Bioinformatics, 35(18), 2019, 3279–3286
doi: 10.1093/bioinformatics/btz060
Advance Access Publication Date: 25 January 2019
Original Paper

OXFORD

Sequence analysis

Haplotype assembly of autotetraploid potato using integer linear programing

Enrico Siragusa[1], Niina Haiminen[1], Richard Finkers[2], Richard Visser[2] and Laxmi Parida[1,*]

[1]IBM T J Watson Research Center, Yorktown Heights, NY, USA and [2]Wageningen UR Plant Breeding, Wageningen, The Netherlands

[*]To whom correspondence should be addressed.

Associate Editor: Inanc Birol

Received on June 28, 2018; revised on January 14, 2019; editorial decision on January 18, 2019; accepted on January 22, 2019

Abstract

Summary: Haplotype assembly of polyploids is an open issue in plant genomics. Recent experimental studies on highly heterozygous autotetraploid potato have shown that available methods do not deliver satisfying results in practice. We propose an optimal method to assemble haplotypes of highly heterozygous polyploids from Illumina short-sequencing reads. Our method is based on a generalization of the existing minimum fragment removal model to the polyploid case and on new integer linear programs to reconstruct optimal haplotypes. We validate our methods experimentally by means of a combined evaluation on simulated and experimental data based on 83 previously sequenced autotetraploid potato cultivars. Results on simulated data show that our methods produce highly accurate haplotype assemblies, while results on experimental data confirm a sensible improvement over the state of the art.

Availability and implementation: Executables for Linux at http://github.com/ComputationalGenomics/HaplotypeAssembler.

Contact: parida@us.ibm.com

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Many plants of agronomic importance are polyploids i.e. their somatic cells contain more than two copies ($p > 2$) of each haploid set of chromosomes. Seedless varieties of watermelon and banana are human-induced triploids ($p = 3$); cultivars of potato, peanut, cotton, tobacco and coffee are naturally occurring tetraploids ($p = 4$); wheat is hexaploid ($p = 6$). N'Diaye *et al.* (2017) demonstrated how haplotype-based analysis results in an increase of the phenotypic variance explained (50.4% on average) compared with single marker analysis when detecting loci associated with color traits in durum wheat. Knowledge of the haplotypes, the distinct sequences of haploid chromosomes, is limited or absent even for common cultivars. This fact limits the effectiveness of plant breeding to selectively develop particular phenotypic traits.

One method for determining haplotypes is bulk DNA sequencing followed by haplotype assembly. The haplotypes are sequenced jointly and then demultiplexed *in silico* by assembling sequenced DNA fragments based on a known reference haplotype. A study by Uitdewilligen *et al.* (2013) demonstrated successful genotyping by a targeted resequencing of a panel of 83 autotetraploid potato cultivars. Assembling haplotypes from their Illumina HiSeq paired-end data should be feasible since potato is highly heterozygous. The authors observed a variant density of 1 SNP/24 bp in exons and 1 SNP/15 bp in introns in the studied 83-cultivar panel. Despite the high heterozygosity, a subsequent experimental study by Motazedi *et al.* (2017) reported that existing computational methods for haplotype assembly are not delivering satisfying results in practice. It is unclear to what extent assemblies are inaccurate because of the heuristics, or because of insufficient or erroneous data.

1.1 Previous work

Haplotype assembly of diploid genomes has been extensively studied over the past two decades. Lancia *et al.* (2001) first introduced the

problem and proposed a combinatorial model called *minimum fragment removal* (MFR) that is solvable in polynomial time for contiguously sequenced fragments (i.e. single-end reads) but NP-hard for gapped fragments (i.e. reads obtained via paired-end or mate-pairs protocols). Subsequently, Lippert *et al.* (2002) refined MFR as *minimum error correction* (MEC), which is NP-hard even for contiguous fragments. MEC became the de-facto model to assemble diploid genomes as several exact and heuristic methods have been proposed for that. We refer the reader to (Schwartz *et al.*, 2010) for a comprehensive treatment of haplotype assembly for diploids.

In recent years, the focus shifted towards assembling polyploid genomes. Aguiar and Istrail (2013) defined an NP-hard problem named *minimum weighted edge removal* in a compass graph and employed a minimum-cost spanning tree heuristic to solve it. Berger *et al.* (2014) defined a probabilistic framework and used heuristic branch-and-bound to find likely haplotypes given the fragments. Das and Vikalo (2015) casted the problem as a correlation clustering problem and derived approximate solutions to the associated semi-definite program via lagrangian relaxation followed by randomized projections and greedy refinement. Xie *et al.* (2016) defined a NP-hard problem called *polyploid balanced optimal partition* and proposed constrained dynamic programing to find heuristic solutions. Very recently, Mazrouee and Wang (2018) proposed a clustering approach to minimize disagreement among fragments. All the above methods are heuristic and deliver solutions without guarantees for a combinatorial problem that is hard to approximate (APX-hard) (Bonizzoni *et al.*, 2016).

Integer linear programing (ILP) is a powerful mathematical programing method to efficiently solve combinatorial problems to optimality (Winston *et al.*, 2003). Schwartz *et al.* (2010) remarked that 'the ILP strategy has thus so far received comparatively little attention in the haplotype assembly field'. The only known ILPs for haplotype assembly are specific to diploids (Chen *et al.*, 2013; Etemadi *et al.*, 2018). In a different setting, Szolek *et al.* (2014) has successfully applied an MFR-based ILP to haplotype human leukocyte antigen genes from short-sequencing reads. To the best of our knowledge, haplotype assembly for polyploids has not been attacked yet using ILP.

1.2 Our contribution

We propose an optimal method to assemble haplotypes in highly heterozygous polyploids from Illumina short-sequencing reads. Our method is based on a generalization of MFR to the polyploid case; we leverage ILP to reconstruct optimal haplotypes. In addition, we propose haplotyping distance as a general method to perform pairwise comparison of polyploids and we apply that to assess the accuracy of haplotype assemblies. We validate our models experimentally through a combined evaluation on simulated and real autotetraploid potato sequencing data extrapolated from the targeted sequencing of 83 cultivars performed by Uitdewilligen *et al.* (2013). Results on simulated data show that MFR-based ILPs achieve mean 98% haplotyping recall and precision, that is a 4–11% improvement over existing tools. Results on experimental data confirm the superiority of our methods in terms of genotyping and read error correction.

2 Materials and methods

2.1 Haplotype assembly

2.1.1 Problem definition

Let us fix $p \in \mathbb{N}$ as organism ploidy and $n \in \mathbb{N}$ as number of observed loci in a genomic region of interest. Such genomic region consists of p latent haplotypes $H = \{h_1, \ldots, h_p\}$ over the n genomic loci. At each locus j we observe $a_j \in \mathbb{N}$ distinct alleles across all haplotypes in H. We univocally encode alleles as integers so that each latent haplotype h_k is a sequence of alleles $h_{k1} \ldots h_{kn}$ with $h_{kj} \in [0, a_j)$. Figure 1a shows an example of latent haplotypes.

We are given as input a multiset (F, W) of *preprocessed* fragments coming from all the latent haplotypes in H, where fragments in $F = \{f_1, \ldots, f_m$ have multiplicities in $W = \{w_1, \ldots, w_m\}$. Each fragment f_i is a gapped sequence of alleles $f_{i1} \ldots f_{in}$ with $f_{ij} \in [0, a_j) \cup \{\cdot\}$ and symbol $\cdot$ denoting unknown alleles, i.e. missing observations due to fragment boundaries, paired sequencing protocols or quality thresholds. Fragments may not exactly match their originating latent haplotype due to sequencing errors. The haplotype assembly problem for polyploids is to find the p latent haplotypes in H given all observations from fragments in (F, W). Figure 1b illustrates an example of input fragments.

2.1.2 Heterozygous loci

In haplotype assembly we ignore *homozygous* loci, i.e. genomic loci with only one observed allele. At homozygous loci either all latent haplotypes have the same observed allele or we have no information whether observations are wrong or missing. For this reason, we perform haplotype assembly only on *heterozygous* loci, i.e. genomic loci for which $a_j > 1$.

2.1.3 Concordance and containment

Before introducing combinatorial models for haplotype assembly, we define basic relations of concordance and containment between sequences of alleles. The following definitions hold for fragments and haplotypes, as they are both defined as sequences of alleles.

Let a and b be two allele observations at the same genomic locus. Given $a \neq \cdot$ and $b \neq \cdot$, we say that a and b are concordant if $a = b$ and discordant if $a \neq b$. If either $a = \cdot$ or $b = \cdot$, we say that a and b are non-discordant and write $a \simeq b$. In addition, if $b \neq \cdot$ and either $a = \cdot$ or $a = b$, we say that a is contained in b and write $a \sqsubset b$.

We generalize this terminology to sequences of alleles. We say that sequences of alleles x and y are non-discordant if all alleles of x and y are non-discordant. Similarly, x is contained in y if all alleles of x are contained in those of y. Two non-discordant fragments contribute to explain a common latent haplotype, while two discordant fragments must either originate from distinct haplotypes or imply some sequencing error.

2.1.4 MFR and MEC for polyploids

Two combinatorial models called MFR (Lancia *et al.*, 2001) and MEC (Lippert *et al.*, 2002) have been proposed for diploids. These models are motivated by the parsimony principle by which, within a set of possible explanations of observations, the simplest one is most likely to be true. In this instance, the closest possible haplotypes to input fragments, according to some predefined distance function, are most likely to be correct. Here we generalize MFR and MEC to polyploids.

We first define an objective function FR that counts the number of erroneous fragments that are not contained in candidate haplotypes H and must be therefore removed from (F, W) to explain a correct assembly of H:

$$FR(F, W; H) := \sum_{i=1}^{m} w_i \min_{k=1}^{p} \delta_{\sqsubset}(f_i, h_k),$$

where function $\delta_{\sqsubset}(f_i, h_k)$ is defined as 0 if $f_i \sqsubset h_k$ and 1 otherwise. Consequently, MFR is computed by finding the haplotypes H^* that minimize the number of fragments to be removed:

(a)

k	h_k
1	0 0 0 0 0 0
2	1 0 0 1 1 1
3	1 0 0 1 1 1
4	1 1 1 2 0 1

l	g_l
0	1 3 3 1 2 1
1	3 1 1 2 2 3
2	0 0 0 1 0 0

Latent haplotypes H and known genotypes G

(b)

i	f_i	w_i
1	0 0 · 0 · ·	2
2	· 0 0 2 1 ·	1
3	· 0 0 · · ·	1
4	· · · 0 0 0	3
5	1 0 · 1 · ·	4
6	· 1 0 · 1 ·	4
7	· · · 1 1 1	5
8	1 1 · 2 · ·	2
9	· · 1 · 0 ·	2
10	· · · 2 · 1	3

Input fragments F and multiplicities W

(c)

$\delta_\sqsubset$	h_1	h_2	h_3	h_4
f_1	0	1	1	1
f_2	1	1	1	1
f_3	0	0	0	1
f_4	0	1	1	1
f_5	1	0	0	1
f_6	1	0	0	1
f_7	1	0	0	1
f_8	1	1	1	0
f_9	1	1	1	0
f_{10}	1	1	1	0

FR has value 1 as f_2 is not contained in any haplotype

(d)

$\delta_\simeq$	h_1	h_2	h_3	h_4
f_1	0	2	2	3
f_2	2	1	1	3
f_3	0	0	0	2
f_4	0	3	3	2
f_5	2	0	0	1
f_6	1	0	0	3
f_7	3	0	0	2
f_8	3	2	2	0
f_9	1	2	2	0
f_{10}	2	1	1	0

EC has value 1 as f_2 has at least 1 discordant allele

(e)

k	h^*_k
1	0 0 0 0 0 0
2	1 0 0 1 1 1
3	1 1 1 2 0 1
4	1 1 1 2 0 1

l	$d_l(H)^*$
0	1 2 2 1 3 1
1	3 2 2 1 1 3
2	0 0 0 2 0 0

Assembled haplotypes H^* and implied genotypes G^*

Fig. 1. Example of haplotype assembly. **(a)** shows a set of latent haplotypes H along with their genotypes. **(b)** shows a multiset (F, W) of fragments coming from H. **(c)** shows values of $\delta_\sqsubset$ between F and H. Objective function FR$(F, W; H)$ has value 1 as fragment f_2 with multiplicity $w_2 = 1$ is not contained in any haplotype. **(d)** shows function $\delta\simeq$. Likewise, objective function EC has value 1 as fragment f_2 has at least 1 discordant allele compared with any latent haplotype. **(e)** shows an alternative assembly of the fragments. Both latent and assembled haplotypes yield MEC and MFR 1. Assembled haplotypes in (e) become infeasible when known genotype constraints G are incorporated into MFR or MEC

$$\mathrm{MFR}(F, W) = \min_{H^*} \mathrm{FR}(F, W; H^*).$$

Analogously, we define MEC as the sum of minimum Hamming distances between haplotypes H and input fragments (F, W). MEC is based on the following objective function:

$$\mathrm{EC}(F, W; H) := \sum_{i=1}^{m} w_i \min_{k=1}^{p} \delta \simeq (f_i, h_k),$$

where function $\delta \simeq (f_i, h_k)$ denotes the Hamming distance between f_i and h_k parameterized by $\simeq$. Hence EC counts the minimum number of base calling errors in (F, W) to explain a correct assembly of H.

Figure 1c and d illustrates tabulated values for objective functions FR and EC. Note how a fragment f_i can be non-discordant with (or have equal distance to) more than one haplotype h_k. Our generalization of MFR and MEC does not partition F in k disjoint subsets as done for instance by Das and Vikalo (2015) and Xie *et al.* (2016).

We developed our haplotype assembly model on MFR rather than MEC for two reasons. First, the simplicity of MFR allowed us to formulate concise and efficient ILPs. Second, the low error rate of Illumina sequencing reads suggests that most fragments are correctly sequenced and do not need to be corrected or removed. Additionally, the high coverage of the sequenced fragments allows for removing erroneous fragments yet retaining enough information for phasing.

2.1.5 Genotype-based MFR

MFR alone is not well-defined in the polyploid case (neither is MEC). Figure 1a and e illustrates this issue. A notion of coverage is necessary to determine the number of copies of each unique haplotype, while these simple combinatorial models are clearly coverage oblivious. For this reason we supplement MFR with genotyping information, that is essentially a surrogate of coverage information. Genotypes are estimated a priori from fragments (F, W) assuming uniform sequencing coverage across the haplotypes. MEC-based models can be easily extended in the same way.

Our extended model takes as additional input a matrix of genotypes G, where $g_{lj} \in \mathbb{N}$ denotes the dosage (i.e. multiplicity) of the l-th allele observed at locus j and $\sum_l g_{lj} = p$. We use this additional information to narrow down the search space of our model. Our genotype-constrained MFR (cMFR) model determines the latent haplotypes as:

$$\arg\min_{H^*: d(H^*) = G} \mathrm{FR}(F, W; H^*),$$

where $d(H)$ denotes the genotypes matrix induced by candidate haplotypes H. Note how this is the natural generalization of the *all-heterozygous assumption* for diploids by which all observed heterozygous alleles are believed to be correct and the two latent haplotypes must be complementary.

We propose also an alternative model that is useful whenever sequencing coverage is too low to call genotypes accurately. We incorporate genotyping information in the objective function as the L^1 norm between induced and input genotypes. Consequently, we prioritize fragment removal over genotyping information. Our genotype-augmented MFR (aMFR) model determines the latent haplotypes according to a pair of objective functions:

$$\arg\min_{H^*} \Big(\mathrm{FR}(F, W; H^*), \|d(H^*) - G\|_1\Big),$$

where FR is used as the primary objective function and the L^1 norm as the secondary objective function.

2.1.6 ILP for genotype-cMFR

Our ILPs are based on containment relation $\sqsubset$ between fragments and haplotypes. In what follows, unless otherwise stated, subscript $i \in [1, m]$, $j \in [1, n]$, $k \in [1, p]$. We define three types of binary variables to encode, respectively, fragments to be removed (Line 2), fragments contained in haplotypes (Line 3) and alleles on each haplotype (Line 4). Variable x_i is 0 if fragment f_i is to be removed and 1 otherwise. Variable y_{ik} is 1 if $f_i \sqsubset h_k$ and 0 otherwise. Variable z_{kjl} is 1 if haplotype h_k has the lth allele at locus j and 0 otherwise. For convenience, our ILP maximizes the inverse of function FR and y_{ik} tabulates function $\delta_\sqsubset$ in negated form with respect to Figure 1c.

Objective function 1 maximizes the number of unremoved fragments in R weighted by multiplicities in W. Constraint 5 marks a fragment for removal unless it is contained in some haplotype. Constraints 6–7 allow a fragment to be contained in a haplotype only if all its alleles are contained in the haplotype; Constraint 8

improves convergence by observing that if $f_i \sqsubset f_o$ and $f_i \not\sqsubset h_k$ then $f_o \not\sqsubset h_k$. Finally, Constraint 9 imposes input genotype information on the haplotypes.

$$\max \sum_{i=1}^{m} w_i \cdot x_i \tag{1}$$

s.t.

$$x_i \in \{0,1\} \qquad \forall k,j \tag{2}$$

$$y_{ik} \in \{0,1\} \qquad \forall i,k \tag{3}$$

$$z_{kjl} \in \{0,1\} \qquad \forall k,j,l \in [0,a_j) \tag{4}$$

$$x_i \leq \sum_{k=1}^{p} y_{ik} \qquad \forall i \tag{5}$$

$$y_{ik} \geq 1 + \sum_{f_{ij} \neq \cdot} (z_{kjfij} - 1) \qquad \forall i,k \tag{6}$$

$$y_{ik} \leq z_{kjfij} \qquad \forall i,k,j : f_{ij} \neq \cdot \tag{7}$$

$$y_{ok} \leq y_{ik} \qquad \forall k,(i,o) : f_i \sqsubset f_o \tag{8}$$

$$\sum_{k=1}^{k=1p} z_{kjl} = g_{lj} \qquad \forall j,l \in [0,a_j) \tag{9}$$

We remark that Constraint (6) removes ambiguous solutions by ensuring that y_{ik} is 1 if fragment f_i and haplotype h_k are concordant. This addresses the case of fragments that are concordant with more than one haplotype. In other words, Constraint (6) maximizes the number of containments $\sum_{i=1}^{m} \sum_{k=1}^{p} y_{ik}$.

The above ILP may become infeasible when some haplotypes lack sequencing coverage locally. If a haplotype has no coverage at locus j with allele l, there will be strictly $< g_{lj}$ haplotypes with allele l. To consider this case, we substitute Constraint 9 with 11–12 (and keep Constraints 2–8). Now we have to minimize also the distance between input and induced genotypes, that is:

$$\sum_{j=1}^{n} \sum_{l=0}^{a_j-1} \left(g_{lj} - \sum_{k=1}^{p} z_{kjl} \right) = \sum_{j=1}^{n} \sum_{l=0}^{a_j-1} g_{lj} - \sum_{k=1}^{p} \sum_{j=1}^{n} \sum_{l=0}^{a_j-1} z_{kjl}.$$

We drop the first term because it is a constant (np). As we prioritize genotypes over fragment removal, instead of normalizing the fragment removal term by $\sum_{i=1}^{m} w_i$, we multiply the second term by $\sum_{i=1}^{m} w_i$. This leads us to Objective function 10.

$$\max \sum_{i=1}^{m} w_i \left(x_i + \sum_{k=1}^{p} \sum_{j=1}^{n} \sum_{l=0}^{a_j-1} z_{kjl} \right) \tag{10}$$

$$\sum_{l=0}^{a_j-1} z_{kjl} \leq 1 \qquad \forall k,j : f_{ij} \neq \cdot \tag{11}$$

$$\sum_{k=1}^{p} z_{kjl} \leq g_{lj} \qquad \forall j,l \in [0,a_j) \tag{12}$$

2.1.7 ILP for genotype-*a*MFR

Genotyping is prone to errors in regions of low coverage or in presence of allelic bias introduced during fragment selection or amplification prior to sequencing. Our model *c*MFR is not robust with respect to genotyping errors as it minimizes fragment removal subject to hard genotype constraints. Here we propose an alternative model *a*MFR that that minimizes fragment removal first and genotyping distance second.

We keep Constraints 2–8 from the previous ILPs and introduce real variables d_{jl} (Line 14) to model absolute distances between input and induced genotypes. We substitute Constraint 12 with 15–16 and incorporate absolute distances in Objective function 13. To prioritize fragment removal over genotyping distance, we add a normalization constant $1/(2np)$. Note that fragment removal (the summand on the left) is a function over $\mathbb{N}$, while genotyping distance (the summand on the right) is at most $2np$. We normalize genotyping distance within $[0,1)$ so that fragment removal is the primary objective function.

$$\max \sum_{i=1}^{m} w_i \cdot x_i - \frac{1}{2np} \sum_{j=1}^{n} \sum_{l=0}^{a_j-1} d_{jl} \tag{13}$$

$$d_{jl} \in \mathbb{R}_0^+ \qquad \forall j,l \in [0,a_j) \tag{14}$$

$$g_{lj} - \sum_{k=1}^{p} z_{kjl} \leq d_{jl} \qquad \forall j,l \in [0,a_j) \tag{15}$$

$$\sum_{k=1}^{p} z_{kjl} - g_{lj} \leq d_{jl} \qquad \forall j,l \in [0,a_j) \tag{16}$$

2.2 Haplotype comparison

We now describe methods to compare two polyploids X, Y over a common genomic region. In particular, we propose *haplotyping distance* as a general method for pairwise comparison of polyploids with arbitrary genotypes. For simplicity, in what follows we assume that all alleles in X and Y are known, i.e. there is no$\cdot$. Handling of unknown calls is covered in Section 3 as they relate to the definitions of recall and precision.

2.2.1 Phasing distance

We call phasing distance the minimal Hamming distance between haplotypes in X and any permutation of the haplotypes in Y:

$$\delta_P(X,Y) := \min_{\sigma \in \Sigma_p} \sum_{k=1}^{p} \delta \simeq \left(x_k, y_{\sigma(k)} \right),$$

where Σ_p denotes the set of all permutations of sequence $\{1,2,\ldots,p\}$ and $\sigma(k)$ the element at position k in permutation σ.

We compute phasing distance in $\mathcal{O}(n^2 + p!)$ time by proceeding in two easy steps. First, we compute the Hamming distance between all pairs of haplotypes in X and Y. Second, we permute haplotypes in Y to minimize the sum of Hamming distances with respect to the ordered haplotypes in X.

2.2.2 Haplotyping distance

Phasing distance does not model haplotype *switch* operations. Therefore, in terms of mismatches, a switch between two pairs of corresponding haplotypes can be more or less costly depending on its genomic position. A switch towards the center of the genomic region is accounted as a long sequence of mismatches rather than a single operation with unitary cost. Figure 2a–c shows examples of switches.

(a)

k	x_k
1	0 0 0 0 0 0
2	1 0 0 1 1 1
3	1 0 0 1 1 1
4	1 1 1 2 0 1

(b)

y_k
1 0 0 0 0 0
0 0 0 1 1 1
1 0 0 1 1 1
1 1 1 2 0 1

(c)

y_k
0 0 0 0 1 1
1 0 0 1 0 0
1 0 0 1 1 1
1 1 1 2 0 1

(d)

y_k
0 0 0 0 1 0
1 0 0 1 1 1
1 0 0 1 1 1
1 1 1 2 0 1

Fig. 2. Example of haplotype comparison. Phasing distance between **(a)** and **(b)** is 2, while it is 4 between (a) and **(c)**. On the contrary, vector and haplotyping distance is 2 in both cases (note how suffixes of haplotypes y_1 and y_2 can be switched in (c). Distance between (a) and **(d)** is undefined in vector error, while it is 1 in phasing and haplotyping distance

Berger *et al.* (2014) introduced *vector error* as a generalization of *switch error* to count the number of switches between two polyploid genomes. Vector error denotes the minimum number of segments on all chromosomes for which a switch must occur between halotypes in X and Y (in the diploid case this is exactly twice the switch error). Vector error has limited applicability on practical instances because it is defined only for genomic regions with equal genotypes (see Fig. 2d). To overcome this limitation, we propose *haplotyping distance* as a generalization of vector error.

Haplotyping distance counts the minimum number of haplotype switches and genotype mismatches between two polyploids. To compute haplotyping distance we extend the dynamic programing algorithm for vector error given by Xie *et al.* (2016) (Supplementary Fig. S3). Algorithm 1 operates on X and Y transposed, i.e. it advances from left to right one column (locus) at time. We denote by $\tau(y_j)$ the column j of Y permuted by τ. Initialization (Line 3) counts the number of mismatching alleles at column $j=1$ between X and each permutation $\sigma \in \Sigma_p$ of the haplotypes in Y. Recurrence (7) minimizes the haplotyping distance for each column $j>1$, building up from the distance precomputed for column $j-1$. The minimum distance obtained by permuting the haplotypes in Y as $\sigma \in \Sigma_p$, the cost of remaining allele mismatches, and the number of switches introduced by going from permutation τ to σ are added to the haplotyping distance for permutation τ at column $j-1$. Algorithm 1 computes haplotyping distance in $\mathcal{O}(n \cdot p!^2)$ time and $\mathcal{O}(p!)$ space.

1: **function** $\delta_H(X, Y)$
2: **for all** $\sigma \in \Sigma_p$ **do**
3: $h_{1\sigma} \leftarrow \delta \simeq \left(x_1, \sigma(y_1)\right)$
4: **for** $j \leftarrow 2$ **to** n **do**
5: **for all** $\sigma \in \Sigma_p$ **do**
6: $h_{j\sigma} \leftarrow \min_{\tau \in \Sigma_p} \{h_{(j-1)\tau} + \delta \simeq \left(x_j, \tau(y_j)\right) + \|\sigma - \tau\|_1\}$
7: **return** $\min_{\sigma \in \Sigma_p} h_{n\sigma}$

Algorithm 1: Dynamic programing calculation of haplotyping distance. Computation proceeds on X and Y transposed, and $\tau(y_j)$ denotes the locus j of Y permuted by τ. Distance $\delta \simeq \left(x_j, \tau(y_j)\right)$ denotes the number of allele mismatches at locus j, while $\|\sigma - \tau\|_1$ is the number of switches to go from permutation τ to σ.

3 Results

3.1 Experimental data

We obtained targeted high-throughput sequencing data for 83 highly heterozygous autotetraploid ($p=4$) potato cultivars by Uitdewilligen *et al.* (2013). Cultivars have been sequenced on Illumina HiSeq 2000 at 63× median coverage with 2×96 bp paired-end reads from fragments of 300 bp mean length and 60 bp standard deviation. We reproduced the analysis pipeline described in (Uitdewilligen *et al.*, 2013) for read mapping, deduplication, variant calling and genotyping. We produced a total of 996 test instances by selecting 12 high-quality genomic regions from the sequencing panel. Supplementary Table S1 shows the genomic coordinates of the selected regions, as well as the median number of heterozygous variants per sample.

3.2 Simulated data

We simulated tetraploid data matching the real genotypes of the 83 potato cultivars by Uitdewilligen *et al.* (2013). First, we used the haplotype simulator SimBA-hap (Siragusa *et al.*, 2017) to generate 80 samples from each genomic region with genotypes fitting those of the sequenced cultivars. We remark that simulated variants and genotypes are biallelic and error-free. Subsequently, we used the read simulator Mason (Holtgrewe, 2010) to produce Illumina-like sequencing reads from each simulated region. We produced an Illumina-like paired-end dataset reflecting the technical specifications of actual Illumina instruments: 90× coverage with 2×150 bp paired-end reads from fragments of 500 bp mean length and 60 bp SD at 0.3 % mean sequencing error. To assess the effect of sequencing errors and fragments length on haplotype assembly, we simulated two supplementary datasets by altering specific simulation parameters. We produced a second paired-end dataset consisting of error-free paired-end reads and a third dataset by simulating mate-pair sequencing fragments of 1000 bp mean length and 400 bp standard deviation (with sequencing errors). Mate-pair sequencing allows for investigating haplotypes across a longer range. We produced a total of 960 test instances per dataset.

3.3 Infrastructure

We implemented our ILPs for MFR in C++ using IBM CPLEX® 12.7.0 and the software library SeqAn 2.3.2 (Reinert *et al.*, 2017). In all experiments we configured CPLEX timeout at 600 s. We ran and evaluated our MFR-based models against HapCompass (Aguiar and Istrail, 2013), SDhaP (Das and Vikalo, 2015), H-PoP (Xie *et al.*, 2016) on all real and simulated instances. Although HapCompass accepted standard BAM and VCF files, for SDhaP and H-PoP we had to produce intermediate fragment files using scripts provided by Motazedi *et al.* (2017) relying on HapCut's tool extractHAIRS (Bansal and Bafna, 2008). Out of 960 instances of experimental data, H-PoP failed to produce results on one and SDhaP on two instances. In those cases we considered their assembled haplotypes to be fully unknown (all·). We were unable to run HapTree (Berger *et al.*, 2014) on any of our instances nor to communicate with its corresponding authors. To insure reproducibility of the results, we wrote a Snakemake pipeline (Köster and Rahmann, 2012) and deployed it on IBM Cloud™ using private single-core instances.

3.4 Recall and precision

Haplotype assembly tools may omit some or all allele calls at certain loci, presumably due to insufficient sequencing data. We therefore measured recall and precision accounting for uncalled alleles in the assembled haplotypes.

Given simulated haplotypes X of ploidy p over n alleles and assembled haplotypes Y with $\mu(Y)$ uncalled alleles, we computed the number of incorrectly called alleles as $\delta(X, Y) - \mu(Y)$, where δ is phasing or haplotyping distance. Analogously, we computed the number of correctly called alleles as $np - \delta(X, Y) - \mu(Y)$. We

Table 1. Haplotype assembly results

	Simulated tetraploid potato data										Experimental tetraploid potato data					
	Haplotyping		Phasing		Genotyping		MEC		Resources		Genotyping		MEC		Resources	
Tool	Recl. [%]	Prec. [%]	Recl. [%]	Prec. [%]	Recl. [%]	Prec. [%]	Recl. [%]	Prec. [%]	Time [s]	Mem. [MB]	Recl. [%]	Prec. [%]	Recl. [%]	Prec. [%]	Time [s]	Mem. [MB]
*c*MFR	**98.0**	**98.1**	**95.8**	**95.8**	**100.0**	**100.0**	**99.9**	**99.9**	17.6	**48**	**99.8**	**100.0**	99.1	99.2	95.0	59
*a*MFR	96.8	97.0	94.5	94.7	99.7	99.7	99.8	**99.9**	45.0	**48**	98.4	98.6	**99.8**	**99.9**	58.9	**49**
H-PoP	87.2	92.4	85.1	90.3	97.2	98.7	94.3	98.9	**1.1**	63	94.0	96.3	88.4	98.7	**1.2**	59
SDhaP	91.1	94.1	87.4	90.5	98.5	99.4	96.6	99.2	34.6	6309	95.2	96.5	91.3	98.9	50.1	6309
HapCompass	86.8	86.9	81.8	82.1	**100.0**	**100.0**	92.6	92.8	4.7	112	99.2	99.7	90.0	93.9	6.8	114

Note: The left panel shows results on simulated data, while the right panel shows results on experimental data. Mean values across all tested instances. Values in bold denote best results in each category.

defined recall as the fraction of correctly called alleles over all alleles, which is equivalent to:

$$1 - \frac{\delta(X, Y) + \mu(Y)}{np},$$

and precision as the fraction of correctly called alleles over all called alleles, which is equivalent to:

$$1 - \frac{\delta(X, Y)}{np - \mu(Y)}.$$

We defined genotyping recall and precision analogously. Both known genotypes G and induced genotypes $d(Y)$ may contain unknown alleles. The number of called alleles is $np - \mu(Y)$. We computed the number of incorrectly called alleles as:

$$\sum_{j=1}^{n} \sum_{l=0}^{a_j - 1} \max\left(d_{jl}(Y) - g_{jl}, 0\right)$$

and the number of correctly called alleles as called minus incorrectly called alleles.

We defined recall and precision also in terms of MEC. Given the assembled haplotypes Y, we computed the number of alleles implied to be incorrect as $\mathrm{EC}(Y, F; W)$, those implied to be correct as:

$$\sum_{i=1}^{m} w_i \left(|f_i| - \min_{k=1}^{p} \delta \simeq (y_k, f_i)\right),$$

and the total number of alleles as $\sum_{i=1}^{m} w_i |f_i|$. Here $|f_i|$ denotes the number of non-missing values in f_i.

3.5 Results on simulated data

Table 1 (left) shows mean precision and recall for each tool on the Illumina-like paired-end dataset with respect to haplotyping, phasing, genotyping and MEC, as well as mean runtime and memory footprint. Model *c*MFR obtained consistently the highest values in all precision and recall categories. In particular, *c*MFR obtained 8.4% higher phasing recall than SDhaP. Model *a*MFR lost 1.1–1.4% recall and precision in haplotyping and phasing. Note that HapCompass achieved perfect genotyping but it obtained the lowest recall and precision in haplotyping, phasing and MEC. We remark how MEC is in agreement with haplotyping and phasing when ranking the tools by recall and precision. Mean runtime for *c*MFR is 17.6 s while for *a*MFR it is 45.0 s. H-PoP and HapCompass achieved significantly lower runtimes compared to our MFR-based ILPs but their recall in all categories is equally lower. Memory footprint stayed within 120 MB for all tools except SDhaP that required 6 GB of main memory.

Figure 3 (left) shows the distribution of haplotyping recall and precision values on the Illumina-like paired-end dataset. Supplementary Figure S1 (left) shows recall and precision under phasing distance. Recall and precision values under phasing distance are lower with respect to haplotyping distance because phasing distance does not employ switch operations. Nonetheless, there is no significant change in the relative performances of haplotyping tools.

Supplementary Table S2 and Supplementary Figure S2 (left) show results on the error-free paired-end dataset. Recall and precision values are in line with those on the Illumina-like paired-end dataset shown in Table 1 (left). All tools show between 0.1 and 0.3% improvement on the error-free dataset. That is in agreement with the 0.3% sequencing error-rate used for Illumina-like reads simulation. None of the tools achieved perfect haplotyping recall or precision.

Supplementary Table S3 and Supplementary Figure S2 (right) show results on the mate-pair dataset. Model *c*MFR obtains 99.5% mean recall and 99.7% mean precision under haplotyping distance, showing improvement over the paired-end dataset. That is an almost perfect assembly. Conversely, none of the other assemblers shows a significant improvement on the mate-pair dataset with respect to the paired-end datasets.

3.6 Results on experimental data

We computed genotyping recall and precision by comparing the genotypes induced by the assembled haplotypes to the genotypes previously computed by the variant caller. In addition, we computed precision and recall values for MEC as this can be done without knowing the true haplotypes. Recall and precision under MEC correlate well with haplotyping and phasing, as seen in Table 1 (left).

Table 1 (right) shows mean precision and recall of each tool across all real instances with respect to genotyping and MEC, as well as mean runtime and memory footprint. MEC precision for all tools is comparable to what observed on simulated data, while MEC recall on experimental data drops significantly for H-PoP and SDhap. Figure 3 (right) shows the distribution of MEC versus genotyping recall and precision on all real instances. As expected, *c*MFR and *a*MFR values are concentrated on the top right corner, with *c*MFR lying on (or very close to) the 100% MEC line and *a*MFR lying on the 100% genotyping line.

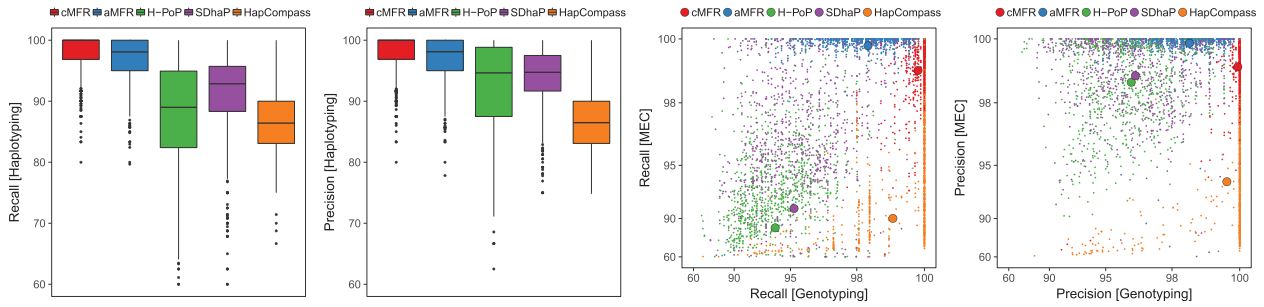

Fig. 3. Haplotype assembly results. The two plots on the left show haplotyping recall and precision on simulated data, while the two plots on the right show MEC versus genotyping on experimental data. Dots in the scatter plot denote results for individual instances, while circles denote mean values per tool

4 Discussion

Motivated by practical limitations in breeding autotetraploid potato cultivars, we investigated methods for haplotype assembly of polyploids. We designed haplotype assembly models that are based on MFR, incorporate sequencing coverage using genotypes and are solved to optimality using ILP. In addition, we proposed haplotyping distance to overcome the limitations of existing methods in comparing polyploid genomes. We applied haplotyping distance to evaluate the accuracy of haplotype assemblies.

Results on simulated data showed that our methods achieve a sensible improvement over the state of the art. Existing methods do not attain our performances either because of inadequate modeling, because of implementation issues, or because of heuristics failing to find optimal solutions. It is hard to assess to which extent each of these factors affects the performances of existing tools. Aggressive heuristics employed to speed up runtimes may significantly lower the accuracy of H-PoP and HapCompass on our datasets. Significantly higher precision compared with recall suggests that H-Pop and SDhaP tend to skip certain loci that might be harder to assemble. This issue is even more evident on experimental data.

Our combined evaluation helped us to interpret performances on experimental data in absence of an experimental validation. Results on experimental data confirmed the improvements of our methods in terms of MEC and genotyping distance. As these two objectives are orthogonal, a simultaneous improvement in MEC and genotyping distance gives us confidence that the quality of the assemblies improved.

In particular, results on simulated mate-pair data show that longer fragments dramatically boost assembly accuracy. Our models produce almost-perfect assemblies, within 0.3% mean haplotyping precision, while assemblies using H-Pop, SDhap or HapCompass plateau at 8–13% or even worsen compared to paired-end data. These results suggest that our methods could be used to derive high-quality haplotype assemblies of tetraploid potato using a combination of mate-pair sequencing reads in order to capture long-range variant interactions and paired-end sequencing reads at higher coverage in order to call genotypes accurately.

Conversely, on Illumina short-sequencing reads, we do not expect MEC-based models to provide significant improvements over MFR. Precision and recall results on error-free sequencing data indicate that residual errors in the assemblies are due to insufficient data (i.e. too short fragments) rather than erroneous sequencing data. Results on simulated mate-pair data show that longer fragments boost assembly accuracy and thus support our hypothesis. Conversely, we expect long noisy reads to be challenging for MFR-based models. We did not investigate this latter hypothesis because we have no access to such experimental data.

Longer runtimes suggested that experimental data has higher complexity than our simulated data, as was to be expected. Our simulation did not faithfully reproduce all possible artifacts that may arise along high-throughput sequencing pipelines and cumulate along sample preparation, base calling, read mapping, variant calling and genotyping steps. In addition, our simulation did not account for the presence of copy number variations (CNVs) or complex structural variations in the genomic regions of interest. While our ILPs are flexible enough to take in consideration known CNVs, we did not explore this direction. We believe that CNV modeling might play an important role in the assembly of experimental data.

Conflict of Interest: none declared.

References

Aguiar,D. and Istrail,S. (2013) Haplotype assembly in polyploid genomes and identical by descent shared tracts. *Bioinformatics*, **29**, i352–i360.

Bansal,V. and Bafna,V. (2008) HapCUT: an efficient and accurate algorithm for the haplotype assembly problem. *Bioinformatics*, **24**, i153–i159.

Berger,E. *et al.* (2014) Haptree: a novel Bayesian framework for single individual polyplotyping using NGS data. *PLoS Comput. Biol.*, **10**, e1003502.

Bonizzoni,P. *et al.* (2016) On the minimum error correction problem for haplotype assembly in diploid and polyploid genomes. *J. Comput. Biol.*, **23**, 718–736.

Chen,Z.Z. *et al.* (2013) Exact algorithms for haplotype assembly from whole-genome sequence data. *Bioinformatics*, **29**, 1938–1945.

Das,S. and Vikalo,H. (2015) SDhaP: haplotype assembly for diploids and polyploids via semi-definite programming. *BMC Genomics*, **16**, 260.

Etemadi,M. *et al.* (2018) Better ILP models for haplotype assembly. *BMC Bioinformatics*, **19**, 52–52.

Holtgrewe,M. (2010) Mason–a read simulator for second generation sequencing data. *Technical report*, FU Berlin.

Köster,J. and Rahmann,S. (2012) Snakemake: a scalable bioinformatics workflow engine. *Bioinformatics*, **28**, 2520–2522.

Lancia,G. *et al.* (2001) SNPs problems, complexity, and algorithms. In: Meyer auf der Heide,F. (ed.) *European Symposium on Algorithms*. Springer, pp. 182–193.

Lippert,R. *et al.* (2002) Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. *Brief. Bioinformatics*, **3**, 23–31.

Mazrouee,S. and Wang,W. (2018) Polycluster: minimum fragment disagreement clustering for polyploid phasing. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, doi: 10.1109/TCBB.2018.2858803.

Motazedi,E. *et al.* (2017) Exploiting next-generation sequencing to solve the haplotyping puzzle in polyploids: a simulation study. *Brief. Bioinform.*, **19**, 387–403.

N'Diaye,A. *et al.* (2017) Single marker and haplotype-based association analysis of semolina and pasta colour in elite durum wheat breeding lines using a high-density consensus map. *PLoS One*, **12**, e0170941.

Reinert,K. *et al*. (2017) The seqan c++ template library for efficient sequence analysis: a resource for programmers. *J. Biotechnol.,* **261**, 157–168.

Schwartz,R. *et al*. (2010) Theory and algorithms for the haplotype assembly problem. *Commun. Inform. Syst.,* **10**, 23–38.

Siragusa,E. *et al*. (2017) Linear time algorithms to construct populations fitting multiple constraint distributions at genomic scales. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, doi: 10.1109/TCBB.2017.2760879.

Szolek,A. *et al*. (2014) Optitype: precision HLA typing from next-generation sequencing data. *Bioinformatics*, **30**, 3310–3316.

Uitdewilligen,J.G. *et al*. (2013) A next-generation sequencing method for genotyping-by-sequencing of highly heterozygous autotetraploid potato. *PLoS One,* **8**, e62355.

Winston,W.L. *et al*. (2003) *Introduction to Mathematical Programming*. **Vol. 1**. Thomson/Brooks/Cole Duxbury, Pacific Grove, CA.

Xie,M. *et al*. (2016) H-PoP and H-PoPG: heuristic partitioning algorithms for single individual haplotyping of polyploids. *Bioinformatics,* **32**, 3735–3744.