Reliable Radiation Hybrid Maps: An Efficient Scalable Clustering-based Approach

Raed I. Seetan, Anne M. Denton, Omar Al-Azzam, Ajay Kumar, M. Javed Iqbal and Shahryar F. Kianian

Abstract—The process of mapping markers from radiation hybrid mapping (RHM) experiments is equivalent to the traveling salesman problem and, thereby, has combinatorial complexity. As an additional problem, experiments typically result in some unreliable markers that reduce the overall quality of the map. We propose a clustering approach for addressing both problems efficiently by eliminating unreliable markers without the need for mapping the complete set of markers. Traditional approaches for eliminating markers use resampling of the full data set, which has an even higher computational complexity than the original mapping problem. In contrast, the proposed approach uses a divide-and-conquer strategy to construct framework maps based on clusters that exclude unreliable markers. Clusters are ordered using parallel processing and are then combined to form the complete map. We present three algorithms that explore the trade-off between the number of markers included in the map and placement accuracy. Using an RHM data set of the human genome, we compare the framework maps from our proposed approaches with published physical maps and with the results of using the Carthagene tool. Overall, our approaches have a very low computational complexity and produce solid framework maps with good chromosome coverage and high agreement with the physical map marker order.

Index Terms—Framework mapping, Radiation Hybrid Mapping, Travelling Salesman Problem, Clustering, Bioinformatics

1 Introduction

Genome mapping [1] is important for finding the order of genes or unique short fragments of DeoxyriboNucleic Acid (DNA) sequence, known as markers, within chromosomes. Map information can also be used in genome sequencing projects [2]. Radiation Hybrid Mapping (RHM) [3] is a mapping technique for ordering markers along a chromosome and estimating the physical distances between them. Markers can be any simple short unique fragment of deoxyribonucleic acid (DNA) sequence that can be amplified and detected using the polymerase chain reaction (PCR). RHM is widely used in mapping projects and has several advantages over alternative mapping techniques: markers need not be polymorphic, i.e. exist as multiple versions, scoring the markers is easier, and the DNA can be kept in cell lines eliminating the cost of animal care and breeding.

- *Raed I. Seetan and Anne M. Denton are with Department of Computer Science, North Dakota State University, Fargo, ND, USA. E-mail: {raed.seetan, anne.denton}@ndsu.edu*
- *Omar Al-Azzam is with Math, Science and Technology Department, University of Minnesota, Crookston, MN, USA. E-mail: oalazzam@crk.umn.edu*
- *Ajay Kumar is with Department of Plant Sciences, North Dakota State University, Fargo, ND, USA. E-mail: ajay.kumar.2@ndsu.edu*
- *M. Javed Iqbal is with Department of Plant Sciences, University of California, Davis, CA, USA. E-mail: mjiqbal@ucdavis.edu*
- *Shahryar F. Kianian is with USDA-ARS Cereal Disease Laboratory, University of Minnesota, St. Paul, MN, USA. E-mail: Shahryar.Kianian@ars.usda.gov*

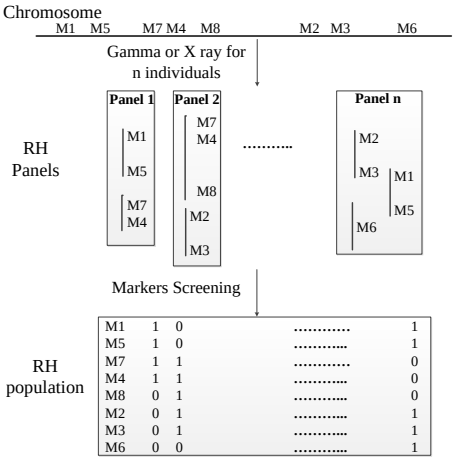

Fig. 1. The process of generating a radiation hybrid population.

The process of constructing radiation hybrid maps can be divided into two sequential stages: 1) the experiment stage and 2) the analysis stage. In the first stage, the chromosomes of interest are irradiated with X- or Gamma-radiation to create deletions. The radiation breaks the chromosome into several fragments, and each fragment contains the attached markers. The presence or absence of each marker in the chromosome fragments is screened and assigned the value 1 if a marker is present, and 0 otherwise. This process is repeated for multiple individuals to construct the RH population. Figure 1 shows the process of generating radiation hybrid populations.

The input for the analysis step is the RH population that is a M X N matrix of {1, 0, ?}, where M is the

number of markers and N is the number of individual organisms in the mapping population. Each marker has an RH vector {C1, C2, C3, ..., Cn}. The value of each attribute in the RH vector Ci indicates if that marker is present (1), absent (0) or ambiguous (?). The basic principle of RH mapping is that markers with similar RH vectors should be close together in the map. The further apart two markers are, the more likely the radiation will create a break between them, thus placing them in two different fragments where one marker may be present and the other one may be absent. The retention pattern is used to find the most likely linear order of the markers and estimate the physical distances between them.

In principle, finding the optimal order of a set of markers on a chromosome could be done by finding all possible marker permutations, and then choosing the permutation with the minimum number of Obligate Chromosome Breaks (OCB), which is the number of times a 1 is followed by 0 or vice versa in the same panel. However, in practice heuristic approaches are used. A toy example of 3 markers on 5 individuals is illustrated in Table 1. The best order is {M1, M2, M3} with four breaks. Any other permutation will have more than four breaks. The mapping process is largely equivalent to the traveling salesman problem of finding the shortest path among markers [4], [5], [6]. The computational complexity is correspondingly high, since for n markers, there are $n!/2$ marker orders. Framework maps consider the subset of most useful

TABLE 1
A toy example of an RH population for 3 markers on 5 panels. Breakages are highlighted in bold.

MARKER/PANEL	P1	P2	P3	P4	P5
M1	1	**1**	1	0	1
M2	1	**0**	**1**	**0**	**1**
M3	1	0	**0**	**1**	**0**

markers that can be ordered with high confidence. Traditional approaches for building framework maps depend mainly on resampling analysis [7] to iteratively filter out markers that cannot be mapped consistently [8], [9]. Such approaches require repeating the already computationally expensive mapping task for each resampled data set. An alternative approach is presented in [10], using an incremental insertion procedure to add one marker at a time to the current framework. However, the number of markers in the final map is too small and the approach may not guarantee full coverage of the whole chromosome. The procedure extends the current map from both sides until a stopping criterion is satisfied, which may happen before coverage of the chromosome is achieved.

To overcome these problems, we propose a fast approach for constructing solid framework maps with good coverage for a large number of markers. The idea of the proposed approach is to group markers based on their LOD (logarithms of odds -base 10) scores. The LOD score [11] was developed as a probabilistic measure for linkage, and has been used consistently throughout the radiation hybrid literature. Using grouping as an initial step is computationally fast and eliminates unreliable markers, which would otherwise reduce the quality of the generated maps [12]. We use agglomerative hierarchical clustering [13], which starts by grouping objects into small clusters and then repeatedly merges neighboring clusters based on a similarity measure to form one big cluster. Overall, hierarchical clustering has been shown to result in high clustering quality [14], [15]. Hierarchical clustering has several benefits in this context. In particular, it allows singletons that represent outlier data points effectively. Furthermore, hierarchical clustering does not require a distance measure that satisfies the properties of a metric. A distance metric has to satisfy the triangle inequality, i.e., A and C cannot be further apart than the sum of the distances from A to B and from B to C. However, in mapping it can happen that some linkage can be observed between markers A and B, and between markers B and C, but no deleted or retained regions extend from marker A to marker C.

Two types of markers can be distinguished that are noisy and cannot be placed reliably. The first type consists of markers far apart from all other markers. These markers fall into singleton groups that are filtered out as they do not have linkage with other markers in the map. The second type consists of markers that are linked with many other markers and it may be possible to map them similarly well into different positions. The grouping helps associate these markers with their most highly linked neighbors. Similarities to other markers, which could otherwise result in unstable maps, are ignored. Figure 2 shows an example of markers that are far apart from each other and still have connections that could result in instabilities of the final map. Problematic connections are {(M1, M5)(M1, M6)(M1, M7)(M2, M5)(M2, M6)(M2, M7)(M3, M6)(M3, M7)(M4, M7)}.

		Cluster B				Cluster A		
		M1	M2	M3	M4	M5	M6	M7
Cluster B	M1	------	13.1	9.5	10	3.9	4.6	3.5
	M2	13.1	------	14.3	13.1	3.9	4.6	4.7
	M3	9.5	14.3	------	14.4	2.6	3.2	4.3
	M4	10	13.1	14.4	------	2.1	2.6	3.6
Cluster A	M5	3.9	3.9	2.6	2.1	------	17.1	11.3
	M6	4.6	4.6	3.2	2.6	17.1	------	13.1
	M7	3.5	4.7	4.3	3.6	11.3	13.1	------

Fig. 2. Neighborhood LOD score matrix for two clusters of markers, Cluster B (M1, M2, M3 and M4) and Cluster A (M5, M6 and M7).

The proposed approach builds framework maps

by first dividing the markers into several linkage groups and then building a framework for each linkage group. As a second step, we concatenate the constructed framework maps of all groups to form the whole chromosome framework. A polishing step is used to create the final framework map.

The remainder of this paper is organized as follows: Section 2 presents the related work in the area of constructing framework maps. In Section 3 our proposed approach is introduced in detail. Section 4 presents the experimental evaluation of the proposed approach, and Section 5 concludes the paper.

2 Related Work

Conventional approaches for building skeleton maps by filtering out unreliable markers [8], [9] depend mainly on resampling analysis [7]. Briefly, the process is done in three iterative steps: resampling, mapping, and removing unreliable markers. A skeleton map from only the reliable markers is constructed after filtering out unreliable markers. Applying the mapping step to every resampled population is computationally expensive and scales exponentially with the number of markers to be mapped.

Several software packages provide options for fast alternatives for constructing framework maps [10], [16], [17]. RHMAPPER [16] is one of the commonly used packages, and it uses a hidden Markov model (HMM) [18] to build framework maps. RHMAPPER provides two ways to build framework maps: 1) using external information, such as a known order from genetic mapping for building an initial framework and then growing the framework map by incrementally adding markers and 2) searching for all available strongly ordered triplets of markers and then combining overlaps between these triplets into a larger framework. Finding the triplets of markers is sensitive to the value of the global parameter FRAMETHRESH and scales poorly to large numbers of markers. RHMAPPER takes several hours for as few as 100 markers. RHMAPPER implements two overlapping mechanisms for assembling the triplets of markers with the goal of merging two consecutive markers. In one strategy a-b-c and b-c-d are merged to a-b-c-d, while in the other a-b-c-d is derived from a-b-c and a-c-d. Once the framework map is constructed, new markers can be added to the framework map by taking each marker and finding its best position, and then keeping the new marker in that position, provided it satisfies some conditions.

Multimap [17] is another package for building framework maps. It builds framework maps in an automated fashion and reduces the amount of user-computer interaction compared to the step-by-step approach that is often used for building maps. Multimap constructs framework maps in three steps: 1) define a solid pair of informative markers, 2) iteratively insert each marker that is not on the framework map in its best position on the framework map, and keep only the set of markers that satisfy some predefined thresholds, and 3) perform some markers order verification to confirm the constructed framework order. Although Multimap reduces the mapping time compared with the step-by-step approach that serves as a comparison method in [17], their reported time for constructing a map for the human Chromosome 21 with 43 markers is a few days. Also, the coverage of the constructed framework maps may not be guaranteed.

A third package is Carthagene [10], which provides several algorithms for ordering a set of markers, i.e. simulated annealing [19], taboo search [20] and LinKernighan heuristic [21]. Carthagene also uses an incremental insertion procedure (Buildfw command) to construct framework maps. The Buildfw command tries to build framework maps as follows: First, an initial set of ordered markers from an external source can be defined as an initial framework; if no such initial framework exists, Carthagene defines the initial framework by finding the triplet of markers that maximizes the difference between the likelihood of the best map and the second best map using only this triplet. Second, Buildfw inserts the remaining markers incrementally at each possible interval on the framework map. Buildfw checks two thresholds for each marker insertion, AddThres and KeepThres. AddThres determines if a marker can be inserted in its position on the map or not. If the loglikelihood difference between the best two insertion positions is greater than the AddThres, then the marker can be inserted on the map in its best interval. The KeepThres is used to determine if the current map with the inserted marker can be used for further steps or not. If the loglikelihood difference between the best two maps is greater than the KeepThres, then the inserted marker will be kept as a part of the map and will be used for the next mapping steps. The Buildfw command may also be adjusted to indicate if post-processing is applied to the markers that cannot be inserted on the framework map. A flag is set to the values of {0, 1, 2} to take actions after generating the framework map: 1) zero, which means no post-processing is applied, 2) one, which means the remaining markers, not on the framework, are tentatively inserted in their best positions on the constructed framework map and the loglikelihood difference between the best two insertions is reported, and 3) two, which indicates no framework is built while the same post-processing procedure is applied.

All the previously discussed mapping packages [10], [16], [17] use an incremental insertion procedure to extend the initial framework. Adding one marker at a time does not scale well with the number of markers. Moreover, it is recommended that an LOD score of at least 3 is used for building a solid framework

map using these packages. However, using such a threshold results in framework maps with only a few markers [22]. Framework maps that are constructed using these techniques have the additional shortcoming that they may not cover the whole chromosome.

3 Proposed Approach

The proposed methods for constructing reliable framework maps are performed in three sequential steps. In the first step, we extract the most reliable markers from the data set and group these markers into clusters in such a way that the markers in the same cluster are closer to each other than those in different clusters. The clustering ensures that the linkage among several markers determines the overall ordering rather than random similarities that may be due to noise. The clustering also speeds up the mapping since mapping algorithms have combinatorial complexity, and the number of markers in each cluster is much smaller than the total. The process of mapping individual clusters can, moreover, be parallelized and be run on multiple compute nodes simultaneously.

The second step aggregates the constructed maps of all small clusters to form a complete framework map. In the third step a local improvement method is applied to fill large gaps between pairs of markers to enhance the overall map. Figure 3 shows the systematic work flow for the proposed approach.

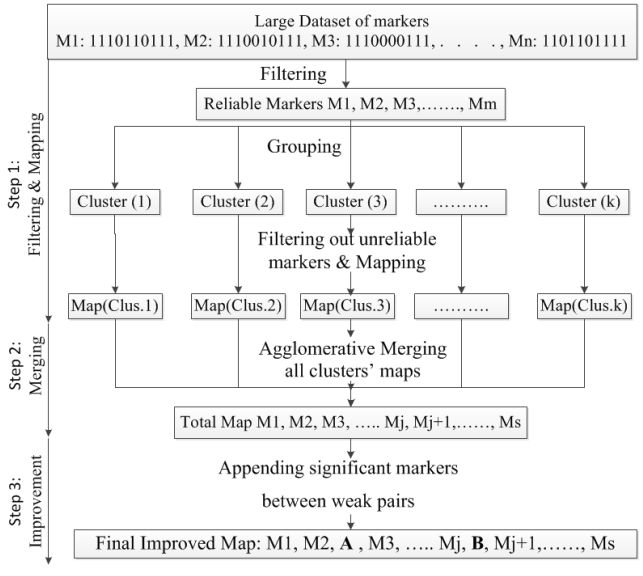

Fig. 3. Systematic Proposed Approach.

3.1 First Method

In this method, we divide the mapping process into three steps. The first step filters out unreliable markers and groups the initial, large set of markers into smaller subsets while eliminating disconnected markers. Solid maps are constructed for each subset. The second step merges all framework maps to get the whole chromosome framework map. The third step applies a local improvement method to strengthen the final constructed framework map.

3.1.1 Filtering and Mapping Task

The first step in the proposed method starts by extracting those pairs of markers that have a high LOD score (greater than threshold T). The T threshold is determined by grouping the markers with thresholds varying from 12 to 15, and then selecting the T threshold that maximizes the number of groups with at least 3 markers. Then the extracted markers are grouped into different clusters, where the distance between any two markers in different clusters is larger than the distance between any one marker and its closest other marker in the same cluster. This corresponds to single linkage clustering which is transitive, i.e., if markers A and B are strongly connected and markers B and C as well, then markers A, B and C are grouped together in one cluster.

The RH mapping process is based on the transitivity property. When radiation is applied, deleted or retained segments of DNA are created that may only extend as far as a few markers. Similarity information becomes meaningless beyond the deletion or retention length. In traditional mapping approaches, it was assumed that deletion or retention lengths might only include two or three markers. This is why conventional mapping algorithms are based on the traveling salesman problem, in which the overall length of the final map is calculated based on nearest-neighbor distances. For the high-resolution mapping considered in this manuscript, it is common that more markers fall within a single deletion or retention region, and we are hence able to still observe linkage even after removing noisy markers. However, it is important to recognize that similarity information is only meaningful for distances shorter than the deletion or retention region, and correspondingly that using anything other than nearest-neighbor information is risky.

Based on the transitive linkage assumption, the proposed method extracts large clusters (with at least three markers) to construct the final map. Then, each marker is labeled according to its cluster, and for each cluster we define the boundary by the two markers which are farthest apart. After that, for each cluster, we use the mapping strategy that is discussed in the next paragraph. The details of the proposed process can be seen in Algorithm 1.

Mapping Strategy

We use the Carthagene tool [10] to order markers. As a first step we represent markers that have identical mapping information by a single marker (double markers). Second, we use the build command (heuristic approach) to build an initial map. The build process starts with the pair of most strongly linked

Algorithm 1 Step 1: Filtering and Mapping

Input: $RHData$ /*NoOfMrk by NoOfIndv matrix*/
Input: T /* LOD threshold */
Result: $BestMaps$ /* Map for each cluster */
Clusters = Group $(RHData, T)$
for each C in $Clusters$ **do**
  if $Count\_Markers(C) > 2$ **then**
    $(E1, E2)$=GetPairWithSmallestLodScore(C)
    BestMap = FindBestMap(C)
    pos1 = FindMarkerPosition(E1)
    pos2 = FindMarkerPosition(E2)
    for each mk in BestMap **do**
      pos3 = FindMarkerPosition(mk)
      if pos3 Not in between (pos1, pos2) **then**
        RemoveMarker($BestMap, mk$)
      end if
    end for
    BestMap = GetMapUniquePositions($BestMap$)
    BestMap $\in$ BestMaps
  end if
end for
Return $BestMaps$

Algorithm 2 Step 2: Merging Maps

Input: $RHData$ /*NoOfMrk by NoOfIndv matrix*/
Input: T /* LOD threshold */
Input: $BestMaps$ /* BestMaps list */
Result: $FWMap$ /* Merged framework map */
MapsBoundaries=GetMapsBoundaries($BestMaps$)
BoundariesLabels=SetLabels($MapsBoundaries$)
while $Count(MapsBoundaries) > 2$ **do**
  Clusters = Group $(MapsBoundaries, T)$
  T = T - 2
  for each C in $Clusters$ **do**
    if $Count\_Markers(C) > 2$ **then**
      ClusterLabels=GetLabels($BoundariesLabels$)
      for $i = 1$ **to** $Count(ClusterLabels) - 1$ **do**
        (E1, E2) = FindStrongestPair(C)
        M1 = GetMap (E1)
        M2 = GetMap (E2)
        $ConnectMaps(M1, M2) \in FWMap$
        /* Delete Merged Boundaries */
        DelBoundaries (MapsBoundaries, E1, E2)
        BoundariesLabels(M1)=GetLabels(M2)
      end for
    end if
  end for
end while
Return $FWMap$

markers and inserts the remaining markers incrementally. Third, a greedy search is used to enhance the map. Fourth, genetic and simulated annealing algorithms are used to find a better map in case a local improvement exists. Finally, a fixed sliding window is applied to try all permutations within the window and checks if a better map can be achieved. Then we remove all inconsistently ordered markers that are mapped outside the cluster boundaries' positions. Also, to improve the constructed map, we keep only one marker in each unique position. Keeping only one marker of close neighbors reduces the number of locally flipped markers.

3.1.2 Merging the Maps of Clusters

In this step we incrementally concatenate the maps of all clusters to form one framework map. The merging step is done by extracting the boundaries of the maps of all clusters. Using the Carthagene tool, we group these boundaries into clusters starting with a high LOD score i. e. T threshold, which was used to divide the markers into the initial clusters, and then reducing it in each iteration until all map-boundaries are merged into one cluster. In each iteration, we group the set of current boundaries. Those grouped boundaries from different maps that fall into one cluster are concatenated to form a bigger map. The grouping process is repeated until all current map-boundaries are in one cluster. The threshold of grouping is decreased in each iteration by a constant i. e. 2, where the closest maps merge first then the next closest ones and so on.

Our proposed method merges the boundaries of different maps grouped in one cluster by concatenating the strongest pair of markers from two different maps, then concatenating the second strongest pair, and so on until all different maps in the cluster are merged. Figure 4 shows a toy example. Suppose we group the current map boundaries, and maps C1 and C2 are in one group, then we merge maps C1 and C2 into one map. To merge the maps C1 and C2, we find the LOD score between each pair (M1, M3, 6), (M1, M4, 3), (M2, M3, 2), (M2, M4, 1) and then merge the boundaries with the highest LOD value (M1, M3, 6). Algorithm 2 shows the detailed process.

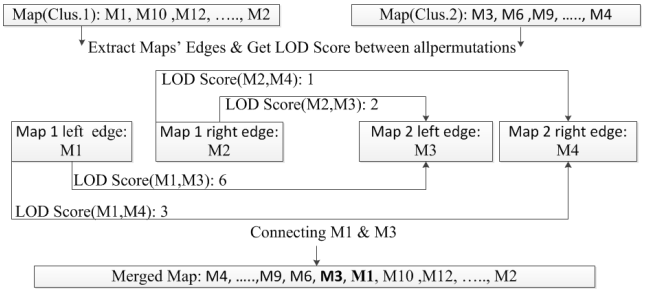

Fig. 4. Merging two clusters.

3.1.3 Local Improvement

After mapping the markers of each cluster and merging these maps to form the whole map, we expect to find some large gaps between pairs of markers. These gaps can result from merging two far apart clusters. Having such gaps in a map may diminish the overall

Algorithm 3 Step 3: Improvement method

Input: $RHData$ /*NoOfMrk by NoOfIndv matrix*/
Input: S /* LOD threshold */
Input: $BestMap$ /* Framework Map */
Result: $ImpMap$ /* Improved framework map */
for $i = 1$ **to** $length(BestMap) - 1$ **do**
  if $GetLodScore(mrk_i, mrk_{i+1}) < S$ **then**
    $N1 = GetNeighbors(mrk_i, RHData)$
    $N1 = Order(N1, "Descending")$
    $N2 = GetNeighbors(mrk_{i+1}, RHData)$
    $N2 = Order(N2, "Descending")$
    $L = GetMutualNeighbors(N1, N2)$
    $SigMrk = GetConnectorMrk(L, S))$
    $ImpMap = FillGap(mrk_i, mrk_{i+1}, SigMrk)$
  end if
end for
Return $ImpMap$.

map quality. Therefore, a local improvement method is used for filling these gaps. Markers that were left out of the original grouping step are added in this step. The process is explained in Algorithm 3. First we scan the whole map and identify the weak pairs of markers with an LOD score less than S, where S is the average LOD score of all consecutive pairs of markers in the final map. Then we find the list of neighbors for each marker in the respective pair. After that we find the mutual neighbors that connect the pairs with an LOD score greater than S. Finally we inject the mutual marker into the gap. If we get multiple mutual neighbors, we pick the one for which the LOD score difference is smallest, so that marker will be placed in the middle of the gap.

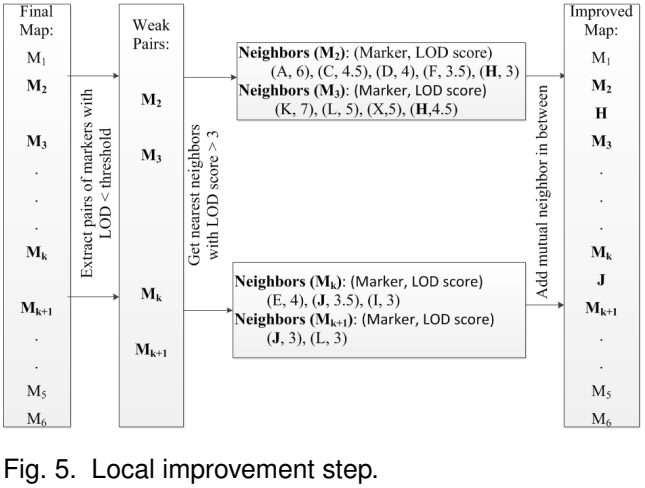

Fig. 5. Local improvement step.

Figure 5 shows a toy example that explains the improvement step. Instead of having (M_2, M_3) with an LOD score of 2, we insert a new marker (H) between M_2 and M_3. The total map is improved to have (M_2, H) with an LOD score of 3 and (H, M_3) with LOD score 4.5. The same applies for the pair M_k and M_{k+1}.

Algorithm 4 Step 1: Filtering and Framework Construction

Input: $RHData$ /*NoOfMrk by NoOfIndv matrix*/
Input: T /* LOD threshold */
Result: $BestMaps$ /* Map for each cluster */
Clusters = Group $(RHData, T)$
for each C in $Clusters$ **do**
  if $Count\_Markers(C) > 2$ **then**
    BestMap = FindBestTripleMrksinOrder(C)
    E1=GetFirstMarker(BestMap)
    E2=GetLastMarker(BestMap)
    if BestMap is $\emptyset$ **then**
      (E1,E2) = FindtheWeakestPair(C)
      BestMap = (E1, E2)
    end if
    BestMap = GetMapUniquePositions($BestMap$)
  end if
  $BestMap \in BestMaps$
end for
Return $BestMaps$

3.2 Second Method

The second proposed method uses a clustering technique that is based on triplets and bases the framework map construction on those. Only the first step is discussed in detail, since the second and third steps are the same for Method 1, as discussed in Sections 3.1.2 and 3.1.3 respectively.

3.2.1 Filtering and Framework Construction

This step finds the reliable groups of markers, and constructs a framework map for each cluster. The process starts with extracting the solid pairs of markers with a high LOD score, and then grouping these markers into different clusters with low intra-cluster distances and high inter-cluster distances. This method only considers large clusters (with at least three markers) to construct the final map. After labeling all large clusters, a framework map is constructed for each cluster by searching for the most solid combinations of three ordered markers in each cluster. To find such solid triplet markers, we use Carthagene. The Buildfw command in Carthagene finds triplets of markers such that all alternative orders have a loglikelihood not within a given threshold of the best order. The Carthagene recommended LOD threshold is 3. After getting a solid triplet of ordered markers for each cluster, we label the first and third markers in each cluster as cluster map boundaries. If Carthagene does not find such a triplet of markers for a cluster, then the cluster boundaries, i.e., the farthest apart two markers, form the cluster's framework. At the end of this step we get a framework map of three (or two) markers for each cluster. The details of this process can be seen in Algorithm 4.

3.3 Third Method

This proposed method uses the same pipeline that is discussed in Method 2, Section 3.2, then applies some post-processing steps to extend the framework map that is constructed using Method 2 to get a more complete framework (mapping more markers) as in Method 1. The results in [23] show that Method 2 constructs solid framework maps. However, considering framework maps with only three marker per cluster may result in constructing framework maps with a relative small number of mapped markers. In this proposed method we use Method 2 to construct framework maps as initial framework maps, and then we iteratively try to extend these framework maps to map as many markers as possible.

Three steps are used to construct the framework maps. The first step groups the set of markers into small clusters and iteratively constructs a framework map for each cluster. The second step merges the constructed framework maps to form the whole chromosome framework map. The third step improves the resulting framework maps. We discuss the first step in detail, while the second and third steps are already discussed in sections 3.1.2 and 3.1.3, respectively.

3.3.1 Iterative Framework Construction

This step starts with the filtering process where only the set of reliable markers are extracted. The Carthagene tool is used to group the markers into smaller groups. Markers in each cluster associate with their most highly linked neighbors, and the noisy similarities between markers in one cluster and other markers in a different cluster are ignored. Once the markers are grouped into several clusters, the large clusters with at least three markers are extracted to be considered for framework map construction.

As in Method 2, the best triplet of markers, such that all alternative orders have a loglikelihood not within a given threshold of the best order, is computed for each large cluster. If such a triplet of markers does not exist in a cluster, then the cluster boundaries, i.e., the farthest apart two markers, form that cluster's framework map and the mapping process ends for that cluster. Otherwise, the solid triplet of markers forms the initial framework map. Up to this point, Method 3 has used the same steps as Method 2. Method 3 then considers each cluster and attempts to extend the framework, which consists of three markers. This extension process attempts to insert the remaining markers that belong to each cluster into that cluster's framework map. Iteratively, each marker that is not yet on the framework map is inserted into each possible interval on the framework map. Two thresholds are checked in each insertion, AddThres and KeepThres. AddThres determines if a marker can be inserted in its position on the map or not. If the loglikelihood difference between the best two insertion positions is greater than the AddThres, then the marker is considered suitable to be inserted on the map in its best position. The KeepThres is used to determine if the current map can be used for further steps or not. If the loglikelihood difference between the best two maps is greater than the KeepThres, then the inserted marker will be kept as a part of the framework map and will be used for the next steps. This iterative step extends the initial three markers' framework map adding one marker in each iteration. The extension process stops when there is no marker left that is suitable to be inserted.

After constructing the framework map for each cluster, the first and last markers in each cluster framework map are defined as cluster map boundaries. The functionality of getting the solid three markers and the incremental insertion of remaining markers are available in the Carthagene tool. The details of this process can be seen in Algorithm 5.

Algorithm 5 Step 1: Iterative Framework construction

Input: $RHData$ /*NoOfMrk by NoOfIndv matrix*/
Input: T /* LOD threshold */
Input: $AddThres, KeepThres$ /*Extend Thres*/
Result: $BestMaps$ /* Map for each cluster */
Clusters = Group $(RHData, T)$
for each C in $Clusters$ **do**
  if $Count\_Markers(C) > 2$ **then**
    BestMap = FindBestTripleMrksinOrder(C)
    for each Mrk in C and not in $BestMap$ **do**
      Mrk_POS=GetBestPosition(Mrk, $BestMap$)
      if $MarkerQuality(Mrk) > AddThres$ **then**
        MapAppend($BestMap$, Mrk_POS, Mrk)
        DL = DeltaLikelihood (BestMap, SecondBestMap)
        if $D < KeepThres$ **then**
          RemoveMarker($BestMap$, Mrk)
        end if
      end if
    end for
    (E1, E2) = GetFirstandLastMarkers($BestMap$)
    if BestMap is $\emptyset$ **then**
      (E1,E2) = FindtheWeakestPair(C)
      BestMap = (E1, E2)
    end if
    BestMap = GetMapUniquePositions($BestMap$)
  end if
  $BestMap \in BestMaps$
end for
Return $BestMaps$

4 Experimental Results

4.1 Datasets

The human genome radiation hybrid data set is used in this study. The three commonly used standard panels of human radiation hybrids are the G3 [24]

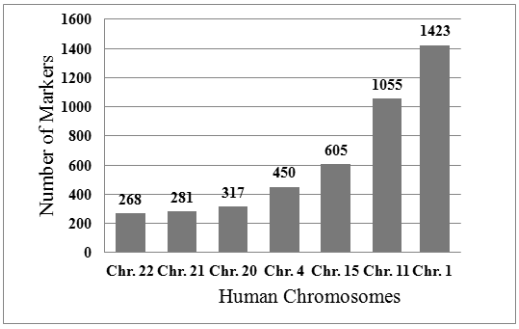

Fig. 6. Experiments Datasets: human chromosomes attached number of markers.

and TNG [25] panels produced by Stanford University and the Genebridge 4 [26] panel by the Sanger Center. In this study, we use both the G3 and Genebridge 4 panels where the numbers of individuals are 83 and 93, respectively. To evaluate the performance of the proposed approach, we have selected 7 different chromosomes with varying number of markers, showing the performance pattern over the increasing of markers number. Figure 6 shows the selected chromosomes and the number of markers in each chromosome. The choice of these chromosomes is determined by the availability of markers in both radiation hybrid data set and physical marker locations. The physical marker locations are extracted one by one from the Ensemble site [27], and the RH data set are downloaded from the EMBL-EBI site [28].

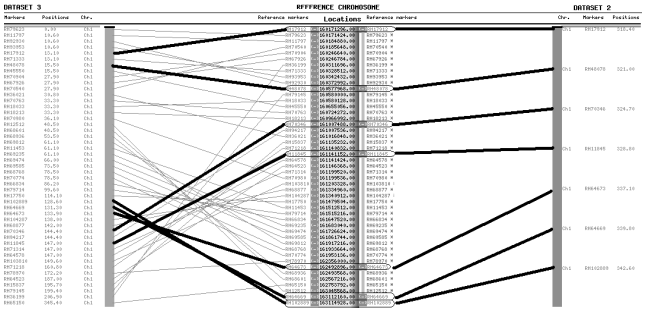

Fig. 7. Comparison between three maps created for the same segment of human Chromosome 1. Left: map created for all markers, the reliable and unreliable markers. Middle: the physical map for all markers. Right: framework map created for only the reliable markers. Good markers order conservation is indicated by parallel lines.

4.2 Evaluation of the approach

To give a high-level overview over the problems that our proposed methods attempt to address, Figure 7 shows a visualization of maps created for a segment of the human Chromosome 1. The maps in Figure 7 illustrate how markers align if all markers are included in the mapping process (left map of Figure 7). It can be seen that the presence of unreliable markers can affect the marker order of other markers. Note that for the left map in the figure, the order of the seven highlighted markers does not align well with the physical map if all markers are mapped in one step. On the other hand, our proposed map (right map of the figure) shows that the order of the same seven markers aligns well with the physical map (middle map of the figure) if the mapping process is carried out for only these seven markers. The Autograph tool [29] is used to visualize the maps.

Figure 8(a) shows the stepwise process of constructing the map for all markers, while Figure 8(b) shows the same process for mapping only the reliable markers. Tracing the map in Figure 8(a) shows that the order of markers does not align well with the true order: Many long forward steps are shown to be followed by several backward steps when mapping the reliable markers (highlighted as black cells) instead of a sequence of consecutive forward steps as shown in Figure 8(b). Two metrics are used to measure the

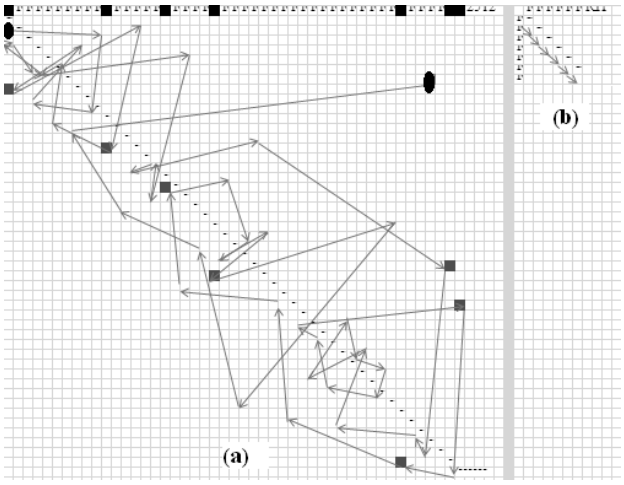

Fig. 8. Neighborhood LOD score matrix for a segment of human Chromosome 1 shows step by step process of constructing the map, (a) for all markers, (b) for the reliable markers. Markers are listed in the matrix according to their true physical order. The first and last markers in the constructed map are highlighted as black ovals. The reliable markers are highlighted as black squares. The best map alignment would be the one that go from a marker to its closest neighbor as shown in (b).

quality of the proposed frameworks. The first one is the agreement, considering the number of markers in the proposed framework map that have the same relative order in the physical map. The second measure reflects the coverage of the framework. To measure the coverage of a framework, we use the distance between real positions for the first and last markers in the constructed framework map and divide that distance by the total physical map length.

Using the proposed two metrics, we compare the constructed framework maps with the physical maps

of the corresponding chromosomes. Also, we compare the proposed framework maps with frameworks generated using the Carthagene tool. The comparison is done by plotting the predicted position of each marker in the proposed framework map relative to the marker orders in the physical map. The plots in Figures 9 to 11 show how well the proposed framework map orders agree with the physical maps for the three variant length chromosomes: short Chromosome 20, medium Chromosome 15 and long Chromosome 1, while the plots for the remaining four chromosomes can be found in the Supplementary document. For each chromosome, four plots are drawn: (a) framework map constructed using the Carthagene method; (b) framework map constructed using Method 1; (c) framework map constructed using Method 2; and (d) framework map constructed using Method 3. Table 2 shows the comparison of the proposed three methods against the Carthagene method.

For a comparison with Carthagene, we use the Buildfw command for building a solid framework map with the Carthagenes recommended LOD score of 3 for both the Adding_threshold and Keeping_threshold parameters [10]. Table 2 shows that our proposed methods substantially outperform Carthagene with regard to the number of markers in the framework maps for all chromosomes, except for Chromosome 11 where only Method 1 outperforms Carthagene. The evaluation of the proposed methods over short chromosomes, i.e Chromosome 20, can be seen in Figure 9, and the other two Chromosomes 22 and 21 can be found in Supplementary Figures S1 and S2, respectively.

For Chromosome 20, Carthagene is only able to map 6 markers, while each of our three methods map more than 10 times as many. Also, the proposed methods map at least twice as many markers than are mapped using the Carthagene method for both Chromosomes 21 and 22.

With regard to the coverage of the physical maps, our three proposed methods outperform Carthagene for both Chromosomes 20 and 22, and show comparable coverage for Chromosome 21. The agreement percentages of our approach with the physical maps were much higher than the Carthagene framework for Chromosome 22. For Chromosomes 20 and 21, the Carthagene agreement percentages are 100% and 95%, respectively, but the coverage of Chromosome 20 is so low that this result is not useful, and the number of markers in the Chromosome 21 framework map is too small for comparing to our proposed framework maps. Supplementary Figures S1(a) and S2(a) show a visual representation of the Carthagene framework maps for Chromosomes 22 and 21, respectively, and Figure 9(a) for Chromosome 20, where the partial coverage with the physical maps can be seen clearly along the y-axis in plot 9(a) 2.3×10^7 to 2.65×10^7 compared to our proposed maps in plots 9(b, c and d) 0 to 6.4×10^7.

For Chromosomes 4 and 15, which have more markers, the three proposed methods outperform the Carthagene method regarding the number of mapped markers in the framework maps. While the Carthagene method agreement percentage for Chromosome 4 is 67%, which is higher than the agreement percentages of the three proposed methods, the coverage of the Chromosome 4 is much lower, 16%, than that of our constructed maps with 91%. Supplementary Figure S3 and Figure 10 show the agreement and coverage of the framework maps constructed for Chromosomes 4 and 15 applying the Carthagene S3(a) and 10(a) and proposed Method 1 (b), Method 2 (c) and Method 3 (d). In Supplementary Figure S3, the marker orders show an overall good agreement between the physical map and the maps for the proposed three methods, with the exception of some local flipping. The agreement and coverage percentages of our proposed framework maps for Chromosome 15 outperform the Carthagene map. Figure 10(a) shows that the agreement of the Carthagene map does not fit well with the physical map, where a relatively long fragment of the map is flipped around at the end of the map. Also, many markers mapped in the wrong positions. In contrast, all of the three other approaches maps, plots 10(b,c and d) show better alignment with the physical map.

For the longer Chromosomes, 11 and 1, the three methods also outperform the Carthagene method regarding the agreement with the physical maps, the coverage of the chromosomes, and the mapping run time. The constructed framework maps for Chromosome 11 show good agreement with the corresponding physical map, where the accuracies are Method 1, 56%, Method 2, 77% and Method 3, 80%. In contrast, the agreement of the Carthagene map is 45%. The coverage of the Carthagene map for Chromosome 11 is too low, 28%, while the coverage of the proposed methods is more than 75%. Supplementary Figure S4 provides a visual representation of the constructed framework maps.

The plots in Figure 11 show the results for Chromosome 1. The constructed maps, using our proposed methods, have more markers compared with the framework map constructed using Carthagene, which has 39 markers. Figure 11(a) shows that the Carthagene map has several misordered markers at the beginning of the map, where some markers are assigned to the same position, while they are not in the original ordering. Also some other markers are flipped close their correct ordering at beginning of the map and in the middle of the map. The plots of our proposed approach 11(b, c and d), agree with the physical map for almost all the markers. The coverage of the map generated by Carthagene and our methods is low, less than 50%. However, the coverage of our maps is much higher than the Carthagene map cov-

erage. Plots 11(b, c and d) show our proposed maps coverage extends from 1.5×10^8 to 2.5×10^8. However, the Carthagene map range is smaller 1.45×10^8 to 1.7×10^8.

4.3 Run Time Comparison

The proposed approach is designed to be fast and scale well with the number of markers, as the initial set of markers is divided into smaller groups, and each group is initially processed separately. Each of the three steps in our algorithm is designed to be efficient. The first step (filtering and mapping) typically groups hundreds of markers into small clusters in less than a second. The mapping is done on clusters, and even for the largest clusters the number of markers is small in comparison with the overall number of markers. Figure 12 shows the mapping run time of Carthagene alone for different marker numbers, to illustrate why a divide-and-conquer approach is so efficient for this problem. Parallel processing decreases the overall run time further. If sufficiently many processors are available, the run time for all clusters is bound by the run time for the largest cluster, which consists of a relatively small number of markers compared with the total number of markers. The second step (merging) is done in a few seconds for all chromosomes. Carthagene calculates LOD scores for all pairs of markers upon loading the dataset and gets the marker_pairs information ready to use for any further process (i.e., agglomerative grouping). The third step (improvement) takes only a few seconds for getting the neighbors for two markers and finding the mutual neighbors. The total run time for constructing a framework map depends on the number of generated clusters and the total number of compute nodes a user has.

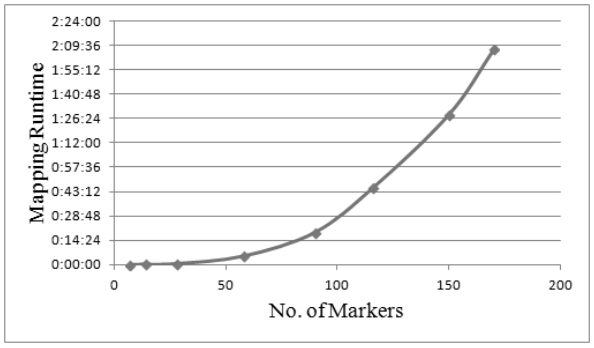

Fig. 12. Relationship between No. of markers and mapping complexity in Carthagene.

To evaluate the performance of the proposed approach, we compared the run time of the proposed methods with the run time of Carthagene. We test the scaling of the methods by mapping different chromosomes with different numbers of markers. We mapped seven chromosomes starting with a short chromosome, Chromosome 22, which has 268 markers, and ending with a long chromosome, Chromosome 1, which has 1423 markers.

Table 2 shows the run time for mapping different chromosomes with a varying number of markers using our proposed methods against the traditional Carthagene approach. The overall pattern of all methods in Table 2 is the same, where the mapping run time increases with the increasing number of markers in a chromosome and that results in constructing framework maps with larger numbers of markers. The mapping run times for Chromosome 1 and 11 are larger than the mapping run time for Chromosomes 22 and 21 for all methods. The results show that our three proposed methods outperform the Carthagene approach regarding the mapping run time and greatly reduce the time complexity of mapping all the chromosomes. While Carthagene maps all markers in one step to get a framework map, the three proposed methods use the grouping process to map only a few markers at a time and thus reducing the mapping run time. The run time of the proposed methods is the sum of three parts. The first part, which constructs a framework for each cluster, varies from one method to another. Method 2 has the shortest run time for constructing the framework maps for each cluster. Finding a triplet of ordered markers among a small number of markers in each cluster can be done in seconds. Method 3 takes more time than Method 2 because after finding the solid triplets of markers an extra step is applied to map the remaining cluster's markers. Method 1 is the most time consuming method among the three proposed methods, where all markers in each cluster are mapped before the process of filtering unreliable markers starts. On the other hand, the second and third parts are the time for merging the framework and improving the final map.

Traditional approaches for removing unreliable markers [8], [9] are based on resampling and have the complete mapping process built in for all resampled datasets. Such approaches require repeating the time consuming mapping task for each resampled data set. Filtering out unreliable markers for even relatively short chromosomes (i.e., Chromosomes 20, 21 and 22 with only 317, 281 and 268 markers, respectively) is a prohibitively slow task. For our approach the number of markers in a cluster is never more than 18 for these three chromosomes. The computation time is approximately doubled for each additional 30 markers. That means that a single iteration would take approximately three orders of magnitude longer for the complete chromosome in comparison with the clusters that are ordered in our approach. The resampling approach furthermore requires that mapping is run on multiple samples. For jackknife resampling this would require as many runs as there are indi-

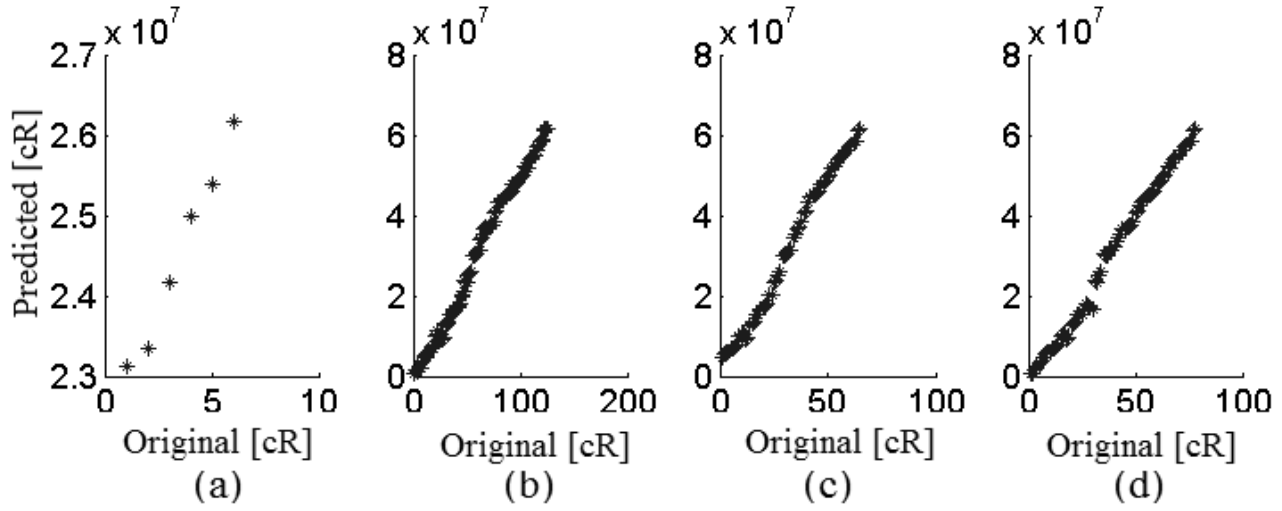

Fig. 9. Original permutation of the markers along the x-axis; predicted marker locations (centiRay) along the y-axis. The four maps are constructed for Chromosome 20 using (a) Carthagene Method, (b) Method 1, (c) Method 2 and (d) Method 3.

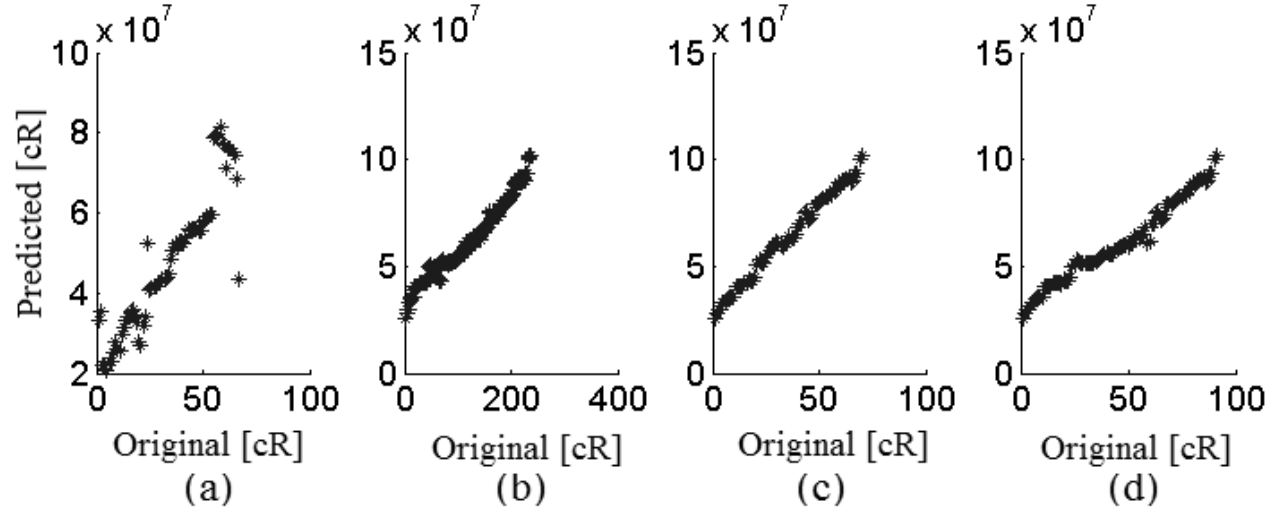

Fig. 10. Original permutation of the markers along the x-axis; predicted marker locations (centiRay) along the y-axis. The four maps are constructed for Chromosome 15 using (a) Carthagene Method, (b) Method 1, (c) Method 2 and (d) Method 3.

viduals, or 83 runs in the case of Chromosome 20. If 5% of the data set, or 16 markers, are to be filtered, one iteration would be necessary for each. Applying the resampling approach for longer chromosomes, i.e. Chromosomes 1 and 11 with larger numbers of markers, would require too many iterations to filter out the unreliable markers to make the approach reasonable. While resampling runs can also be parallelized, any resampling-based approach would clearly take several orders of magnitude longer than our proposed approach.

The run time for neither the RHMapper nor the Multimap packages scales well with the number of markers. RHMapper generates framework maps in two steps: the first step finds the solid triplets markers; the second step assembles the triplets to form framework maps. Finding the set of triplet markers is a time consuming process: a run based on one hundred markers takes several hours, and having more markers increases the run time dramatically [16]. We tried to generate framework maps using the RHMapper package. Three jobs were run on three different machines for the shortest three Chromosomes 20, 21 and 22, and none of the three jobs had completed after 6 days. After that time the jobs had only finished the first step of finding the strong connected triplets markers which took approximately 7 hours. Using RHMapper to construct framework maps for the remaining longer chromosomes, i.e. Chromosomes 11 or 1 would take a prohibitively long time to complete. The Multimap package takes n steps to construct a

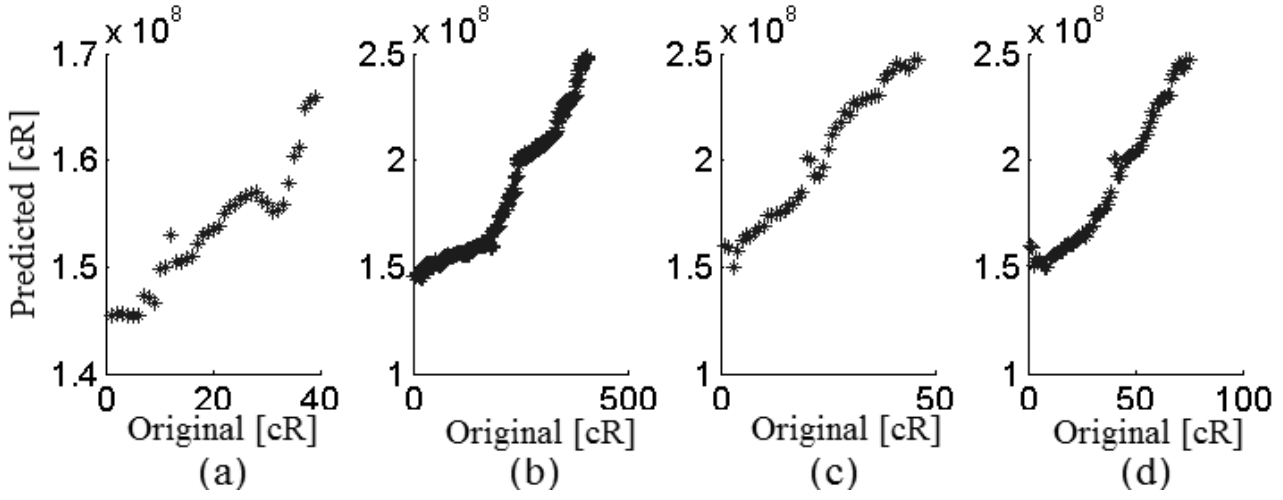

Fig. 11. Original permutation of the markers along the x-axis; predicted marker locations (centiRay) along the y-axis. The four maps are constructed for Chromosome 1 using (a) Carthagene Method, (b) Method 1, (c) Method 2 and (d) Method 3.

TABLE 2
Comparison of the running time and number of mapped markers in framework maps generated by Method 1, Method 2, Method 3 and the Carthagene method.

Chomosome Markers		**Chr. 22** 268	**Chr. 21** 281	**Chr. 20** 317	**Chr. 4** 450	**Chr. 15** 605	**Chr. 11** 1055	**Chr. 1** 1423
Carthagene Method	Mapped Markers	21	20	6	40	67	140	39
	Run Time	0:04:27	0:04:31	0:08:25	0:23:12	1:28:40	14:52:53	10:30:53
	Agreement Percentage	0.66	0.95	100	0.67	0.53	0.45	0.66
	Coverage Percentage	0.38	0.74	0.05	0.16	0.13	0.28	0.08
First Method	Mapped Markers	98	65	125	191	236	513	407
	Run Time	0:00:21	0:00:13	0:00:45	0:00:49	0:02:20	0:10:37	0:44:52
	Agreement Percentage	0.79	0.76	0.83	0.59	0.58	0.56	0.58
	Coverage Percentage	0.94	0.72	0.98	0.91	0.94	0.75	0.41
Second Method	Mapped Markers	51	45	65	71	70	80	62
	Run Time	0:00:09	0:00:11	0:00:12	0:00:05	0:00:12	0:01:16	0:04:53
	Agreement Percentage	0.98	0.75	0.98	0.63	0.78	0.77	0.76
	Coverage Percentage	0.94	0.72	0.91	0.91	0.94	0.95	0.35
Third Method	Mapped Markers	51	49	78	76	91	105	75
	Run Time	0:00:09	0:00:11	0:00:15	0:00:06	0:00:12	0:01:19	0:04:53
	Agreement Percentage	0.98	0.88	0.87	0.57	0.77	0.80	0.76
	Coverage Percentage	0.93	0.72	0.98	0.91	0.94	0.75	0.35

framework map of n markers, where all unmapped markers are successively inserted into the current framework map, and if the added marker satisfies predefined conditions then the new marker becomes a part of the current framework. Each time only one marker is added to its best interval in the map, checking all intervals for a large number of markers is a time consuming process. Both packages are far more time consuming than our proposed approach.

5 Conclusion and Future Work

Several fast methods have been proposed for constructing radiation hybrid framework maps. Given a large number of markers, the proposed methods aim to select a subset of markers and build a solid framework map. The proposed methods efficiently construct high-quality frameworks by using a divide-and-conquer strategy to construct framework maps. The proposed methods work in three steps: 1) divide the set of markers into smaller subsets and construct a framework map for each subset, 2) merge all framework maps, and 3) polish the map to fill the large gaps. The proposed methods differ in the first step, the way of constructing a framework map for a subset of markers.

The first proposed method constructs maps with larger numbers of markers, while the second method constructs maps with less numbers of markers and

a higher percentage of agreement. The third method combines the strengths of both the first and second methods, where it constructs maps with large numbers of markers and high agreement percentages.

To validate our methods, we use human radiation hybrid data and compare the framework maps with published physical maps. As a comparison technique we consider the framework maps generated by the Carthagene tool. We use two metrics in the comparison: 1) the agreement between the maps and 2) the coverage of the frameworks. The comparison results show that the proposed methods can produce solid framework maps that have high marker order agreement and good coverage for chromosomes with a varying number of markers. We show that the total computation time is lower than for Carthagene and far lower than for other approaches.

In this paper, we addressed the process of constructing solid framework maps. However, it is often desirable to construct comprehensive maps that provide information about a larger number of markers. We are currently working on developing an integrated approach for constructing comprehensive maps that integrates the two steps of performing framework mapping and mapping of remaining markers. Traditional approaches map all markers in a single-iteration. However, some markers may disrupt the whole mapping process. Our comprehensive mapping algorithm maps markers incrementally, where in the first step we consider the subset of reliable markers for constructing a solid framework map by applying the exhaustive and heuristic algorithms. Then in the second step we iteratively map and validate the remaining markers, while excluding problematic markers.

ACKNOWLEDGMENT

This work was supported by funding from the National Science Foundation, Plant Genome Research Program (NSF-PGRP) grant No. IOS-0822100 to SFK.

REFERENCES

[1] V. Kalavacharla *et al.*, "Radiation hybrid mapping in crop plants," *Advances in Agronomy*, vol. 102, pp. 201–222, 2009.

[2] P. H. Dear, "Genome mapping," *eLS*, 2001.

[3] S. J. Goss and H. Harris, "New method for mapping genes in human chromosomes," *Nature*, vol. 255, pp. 680–684, 1975.

[4] D. E. Weeks and K. Lange, "Preliminary ranking procedures for multilocus ordering," *Genomics*, vol. 1, no. 3, pp. 236–242, 1987.

[5] A. Ben-Dor, B. Chor, and D. Pelleg, "Rhoradiation hybrid ordering," *Genome Research*, vol. 10, no. 3, pp. 365–378, 2000.

[6] D. Hall, S. M. Bhandarkar, J. Arnold, and T. Jiang, "Physical mapping with automatic capture of hybridization data," *Bioinformatics*, vol. 17, no. 3, pp. 205–213, 2001.

[7] P. Good, *Resampling methods: A practical guide to data analysis*. Springer, 2001.

[8] Y. Ronin, D. Mester, D. Minkov, and A. Korol, "Building reliable genetic maps: different mapping strategies may result in different maps," *Nat. Science*, vol. 2, pp. 576–589, 2010.

[9] D. I. Mester *et al.*, "Multilocus consensus genetic maps (mcgm): formulation, algorithms, and results," *Computational biology and chemistry*, vol. 30, no. 1, pp. 12–20, 2006.

[10] S. De Givry, M. Bouchez, P. Chabrier, D. Milan, and T. Schiex, "Carh ta gene: multipopulation integrated genetic and radiation hybrid mapping," *Bioinformatics*, vol. 21, no. 8, pp. 1703–1704, 2005.

[11] N. E. Morton, "Sequential tests for the detection of linkage," *American journal of human genetics*, vol. 7, no. 3, p. 277, 1955.

[12] O. Al-Azzam *et al.*, "Network-based filtering of unreliable markers in genome mapping," in *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, vol. 1. IEEE, 2011, pp. 19–24.

[13] P. H. Sneath, R. R. Sokal *et al.*, *Numerical taxonomy. The principles and practice of numerical classification.*, 1973.

[14] P. Willett, "Recent trends in hierarchic document clustering: a critical review," *Information Processing & Management*, vol. 24, no. 5, pp. 577–597, 1988.

[15] A. V. Leouski and W. B. Croft, "An evaluation of techniques for clustering search results," DTIC Document, Tech. Rep., 2005.

[16] L. Stein, L. Kruglyak, D. Slonim, and E. Lander, "Rhmapper," *unpublished software, Whitehead Institute/MIT Center for Genome Research. Available at, and,, directory/pub/software/rhmapper. http://www.genome.wi.mit.edu/ftp/pub/software/rhmapper/, ftp. genome. wi. mit. edu*, 1995.

[17] T. C. Matise, M. Perlin, and A. Chakravarti, "Automated construction of genetic linkage maps using an expert system (multimap): a human genome linkage map," *Nature genetics*, vol. 6, no. 4, pp. 384–390, 1994.

[18] M. Boehnke, K. Lange, and D. R. Cox, "Statistical methods for multipoint radiation hybrid mapping." *American journal of human genetics*, vol. 49, no. 6, p. 1174, 1991.

[19] W. H. Press, *Numerical recipes in Fortran 77: the art of scientific computing*. Cambridge university press, 1992, vol. 1.

[20] D. Cvijovic and J. Klinowski, "Taboo search: an approach to the multiple minima problem," *Science*, vol. 267, no. 5198, pp. 664–666, 1995.

[21] K. Helsgaun, "An effective implementation of the lin–kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126,

no. 1, pp. 106–130, 2000.

[22] O. G. Al-Azzam and A. M. Adviser-Denton, "Mining for significant information from unstructured and structured biological data and its applications," 2012.

[23] R. I. Seetan *et al.*, "A fast and scalable clustering-based approach for constructing reliable radiation hybrid maps," in *Proceedings of the 12th International Workshop on Data Mining in Bioinformatics*. ACM, 2013, pp. 34–41.

[24] E. A. Stewart *et al.*, "An sts-based radiation hybrid map of the human genome," *Genome research*, vol. 7, no. 5, pp. 422–433, 1997.

[25] K. L. Lunetta, M. Boehnke, K. Lange, and D. R. Cox, "Selected locus and multiple panel models for radiation hybrid mapping." *American journal of human genetics*, vol. 59, no. 3, p. 717, 1996.

[26] G. Gyapay *et al.*, "A radiation hybrid map of the human genome," *Human Molecular Genetics*, vol. 5, no. 3, pp. 339–346, 1996.

[27] R. J. Kinsella *et al.*, "Ensembl biomarts: a hub for data retrieval across taxonomic space," *Database: the journal of biological databases and curation*, vol. 2011, 2011.

[28] C. Amid *et al.*, "Major submissions tool developments at the european nucleotide archive," *Nucleic acids research*, vol. 40, no. D1, pp. D43–D47, 2012.

[29] T. Derrien, C. André, F. Galibert, and C. Hitte, "Autograph: an interactive web server for automating and visualizing comparative genome maps," *Bioinformatics*, vol. 23, no. 4, pp. 498–499, 2007.

Raed Seetan received the MS degree in Computer Science from University of Science and Technology, Jordan, in 2009 and he is currently working toward the PhD degree under the supervision of Dr. Anne Denton at North Dakota State University. His research interests include: Clustering analysis, Radiation Hybrid Mapping and Optimization. He has been working with the Bioinformatics and Data Mining team on an NSF funded project on Radiation Hybrid Mapping.

Anne Denton received the MS in Computer Science from North Dakota State University in 2003 and the PhD in Physics from the Johannes Gutenberg University, Mainz, Germany, in 1996. She is currently an associate professor in the Computer Science Department at North Dakota State University. Her research interests are in data mining of diverse scientific data sets that are too complex to be analyzed using classical statistics or machine learning techniques, and for which rigorous significance evaluations are required. She works with collaborators in plant sciences, microbiology, and the chemistry of coatings.

Omar Al-Azzam received the PhD degree in Computer Science from North Dakota State University, in 2012. He is currently an assistant professor in the Math, Science, and Technology Department at University of Minnesota Crookston (UMC). His research interests are in data mining and bioinformatics.

Ajay Kumar received the MS degree in Agricultural Botany in 2001 and the PhD degree in Genetics and Plant Breeding in 2009, both from the Department of Agricultural Botany, Ch. Charan Singh University Meerut, India. Since 2009, He is working as a research scientist in the department of Plant Sciences, North Dakota State University, USA. My research is focused mainly on plant genomics. This include identification of QTL/genes for important traits, utilization of these genes in breeding programmes through marker assisted selection, QTL cloning and physical mapping, in both tetraploid and hexaploid wheat.

M. Javed Iqbal received the PhD in genetics and molecular biology from University of Illinois. He did postdoctoral research at Texas A & M University and Southern Illinois University and has been Adjunct Faculty at the SIU and Virginia Tech in the departments of Horticulture and Forestry. He is been PI and CoPI on several federal and state funded research projects. He is working on gene mapping in various crops and genomic analysis of plant pathogen interactions.

Shahryar F. Kianian received the BS in Biological Sciences from UC-Irvine, and the PhD in Genetics from UC-Davis. He is currently the Research Leader for the Cereal Disease Research Unit in St. Paul., MN. He has been engaged in improvement of wheat for the past 15 years. Dr. Kianian is recognized nationally and internationally as an authority of wheat genetics, alloplasmic, and genomics research. He has been the leader in the use of radiation hybrid (RH) mapping to generate high-resolution physical maps. His research also identified and mapped quantitative trait loci from wild type and wheat land races which imparted resistance to Fusarium head blight. He is author or coauthor on over 80 peer-reviewed journal publications, numerous edited proceedings, and book chapters.