A Fast and Scalable Clustering-based Approach for Constructing Reliable Radiation Hybrid Maps

Raed I. Seetan
Department of Computer Science
North Dakota State University
Fargo, ND, USA
raed.seetan@ndsu.edu

Anne M. Denton
Department of Computer Science
North Dakota State University
Fargo, ND, USA
anne.denton@ndsu.edu

Omar Al-Azzam
Math, Science and Technology Department
University of Minnesota
Crookston, MN, USA
oalazzam@crk.umn.edu

Ajay Kumar
Department of Plant Sciences
North Dakota State University
Fargo, ND, USA
ajay.kumar.2@ndsu.edu

M. Javed Iqbal
Department of Plant Sciences
North Dakota State University
Fargo, ND, USA
muhammad.iqbal@ndsu.edu

Shahryar F. Kianian
Department of Plant Sciences
North Dakota State University
Fargo, ND, USA
s.kianian@ndsu.edu

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. 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 Carthagene tool. Overall, our approach has a very low computational complexity and produces solid framework maps with good chromosome coverage and high agreement with the physical map marker order.

Keywords

Framework mapping, Radiation Hybrid Mapping, Clustering, Bioinformatics

BIOKDD'13 Chicago, IL, USA.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Clustering; H.2.8 [**Database Applications**]: Data mining

1. INTRODUCTION

Genome mapping [1] is important for finding the order of markers within chromosomes. Map information can also be used in genome sequencing projects [2]. Radiation Hybrid mapping (RHM) [3] is a mapping technique, in which deletions in chromosomes are created using radiation. 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). Presence or absence of a marker in each individual organism in the mapping population can be viewed as an attribute of that marker. Physical distances between markers can be calculated on that basis. The mapping process is largely equivalent to the traveling salesman problem of finding the shortest path among markers. The computational complexity is correspondingly high, since for n markers, there are $n!/2$ marker orders.

Framework maps consider the subset of most useful markers that can be ordered with high confidence. Traditional approaches for building framework maps depend mainly on resampling analysis [4] to iteratively filter out markers that cannot be mapped constantly [5, 6]. Such approaches require repeating the already computationally expensive mapping task for each resampled data set. An alternative approach is presented in [7], 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 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 to 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 [8] is a widely used

measure of the likelihood that two markers are linked. The grouping process is computationally fast and eliminates unreliable markers, which would otherwise reduce the quality of the generated maps [9].

Two types of unreliable markers can be distinguished: 1) Some markers are 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. 2) Another scenario is that markers are linked with many other markers and may be mapped in different positions. The grouping helps associate these markers with their most highly linked neighbors. The process ignores similarities to other markers which could otherwise result in unstable maps. Figure 1 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	------

Figure 1: 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 [5, 6] depend mainly on resampling analysis [4]. 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 [7, 10, 11]. RHMAPPER [10] is one of the commonly used packages. This package builds the framework by searching for all available strongly ordered triples of markers and then combines overlaps between these triples into a larger framework. The process of finding the triples is expensive for large data sets of markers.

Multimap [11] is another package to build framework maps by initially defining a solid pair of markers then iteratively adding one marker at a time to the initial framework, which may result in incomplete coverage of the chromosome.

Carthagene [7] uses a stepwise marker insertion method called Buildfw to construct framework maps. The Buildfw command map starts constructing framework maps with an empty map or an initial set of ordered markers from an external source. Then it tries to insert markers at all possible positions of the current map by testing if the difference in loglikelihood between the best insertion position and the second best insertion is greater than the Adding_threshold. If so, the marker is placed in its best position. Any markers for which the insertion results in a loglikelihood difference greater than the Keeping_threshold will be kept for the next iterations. This process stops when no marker results in a difference in loglikelihood greater than the Adding_threshold. We compare results from our method with the Carthagene result in the evaluation section.

All the previously discussed mapping packages [7, 10, 11] 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 few markers. In some cases, only approximately 5% of markers remain as the framework map. A framework map that is constructed using these techniques is neither contain enough of the markers, nor is there a guarantee for the constructed framework to cover the whole chromosome.

3. PROPOSED APPROACH

The proposed methods for constructing reliable framework maps is 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 other clusters. The clustering method allows parallelizing the process such that the mapping can be done on multiple computing nodes. The second step aggregates the constructed maps of all 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. Figure 2 shows the systematic work flow for the 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.

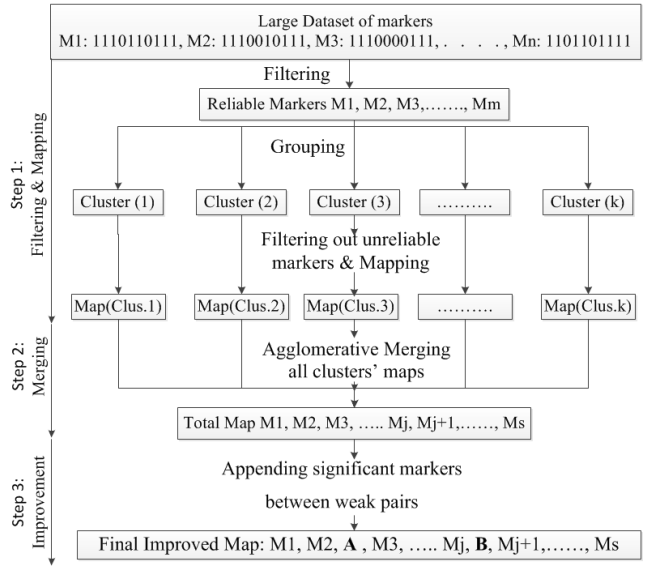


Figure 2: Systematic Proposed Approach.

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). 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 two markers 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.

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 [7] 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 markers and inserts the remaining markers incrementally. Third, the greedy search algorithm 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 check 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 impact of the local flipping problem.

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 $BestMaps$

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 all clusters maps. Using the Carthagene tool, we group these boundaries into clusters starting with a high LOD score and then releasing 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 T, 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 3 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 C1 and C2 maps, 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.

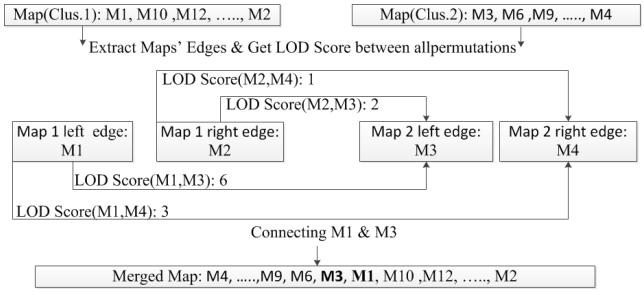


Figure 3: Merging two clusters.

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 = ExtractMapsBoundaries ($BestMaps$)
MapsBoundariesLabels=AssignLabels($MapsBoundaries$)
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($MapsBoundariesLabels$)
      for $i = 1$ **to** $Count(ClusterLabels) - 1$ **do**
        (E1, E2) = FindStrongestPair(C)
        M1 = GetMap (E1)
        M2 = GetMap (E2)
        $ConnectMaps(M1, M2) \in FWMap$
        RemoveConnectedBoundaries(MapsBoundaries, E1, E2)
        BoundariesLabels(M1)=BoundariesLabels(M2)
      end for
    end if
  end for
end while
Return $FWMap$

3.1.3 Local Improvement

After mapping the markers of each cluster and merging these maps together 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 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 LOD score less than S. Then we find the list of neighbors for each marker in that pair. After that we find the mutual neighbors that connect the pairs with 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.

Figure 4 shows a toy example that explains the improvement step. Instead of having (M_2, M_3) with 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 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}.

3.2 Second Method

The second proposed method uses a clustering technique that is based on triples 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 those for Method 1, as discussed in Sections 3.1.2 and 3.1.3 respectively.

Algorithm 3 Step 3: Improvement method

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

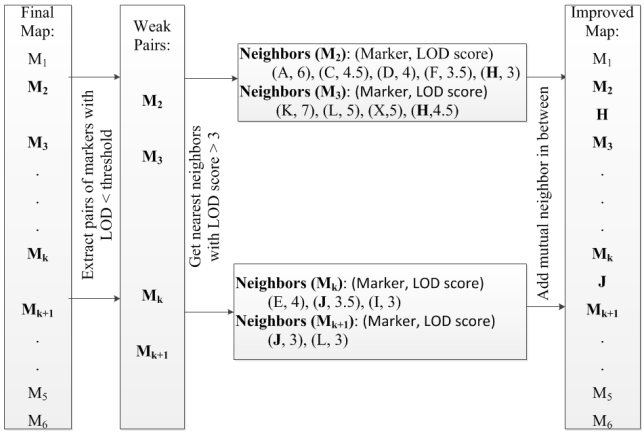


Figure 4: Local improvement step.

3.2.1 Filtering and Framework constructing

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 triple markers, we use Carthagene. The Buildfw command in Carthagene finds triples of markers such that all alternatives orders have a loglikelihood not within a given threshold of the best order. The recommended LOD threshold is three. After getting a solid triple 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 triple of markers for a cluster, then the cluster boundaries, the farthest apart two markers, form the clusters' 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.

Algorithm 4 Step 1: Filtering and Framework constructing

```
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 ∅ then
      (E1,E2) = FindtheWeakestPair(C)
      BestMap = (E1, E2)
    end if
    BestMap = GetMapUniquePositions(BestMap)
  end if
  BestMap ∈ BestMaps
end for
Return BestMaps
```

4. EXPERIMENTAL RESULTS

4.1 Datasets

The Human genome radiation hybrids data set are used in this study. The common three standard panels of human radiation hybrids are the G3 and TNG panels produced by Stanford University and the Genebridge 4 panel by the Sanger Center. In this study we used the G3 panels. Due to time constraints, we performed our experiments on three human chromosomes 20, 21 and 22. The choice of these chromosomes is determined by the availability of markers in both radiation hybrid data set and NCBI_RH map. Chromosome 20 has 317 markers with 83 RH lines, Chromosome 21 has 281 markers with 83 RH lines and chromosome 22 has 268 markers with 83 RH lines. The physical marker locations are extracted one by one from Ensemble site [12], and the RH data set are downloaded from EMBL-EBI site [13].

4.2 Evaluation of the approach

Two metrics were used to measure the quality of the proposed frameworks. The first one is the agreement, considering the number of markers in the proposed framework map having the same relative order in the physical map. The second measurement 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.

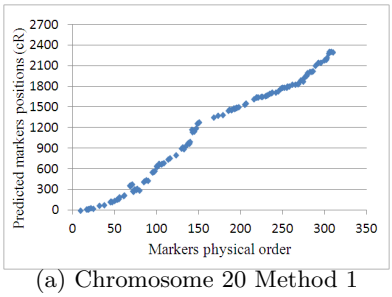
Using the proposed two metrics we compared the constructed framework maps with the physical maps of the corresponding chromosomes. Also we compare the proposed framework maps with frameworks generated using the Carthagene tool. The comparison was done by plotting the predicted position of each marker in the proposed framework map relative to the marker orders in the physical map. The plots in Figures 5, 6 and 7 show how well the proposed framework map orders agree with the physical maps for the three chromosomes 20, 21 and 22 respectively, where Figures 5(a) and 5(b) show the results for maps generated for chromosome 20 applying Methods 1 and 2 respectively, Figures 6(a) and 6(b) show the generated framework maps for chromosome 21 applying Methods 1 and 2 respectively, and Figures 7(a) and 7(d) show chromosome 22 framework map orders applying Methods 1 and 2 respectively.

The results obtained for human chromosomes 20, 21 and 22 are summarized in Table 1. The results show that the proposed framework for chromosome 20 using Method 2 agrees 98.4% with the physical map, while the agreement is 83.2% for Method 1. For chromosome 21 the agreement of both methods with the physical map is comparable, 76% for Method 1, and 75% for Method 2. For chromosome 22 the agreement with the physical map of applying Method 2 is 98% while it is 79.0% for Method 1. Method 2, thereby, gives overall more reliable results than Method 1.

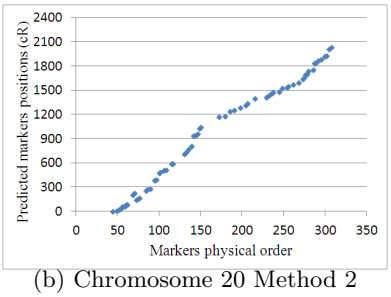
For a comparison with Carthagene we use the Buildfw command for building a solid framework map, with the recommended LOD score of 3 for both the Adding_threshold and Keeping_threshold parameters. Table 1 shows that our proposed algorithms substantially outperforms Carthagene with regard to the number of markers in framework maps for all chromosomes. For chromosome 20, Carthagene was only able to map 6 markers, while our methods both map more than 10 times as many. With regard to the coverage of the physical maps our approaches 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 for 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. Figures 5(c), 6(c) and 7(c) show a visual representation of the Carthagene framework maps, where the partial coverage and the marker order disagreement with the physical maps can be seen clearly.

4.3 Run Time Comparison

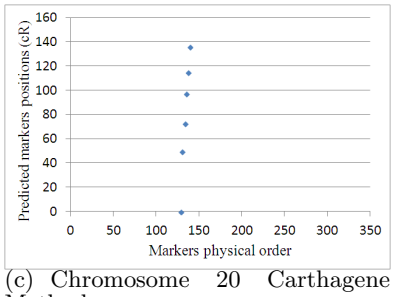
The proposed approach is designed to work fast and scale well with the number of markers, as the initial set of markers is divided into smaller groups. Working on each group separately helps reduce the run time for the mapping problem. Our three-step algorithm builds framework maps within short time (less than a minute). The first step (filtering and mapping) can group hundreds of markers into small clusters in less than a second, and the parallel mapping process for all generated clusters can be done in a few seconds. Even the largest group consists of a relative small number of markers in comparison with the total number of markers. Mapping the largest cluster we have in our experiments (18 markers) takes only 35 seconds, and the maximum number of clusters is 32 (see Table 1). The parallel processing decreases the overall running time further, since the running time for all clusters equals the running time for the largest cluster. The second step (merging) is done in a few seconds for all chromosomes, as Carthagene calculates LOD scores for all pairs of markers upon loading the dataset, and gets the markers_pairs information ready to use for any further process (i.e., agglomerative grouping). The third step (improvement) takes only a few seconds to get the neighbors for two markers and find the mutual neighbors. The total run time for constructing a framework map depends on the number



(a) Chromosome 20 Method 1

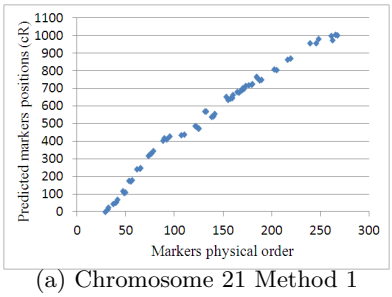


(b) Chromosome 20 Method 2

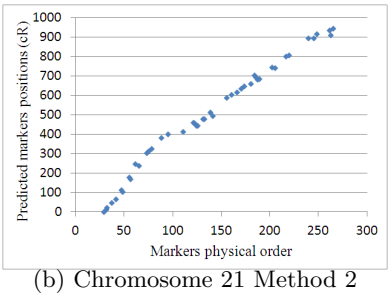


(c) Chromosome 20 Carthagene Method

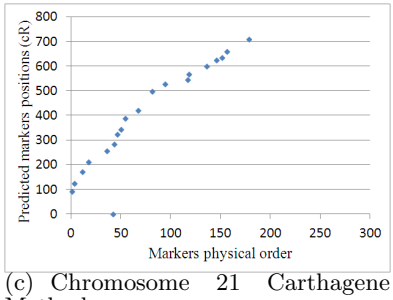
Figure 5: Plot of constructed framework maps for chromosome 20 applying Methods 1, 2 and Carthagene. The actual order of the markers is shown along the x-axis and the predicted order along the y-axis.



(a) Chromosome 21 Method 1

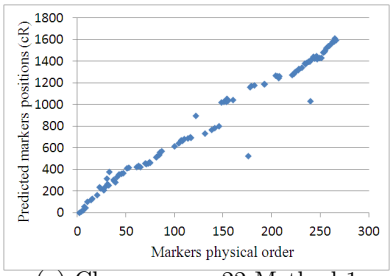


(b) Chromosome 21 Method 2

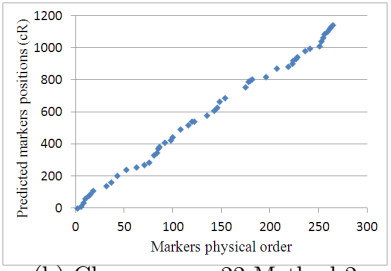


(c) Chromosome 21 Carthagene Method

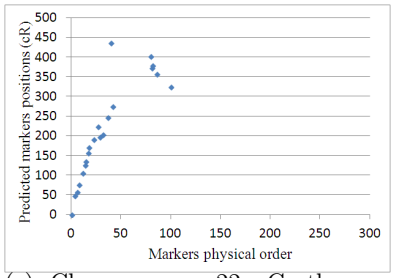
Figure 6: Plot of constructed framework maps for chromosome 21 applying Methods 1, 2 and Carthagene. The actual order of the markers is shown along the x-axis and the predicted order along the y-axis.



(a) Chromosome 22 Method 1



(b) Chromosome 22 Method 2



(c) Chromosome 22 Carthagene Method

Figure 7: Plot of constructed framework maps for chromosome 22 applying Methods 1, 2 and Carthagene. The actual order of the markers is shown along the x-axis and the predicted order along the y-axis.

Table 1: Comparison the Framework maps of Method 1, Method 2 and Carthagene frameworks for Chromosomes 20, 21 and 22.

Chromosome 20	**Input dataset**	**First Method**	**Second Method**	**Carthagene Method**
No. of markers in map	317	125	65	6
No. of Markers in correct order	-	104	64	6
Agreement with physical map	-	83.2%	98.4%	100%
Coverage of physical map	198,257 - 62,730,200	98%	91%	04%
Number of clusters	-	32	32	-
Approximate run time	-	00:00:45	00:00:12	00:08:25
Chromosome 21				
No. of markers in map	281	65	45	20
No. of Markers in correct order	-	50	34	19
Agreement with physical map	-	76%	75%	95%
Coverage of physical map	14,756,329 - 48,078,314	72%	72%	74%
Number of clusters	-	24	24	-
Approximate run time	-	00:00:13	00:00:11	00:04:31
Chromosome 22				
No. of markers in map	268	98	51	21
No. of Markers in correct order	-	77	50	14
Agreement with physical map	-	79%	98%	66%
Coverage of physical map	17,172,704 - 51,007,300	94%	94%	38%
Number of clusters	-	19	19	-
Approximate run time	-	00:00:21	00:00:09	00:04:27

of generated clusters and the total number of computational nodes a user has.

Figure 8 shows the running time of the Carthagene mapping alone for different marker numbers to illustrate why a divide and conquer approach is so important in this problem.

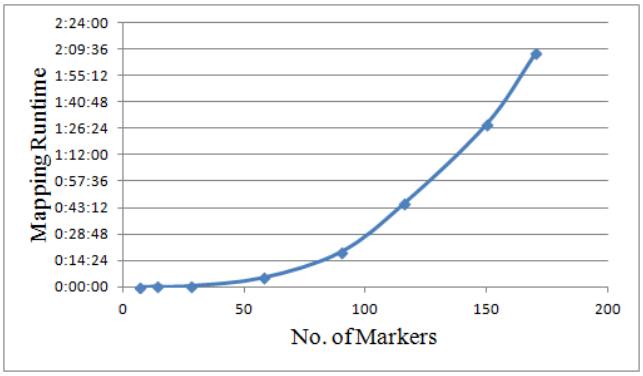


Figure 8: Relationship between No. of markers and mapping complexity in Carthagene.

Traditional approaches for removing unreliable markers are based on resampling and have the complete mapping process built in for all resampled datasets. Chromosomes 20, 21 and 22 have 317, 281 and 268 markers respectively, making a resampling approach on their basis prohibitively slow. The number of markers in a cluster for our approach is never more than 18. 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 that would require as many runs as there are individuals, 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. While resampling runs can also be parallelized, the overall result is clearly that any resampling-based approach would take several orders of magnitude longer than our proposed approach.

The running time for both the RHMapper and Multimap packages does not scale well with both number of markers and individuals. RHMapper generates framework maps in two steps: the first step finds the solid triples markers; the second step assembles the triples to form framework maps. Finding the set of triple markers is a time consuming process: a run based on one hundred markers takes several hours, and having more markers increases the running time dramatically [10]. We tried to generate framework maps using the RHMapper package. Three jobs that were running on three different machines for the three chromosomes 20, 21 and 22, had not completed after 6 days. After that time the jobs had only finished the first step of finding the strong connected triples markers which took 7 hours. The Multimap package takes n steps to construct a framework map of n markers, where all unmapped markers are successively inserted in 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 all markers is a time consuming process. Both packages are far more time consuming than our proposed approach.

5. CONCLUSION

Two 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 two proposed methods efficiently construct high-quality frameworks by applying parallel processing on the generated groups of markers. 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 two proposed methods differ in the first step, the way of constructing a framework map for a subset of markers.

To validate our methods, we applied them to human radiation hybrid data and compared the framework maps with published physical maps. As comparison technique we considered the framework maps generated by the Carthagene tool. We used two metrics in the comparison: the agreement between the maps and 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. We show that the total computation time is lower than for Carthagene and far lower than for other approaches.

Acknowledgment

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

6. REFERENCES

[1] V. Kalavacharla, K. Hossain, O. Riera-Lizarazu, Y. Gu, S. S. Maan, and S. F. Kianian. Radiation hybrid mapping in crop plants. In *Advances in Agronomy*, pages 201–222. Academic Press, 2009.

[2] P. H. Dear. Genome mapping. *eLS.John Wiley and Sons Ltd*, 2001.

[3] S. J. Goss and H. Harris. New method for mapping genes in human chromosomes. *Nature*, 255(5511):680–684, 1975.

[4] P. I. Good. *Resampling Methods A Practical Guide to Data Analysis, 2nd edition*. Birkhduser Boston, 2001.

[5] Y. Ronin, D. Mester, D. Minkov, and A. Korol. Building reliable genetic maps: different mapping strategies may result in different maps. *Natural Science*, 02(06):576–589, 2010.

[6] D. I. Mester, Y. I. Ronin, M. A. Korostishevsky, V. L. Pikus, A. E. Glazman, and A. B. Korol. Multilocus consensus genetic maps (mcgm): Formulation, algorithms, and results. *Computational Biology and Chemistry*, 30(1):12–20, 2006.

[7] S. De Givry, M. Bouchez, P. Chabrier, D. Milan, and T. Schiex. Carhta gene: multipopulation integrated genetic and radiation hybrid mapping. *Bioinformatics*, 21(8):1703–1704, Apr. 2005.

[8] N. E. MORTON. Sequential tests for the detection of linkage. *American journal of human genetics*, 07:277–318, 1955.

[9] O. A. Azzam, L. A. Nimer, C. Chitraranjan, A. M. Denton, A. Kumar, F. M. Bassi, M. J. Iqbal, and S. F. Kianian. Network-based filtering of unreliable markers in genome mapping. In *ICMLA (1)'11*, pages 19–24, 2011.

[10] D. S. L. Stein, L. Kruglyak and E. Lander. Rhmapper, unpublished software. *Whitehead Institute/MIT Center for Genome Research*, 1995.

[11] 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*, 6:384–90, 1994.

[12] R. J. Kinsella, A. KÃd'hÃd'ri, S. Haider, J. Zamora, G. Proctor, G. Spudich, J. Almeida-King, D. Staines, P. Derwent, A. Kerhornou, P. Kersey, and P. Flicek. Ensembl biomarts: a hub for data retrieval across taxonomic space. *Database*, 2011, 2011.

[13] C. Amid. Major submissions tool developments at the european nucleotide archive. *Nucleic acids research*, 21(8):D43–7, Apr. Jan 2012.