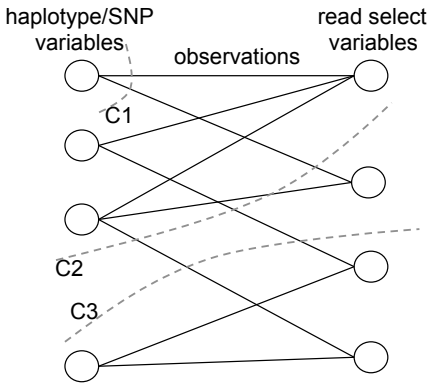

Figure 4.5: A sample bipartite graph representation of the fragment matrix. The nodes on the left represent SNPs while the nodes on the right represent read select variables. The edge between a SNP and a read select node exists if the read covers the SNP location. Three cuts are illustrated in the figure: C1, C2 and C3. We refer to the cut C1 as isolating cut since it contains all the edges emanating from a single SNP, and define the cardinality of the cut as the coverage of the corresponding SNP. We refer to the cuts C2 and C3 as separating cuts since they partition SNPs into two non empty sets.

haplotype is different from the true one, is given by

$$P(\hat{\mathbf{h}} \neq \mathbf{h}) \geq 1/2 \sum_{i=\lfloor k/2 \rfloor + 1}^{k} \binom{k}{i} p^i (1-p)^{k-i},$$

where p denotes the SNP calling error rate and k is the smallest coverage over all SNP positions (the SNP with the lowest coverage is the most error prone). If the number of SNP calling errors is large ($> \lfloor k/2 \rfloor + 1$), it is likely that the algorithms will arrive at a haplotype sequence with an erroneous SNP position. In the bipartite graph representation in Fig. 4.5, this bound corresponds to the errors introduces along the smallest isolating cut.

Accuracy of haplotype assembly is often expressed in terms of the

switch probability. Switch refers to the event where a subsequence of consecutive errors starts at a site in the assembled haplotype; the errors essentially imply mistaking a segment of $\hat{\mathbf{h}}$ for that of of $\mathbf{h}$. In the bipartite graph representation each separating cut potentially leads to the switch in the decoded haplotype. Now, looking at all possible locations of switches, it is straightforward to show that the probability of a switch can be bounded by

$$P_{\text{switch}} \geq 1/2 \sum_{i=\lfloor s/2 \rfloor+1}^{s} \binom{s}{i} p^i (1-p)^{s-i},$$

where s denotes the minimum separating cut of the bipartite graph representation of fragment matrix. Notice that the switch probability is always greater than the probability of an erroneous SNP, as the smallest isolating cut is just one possible separating cut in the graph.

Next, we examine the expected number of erroneous (i.e., inverted) SNPs. This expectation is lower bounded by the coverage errors as

$$\begin{aligned} E[\|\hat{\mathbf{h}} - \mathbf{h}\|_0] &= E[\sum_{i=1}^{n} \mathbf{1}_{\hat{h}_i \neq h_i}] = \sum_{i=1}^{n} P(\hat{h}_i \neq h_i) \\ &\geq \sum_{i=1}^{n} \frac{1}{2} \sum_{j=\lfloor \frac{k_i}{2} \rfloor+1}^{k_i} \binom{k_i}{j} p^j (1-p)^{k_i-j} \end{aligned} \tag{4.6}$$

where k_i denotes the coverage of the i^{th} SNP. Note that on the right-hand side in (4.6) we sum up the probabilities of each SNP being inverted due to the errors introduced into the corresponding isolating cut. This leads to a lower bound because all isolating cuts in the bipartite graph are independent, i.e., each edge of the bipartite graph appears in at most one isolating cut.

For arbitrary separating cuts in the graph, the independence assumption does not hold (for example, cuts C2 and C3 are not independent since they share two edges). We need to take that into account when characterizing the lower bound on the expected number of switches,

$$\begin{aligned} E[\text{SWER}] &= \frac{1}{n}\sum_{i=2}^{n} P(\text{switch between } i-1 \text{ and } i) \\ &\geq \frac{1}{n}\sum_{i=2}^{n}\frac{1}{2}\sum_{j=\lfloor\frac{s_i}{2}\rfloor+1}^{s_i}\binom{s_i}{j}p^j(1-p)^{s_i-j}. \end{aligned}$$

Here s_i denotes the smallest cut separating the i^{th} SNP node that is independent of all other separating cuts $s_j \neq s_i$. It readily follows that

$$E[\text{SWER}] \geq \sum_{i=1}^{n}\frac{1}{2}\sum_{j=\lfloor\frac{k_i}{2}\rfloor+1}^{k_i}\binom{k_i}{j}p^j(1-p)^{k_i-j},$$

where k_i denotes the coverage of the i^{th} SNP.

We should point out that the lower bound on the probability of switching is always greater than the lower bound on the probability of error under maximum a posteriori decoding. For the scenario where the data is characterized by long reads and low coverage (as in, e.g., Fosmid datasets), the computed bounds are fairly tight; however, when the coverage is high and the reads are relatively short (resembling reads in *1000 Genomes Project* datasets), they tend to be loose. Either way, they provide a useful insight about the achievable accuracy of single individual haplotyping.

4.6 Polyploid haplotype assembly

In previous sections, we focused on haplotype assembly for diploids. We now turn our attention to the polyploid version of the haplotype assembly problem. For simplicity of presentation, we constrain SNPs to be bi-allelic; extension to the multi-allelic scenario is relatively straightforward (albeit somewhat cumbersome) and involves replacing vectors of beliefs by matrices. Hence, the alphabet used to denote alleles in the SNP data matrix remains $\{0, 1, \times\}$, where $\times$ marks the positions in a read not covering the corresponding SNP sites.

The goal of polyploid haplotype assembly concerned with optimizing the MEC criterion[2] is to minimize Z over the set $(\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{k-1})$,

$$Z = \sum_{i=1}^{m} \min(\mathrm{hd}(\mathbf{r}_i, \mathbf{h}_0), \mathrm{hd}(\mathbf{r}_i, \mathbf{h}_1), \ldots, \mathrm{hd}(\mathbf{r}_i, \mathbf{h}_{k-1})), \tag{4.7}$$

where k denotes the ploidy and $\mathrm{hd}(\cdot, \cdot)$ is the generalized Hamming distance between its arguments.

The above minimization can be rewritten using a select variable vector $\mathbf{s}$. The definition of the select variables is somewhat modified in the polyploid case: the i^{th} component of $\mathbf{s}$, s_i, is the index of the haplotype that the read is nearest to in terms of the generalized Hamming distance. This means that

[2]It was remarked in [12] that the MEC objective may not be suited for use in the polyploid case due to ambiguity when phasing with reads that cover the same collection of SNP sites. However, when the reads are long and cover diverse subsets of SNP sites, as is the case with recent high-throughput technologies such as fosmid, the MEC objective facilitates successful assembly as indicated by the results presented in Section 7.

the optimization of the MEC objective can be rewritten as

$$\min_{\mathbf{s}\in\{0,\ldots,k-1\}^m} \sum_{i=1}^{m}\sum_{j=0}^{k-1} \mathbf{1}_{\{s_i==j\}} \mathrm{hd}(\mathbf{r}_i, \mathbf{h}_j). \tag{4.8}$$

The graphical model that describes the data used for polyploid assembly is illustrated in Fig. 4.6. Since the read select variables (i.e., components of $\mathbf{s}$) no longer take the binary values $\{0, 1\}$, the linear coding analogy does not extend to this scenario. However, relying on the graph in Fig. 4.6, we can still design a belief propagation algorithm for polyploid haplotype assembly. Note that the select function nodes of the graph shown in Fig. 4.6 act as a simple multiplexer, allowing only one of the entries $\{\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_k\}$ to "pass" depending on the value of the select variable $\mathbf{s}$.

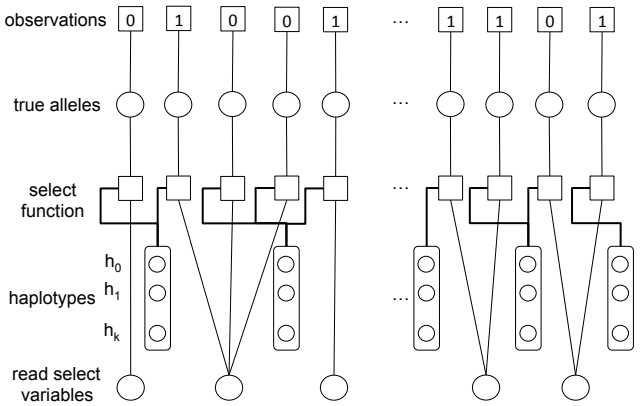

Figure 4.6: An illustration of the graphical model used for polyploid haplotype assembly. This model is an extension of the model in Fig. 4.3. The modified belief propagation algorithm is implemented on this graph.

4.6.1 The polyploid belief propagation algorithm

To extend the belief propagation algorithm from Section 4 to the assembly of haplotypes in polyploids, we need to redefine the nodes that participate in the exchange of the beliefs as well as specify messages that are being exchanged. For each SNP location, we need to infer the probabilities of the alleles for each of the k haplotypes. To this end, for the i^{th} component of $\mathbf{h}$ we define a variable $p_{\mathbf{h}}^{(i)}$ as the probability vector with k entries. The value of the j^{th} entry in this vector, $p_{\mathbf{h},j}^{(i)}$, is the probability of the j^{th} haplotype having a reference allele at the i^{th} position. Similarly, for the i^{th} component of $\mathbf{s}$ we define a probability vector $p_{\mathbf{s}}^{(i)}$ of length k, where the j^{th} component of $p_{\mathbf{s}}^{(i)}$, $p_{\mathbf{s},j}^{(i)}$, is the beliefs that the i^{th} read is associated with the j^{th} haplotype ($1 \leq j \leq k$).

Note a major distinction between the probability vectors $p_{\mathbf{h}}^{(i)}$ and $p_{\mathbf{s}}^{(i)}$. On one hand, an allele may occur at a particular position in multiple haplotype strands, and the information about an allele in one haplotype may not help infer alleles at the same position in other haplotypes. On the other hand, each read should eventually be associated with a single haplotype strand, thus helping uniquely determine multiple SNP positions in the haplotype. This difference between the probability vectors motivates different ways of aggregating messages at the graph nodes. Since SNP variables act as factors in the belief propagation factor graphs, imposing the presence of an allele in a haplotype, messages at the corresponding nodes are computed via sum aggregation. The read select variables are the ones for which we want to find the most likely configuration via marginalization, leading us to the product form.

The auxiliary variable $y_{i,j}$ has the same definition as in the belief propagation algorithm for diploids (see eq. (4.5)). However, we modify the notation $\bar{}$ to denote the belief complement of a vector variable; in particular, for any probability vector $\mathbf{x}$ of size k, $\bar{\mathbf{x}} = (1 - \mathbf{x})/(k - 1)$.

The starting point of the algorithm is restricted to a read, and a separate starting point is chosen for each strand of the haplotype. A different read will be uniquely associated with each one of the haplotype strands, as seen in Step 22 of the polyploid belief propagation algorithm.

The algorithm is formalized as Algorithm 6.

4.7 Simulation Results and Discussion

We test the performance of the proposed algorithms on both real and simulated data sets. The experimental data include the *1000 Genomes Project* [76] and *Fosmid* [28] data sets, on which we compare the MEC scores and runtimes of bit-flipping and belief propagation algorithms with those of HapCUT [9], HapCompass [3] and RefHap [26]. RefHap is part of the Single Individual Haplotyping (SIH) package that includes DGS [54], FastHare [66], SHRThree [18], Speedhap [33], 2d-MEC [89], and WMLF [45]. These algorithms were compared against each other as well as against HapCUT [9] in [27]. We also tested all of the aforementioned algorithms from the SIH package on the Fosmid dataset and found that RefHap outperforms others in terms of both speed and accuracy. On another note, simulated data mimics long (fosmid-like) pair-end reads with varying coverages, allowing comprehensive

Algorithm 6 Polyploid Belief Propagation Haplotype Assembly

procedure PBP
  $sp \leftarrow randomStartingPoint$
  $order \leftarrow 1$
  $r_i \leftarrow sp$
  $A_R^{(0)} \leftarrow r_i$ (set active reads)
  $b_{r_i}^{(0)} \leftarrow \begin{bmatrix} 1-p_e & \frac{p_e}{k-1} & \dots & \frac{p_e}{k-1} \end{bmatrix}$ (set initial beliefs)
  $m_{r_i \to h_j}^{(0)} \leftarrow \begin{bmatrix} 1-p_e & \frac{p_e}{k-1} & \dots & \frac{p_e}{k-1} \end{bmatrix}$ (set initial messages)
  $t \leftarrow 1$ iterations
  repeat
    $A_H^{(t)} \leftarrow \Gamma(A_R^{(t-1)})$ update active haplotype
    if $order = k$ **then**
      $bias_j \leftarrow$ vector with h_j degree on largest beliefs locations
    else
      $bias_j \leftarrow [0 \;\; 0 \;\; \dots \;\; 0]$
    end if
    $m_{h_j \to r_i}^{(t)} \leftarrow \frac{1}{Zh_{i,j}} \left[bias_j + \sum_{r_l \in A_R^{(t)} \setminus r_i} \left(m_{r_l \to h_j}^{(t)} y_{l,j} + \bar{m}_{(r_l \to h_j)}^{(t)} \bar{y}_{l,j} \right) \right]$
    $b_{h_j}^{(t)} \leftarrow \frac{1}{Zh_{b_j}} b_{h_j}^{(t-1)} \left[bias_j + \sum_{r_i \in A_R^{(t)}} \left(m_{r_i \to h_j}^{(t)} y_{i,j} + \bar{m}_{r_i \to h_j}^{(t)} \bar{y}_{i,j} \right) \right]$
    $A_R^{(t)} \leftarrow \Gamma(A_H^{(t-1)})$
    $b_{r_i}^{(t)} \leftarrow \frac{1}{Zr_{b_i}} \prod_{h_j \in A_H^{(t-1)}} \left(m_{h_j \to r_i}^{(t-1)} y_{i,j} + \bar{m}_{h_j \to r_i}^{(t-1)} \bar{y}_{i,j} \right)$
    $m_{r_i \to h_j}^{(t)} \leftarrow \frac{1}{Zr_{i,j}} \prod_{h_l \in A_H^{(t-1)} \setminus h_j} \left(m_{h_l \to r_i}^{(t-1)} y_{i,l} + \bar{m}_{h_l \to r_i}^{(t-1)} \bar{y}_{i,l} \right)$
    if $order < k$ **then**
      $r_c \leftarrow$ read with lowest margin of belief
      $b_{r_c}^{(t)} \leftarrow \begin{bmatrix} \frac{p_e}{k-1} & \dots & 1-pe & \dots & \frac{p_e}{k-1} \end{bmatrix}$
      $m_{r_c \to h_j}^{(t)} \leftarrow \begin{bmatrix} \frac{p_e}{k-1} & \dots & 1-pe & \dots & \frac{p_e}{k-1} \end{bmatrix}$ update messages
      $order \leftarrow order + 1$
    end if
    $t \leftarrow t + 1$
  until $(t >$ MAXITER$) \vee (\|\mathbf{b}_\mathbf{h}^{(t)} - \mathbf{b}_\mathbf{h}^{(t-1)}\|_2 \leq \epsilon)$
end procedure

study of the algorithms' performance in terms of switch error rates (SWER) which we can compute due to availability of the ground truth. The algorithms were run on MacBook Air with 2.13GHz Intel Core Duo processor with 4 GB of DDR3 RAM and MacBook with 2.4GHz Intel Core i5 processor with 4 GB of DDR3 RAM. Implementation of the algorithms is available for download from http://users.ece.utexas.edu/~hvikalo/DecodingBFBP.html and http://sourceforge.net/projects/bfbp.

4.7.1 Benchmarking on 1000 Genomes Project data

For the first test of the proposed algorithms we relied on a data set from the *1000 Genomes Project* – in particular, the sample NA12787 sequenced at high coverage using 454 sequencing platform, also considered in several recent studies including [3]. We compared the MEC scores achieved by our algorithms to those of the widely used HapCUT [9] and the more recent RefHap [26] and HapCompass [2]. HapCompass, RefHap and HapCUT were called with their default settings; for HapCUT that means using software version v.05 and running standard 100 iterations, for RefHap it means version of SIH package 1.0, while for HapCompass we ran software version 0.7.1. HapCompass processed BAM and VCF files downloaded from the 1000 Genomes Project data repository and its resulting fragment matrix was partitioned into connected components. Each component was separately fed into HapCUT along with the corresponding quality scores, ensuring that both algorithms take into account the same SNP locations and provide a fair comparison of the MEC scores (i.e.,

the number of SNPs, reads, alleles and blocks was the same for all the algorithms). The resulting MEC scores are reported in the second super-column of Table 4.2. As can be seen there, among all algorithms belief propagation provides the smallest MEC score for all chromosomes. Bit-flipping (BF) is slightly worse than belief propagation (BP), matching its performance on 4 chromosomes, but better than HapCUT on all and better than HapCompass and RefHap on all but one chromosome. Note that RefHap does not phase all the SNPs and discards a fraction of reads which leads to an unrealistic MEC score that is artificially lower than those achieved by the other algorithms. To make a fair comparison, we assigned the reads discarded by RefHap to either $\mathbf{h}$ or $\bar{\mathbf{h}}$ using the phased SNPs and then phased the remaining SNPs in such a way that the resulting total MEC score of RefHap is the lowest possible. The computational overhead due to these additional operations is not included in the reported RefHap runtimes; for completeness, the original RefHap MEC scores evaluated on a reduced set of reads are reported in parenthesis.

When comparing the speed of the algorithms, we wanted to exclude the bias due to the system calls for I/O operations; therefore, in Table 4.3 we report the user time provided by the UNIX time command. For HapCUT, the total time including the system calls, operations and context switches is around an order of magnitude greater than the times reported in Table 4.3. For HapCompass, the total time in only marginally greater than the user time given in the table. The BP and BF algorithms also have the total time only marginally greater than the user time. As evident from the table, RefHap is

Table 4.2: MEC scores HapCUT (HCUT), HapCompass (HCom), bit-flipping (BF) and belief propagation (BP) algorithms for the *1000 Genomes Project* individual NA12787.

	Properties				MEC score				
Chr.	#SNP	#Read	#Allele	#blocks	HCUT	HCom.	RefHap	BF	BP
1	122960	180199	403138	21825	12675	12390	12380 (11203)	12312	**12310**
2	139475	211311	475770	24836	15363	15015	14991 (13650)	14933	**14916**
3	117657	180572	407084	20855	13243	12970	12947 (11777)	12882	**12872**
4	119330	190029	437636	20802	14980	14642	14631 (13361)	14536	**14532**
5	112643	171881	387704	20049	12611	12266	12260 (11070)	12200	**12196**
6	116414	189932	463272	19579	17057	16805	16800 (15474)	16924	**16769**
7	94511	148305	340748	16624	11445	11174	11165 (10098)	11110	**11108**
8	94024	152864	34966	16571	11088	10817	10797 (9750)	10735	**10732**
9	71898	115722	263419	12979	8481	8319	8279 (7503)	8237	**8236**
10	85499	136288	310879	15001	10167	9899	9893 (9030)	9833	**9828**
11	81018	126027	288307	14225	9308	9104	9111 (8273)	9047	**9042**
12	78146	117673	265958	13849	8659	8418	8400 (7651)	8362	**8361**
13	63689	100081	230321	11241	7848	7695	7687 (7041)	7652	**7631**
14	53934	82139	185435	9598	5866	5715	5698 (5154)	5662	**5661**
15	46254	73559	166860	8191	5244	5137	5120 (4660)	5102	**5100**
16	51786	86684	201865	9043	6400	6231	6205 (5708)	6177	**6175**
17	38839	58363	134103	6696	4631	4550	4537 (4201)	**4506**	**4506**
18	49873	77291	175107	8821	5531	5371	5363 (4868)	5342	**5340**
19	31760	46397	105525	5602	3571	3421	3403 (3086)	**3392**	**3392**
20	38044	58879	134089	6831	4213	4122	4103 (3754)	4087	**4086**
21	24342	40107	92582	4379	3021	2987	2972 (2695)	**2953**	**2953**
22	22801	33671	77122	4076	2395	2324	2316 (2112)	**2301**	**2301**

Table 4.3: Execution times for HapCUT (HCUT), HapCompass (HCom), bit-flipping (BF) and belief propagation (BP) algorithms for the *1000 Genomes Project* individual NA12787.

	Properties				Execution time (s)				
Chr.	#SNP	#Read	#Allele	#blocks	HCUT	HCom.	RefHap	BF	BP
1	122960	180199	403138	21825	115	214	28	59	39
2	139475	211311	475770	24836	134	243	61	117	99
3	117657	180572	407084	20855	113	203	37	70	60
4	119330	190029	437636	20802	123	202	45	116	89
5	112643	171881	387704	20049	122	167	30	62	48
6	116414	189932	463272	19579	202	16290	987	1193	1073
7	94511	148305	340748	16624	97	164	26	96	51
8	94024	152864	34966	16571	97	133	21	60	60
9	71898	115722	263419	12979	72	149	32	140	43
10	85499	136288	310879	15001	86	122	21	61	47
11	81018	126027	288307	14225	83	131	25	64	68
12	78146	117673	265958	13849	76	131	17	40	32
13	63689	100081	230321	11241	64	108	40	80	67
14	53934	82139	185435	9598	51	84	11	33	29
15	46254	73559	166860	8191	45	77	9	25	18
16	51786	86684	201865	9043	54	83	15	36	51
17	38839	58363	134103	6696	37	84	17	69	20
18	49873	77291	175107	8821	47	75	8	34	23
19	31760	46397	105525	5602	29	54	10	15	11
20	38044	58879	134089	6831	38	61	7	19	14
21	24342	40107	92582	4379	26	40	6	14	13
22	22801	33671	77122	4076	24	37	5	11	7

the fastest among the considered schemes but its speed comes at the cost of reduced accuracy. On the other hand, the BP and BF algorithms are faster than HapCUT and HapCompass except for chromosome 6 where HapCUT incurs the least amount of runtime.

4.7.2 Fosmid data

Fosmid pool-based sequencing provides very long fragments, characterized by much higher ratio of the number of SNPs to the number of reads than in standard high-throughput sequencing platforms. We consider the fosmid sequence data for a HapMap NA12878, also studied in [28]. As an example, chromosome 1 of this dataset has $22,737$ reads and $122,960$ SNPs.

We compared the performance of our bit-flipping (BF) and belief propagation (BP) algorithms to that of HapCUT and RefHap and report the results in Table 4.4 (we attempted running HapCompass on the same dataset but that algorithm was running out of memory or stopping for unknown reasons). As can be seen from Table 4.4, BF or BP outperform HapCUT on half of the chromosomes; the MEC scores of all three algorithms are within 1% margin for all chromosomes.

In Table 4.5 we see the execution time of tested algorithms as reported user time in UNIX time command. Both bit-flipping and belief propagation are significantly faster than HapCUT – the widest gap in speed is seen on chromosome 6 where belief propagation is about 40 times faster than HapCUT. RefHap is again the fastest among the considered schemes but its speed is

Table 4.4: MEC scores for HapCUT, bit-flipping and belief propagation algorithms on the Fosmid dataset for NA12878 individual.

	Properties				MEC score			
Chr.	#SNP	#Read	#Allele	#blocks	HapCUT	RefHap	BF	BP
1	122960	22736	393201	1316	9555	9644 (8051)	9589	**9552**
2	129732	22602	413205	1519	9668	9728 (7910)	9734	**9661**
3	108204	18722	330763	1285	7566	7635 (6111)	7606	**7554**
4	107430	16027	306639	1372	6267	6317 (4880)	**6262**	6267
5	103442	17013	317206	1196	**6922**	6963 (5558)	6960	6946
6	107882	16451	332388	1072	**7957**	8038 (6341)	8002	7965
7	87563	14936	275308	1011	**6071**	6098 (4961)	6107	6078
8	86708	13733	275302	959	6260	6289 (5092)	6282	**6250**
9	66996	11528	224982	678	5464	5505 (4591)	**5460**	5462
10	79978	14064	265142	779	**6446**	6489 (5357)	6475	6475
11	75235	13519	246499	802	**5560**	5602 (4620)	5575	5567
12	72917	13377	238070	793	**5666**	5691 (4686)	5692	5670
13	57287	8800	168625	671	3968	4030 (3155)	**3961**	3967
14	50219	9030	165775	523	**3979**	4017 (3244)	4016	3989
15	43578	8306	149536	481	4009	4053 (3341)	4030	**4003**
16	49736	9655	191480	400	**5087**	5102 (4438)	5128	5099
17	37820	8776	146019	426	4744	4819 (4159)	4773	**4740**
18	46313	7704	146353	497	3448	3473 (2801)	3475	**3441**
19	30777	7431	119629	266	**3900**	3940 (3406)	4037	3902
20	36398	7447	135745	317	3811	3863 (3295)	3883	**3810**
21	22756	3760	73711	222	**1953**	1967 (1601)	1958	**1953**
22	22083	5567	92889	141	**3261**	3343 (2876)	3375	3278

Table 4.5: Execution time for HapCUT, bit-flipping and belief propagation algorithms on the Fosmid dataset for NA12878 individual.

	Properties				Execution time (s)			
Chr.	#SNP	#Read	#Allele	#blocks	HapCUT	RefHap	BF	BP
1	122960	22736	393201	1316	2423	2.3	319	268
2	129732	22602	413205	1519	3089	2.22	254	282
3	108204	18722	330763	1285	2124	2.01	220	213
4	107430	16027	306639	1372	2431	1.81	159	234
5	103442	17013	317206	1196	2202	1.91	238	213
6	107882	16451	332388	1072	11260	2.02	286	283
7	87563	14936	275308	1011	2242	1.79	274	208
8	86708	13733	275302	959	2670	1.78	332	158
9	66996	11528	224982	678	2309	1.56	286	163
10	79978	14064	265142	779	2006	1.62	180	199
11	75235	13519	246499	802	1752	1.64	177	182
12	72917	13377	238070	793	1610	1.71	183	189
13	57287	8800	168625	671	1276	1.28	108	92
14	50219	9030	165775	523	1302	1.26	168	186
15	43578	8306	149536	481	1083	1.17	120	150
16	49736	9655	191480	400	2481	1.58	220	260
17	37820	8776	146019	426	1069	1.21	151	191
18	46313	7704	146353	497	1021	1.17	83	88
19	30777	7431	119629	266	773	1.07	137	136
20	36398	7447	135745	317	951	1.2	164	115
21	22756	3760	73711	222	587	0.72	48	33
22	22083	5567	92889	141	752	1.24	218	188

traded off for accuracy. As before, the reported runtimes exclude the system calls time.

4.7.3 Simulation results: the diploid case

We simulate two scenarios, one with high coverage paired-end reads that resemble those in *1000 Genomes Project* datasets and the other with long reads and low coverage similar to what may be available in a *Fosmid* dataset.

To emulate the short-read high-coverage scenario, we generated reads that span 500 basis with inserts having 10k mean length and 10% deviation. The rate of SNPs is assumed to be 1 in 300 basis as reported in [34]. We simulate sampling of the entire genome with the paired-end reads, marking each base as a SNP location with probability 1 in 300. This means that the number of basis between two neighboring SNP locations is a geometrically distributed random variable (as assumed in, e.g., [12]).

We study the dependence of the switch error rate (SWER) on coverage. Moreover, we are interested in understanding how haplotype assembly depends upon the errors in the SNP fragment matrix. To this end, we consider haplotype block lengths of 1000 and 5000, with the coverage of 10, 15, 20, 25 and 30. Fore each pair of parameters (block length and coverage), we simulate error rates of 1%, 2% and 5% in the SNP fragment matrix. The error rate of 2% is the closest to what we observed in experimental data sets (both 1000 Genomes Project and Fosmid). Each experiment is repeated 10 times. The mean values of SWER for the BP algorithm are reported in Fig. 4.7. As can be seen from the figure, increase in the block length leads to the increase in the SWER value for the same value of the erroneous data rate and coverage. Increasing the fraction of errors in the SNP fragment matrix causes deterioration of the SWER performance, while increasing the coverage improves the SWER. Note that even in the worst considered scenario (highest rate of sequencing errors, largest block size, and smallest coverage), SWER remains below 2%. For a comparison, we include the SWER of RefHap and HapCUT for block

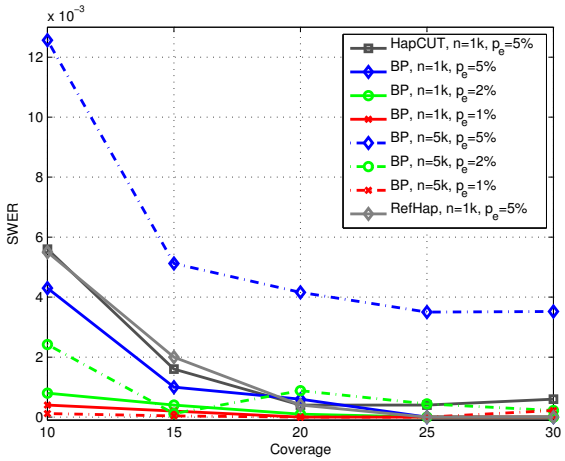

Figure 4.7: Simulated SWER rates for haplotype assembly with short reads using the BP algorithm (diploid). Results are averaged over 10 different fragment matrices, and reported with respect to varying coverage, block length and probability of error.

lengths $n = 1000$ and error rate 5%. Except for the coverage $c = 20$, the BP algorithms leads to lowest SWERs for the considered set of parameters.

Next, we consider the long reads, small coverage scenario (similar to the Fosmid data). The reads are generated with lengths distributed following Poisson distribution with specified average read length. Within each read, a single SNP is covered independently with probability 0.9. We study the performance of our algorithm in terms of SWER. The parameters of the simulation are the number of SNPs, the sequencing error rate, and the average read length. The number of SNPs (i.e., haplotype block length) is 5,000 and 25,000. The error rate in the SNP fragment matrix is set to 0.5%, 1% and 2%. The average read length was set to 40, 80, and 120. The coverage was set to 8. Each experiment

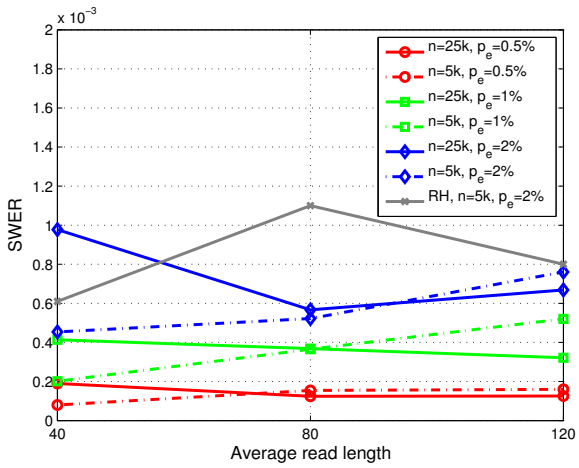

Figure 4.8: Simulated SWER rates for haplotype assembly with long reads using the BP algorithm (diploid). Results are averaged over 10 different fragment matrices, and reported with respect to varying read length, block length and probability of error.

was repeated 10 times and the mean value and standard deviation of SWER are shown in the Fig. 4.8. We again include the comparison with RefHap for block lengths $n = 5000$ and error rate 2%. As can be seen from the figure, the BP algorithms provides better SWERs for all coverage levels.

We see from Fig. 4.8 that, as expected, higher data error rates lead to higher SWER. Interestingly, the blocks of length 5,000 seem to have similar SWER to those for the blocks of length 25,000. Moreover, long reads appear to lead to SWER similar to those provided by short reads. In order to gain further insight, we examined the cause of the switches. As it turns out, majority of the switches are actually single SNPs that got inverted, implying that most of the reported errors in haplotype assembly are due to errors in the isolated

cuts (discussed in Section 5).

4.7.4 Simulation results: the polyploid case

We implemented our belief propagation algorithm for haplotype assembly of polyploids in C, and compared its performance in terms of MEC, SWER and execution time to the polyploid version of the HapCompass algorithm [3]. [We attempted to compare its performance to HapTree [12] as well, however, for the considered block sizes HapTree runs out of memory.] We simulated pair-end reads with the same setup as in the diploid simulations, for haplotype block lengths 200 and 1000. The coverage was varied and the error rate in the SNP fragment matrix was 1% and 2%. The plotted lines are obtained by averaging results of 10 simulation runs.

The MEC scores obtained by applying the algorithms to the assembly of a triploid are given in Fig. 4.9. As can be seen there, the BP algorithm achieves significantly lower MEC scores than HapCompass. As the data error rates increase, the MEC scores increase by approximately the same factor.

Next, we study the switch error rate (SWER) and compare the performance of the algorithms for various polyploid orders. The results are shown in Fig. 4.10. As we can see from the figure, increasing the coverage reduces the SWER while as the ploidy increases the SWER deteriorates.

Finally, we study running times of the belief propagation algorithm for haplotype assembly of polyploids and report them in Fig. 4.11. As can be seen there, the run time increases with coverage, ploidy and block size, while

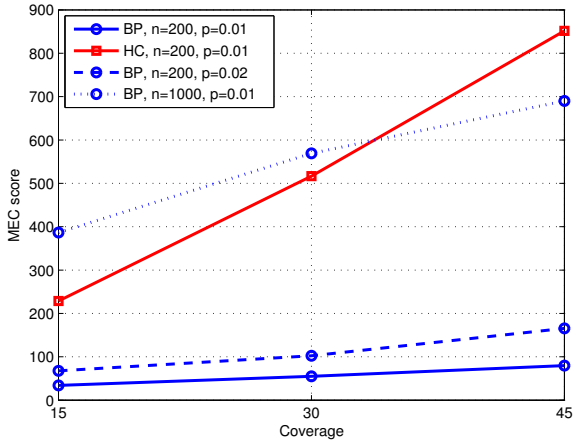

Figure 4.9: Simulated MEC scores for haplotype assembly of a triploid (the belief propagation algorithm and HapCompass).

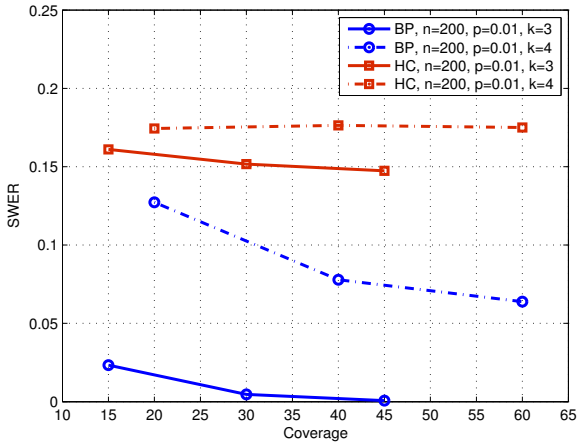

Figure 4.10: Simulated SWER rates for haplotype assembly of various polyploids (the belief propagation algorithm and HapCompass). Haplotype block length is set to 200.

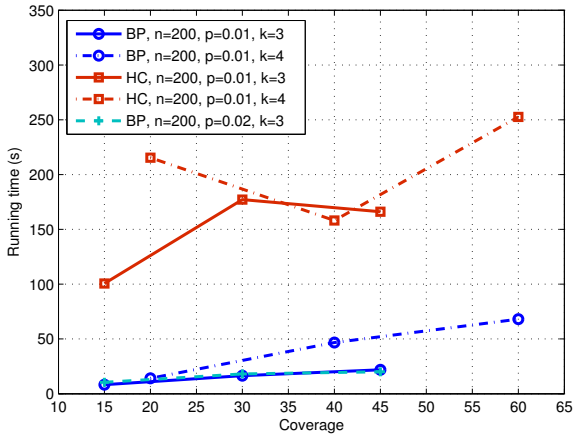

Figure 4.11: Comparison of running times for haplotype assembly of various polyploids (the belief propagation algorithm and HapCompass). Haplotype block length is set to 200.

it appears independent of data error rates. Moreover, the belief propagation algorithm is significantly faster than HapCompass for the same sets of simulation parameters. Note that the reported run times are obtained using the UNIX time command (i.e., we are reporting the user time).

4.8 Conclusion and Future work

We proposed and studied formulation of the haplotype assembly problem that relies on a novel graphical representation and draws upon parallels between haplotype assembly and decoding in data communication systems. We proposed two algorithms, namely the bit-flipping and belief propagation algorithm, both highly accurate and fast heuristics. The complexity of both algorithms is only linear in the length of the haplotype block. Their accu-

racy compares favorably with HapCUT, HapCompass and RefHap on both simulated and experimental data. When applied to fosmid data characterized by long fragments and small ratio between the number of reads and haplotype length, our proposed methods are often more than 10 times faster than HapCUT. Moreover, we extended the belief propagation algorithm to the haplotype assembly of polyploids, focusing on the bi-allelic case, and demonstrated significant performance improvements over HapCompass.

As part of the future work, it is of interest to explore other, more sophisticated, decoding algorithms in the context of haplotype assembly (e.g., belief propagation with soft thresholding). There is also a number of potentially interesting fundamental questions such as performance vs. complexity tradeoffs and further analysis of the achievable limits of performance.

Chapter 5

Conclusion and Future Work

In this thesis we explore the recovery of the high-dimensional signal from partial observations. We look into this problem from theoretical viewpoint and define the necessary and sufficient conditions of resynchronization of time varying systems with linear observations in case of both full and partial view of the system. However, the recovery procedure in general is exponential which makes it inefficient for real systems. As a result, we give a heuristic and extensively simulate its performance on time varying system.

As part of the future work, it would be interesting to expanding the existing framework with theoretical guarantees on the performance of the designed heuristics. In addition, still open problem in this framework is a design of measurement schedule with multiple observers that would lead to the most efficient signal recovery. Both of these would address in more detail the tradeoffs between the complexity of recovery procedure with number of measurements required and guarantees of the validity of the solution.

Next, we introduce two applications of high-dimensional signal recovery from partial observations. First one is related to massive MIMO wireless system where the goal is to reduce the channel contamination. The second

application is a biomedical application with focus on recovery of the genetic variation for both diploid and polyploid species from incomplete observations that is known as haplotype assembly problem.

The pilot contamination problem exists in massive MIMO systems. In this system the performance bottleneck is the ability to recovery the channel coefficients from sent pilots. By exploiting the time correlation and using non-orthogonal pilot sequences we can reduce the pilot contamination, as well as allow packing of more users in the system. This is achievable without any cooperation between the base stations, and with minimal user management.

In the future it would be useful to see if base station collaboration can lead to better performance. In addition, it would be useful to explore how to pack higher number of users in the system, while keeping pilot contamination under a threshold.

Haplotype assembly problem is a problem of recovering the genetic variation from partial and erroneous observations. We give two algorithms, namely the bit-flipping and belief propagation algorithm, both highly accurate and fast heuristics. The complexity of both algorithms is only linear in the length of the haplotype block. We expand the belief propagation algorithm for polyploid case. All of the algorithms perform favorably when compared to the state of the art algorithms on both simulated and real data sets.

As part of the future work in the area of haplotype assembly, it is of interest to explore other, more sophisticated, decoding algorithms in the

context of haplotype assembly (e.g., belief propagation with soft thresholding). There is also a number of potentially interesting fundamental questions such as performance vs. complexity tradeoffs and further analysis of the achievable limits of performance.

Bibliography

[1] Shuchin Aeron, Manqi Zhao, and Venkatesh Saligrama. On sensing capacity of sensor networks for a class of linear observation models. In *Statistical Signal Processing, 2007. SSP'07. IEEE/SP 14th Workshop on*, pages 388–392. IEEE, 2007. 10

[2] Derek Aguiar and Sorin Istrail. Hapcompass: a fast cycle basis algorithm for accurate haplotype assembly of sequence data. *Journal of Computational Biology*, 19(6):577–590, 2012. 99

[3] Derek Aguiar and Sorin Istrail. Haplotype assembly in polyploid genomes and identical by descent shared tracts. *Bioinformatics*, 29(13):i352–i360, 2013. 71, 97, 99, 109

[4] D. Angelosante, G.B. Giannakis, and E. Grossi. Compressed sensing of time-varying signals. In *Digital Signal Processing, 2009 16th International Conference on*, pages 1 –8, july 2009. 10

[5] Kumar Appaiah, Alexei Ashikhmin, and Thomas L Marzetta. Pilot contamination reduction in multi-user tdd systems. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5. IEEE, 2010. 42

[6] A. Ashikhmin and T. Marzetta. Pilot contamination precoding in multi-cell large scale antenna systems. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 1137–1141, 2012. 42

[7] M.S. Asif and J. Romberg. Dynamic updating for sparse time varying signals. In *Information Sciences and Systems, 2009. CISS 2009. 43rd Annual Conference on*, pages 3 –8, march 2009. 10

[8] Waheed Bajwa, Jarvis Haupt, Akbar Sayeed, and Robert Nowak. Compressive wireless sensing. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 134–142. ACM, 2006. 10

[9] Vikas Bansal and Vineet Bafna. Hapcut: an efficient and accurate algorithm for the haplotype assembly problem. *Bioinformatics*, 24(16):i153–i159, 2008. 71, 97, 99

[10] R.G. Baraniuk. Compressive sensing [lecture notes]. *Signal Processing Magazine, IEEE*, 24(4):118 –121, july 2007. 9

[11] D. Baron, M.B. Wakin, M.F. Duarte, S. Sarvotham, and R.G. Baraniuk. Distributed compressed sensing, 2005. 10

[12] Emily Berger, Deniz Yorukoglu, Jian Peng, and Bonnie Berger. Haptree: A novel bayesian framework for single individual polyplotyping using ngs data. *PLoS Comput Biol*, 10(3):e1003502, 03 2014. 71, 94, 106, 109

[13] E.J. Candès. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9):589–592, 2008. 24

[14] E.J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489 – 509, feb. 2006. 9

[15] E.J. Candes and M.B. Wakin. An introduction to compressive sampling. *Signal Processing Magazine, IEEE*, 25(2):21 –30, march 2008. 9

[16] Emmanuel Candes and Terence Tao. The dantzig selector: Statistical estimation when p is much larger than n. *The Annals of Statistics*, 35(6):2313–2351, 2007. 9

[17] Zhixiang Chen, Bin Fu, Robert Schweller, Boting Yang, Zhiyu Zhao, and Binhai Zhu. Linear time probabilistic algorithms for the singular haplotype reconstruction problem from snp fragments. *Journal of Computational Biology*, 15(5):535–546, 2008. 71

[18] Zhixiang Chen, Bin Fu, Robert Schweller, Boting Yang, Zhiyu Zhao, and Binhai Zhu. Linear time probabilistic algorithms for the singular haplotype reconstruction problem from snp fragments. *Journal of Computational Biology*, 15(5):535–546, 2008. 97

[19] Albert Cohen, Wolfgang Dahmen, and Ronald DeVore. Compressed sensing and best k-term approximation. *J. Amer. Math. Soc*, 22(1):211–231, 2009. 9

[20] Francis S Collins, Michael Morgan, and Aristides Patrinos. The human genome project: lessons from large-scale biology. *Science*, 300(5617):286–290, 2003. 69

[21] Thomas M. Cover and Joy Thomas. *Elements of Information Theory*. Wiley, 1991. 71

[22] David L. Donoho and Jared Tanner. Sparse nonnegative solution of underdetermined linear equations by linear programming. *Proceedings of the National Academy of Sciences*, 102(27):9446–9451, 2005. 9

[23] D.L. Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289 –1306, april 2006. 9

[24] D.L. Donoho and J. Tanner. Thresholds for the recovery of sparse solutions via l1 minimization. In *Information Sciences and Systems, 2006 40th Annual Conference on*, pages 202 –206, march 2006. 30

[25] D.L. Donoho and J. Tanner. Precise undersampling theorems. *Proceedings of the IEEE*, 98(6):913–924, June. 30

[26] Jorge Duitama, Thomas Huebsch, Gayle McEwen, Eun-Kyung Suk, and Margret R Hoehe. Refhap: a reliable and fast algorithm for single individual haplotyping. In *Proceedings of the First ACM International*

Conference on Bioinformatics and Computational Biology, pages 160–169. ACM, 2010. 97, 99

[27] Jorge Duitama, Gayle K McEwen, Thomas Huebsch, Stefanie Palczewski, Sabrina Schulz, Kevin Verstrepen, Eun-Kyung Suk, and Margret R Hoehe. Fosmid-based whole genome haplotyping of a hapmap trio child: evaluation of single individual haplotyping techniques. *Nucleic acids research*, page gkr1042, 2011. 97

[28] Jorge Duitama, Gayle K McEwen, Thomas Huebsch, Stefanie Palczewski, Sabrina Schulz, Kevin Verstrepen, Eun-Kyung Suk, and Margret R Hoehe. Fosmid-based whole genome haplotyping of a hapmap trio child: evaluation of single individual haplotyping techniques. *Nucleic acids research*, 40(5):2041–2053, 2012. 71, 97, 103

[29] YC Eldar and G. Kutyniok. Compressed sensing: Theory and applications. *New York: Cambridge Univ. Press*, 20:12, 2012. 9

[30] Marvin E Frazier, Gary M Johnson, David G Thomassen, Carl E Oliver, and Aristides Patrinos. Realizing the potential of the genome revolution: the genomes to life program. *Science*, 300(5617):290–293, 2003. 69

[31] Robert G. Gallager. Low-density parity-check codes. *Research Monograph Series*, 1963. 71, 85

[32] Harish Ganapathy, Constantine Caramanis, and Lei Ying. Exploiting sparse dynamics for bandwidth reduction in cooperative sensing systems.

arXiv preprint arXiv:1210.7543, 2012. 10

[33] Loredana M Genovese, Filippo Geraci, and Marco Pellegrini. 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 (TCBB)*, 5(4):492–502, 2008. 97

[34] Richard A Gibbs, John W Belmont, Paul Hardenbol, Thomas D Willis, Fuli Yu, Huanming Yang, Lan-Yang Ch'ang, Wei Huang, Bin Liu, Yan Shen, et al. The international hapmap project. *Nature*, 426(6968):789–796, 2003. 106

[35] Balasubramanian Gopalakrishnan and Nihar Jindal. An analysis of pilot contamination on multi-user mimo cellular systems with many antennas. In *Signal Processing Advances in Wireless Communications (SPAWC), 2011 IEEE 12th International Workshop on*, pages 381–385. IEEE, 2011. 42

[36] Neil Hall. Advanced sequencing technologies and their wider impact in microbiology. *Journal of Experimental Biology*, 210(9):1518–1525, 2007. 69

[37] P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, s1-10(1):26–30, 1935. 21

[38] Babak Hassibi and Bertrand M Hochwald. How much training is needed in multiple-antenna wireless links? *Information Theory, IEEE Transactions on*, 49(4):951–963, 2003. 49

[39] Jakob Hoydis, Stephan Ten Brink, and Mérouane Debbah. Massive mimo: How many antennas do we need? In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, pages 545–550. IEEE, 2011. 42

[40] Hoon Huh, Giuseppe Caire, Haralabos C Papadopoulos, and Sean A Ramprashad. Achieving large spectral efficiency with tdd and not-so-many base-station antennas. In *Antennas and Propagation in Wireless Communications (APWC), 2011 IEEE-APS Topical Conference on*, pages 1346–1349. IEEE, 2011. 42

[41] J. Jose, A. Ashikhmin, T.L. Marzetta, and S. Vishwanath. Pilot contamination problem in multi-cell tdd systems. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, pages 2184–2188, 2009. 4, 41

[42] Jubin Jose, Alexei Ashikhmin, Thomas L Marzetta, and Sriram Vishwanath. Pilot contamination and precoding in multi-cell tdd systems. *Wireless Communications, IEEE Transactions on*, 10(8):2640–2651, 2011. 42, 43, 48

[43] T. Kailath, A.H. Sayed, and B. Hassibi. *Linear Estimation*. Prentice-Hall information and system sciences series. Prentice Hall, 2000. 47,

56

[44] R. E. Kalman. A new approach to linear filtering and prediction problems. 1960. 9

[45] Seung-Ho Kang, In-Seon Jeong, Hwan-Gue Cho, and Hyeong-Seok Lim. Hapassembler: A web server for haplotype assembly from snp fragments using genetic algorithm. *Biochemical and biophysical research communications*, 397(2):340–344, June 2010. 97

[46] Evripidis Karseras and Wei Dai. Tracking dynamic sparse signals: a hierarchical kalman filter. *Information theory and applications workshop, ITA*, 2013. 9, 10

[47] Jong Hyun Kim, Michael S Waterman, and Lei M Li. Accuracy assessment of diploid consensus sequences. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 4(1):88–97, 2007. 71

[48] Jong Hyun Kim, Michael S Waterman, and Lei M Li. Diploid genome reconstruction of ciona intestinalis and comparative analysis with ciona savignyi. *Genome research*, 17(7):1101–1110, 2007. 71

[49] Kiran and D.N.C. Tse. Effective interference and effective bandwidth of linear multiuser receivers in asynchronous cdma systems. *Information Theory, IEEE Transactions on*, 46(4):1426–1447, Jul 2000. 42

[50] Shrinivas Kudekar, Thomas J Richardson, and Rüdiger L Urbanke. Threshold saturation via spatial coupling: Why convolutional ldpc ensembles

perform so well over the bec. *Information Theory, IEEE Transactions on*, 57(2):803–834, 2011. 71

[51] Shrinivas Kudekar, Tom Richardson, and Rüdiger Urbanke. Spatially coupled ensembles universally achieve capacity under belief propagation. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 453–457. IEEE, 2012. 71

[52] Santhosh Kumar, Andrew J. Young, Nicolas Macris, and Henry D. Pfister. A proof of threshold saturation for spatially-coupled ldpc codes on bms channels. *CoRR*, abs/1301.6111, 2013. 71

[53] Eric S Lander, Lauren M Linton, Bruce Birren, Chad Nusbaum, Michael C Zody, Jennifer Baldwin, Keri Devon, Ken Dewar, Michael Doyle, William FitzHugh, et al. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 2001. 69

[54] Samuel Levy, Granger Sutton, Pauline C Ng, Lars Feuk, Aaron L Halpern, Brian P Walenz, Nelson Axelrod, Jiaqi Huang, Ewen F Kirkness, Gennady Denisov, et al. The diploid genome sequence of an individual human. *PLoS biology*, 5(10):e254, 2007. 71, 97

[55] Lei M Li, Jong Hyun Kim, and Michael S Waterman. Haplotype reconstruction from snp alignment. *Journal of Computational Biology*, 11(2-3):505–516, 2004. 71

[56] Sarah J Lindsay, James K Bonfield, and Matthew E Hurles. Shotgun haplotyping: a novel method for surveying allelic sequence variation. *Nucleic acids research*, 33(18):e152–e152, 2005. 71

[57] Ross Lippert, Russell Schwartz, Giuseppe Lancia, and Sorin Istrail. Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. *Briefings in bioinformatics*, 3(1):23–31, 2002. 70

[58] Elaine R Mardis. The impact of next-generation sequencing technology on genetics. *Trends in genetics*, 24(3):133–141, 2008. 69

[59] Thomas L Marzetta. How much training is required for multiuser mimo? In *Signals, Systems and Computers, 2006. ACSSC'06. Fortieth Asilomar Conference on*, pages 359–363. IEEE, 2006. 48

[60] Thomas L Marzetta. Noncooperative cellular wireless with unlimited numbers of base station antennas. *Wireless Communications, IEEE Transactions on*, 9(11):3590–3600, 2010. 42

[61] Hlaing Minn and Naofal Al-Dhahir. Optimal training signals for mimo ofdm channel estimation. *Wireless Communications, IEEE Transactions on*, 5(5):1158–1168, 2006. 62

[62] Andrea Montanari. Tight bounds for ldpc and ldgm codes under map decoding. *Information Theory, IEEE Transactions on*, 51(9):3221–3246, 2005. 71

[63] Rafael Rios Muller, Laura Cottatellucci, and Mikko Vehkapera. Blind pilot decontamination. *Selected Topics in Signal Processing, IEEE Journal of*, 8(5):773–786, 2014. 42

[64] Hien Quoc Ngo, Thomas L Marzetta, and Erik G Larsson. Analysis of the pilot contamination effect in very large multicell multiuser mimo systems for physical channel models. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 3464–3467. IEEE, 2011. 42

[65] Dapo Omidiran and Martin J. Wainwright. High-dimensional variable selection with sparse random projections: Measurement sparsity and statistical efficiency. *J. Mach. Learn. Res.*, 11:2361–2386, August 2010. 10

[66] Alessandro Panconesi and Mauro Sozio. Fast hare: A fast heuristic for single individual snp haplotype reconstruction. In *Algorithms in Bioinformatics*, pages 266–277. Springer, 2004. 97

[67] Yagyensh Chandra Pati, Ramin Rezaiifar, and PS Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44. IEEE, 1993. 55

[68] Z. Puljiz and H. Vikalo. Decoding genetic variations: Communications-inspired haplotype assembly. *Computational Biology and Bioinformatics,*

IEEE/ACM Transactions on, PP(99):1–1, 2015. 69

[69] Tom Richardson and Urbanke. Ruediger. *Modern Coding Theory*. Cambridge University Press, 2008. 71

[70] Mostafa Ronaghi, Samer Karamohamed, Bertil Pettersson, Mathias Uhlén, and Pål Nyrén. Real-time dna sequencing using detection of pyrophosphate release. *Analytical biochemistry*, 242(1):84–89, 1996. 69

[71] Stephan C Schuster. Next-generation sequencing transforms todays biology. *Nature*, 200(8), 2007. 69

[72] Russell Schwartz. Theory and algorithms for the haplotype assembly problem. *Communications in Information & Systems*, 10(1):23–38, 2010. 5

[73] Russell Schwartz. Theory and algorithms for the haplotype assembly problem. *Communications in Information & Systems*, 10(1):23–38, 2010. 70, 86

[74] B. Shahrasbi, A. Talari, and N. Rahnavard. Tc-csbp: Compressive sensing for time-correlated data based on belief propagation. In *Information Sciences and Systems (CISS), 2011 45th Annual Conference on*, pages 1 –6, march 2011. 10

[75] M. Shamaiah and H. Vikalo. Compressed sensing for bandwidth constrained systems. In *Acoustics Speech and Signal Processing (ICASSP),*

2010 IEEE International Conference on, pages 2650 –2653, march 2010. 9

[76] The 1000 Genomes Project Consortium. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061–1073, 10 2010. 97

[77] J Tropp and Anna C Gilbert. Signal recovery from partial information via orthogonal matching pursuit, 2005. 55

[78] Joel Tropp, Anna C Gilbert, et al. Signal recovery from random measurements via orthogonal matching pursuit. *Information Theory, IEEE Transactions on*, 53(12):4655–4666, 2007. 55

[79] David Tse and Pramod Viswanath. *Fundamentals of Wireless Communication*. Cambridge University Press, New York, NY, USA, 2005. 46

[80] N. Vaswani. Kalman filtered compressed sensing. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 893 –896, oct. 2008. 9

[81] N. Vaswani. Analyzing least squares and kalman filtered compressed sensing. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 3013 –3016, april 2009. 9

[82] N. Vaswani. Ls-cs-residual (ls-cs): Compressive sensing on least squares residual. *Signal Processing, IEEE Transactions on*, 58(8):4108 –4120, aug. 2010. 10

[83] N. Vaswani and Wei Lu. Modified-cs: Modifying compressive sensing for problems with partially known support. *Signal Processing, IEEE Transactions on*, 58(9):4595 –4607, sept. 2010. 10

[84] J Craig Venter, Mark D Adams, Eugene W Myers, Peter W Li, Richard J Mural, Granger G Sutton, Hamilton O Smith, Mark Yandell, Cheryl A Evans, Robert A Holt, et al. The sequence of the human genome. *science*, 291(5507):1304–1351, 2001. 69

[85] Pramod Viswanath, Venkat Anantharam, and David NC Tse. Optimal sequences, power control, and user capacity of synchronous cdma systems with linear mmse multiuser receivers. *Information Theory, IEEE Transactions on*, 45(6):1968–1983, 1999. 42

[86] M.J. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ_1 -constrained quadratic programming (lasso). *Information Theory, IEEE Transactions on*, 55(5):2183 –2202, may 2009. 9

[87] Rui-Sheng Wang, Ling-Yun Wu, Zhen-Ping Li, and Xiang-Sun Zhang. Haplotype reconstruction from snp fragments by minimum error correction. *Bioinformatics*, 21(10):2456–2462, 2005. 71

[88] Wei Wang, Martin J. Wainwright, and Kannan Ramchandran. Information-theoretic limits on sparse signal recovery: dense versus sparse measurement matrices. *IEEE Trans. Inf. Theor.*, 56(6):2967–2979, June 2010. 10

[89] Ying Wang, Enmin Feng, and Ruisheng Wang. A clustering algorithm based on two distance functions for mec model. *Computational biology and chemistry*, 31(2):148–150, 2007. 97

[90] Haifan Yin, David Gesbert, Miltiades C Filippou, and Yingzhuang Liu. Decontaminating pilots in massive mimo systems. In *Communications (ICC), 2013 IEEE International Conference on*, pages 3170–3175. IEEE, 2013. 42

[91] Zhilin Zhang and Bhaskar D Rao. Exploiting correlation in sparse signal recovery problems: Multiple measurement vectors, block sparsity, and time-varying sparsity. Technical Report arXiv:1105.0725, May 2011. Comments: Extended abstract for ICML 2011 Structured Sparsity: Learning and Inference Workshop. 10

[92] J. Ziniel, L.C. Potter, and P. Schniter. Tracking and smoothing of time-varying sparse signals via approximate belief propagation. In *Signals, Systems and Computers (ASILOMAR), 2010 Conference Record of the Forty Fourth Asilomar Conference on*, pages 808 –812, nov. 2010. 10