Efficient and Accurate Clustering for Large-Scale Genetic Mapping

Veronika Strnadová[*,†], Aydın Buluç[†], Jarrod Chapman[♭], John R. Gilbert[*]
Joseph Gonzalez[‡], Stefanie Jegelka[‡], Daniel Rokhsar[♭,¶], Leonid Oliker[†]
[†]*Computational Research Division /* [♭]*Joint Genome Institute, Lawrence Berkeley National Laboratory, USA*
[‡]*EECS Dept /* [¶]*Molecular and Cell Biology, University of California, Berkeley, USA*
[*]*Computer Science Department, University of California, Santa Barbara, USA*

***Abstract*—High-throughput "next generation" genome sequencing technologies are producing a flood of inexpensive genetic information that is invaluable to genomics research. Sequences of millions of genetic markers are being produced, providing genomics researchers with the opportunity to construct high-resolution genetic maps for many complicated genomes. However, the current generation of genetic mapping tools were designed for the small data setting, and are now limited by the prohibitively slow clustering algorithms they employ in the genetic marker-clustering stage. In this work, we present a new approach to genetic mapping based on a fast clustering algorithm that exploits the geometry of the data. Our theoretical and empirical analysis shows that the algorithm can correctly recover linkage groups. Using synthetic and real-world data, including the grand-challenge wheat genome, we demonstrate that our approach can quickly process orders of magnitude more genetic markers than existing tools while retaining — and in some cases even improving — the quality of genetic marker clusters.**

I. Introduction

Genetic maps are essential for organizing DNA sequence information along chromosomes, and they enable diverse applications of genetics to problems in health, agriculture, and the study of biodiversity. Early genetic maps were constructed using only a few hundred *genetic markers* (chromosomal regions with two or more sequence variants in a population), and with such limited data, their construction was accordingly computationally inexpensive. With the advent of inexpensive high-throughput "next generation" sequencing [1], however, it is becoming a simple matter to generate data corresponding to millions of genetic markers across a genome. This flood of data rules out many standard, slow genetic mapping algorithms and poses a new challenge: to produce accurate high-density genetic maps in a computationally efficient manner.

A genetic map is a linear ordering of genetic markers that is consistent with observed patterns of inheritance in a population. An essential concept is the *linkage group* which collects together markers that are found on a single chromosome. Genetic maps are therefore organized into multiple linkage groups, with the number of groups equal to the number of chromosomes in the species. Within a linkage group, there is a natural measure of proximity which arises from the linear structure of chromosomes and the mechanics of their transmission from generation to generation.

Given a pair of markers in the same linkage group, we can estimate their proximity on the chromosome by comparing their sequence across a *mapping population* of related individuals. This estimate is made based on the LOD score, a logarithm of odds that two markers are genetically linked, based on the similarities and differences across each individual's genotype. The fundamental problem of genetic map construction is to take as input the sequences of n related individuals at m genetic markers, with low genotyping errors and often missing data (unknown genotypes of particular markers for particular individuals), and to organize these markers into linear chains that represent the structure of chromosomes.

The first step of genetic mapping involves clustering markers into linkage groups. This is traditionally performed by various standard clustering algorithms applied to a similarity graph of the markers. The similarity score between two markers can be a simple attribute comparison or a computationally intensive procedure, such as estimating the recombination rate of two genetic markers via nonlinear regression [2], [3]. On large datasets, computing the $O(m^2)$ pairwise similarities between all m markers quickly becomes prohibitive. In addition, the abundance of missing entries in genome sequencing data makes it challenging to translate the LOD score into a distance function that respects the triangle inequality, a requirement imposed on the data by many popular fast clustering algorithms.

To the best of our knowledge, no faster clustering method than single linkage has been successfully applied to the linkage group finding phase of genetic mapping, resulting in a troublesome gap between the amount of sequence data available for analysis and the amount that can be efficiently processed by current mapping tools. The ordering stage of the genetic mapping pipeline can be solved efficiently using heuristics such as a minimum-spanning tree finding algorithm [3]. Our initial efforts at benchmarking the popular genetic mapping tools JoinMap and MSTMap revealed that the clustering stage is indeed a severe bottleneck to the genetic mapping process.

In this paper, we propose a fast clustering algorithm that circumvents the computation of all similarities by exploiting prior knowledge about the specific structure of the marker data: linkage groups (i.e., chromosomes) have an intrinsically linear substructure that remains reflected in the similarity measure. After sorting, the algorithm creates a specific sketch that respects both the geometry and quality of the data. We show correctness of the algorithm under mild assumptions, and our empirical evaluation on synthetic and real-world data demonstrates its scalability and accuracy in practice.

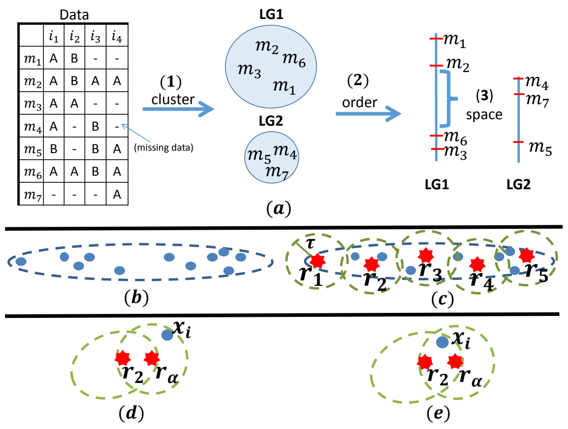

Fig. 1. **(a)** Genetic map construction pipeline: The markers are clustered into linkage groups (**LG**'s), ordered within each linkage group, and finally spaced according to their genetic distance. **(b)** The linear structure of markers within a linkage group; **(c)** representative points r_i are shown as red stars; **(d)**, **(e)** difference between a point x_i that is added as a new boundary point **(d)** and one that is not **(e)** based on $LOD(r_\alpha, r_2)$, $LOD(r_\alpha, x_i)$ & $LOD(x_i, r_2)$.

II. Problem Definition

Computational tools for genetic mapping follow three phases: (1) grouping markers into linkage groups (typically chromosomes), (2) ordering markers within chromosomes and (3) map distance estimation (Fig. 1(a)). Current software tools, such as the popular MSTMap [3] and JoinMap [4], typically fail to scale beyond tens of thousands of markers, especially when there is a high missing data rate. Our initial benchmarks revealed that a severe bottleneck is the pairwise similarity calculation step in the linkage group construction phase, and we therefore focus on this bottleneck here.

We attempt to solve the following problem: given m markers out of a population of n individuals, with a low genotyping error rate and a known missing data rate μ, cluster the markers into a (possibly unknown) number k of clusters. Each cluster represents one linkage group from the species whose sequence data is obtained from the mapping population. Formally, we are given m markers measured across n individuals in the mapping population and aim to find ordered, connected clusters (linkage groups) $C_1, ..., C_k$. The entries of a marker feature vector x_i are individual genotypes at marker sites. In this paper, we will explain genetic marker clustering in terms of homozygous genotypes. Hence, the n entries of a marker feature vector can take only two values A and B, or '$-$' for missing values.

Linkage groups are typically constructed by single linkage clustering based on LOD score similarities. The LOD score is the logarithm of a ratio of odds that two markers are genetically linked. A critical LOD score (*linklod* [4]) is estimated and serves as the cut-off threshold for constructing clusters from the single linkage dendrogram.

The LOD score for two markers x_i, x_j is $LOD(x_i, x_j) = \log_{10}\left(\frac{(1-\theta)^{NR}\theta^R}{0.5^{NR+R}}\right)$, where $\theta = \frac{R}{NR+R}$ is the recombination fraction, R is the number of recombinant individuals between the two markers, and NR is the number of non-recombinant individuals. Thus the LOD score is the logarithm of odds that two markers are genetically linked, under a null hypothesis of independent assortment. It is easy to show from the definition above, that the LOD score takes on a minimum value of 0 over the interval $0 \leq \theta \leq 1$ at $\theta = 1/2$ (where $R = NR$). The LOD score is symmetric about $\theta = 1/2$, taking on its maximum value of $(R + NR)\log_{10}(2)$ at $\theta = 0$ (where $R = 0$) and $\theta = 1$ (where $NR = 0$). Note that $R + NR$ is not necessarily equal to the number of individuals in the population, because the sequence data for either marker may be missing for a particular individual.

The LOD score is minimized at 0, and large positive values indicate that it is very unlikely that the markers happen to (either mismatch or) match in genotype for a large number of individuals by chance. Changes in population type (DH, RIL, F2, etc...) only affect the computation of R and NR in the LOD calculation. Thus our algorithm generalizes to more complex populations. In other words, heterozygosity will not change the fact that we depend on the LOD score to evaluate marker-marker similarity.

We point out that the fixed order of genetic markers along chromosomes is a key property of the data. Exploiting this linear, one-dimensional structure enables us to design a specialized procedure for finding linkage groups that is faster than generic clustering algorithms. We use the LOD score to quickly build representative sketches of the structure of each cluster. These sketches enable us to efficiently assign each marker to its proper linkage group.

III. The BubbleCluster Algorithm

Algorithm 1 clusters in three phases: (1) perform an initial clustering $\mathcal{C}$ using high-quality markers (lines 1–17); (2) assign low-quality markers to their most likely cluster $C \in \mathcal{C}$ (lines 18–22); and finally (3) merge unrealistically small clusters with large clusters (lines 23–25). This is a coarse-to-fine approach: it relies on a good clustering of reliable high-quality data points in Phase 1 as a skeleton to assign the low-quality points in Phase 2. Such a hierarchical approach relates to theoretically well-grounded clustering techniques like core sets [5], [6] or nearest neighbor clustering [7]. In Phase 3, we identify clusters which are too small to be considered true linkage groups. We attempt to merge all such clusters with the larger clusters from Phases 1 and 2.

Our algorithm takes four parameters as input: the threshold LOD score τ, the non-missing data threshold η, a cluster size threshold σ, and an odds difference threshold c. The selection and significance of τ and η will be explained in Section III-B. Briefly, τ represents the LOD score that a marker must achieve with at least one representative marker, or *sketch point*, r_j in order to join the cluster that contains r_j. Because missing data makes it impossible or very unlikely that markers with many missing entries will ever achieve a LOD score of τ, we use η to limit the number of missing entries allowed for markers included in Phase I of our algorithm. The σ input places a lower bound on the size of a cluster that the user expects could represent a true linkage group. The constant c determines whether the odds that a marker belongs to one particular cluster is much greater than the odds that it belongs to another cluster. For ease of reference, we provide a table of these and other variables that we refer to throughout the paper in Table I.

Backbone Clustering. The first, most important phase of the algorithm exploits the structure of genetic linkage groups

Algorithm 1: BubbleCluster Algorithm

```
   Inputs: $\mathcal{X} = \{x_1 \dots x_M\}$, $\tau, \eta, c, \sigma$
1  $\mathcal{C} \leftarrow \emptyset$; $\mathcal{R} \leftarrow \emptyset$;      // Lists of cluster and representative sets
2  sort $\mathcal{X}$ by increasing missing data;
3  $\mathcal{H} = \{x_i \in \mathcal{X} | \text{ nonmissing}(x_i) > \eta\}$ ;
4  if $|\mathcal{H}| == 0$ then return $\mathcal{C}, \mathcal{R}$;
5  for point $x \in \mathcal{H}$ in sequence do
6      if $\mathcal{R} = \emptyset$ then
7          define new cluster $C_\alpha$: $C_\alpha \leftarrow \{x\}$;
8          define new rep. set $R_\alpha$: $R_\alpha \leftarrow \{x\}$;
9          $\mathcal{C} \leftarrow \mathcal{C} \cup C_\alpha, \mathcal{R} \leftarrow \mathcal{R} \cup R_\alpha$;
10     else
11         $r_{\max} = \underset{r}{\text{argmax}}\ LOD(x, r)$ s.t. $r \in R_\alpha \in \mathcal{R}$;
12         if $LOD(x, r_{\max}) > \tau$ then
13             $r_{\max_2} = \underset{r \notin R_\alpha}{\text{argmax}}\ LOD(x, r)$ ;
14             assign $x$ to cluster $C_\alpha$: $C_\alpha \leftarrow C_\alpha \cup \{x\}$;
15             if isBdryPt $(x, R_\alpha)$ then add $x$ to the correct end of $R_\alpha$ ;
16             if $LOD(x, r_{\max_2}) > \tau$ then MergeRs $(R_\alpha, R_\beta)$; MergeCs $(C_\alpha, C_\beta)$ ;
17         else set up new cluster:
           $\mathcal{C} \leftarrow \mathcal{C} \cup \{x\}, \mathcal{R} \leftarrow \mathcal{R} \cup \{x\}$ ;
18 for $y \in \mathcal{X} \setminus \mathcal{H}$ do
19     $r_{\max} = \underset{r}{\text{argmax}}\ LOD(y, r)$ s.t. $r \in R_\alpha \in \mathcal{R}$;
       $r_{\max_2} = \underset{r \notin R_\alpha}{\text{argmax}}\ LOD(y, r)$ ;
20     $l_{\max_2} = LOD(y, r_{\max_2})$; $l_{\max} = LOD(y, r_{\max})$;
21     if $(l_{\max} - l_{\max_2}) > c$ then $C_\alpha \leftarrow C_\alpha \cup \{y\}$ ;
22     else set up new cluster:
       $\mathcal{C} \leftarrow \mathcal{C} \cup \{y\}, \mathcal{R} \leftarrow \mathcal{R} \cup \{y\}$ ;
23 for all $C$ in $\mathcal{C}$ s.t. $|C| < \sigma$ do
24     pick a $c_j \in C$;
25     if $LOD(c_j, r_k) > \tau$ for any $r_k$ in any $R_\alpha \in \mathcal{R}$ then
       MergeCs $(C, C_\alpha)$ ;
```

τ	LOD threshold
η	non-missing data threshold (applied to marker vector entries)
σ	cluster size threshold
c	odds difference threshold
μ	missing data rate in the dataset
n	population size (number of individuals in the mapping population)
k	number of clusters
ϵ	error tolerance for misassignment of markers to clusters
n_{nm}	number of non-missing entries in a particular marker vector
r	number of representative points aka sketch points
b	number of bins, i.e. unique locations on the genetic map, always $O(nk)$
$\mathcal{H}$	high-quality set, defined as the set of markers with $n_{\text{nm}} > \eta$

TABLE I. LIST OF PARAMETERS/VARIABLES USED

for quickly ordering genetic markers. In Phase 1, we only process high-quality markers, that is markers with at least η non-missing entries. The algorithm establishes clusters on the fly: each incoming point is either close to, and hence assigned, to an existing cluster, or it creates a new cluster. Two clusters are merged if they are "close" in genetic distance, i.e. if points on the boundary of the clusters obtain a LOD score greater than a given threshold τ.

To avoid storing and comparing distances between a new point and all previous points, we only keep a representative sketch R_α for each cluster C_α. To create and maintain sketches, we exploit the special linear structure of the data, illustrated in Fig. 1(**b**). The resulting sketch is therefore an *ordered* list of representative points (Fig. 1(**c**)) where for every point in C_α, there is a point $r_\alpha(x)$ in R_α with $LOD(x, r_\alpha(x)) > \tau$.

For each incoming point x, we find the closest sketch point $r_{\max}$. If $LOD(x, r_{\max}) > \tau$, then x is assigned to the cluster of $r_{\max}$. Otherwise, it sprouts a new cluster (line 17). If x is added to an existing cluster, we check whether it is well represented by the current sketch, or whether we need to augment R_α. Here, we use the linearity assumption. If x is outside of the boundaries specified by R_α (the `isBdryPt()` function, line 15), we add x as a new (boundary) sketch point. If $r_{\max}$ is the only sketch point, x becomes a new sketch point automatically. If not, we compare the LOD score between x and the point $r_2 \in R_\alpha$ immediately next to r_α in the ordered list R_α, and the LOD score between r_α and r_2 as illustrated in Figures 1(**d**) and 1(**e**): if $LOD(x, r_2) < LOD(r_2, r_\alpha)$ and $LOD(x, r_2) < LOD(x, r_\alpha)$, then x extends the boundary.

When x becomes a new sketch point, it extends C_α in the linear dimension along which we assume the sketch points to lie. It succeeds r_α and becomes a new end of R_α. Finally, we determine whether x connects two clusters (line 16) by finding the nearest sketch point $r_{\max_2}$ that is *not* in the cluster to which x was assigned. If $LOD(x, r_{\max_2}) > \tau$, then x forms a bridge $r_{\max}, x, r_{\max_2}$ between the two clusters. When merging clusters, we also merge their sketches R_α and R_β. To do so, we compare the four boundary points of R_α and R_β and append R_β to the end of R_α in the order which maintains the greatest LOD score between boundary two points, one from either cluster.

Low quality marker assignment. At the completion of Phase 1, we have an initial clustering $\mathcal{C}$ of all the high-quality data points $x \in \mathcal{H}$, along with their ordered sketches. In Phase 2 (lines 18–22), we rely on the sketches to assign the remaining low quality markers $y \in \mathcal{X} \setminus \mathcal{H}$ to one of the existing clusters. We use a simple heuristic: for each low-quality marker, we find the difference between $l_{\max_2} = LOD(y, r_{\max_2})$ and $l_{\max} = LOD(y, r_{\max})$, where $r_{\max}$ and $r_{\max_2}$ are defined as above. If this difference is greater than a threshold c, then we add y to the cluster C_α containing $r_{\max}$. Otherwise, we simply create a new, temporary *singleton* cluster containing only the point y. This choice of difference threshold means that the odds that y belongs to cluster C_α should be by a factor of 10^c greater than the odds that y belongs to any other cluster. Moreover, we only assign points to existing clusters for which we have high confidence in our assignment.

Merging small clusters with large clusters. At this stage, we rely on further assumptions about the underlying structure of our clusters, based on the following prior knowledge of true linkage groups: we know that each marker comes from exactly one linkage group, and that these groups tend to be relatively large. We attempt to merge all clusters C with $|C|$ smaller than a user-specified σ with larger clusters, by picking a random point within each small cluster and comparing its distance to all the sketch points r in large clusters. If this point is found to lie within the threshold distance of a sketch point, then we merge the small cluster with the large cluster. We can estimate σ based on the number of markers and the number of expected linkage groups – σ is the largest cluster size that the user would consider too small to be a true linkage group.

Running time.: The BubbleCluster algorithm runs in time $O(m\log(m) + mr)$ for m markers and r sketch points. If we have chosen a threshold less than $LOD_{\max}$, which is the maximum achievable LOD score between any two markers in our dataset (and is bounded by $n\log_{10} 2$) then the number of sketch points is bounded by the number of uniquely identifiable locations in the genome, which we refer to as *bins*[1] and whose number we denote with b. For a fixed number k of chromosomes in the organism and n individuals in the mapping population, the number of bins is proportional to n and k: $b = O(nk)$ [8]. Thus $r = O(kn)$. For the organisms of interest in our work, under typical experimental conditions, the number of bins is in the thousands. In our experiments, the number of sketch points never exceeded 7% of the linkage group size, and was in fact much lower than nk.

A. Correctness of the Algorithm

We make the following assumptions on the true underlying linkage groups $C_1^*, \ldots C_K^*$ that are roughly reasonable for real data.

- A1. Separation: there exists a $\lambda_{\text{sep}} > 0$ such that for any C_α^* and any two points $x \in C_\alpha^*$, $y \notin C_\alpha^*$, it holds that $LOD(x, y) < \lambda_{\text{sep}}$.
- A2. Connectedness: there exists a constant λ_{conn} with $0 < \lambda_{\text{sep}} < \lambda_{\text{conn}}$ such that for every C_α^* and each $x_1, x_2 \in C_\alpha^*$, there is a *path* of points $y_1, \ldots y_m \in C_\alpha^* \cap \mathcal{H}$ with $LOD(x_1, y_1) > \lambda_{\text{conn}}$, $LOD(y_m, x_2) > \lambda_{\text{conn}}$ and $LOD(y_j, y_{j+1}) > \lambda_{\text{conn}}$ for all $1 \leq j \leq m$.
- A3. Local linear ordering: If for three points $x_1, x_2, x_3 \in C_\alpha^*$, $LOD(x_1, x_2) > \lambda_{\text{conn}} - \delta$ and $LOD(x_2, x_3) > \lambda_{\text{conn}} - \delta$ for a $\delta > 0$, then the true order of these points is x_1, x_2, x_3 if and only if $LOD(x_1, x_3) < \min(LOD(x_1, x_2), LOD(x_2, x_3))$[2].

Lemma 3.1: If $\lambda_{\text{conn}} > \tau \geq \lambda_{\text{conn}} - \delta > \lambda_{\text{sep}}$ and if A1-A3 hold, then the algorithm identifies the correct clusters for all points in $\mathcal{H}$ within one pass over the sorted data.

Proof: First, the following invariant holds throughout and after Phase 1: any existing cluster C'_α is a subset of a true cluster, i.e., $C'_\alpha \subseteq C^*_\beta$ for some β. When a cluster is created, it consists of one point and therefore certainly is contained in a single true cluster. If a new point x gets added to C'_α, that point is within a LOD score of $\tau > \lambda_{\text{sep}}$ of $r_{\max} \in C^*_\beta$, and hence by A1, x and $r_{\max}$ must be in the same true cluster. Two clusters are merged only if there is a path $(r_{\max}, x, r_{\max_2})$ between them with a LOD score of at least τ at each hop. By A1, these clusters must therefore belong to the same true cluster.

Second, we see that if $C'_\alpha \subseteq C^*_\gamma$ and $C'_\beta \subseteq C^*_\gamma$, then $\alpha = \beta$, i.e., no true cluster is split: If C^*_γ was split, then, by A2, there would be points $y_\alpha \in C'_\alpha$ and $y_\beta \in C'_\beta$ with $LOD(y_\alpha, y_\beta) > \lambda_{\text{conn}}$. Let r_α be the point that y_α was assigned to, and r_β the point that y_β was assigned to. Then $LOD(r_\alpha, y_\alpha) > \tau$ and $LOD(r_\beta, y_\beta) > \tau$. Without loss of generality, let us assume that y_β was encountered after y_α by the algorithm, and that y_α is the closest point (highest LOD score) to y_β in C_α. Then, y_α must be a boundary point when it is added to its cluster. To see this, consider 2 cases:
(i) r_α is the only sketch point in its cluster at the time y_α is seen. In this case, y_α automatically becomes a new representative point.
(ii) There are other sketch points in the cluster of r_α. By the way we merge clusters, there must be at least one sketch point r'_α with $LOD(r'_\alpha, r_\alpha) > \tau$. Since $LOD(y_\alpha, r_\alpha) > LOD(y_\alpha, r'_\alpha), LOD(y_\alpha, r_\alpha) > \tau$, and $LOD(r'_\alpha, r_\alpha) > \tau$, then by A3 y_α will be made a boundary point when it is encountered.
Since y_α is a boundary point, then when y_β is encountered, its LOD score will be highest with the sketch points r_α and y_β, with both of these LOD scores above τ. Hence, C'_α and C'_β will be merged.

■

In the presence of high missing data rates, the algorithm still provably achieves perfect precision but not perfect recall.

B. Parameter selection

In theory and in practice, the LOD threshold τ and non-missing data threshold η will affect both the running time and the accuracy of our algorithm. In this section, we show that τ and η are interdependent, and we explain how we choose τ and η given a population size n and a missing data rate μ. The effect of c and σ is easily explained and will be addressed at the end of the section.

Recall that a marker must achieve a LOD score above τ with a representative point in order to join that representative point's cluster, and that η limits the number of missing entries a marker vector can have in order to be included in the high-quality marker set $\mathcal{H}$. Our choices of τ and η were made to maximize the probability that each marker will be assigned to the correct cluster (set the LOD threshold τ high enough), but to also include enough points in $\mathcal{H}$ to build a reliable sketch of each cluster (set the nonmissing threshold η low enough).

Suppose that the number of observed entries n_{nm} in a marker x_i is less than the number of observed entries in another marker x_j. As the number of nonrecombinant individuals NR in this pair of markers (x_i, x_j) approaches n_{nm}, the maximum achievable LOD score for this pair approaches $n_{\text{nm}}\log_{10} 2$ (by the definition of the LOD score in Section II):

$$\lim_{NR\to n_{\text{nm}}} \log_{10}\left(\frac{\left(1 - \frac{R}{R+NR}\right)^{NR}\left(\frac{R}{R+NR}\right)^{R}}{0.5^{R+NR}}\right) = n_{\text{nm}}\log_{10} 2$$

Thus, the maximum achievable LOD score for a marker is dependent upon the number of nonmissing entries in that marker. For a LOD threshold τ, if $n_{\text{nm}} \leq \tau/\log_{10} 2$ in x_i, then line 12 will evaluate to `false` for any choice of $r_{\max}$. We want to set η high enough to prevent an overabundance of clusters from sprouting, which would necessarily raise the number of representative points and hurt efficiency – thus η should be at least $\tau/\log_{10} 2$.

[1]Many markers may map to the same location on a chromosome.

[2]This assumption is supported by the fact that the recombination fraction between markers very close together on the chromosome is a reliable estimate of genetic distance [9],[10]

We can be even more aggressive in limiting missing entries in the high-quality set, however. Given a missing rate μ, and a marker x_i with n_{nm} non-missing entries, we expect the number of non-missing entries x_i shares with any other marker will be: $E\left[\text{shared nonmissing entries}(x_i)\right] = (1-\mu)n_{\text{nm}}$. Thus if

$$(1-\mu)n_{\text{nm}} > \tau/\log_{10} 2 \Rightarrow n_{\text{nm}} > \tau/(1-\mu)\log_{10} 2$$

then we expect x_i to achieve a LOD score greater than the threshold τ with at least one other marker x_k in the high-quality set. If the structure of each cluster is indeed approximately linear, then we expect that the sketch point r nearest x_k will also achieve a high LOD with x_i, allowing x_i to be placed in the appropriate cluster. If, on the other hand, $n_{\text{nm}} \leq \tau/(1-\mu)\log_{10} 2$, we can choose to eliminate x_i from $\mathcal{H}$ because it is unlikely that x_i will score higher than τ with any sketch point.

Here the interplay between the efficiency and accuracy of our algorithm becomes apparent. For a high missing data rate, the gap between λ_{conn} and λ_{sep} may be small, and τ must be set very high. If τ is greater than λ_{sep}, we can guarantee perfect precision, but we may eliminate so many markers from the high-quality marker set that we will not have enough markers to guarantee coverage of all clusters, resulting in low recall. Therefore, with a high missing data rate we seek to minimize the probability that we assign a marker to the wrong cluster, allowing τ to be less than λ_{sep}. Let p represent this probability:

$$p = P(LOD(x_i, x_j) > \tau | x_i \in C_i, x_j \in C_j, i \neq j)$$

By the definition of the LOD score, $p \leq 1/10^{\tau}$. Let n_{comp} be the number of LOD comparisons that we make in line 12 of our algorithm. The probability that we make no mistakes in assigning a marker to its cluster is then:

$$P(\text{no mistakes}) = (1-p)^{n_{\text{comp}}}$$

Therefore, to ensure that $P(\text{no mistakes}) > 1-\epsilon$ for $\epsilon > 0$, we need:

$$\left(1 - \frac{1}{10^{\tau}}\right)^{n_{\text{comp}}} > 1 - \epsilon$$

$$\Rightarrow \tau > \log_{10}\left(\frac{1}{1-(1-\epsilon)^{1/n_{\text{comp}}}}\right)$$

Recall that *bins* are uniquely identifiable locations on the genetic map, and their number b is $O(kn)$ for k linkage groups and n individuals in the mapping population. An upper bound on n_{comp} is thus $\binom{b}{2}$, corresponding to the grossly pessimistic situation where every bin is represented by a representative point, and the LOD score must therefore be evaluated once for every bin pair. Although this is a worst-case scenario, we can use this upper bound to estimate τ given an $\epsilon > 0$.

Our selection of τ and η is thus a balancing act, where we want to ensure that τ is high enough to guarantee a high $P(\text{no mistakes})$, but at the same time is not so high that a large fraction of our data would be excluded from Phase 1, which we rely on to build a sketch of each cluster. For example, if we are given a population size of $n = 300$, we expect $k = 10$ linkage groups, and we want to achieve $P(\text{no mistakes}) > 0.99$, then $\tau > \log_{10}\left(1/1 - 0.99^{1/\binom{3000}{2}}\right) = 8.6509$ would achieve perfect precision with 99% confidence. Given a missing data rate $\mu = 35\%$, we require $n_{\text{nm}} > 8.6509/(0.65)\log_{10} 2 = 44.2117$, so we would set η to 44. If $\mu = 65\%$, then by the same calculation we would require $\eta \geq 82.1076$. If this choice of η excludes too many markers from $\mathcal{H}$, we might choose to either raise ϵ or allow less-than-perfect precision (achieve a low $P(\text{number of mistakes} > \text{constant})$) in exchange for greater coverage of the linkage groups by the markers in $\mathcal{H}$. In practice, we achieve extremely high precision as well as recall using these crude estimates.

Although c and σ also influence the resulting cluster quality, their effect is quite obvious. Higher c values prevent markers with many missing entries to be assigned to any cluster, resulting in more small clusters in the output. These small clusters can be left out of the final genetic map or assigned to larger clusters by more careful analysis by the user. Similarly, high values of σ can cause unnecessary comparisons to be made between large clusters that already represent linkage groups, while extremely small values may miss the opportunity to merge small clusters with larger clusters.

C. Related Work

Several computational tools exist for the construction of genetic linkage maps, as explained in the survey by Cheema and Dicks [8]. Since then, OneMap [11] and Lep-MAP [12] have also been proposed. All these tools, without exception, perform all-pairs comparisons among markers, making them unsuitable for datasets with millions of markers. Structural clustering methods that have been applied to genetic mapping include connected components [13], [14], [3] and single linkage clustering [15]. Differently from single linkage, we construct and merge clusters on the fly, requiring only one pass through the data after sorting.

General compressed representations for clustering problems have been addressed by core sets [5], [6], and by hierarchical re-clustering ideas for streaming and distributed clustering [16], [17]. As opposed to general sampling techniques, we extract a problem-specific representative core set deterministically within one pass, and exploit the specific structure of the marker data.

Our algorithm maintains an ordering of the dataset that is similar in spirit to the OPTICS [18], DBSCAN [19], or BIRCH [20] algorithms. However, our algorithm is not density based. Applying density-based approaches to genetic marker data would be difficult if not impossible, due to the lack of a distance metric with which to compute inter-marker distances. The challenge is converting the LOD similarity score into a valid distance metric which respects the triangle inequality. We cannot use density-based approaches which rely on the notion of an "$\epsilon-$neighborhood" around data points in order to find a dense region of the space in which the data lie.

Our algorithm uses several representative points to provide an accurate coverage of the underlying cluster. In that sense, our approach is closest to the CURE algorithm [21], which also maintains representative points. The specific insight we draw from the genetic mapping problem enables our algorithm to maintain a better performance bound than CURE's $O(m^2 \log m)$ bound, and allows us to prove correctness with mild assumptions.

IV. Empirical Evaluation

We compare our algorithm to two popular genetic mapping tools: JoinMap and MSTMap. We also provide a comparison with PIC, a spectral clustering approach [22]. Most of the experiments were run on Neumann, a quad-core server with AMD Opteron 8378 Processors running at 2.4GHz. Because JoinMap requires the Windows OS, experiments with JoinMap were performed on a Windows desktop with Intel 2.93GHz Core 2 Duo processors. Our code was written in C++ and compiled with gcc 4.4.7. All experiments are single threaded and use a single core.

A. Data

We evaluate BubbleCluster on both real and synthetic datasets. The first dataset, *barley*, consists of 65,357 genetic markers from a population of 90 individuals with 20% missing data. This species of barley has 7 true linkage groups. The second, larger *switchgrass* dataset of 548,281 genetic markers comes from a population of 500 individuals (with some replicated individuals for better coverage), with 65% missing data and 18 true linkage groups. Due to its size, previous clustering efforts on this data focused only on the 113,326 highest-quality markers. We cluster both the 113K subset of markers and the complete 548K dataset in our experiments to demonstrate the scalability of our algorithm. Our third timing result on real data is for the grand-challenge *wheat* genome, containing 1,582,606 markers from a population of 88 individuals and 21 true linkage groups with 39% missing data entries. We do not report accuracy results on wheat because single-linkage clustering failed to provide a golden standard result to compare to after a week of computation given all our resources.

For scaling studies, we rely primarily on synthetic data generated by the Spaghetti software, which simulates genetic marker data with real-life complications [23]. In particular, we created datasets for a range of missing data rates, from 0 to 65%. We varied the number of markers from 12.5K to 400K, doubling the size at each increment. The population size was fixed at 300, the sequencing error rate at 0.1%, and the number of linkage groups at 10 in all experiments.

B. Evaluation Metric

We use the overall F-score to evaluate the quality of each clustering. The F-score ranges from 0 (no correspondence) to 1 (perfect match), and evaluates a test cluster C_i with respect to a "golden standard" cluster G_j in terms of precision and recall [24]. Formally, if $G_j \in \mathcal{G}$ is a golden standard cluster, then the F-score for a test cluster C_i with respect to G_j is defined as: $F(G_j, C_i) = \frac{2p_{ij}r_{ij}}{r_{ij}+p_{ij}}$, for recall $r_{ij} = \frac{|C_i \cap G_j|}{|G_j|}$ and precision $p_{ij} = \frac{|C_i \cap G_j|}{|C_i|}$. The *overall* F-score is a normalized, weighted sum of the F-scores for each golden standard cluster $G_j \in \mathcal{G}$: $F(\mathcal{G}, \mathcal{C}) = \frac{1}{m}\sum_{j=1}^{k} |G_j| \max_{i=1\ldots l} F(G_j, C_i)$, where k is the number of true clusters, l is the number of test clusters, and m is the total number of datapoints. "True" clusters are generated directly in simulated data experiments; for real data, if assumption A1 holds, then single linkage clustering will provably find the correct clusters given a threshold $\tau > \lambda_{\text{sep}}$. We thus rely on the outcome of single linkage clustering to measure accuracy on real data.

Dataset	Markers	BubbleCluster	
		F-score	Time
Barley	64K	0.9993	15 sec
S-grass	113K	0.9745	8.9 min
S-grass	548K	0.9894	1.9 hrs
Wheat	1.582M	N/A	1.22 hrs

TABLE II. Clustering performance on Barley, Switchgrass, and Wheat from the Joint Genome Institute using BubbleCluster. MSTmap and JoinMAP are unable to cluster data sets at this scale.

Clustering	12.5K Markers		25K Markers	
	F-Score	Time	F-Score	Time
JoinMAP	0.9996	14 min	0.9998	46 min
MSTMap	0.9996	4.5 min	0.9998	20 min
PIC	0.4702	11 sec +(4 min)	0.6078	44 sec +(16.5 min)
Bubble	**0.9996**	**6 sec**	**0.9998**	**15 sec**

TABLE III. Performance comparison of clustering algorithms using synthetic Spaghetti with 35% missing data and 0.1% error rate. All experiments ran on Neumann (Linux/AMD), except JoinMap ran on the Windows machine. The parenthesized PIC algorithm number is preprocessing time for pairwise calculations.

C. Results

Table II summarizes the running time and F-score on the real data sets. A LOD threshold of $\tau = 8$ with nonmissing data threshold $\eta = 66$ was used to cluster the 65K barley dataset; for switchgrass, we used thresholds $\tau = 20$ and $\eta = 132$; for wheat, we fixed $\tau = 9$ and $\eta = 44$. The c parameter was set to 5, and σ to 100, in all experiments. Our selection of τ and η was based on two objectives, as explained in Section III-B: (1) maximize the probability that each marker is assigned to the correct cluster, but at the same time (2) ensure that Phase 1 contains sufficiently many markers to build a reliable sketch of each cluster. No mapping tool we know of, including the popular MSTMap or JoinMap, has been successful in clustering genetic marker datasets at this scale.

Table II demonstrates that we achieve very accurate clusters in $O(m \log m)$ time for m markers, a significant improvement over the $O(m^2)$ algorithms used by other genetic marker clustering tools. We emphasize that these datasets come from real-world sequence data, where missing data entries do not have a simple known distribution. Nonetheless, our algorithm recovers the linkage groups with both precision and recall above 97%. We omit comparisons with MSTMap on all but the smallest dataset of 64K markers, where it took MSTMap almost a *week* to ultimately place all the markers in the same cluster. The clustering accuracy of the recently sequenced wheat genome, with 1.582 million markers, has been independently validated in a recent study [25]. In summary, our algorithm achieves very high accuracy at fast running times.

Table III underscores our ability to outperform existing genetic clustering methods as well as more general clustering methods applied to synthetic genetic marker data. Even on relatively small datasets, where JoinMap and MSTMap complete the clustering stage in a reasonable amount of time, we achieve identical F-scores as those methods, but within a fraction of the time. In fact, the results are slightly biased in favor of JoinMap. Although this tool will automatically construct the single-linkage dendrogram from an input data matrix, the user must select which dendrogram edges to cut in order to produce the final clusters. We selected the edges that resulted

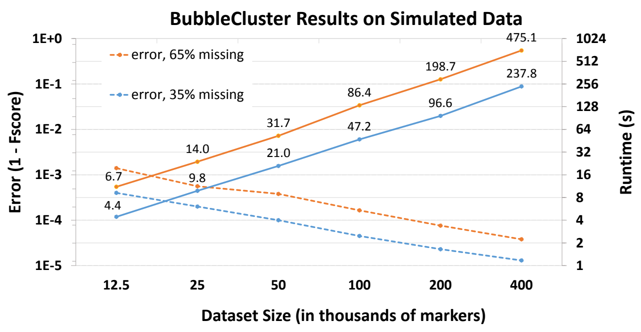

Fig. 2. Average errors ($1 - F$-score, left axis) and runtimes (right axis) for increasing dataset sizes.

in the best clustering based on our prior knowledge of the simulated linkage groups. The time reported is only the time for generating the JoinMap dendrogram.

The best run out of several re-runs with different counts of pseudo-eigenvectors for the PIC algorithm is reported in Table III. The clustering was performed using k-means++ on the data projected onto the two-dimensional space spanned by two pseudo-eigenvectors. This procedure performed empirically the best. We include the running time of PIC given the similarity matrix as input, and indicate the $O(m^2)$ time required to construct this matrix in parentheses. The inability of PIC to produce competitive results in terms of clustering quality motivates the need for a more application-specific approach in this domain. We point out the drawbacks in applying general clustering methods to a problem with missing data, a similarity function that cannot be expressed as an inner product, and an underlying structure that can only be exploited if the application-specific problem is well understood.

Our ability to scale, while simultaneously making use of more data availability, is demonstrated in Fig. 2. Here, we increase the size of our synthetic dataset from 12.5K to 400K genetic markers. We report the clustering quality for both 35% and 65% missing data in terms of the errors we make; the error is reported in terms of ($1 - F$-score) for each (dataset size, missing data rate) pair (left-hand axis of Fig. 2). The running times for the same data are plotted with respect to the right-hand axis of Fig. 2. Both the running time and errors are averages of five trials on each dataset size. We make two points about these empirical results: (1) the error we make in clustering decreases linearly, in almost exact correlation with the size of the data, and (2) the running time increases with $O(m \log m)$, promising reliable performance up to almost half a million markers, even with an enormous amount of missing data. Comparing Table II with these results, we believe the behavior of our algorithm in these experiments is predictive of its performance in the real world.

Lastly, we make a note on the impact of our choices of thresholds on the cluster quality and the running time. Tables IV and V capture the behavior of BubbleCluster on a 200K simulated dataset with 300 individuals with fixed missing data rates of 65% and 35%, respectively. In the top half of Table IV, we fix η at 66 and vary τ. With this choice of η, any marker that can achieve a maximum LOD score of at least 20 will be included in the high-quality set $\mathcal{H}$. The running time

200K Markers, 300 individuals, 65% missing, $\eta = 66$

τ	5	6	7	8	9	10
F-Score	0.1894	0.5240	0.9261	0.9999	0.9998	0.9988
Time (s)	82.5	124	155	183	242	307

200K Markers, 300 individuals, 65% missing, $\tau = 8$

η (self-lod)	66 (20)	83 (25)	99 (30)	116 (35)	132 (40)	149 (45)
F-Score	0.9999	0.9982	0.9004	0.6169	0.4986	N/A
Time (s)	183	190	180	101	42.3	N/A
# markers with $n_{\text{nm}} > \eta$	200,000	199,205	148,914	16,551	99	0

TABLE IV. F-SCORES & RUNNING TIMES FOR VARYING CHOICES OF η & τ ON A 200K DATASET WITH 65% MISSING DATA.

200K Markers, 300 individuals, 35% missing, $\eta = 132$

τ	5	10	15	20	25	30
F-Score	0.6225	0.9999	0.9999	0.9999	0.9999	0.9999
Time (s)	48.6	67.0	70.9	82.0	106	170

200K Markers, 300 individuals, 35% missing, $\tau = 20$

η (self-lod)	132 (40)	166 (50)	172 (52)	179 (54)	186 (56)	192 (58)
F-Score	0.9999	0.9999	0.9992	0.9930	0.9610	0.8948
Time (s)	82.0	84.6	82.7	83.0	81.7	82.0
# markers with $n_{\text{nm}} > \eta$	200,000	199,920	199,296	193,701	169,648	124,263

TABLE V. F-SCORES & RUNNING TIMES FOR VARYING CHOICES OF η & τ ON A 200K DATASET WITH 35% MISSING DATA.

increases with increasing τ, as expected; a higher τ will cause more clusters and sketch points to be created, increasing the number of LOD comparisons the algorithm needs to make. The F-score also increases up to $\tau = 8$, then drops off slightly for increasing values of τ. This is due to Phase III of our algorithm – at $\tau > 8$, small clusters are created that cannot be merged with any large existing cluster because no marker within the small clusters achieves a high enough LOD score with any of the large clusters' sketch points.

In the bottom half of Table IV, we reverse the roles of η and τ. Here, τ remains fixed at 8 and we increase η from 66 (22% observed entries) to 149 (49.67% observed). In parentheses we show what we call the *self-lod* of a marker for the given value of η, which is the LOD score a marker would achieve with itself if it had $n - \eta$ missing entries, i.e. the maximum achievable LOD score for a marker with η observed entries. We see that at $\eta = 66$, every marker is included in the high-quality set $\mathcal{H}$, and the F-score is very high. As we increase η, more markers are excluded from $\mathcal{H}$ and the F-score suffers. Note that at $\eta = 99$, more than 50K markers are excluded from $\mathcal{H}$, resulting in poorer coverage of the clusters with sketch points and a greater chance that low-quality markers will not achieve a high LOD score with any sketch point. These low-quality markers are left out of large clusters, decreasing the recall in the $F-$score. As we increase η to exclude a majority of the markers, the F-score drops off dramatically. These results show that while we attempt to eliminate very "low-quality" markers from our dataset, a high enough LOD threshold allows us to include many markers with high amounts of missing data in our "high-quality" set $\mathcal{H}$, producing very accurate clusters.

Table V, with analogous results to Table IV for 35% missing data, tells a similar story. The running time increases with increasing τ and fixed η. However, here we see a much wider range of τ values will give accurate results very quickly. We can afford to set τ to a much higher value than in the case of 65% missing data, allowing η to be low enough to include all the markers in our dataset in $\mathcal{H}$ for higher F-scores.

V. Discussion

Current approaches to genetic mapping were designed for a small data setting, and use algorithms that scale quadratically in the number of markers. We propose an approach that exploits the underlying linear structure of chromosomes to avoid expensive comparisons between (quadratically) many pairs of markers. The resulting linkage groups (i.e., marker clusters) are highly concordant with computationally expensive quadratic calculations, but our improved scaling allows far denser maps to be constructed with minimal computation.

After the formation of linkage groups, the next step in constructing a high quality genetic map is inferring the detailed ordering of markers along chromosomes. Since our method takes into account the linear structure of chromosomes from the start, the result is an approximate marker ordering that is an excellent starting point for detailed marker order by simulated annealing or other methods that explore short-range perturbations of our approximate ordering. In fact, the sketch point order found by our algorithm for the barley dataset (Sec. IV-A), was highly correlated with the true marker order in barley linkage groups: for 6 out of 7 groups, the Spearman Rank Correlation Coefficient ρ was above 0.9. In simulated data experiments, we also found a high concordance between sketch point order and the simulated map order with a high ρ in most cases and a perfect order in many examples with 35% missing data. We are currently working on an efficient ordering algorithm that can use the results of BubbleCluster to infer missing data and to quickly order markers.

Our algorithm can significantly speed up current genetic mapping efforts on large datasets. Though the ordering phase of genetic mapping has been shown to be NP-hard, efficient heuristic algorithms have been proposed to quickly order markers within a linkage group [3]. BubbleCluster eliminates the bottleneck in current genetic mapping tools in the big data setting. An important application of our method is in the efficient construction of ultra-dense genetic maps for large and complex genomes that are filled with repetitive sequences that frustrate genome assembly but do not limit the number of genetic markers. The most economically important of these genomes are various grasses, including crops grown for food (e.g., barley and wheat, whose genome sizes are two- to seven-fold larger than the human genome) or as biofuel feedstocks (e.g, switchgrass and miscanthus, polyploids that contain multiple, subtly different copies of a basic genome).

Acknowledgment

The authors thank Nicholas Tinker for providing us with his Spaghetti software. The work conducted by the Lawrence Berkeley National Laboratory and the U.S. DOE Joint Genome Institute is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This research is supported in part by NSF CISE Expeditions award CCF-1139158 and DARPA XData Award FA8750-12-2-0331, and gifts from Amazon Web Services, Google, SAP, Cisco, Clearstory Data, Cloudera, Ericsson, Facebook, FitWave, General Electric, Hortonworks, Huawei, Intel, Microsoft, NetApp, Oracle, Samsung, Splunk, VMware, WANdisco and Yahoo!.

References

[1] ML Metzker. Sequencing technologies – the next generation. *Nature Reviews Genetics*, 11(1):31–46, 2009.

[2] M. V Rockman and L. Kruglyak. Recombinational landscape and population genomics of caenorhabditis elegans. *PLoS Genetics*, 5(3):e1000419, 2009.

[3] Y. Wu, P.R. Bhat, T.J. Close, and S. Lonardi. Efficient and accurate construction of genetic linkage maps from the minimum spanning tree of a graph. *PLoS Genet.*, 4(10), 2008.

[4] P. Stam. Construction of integrated genetic linkage maps by means of a new computer package: Join map. *The Plant Journal*, 3(5):739–744, 1993.

[5] M. Bādoiu, S. Har-Peled, and P. Indyk. Approximate clustering via core-sets. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 250–257, New York, NY, USA, 2002. ACM.

[6] D. Feldman and M. Langberg. A unified framework for approximating and clustering data. In *STOC*, 2011.

[7] U. von Luxburg, S. Bubeck, S. Jegelka, and M. Kaufmann. Consistent minimization of clustering objective functions. In *NIPS*, 2007.

[8] J. Cheema and J. Dicks. Computational approaches and software tools for genetic linkage map estimation in plants. *Briefings in Bioinformatics*, 10(6):595–608, 2009.

[9] J. B. S. Haldane. The combination of linkage values and the calculation of distances between the loci of linked factors. *J Genet*, 8(29):299–309, 1919.

[10] DD Kosambi. The estimation of map distances from recombination values. *Annals of Eugenics*, 12(1):172–175, 1943.

[11] GRA Margarido, AP Souza, and AAF Garcia. Onemap: software for genetic mapping in outcrossing species. *Hereditas*, 144(3):78–79, 2007.

[12] P. Rastas, L. Paulin, I. Hanski, R. Lehtonen, and P. Auvinen. Lep-MAP: fast and accurate linkage map construction for large SNP datasets. *Bioinformatics*, page advance access, 2013.

[13] B.N. Jackson, P.S. Schnable, and S. Aluru. Consensus genetic maps as median orders from inconsistent sources. *IEEE Trans. on Comp. Biology and Bioinformatics*, 5(2), 2008.

[14] A. Kozik and R. Michelmore. MadMapper and CheckMatrix – python scripts to infer orders of genetic markers and for visualization and validation of genetic maps and haplotypes. In *Proceedings of the Plant and Animal Genome XIV Conference, San Diego*, 2006.

[15] E. S. Lander, P. Green, J. Abrahamson, A. Barlow, M.J. Daly, S.E. Lincoln, and L.A. Newberg. MAPMAKER: an interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics*, 1:174–181, 1987.

[16] A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: Theory and practice. *IEEE TKDE*, 15(3):515–528, 2003.

[17] M. Shindler, A. Wong, and A. Meyerson. Fast and accurate k-means for large datasets. In *NIPS*, 2011.

[18] M. Ankerst, M. M. Breunig, H. Kriegel, and J. Sander. OPTICS: ordering points to identify the clustering structure. *ACM SIGMOD Record*, 28(2):49–60, 1999.

[19] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, 1996.

[20] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases, 1996.

[21] S. Guha, R. Rastogi, and K. Shim. CURE: an efficient clustering algorithm for large databases. *ACM SIGMOD Record*, 27(2):73–84, 1998.

[22] F. Lin and W. W. Cohen. Power iteration clustering. In *Proc. of ICML*, volume 10, 2010.

[23] N.A. Tinker. Spaghetti: Simulation software to test genetic mapping programs. *Journal of Heredity*, 101(2):257–259, 2010.

[24] S. Wagner and D. Wagner. *Comparing Clusterings – An Overview*. Universität Karlsruhe, Fakultät für Informatik, 2007.

[25] J. Chapman et al. Chromosome-scale assembly of the hexaploid wheat genome from whole genome shotgun sequencing. In *submission to Genome Biology*, 2014.