# A DATA MINING APPROACH TO RADIATION HYBRID MAPPING

**A Dissertation**
**Submitted to the Graduate Faculty**
**of the**
**North Dakota State University**
**of Agriculture and Applied Science**

**By**

**Raed Issa Seetan**

**In Partial Fulfillment of the Requirements**
**for the Degree of**
**DOCTOR OF PHILOSOPHY**

**Major Department:**
**Computer Science**

**May 2014**

**Fargo, North Dakota**

# North Dakota State University
## Graduate School

**Title**

A Data Mining Approach to Radiation Hybrid Mapping

**By**

Raed Issa Seetan

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

**DOCTOR OF PHILOSOPHY**

SUPERVISORY COMMITTEE:

Dr. Anne Denton
Chair

Dr. Jun Kong

Dr. Simone Ludwig

Dr. Limin Zhang

Approved:

05/09/2014
Date

Dr. Brian M. Slator
Department Chair

# ABSTRACT

The task of mapping markers from Radiation Hybrid (RH) mapping experiments is typically viewed as equivalent to the traveling-salesman problem, which has combinatorial complexity. As an additional problem, experiments commonly result in some unreliable markers that reduce the overall map quality. Due to the large numbers of markers in current radiation hybrid populations, the use of the data mining techniques becomes increasingly important for reducing both the computational complexity and the impact of noise of the original data.

In this dissertation, a clustering-based approach is proposed for addressing both the problem of filtering unreliable markers (framework maps) and the problem of mapping large numbers of markers (comprehensive maps) efficiently. Traditional approaches for eliminating unreliable markers use resampling of the full data set, which has an even higher computational complexity than the original mapping problem. In contrast, the proposed algorithms use a divide-and-conquer strategy to construct framework maps based on clusters that exclude unreliable markers. The clusters of markers are ordered using parallel processing and are then combined to form the complete map. Three algorithms are presented that explore the trade-off between the number of markers included in the framework map and placement accuracy. Since the mapping problem is susceptible to noise, it is often beneficial to remove markers that are not trustworthy. Traditional mapping techniques for building comprehensive maps process all markers together, including unreliable markers, in a single-iteration approach. The accuracy of the constructed maps may be reduced. In this research work, two-stage algorithms are proposed to mapping most markers by first constructing a

framework map of the reliable markers, and then incrementally adding the remaining markers to construct high quality comprehensive maps.

All proposed algorithms have been evaluated on several human chromosomes using radiation hybrid datasets with varying sizes, and also the performance of our proposed algorithms is compared with state-of-the-art RH mapping softwares. Overall, the proposed algorithms are not only much faster than the comparative approaches, but that the quality of the resulting maps is also much higher.

# ACKNOWLEDGMENTS

First, praise is to Allah, the Almighty, on whom ultimately we depend for sustenance and guidance. All praise and gratitude to Allah for endowing me the opportunity, strength, knowledge and ability to completing this dissertation.

Second, I would like to express my special appreciation and thanks to my advisor, Dr. Anne denton for her guidance and patience. I am grateful for her valuable comments that moved the simple ideas into a completed study. Also I would like to thank her for giving me the opportunity to be part of her bioinformatics team, where I learned a lot about my research area. Special thanks to my dissertation committee Dr. Simone Ludwig, Dr. Jun Kong and Dr. Limin Zhang for investing time to review my dissertation and providing me helpful suggestions and comments.

I would like to thank Dr. Shahryar Kianian, Dr. Mohammad Javed Iqbal, Dr. Ajay Kumar, Dr. Filippo Bassi and all the other members of the RH-mapping group for their support and guidance; I gained valuable knowledge and experience from meeting with them. Also I thank Dr. Omar Al-Azzam and Dr. Loai Al-Nemer for giving me a push to get started.

Finally, I owe everything to my family, my parents, my sisters and my brothers who supported and encouraged me at every moment in my life. Without their encouragement and support this study would not have been completed. Also I cannot forget to thank all my friends who encouraged me to continue my research every time I was ready to give up. A special thanks goes to my father, Abu Wael and my uncle, Abu Al-Waleed for everything they have done for me.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ALGORITHMS

# 1. GENERAL INTRODUCTION

This chapter provides the information theoretic context for the bioinformatics algorithm development that was done, in particular, for the Radiation Hybrid Mapping problem. The organization of this chapter is as follows: Section 1.1 introduces the background to the research area and the concepts that are used in this dissertation. Section 1.2 presents the Radiation Hybrid Mapping process. In Section 1.3 the motivation and problem statement are introduced in detail. Section 1.4 presents our contributions of this dissertation, and Section 1.5 presents the organization of this dissertation.

## 1.1. Background

The Traveling Salesman Problem (TSP) is an important optimization problem in computational mathematics [17, 37]. The question to address in TSP is: given a set of cities and the distance between each possible pair, find the best possible way of visiting all the cities exactly once and returning to the starting city. The solution requires an exhaustive evaluation of all possible paths to find the cheapest way of visiting all the cities. However, this solution is usually computationally expensive and unfeasible for a large number of cities. Typically heuristics algorithms are used to find optimal solutions. Traveling-salesman-based algorithms have been used for the Radiation Hybrid (RH) mapping problem for many years [4, 9, 60]. More precisely, the RH mapping problem is equivalent to the symmetric unidimensional wandering salesman problem (a variant of the symmetric traveling salesman problem). The unidimensional wandering salesman problem is a special case of the general TSP where the choice of the first and last cities is free, and all the cities are placed on one coordinate axis only. In this dissertation, we will use the TSP term instead of the unidimensional wandering salesman, because the term TSP is common in most of the genetic and physical mapping literature [11, 18, 44, 48]. In Radiation Hybrid Mapping, the concept of the city is represented by markers, and the concept of distance by the measures of dissimilarity of RHvectors. The objective for RH mapping is to map the markers in a way that maximizes the sum of similarities along the map.

Due to the large numbers of markers in radiation hybrid populations, the use of the data mining techniques becomes increasingly important. A central goal of data mining [19, 25, 62] is to recognize common properties and enumerate patterns from a large data set. Data mining techniques include association rule mining, data classification, and data clustering. Data Clustering is one of the most commonly practiced tasks in data mining, and it groups the set of objects into meaningful clusters, such that similar or related objects are in same cluster [35]. This technique is widely used in bioinformatics applications [59], document categorization [65], pattern recognition [5] and social networking [50].

The goal of clustering is to split objects into smaller groups, such that the similarity between objects are maximized within the same cluster and minimized across clusters. The clustering algorithms can be divided into three main groups [25]: partitional clustering, density clustering and hierarchical clustering. Partitional clustering (e.g., K-means) [40] constructs several disjoint clusters of objects with no hierarchical structure. Density based clustering approaches (e.g., DBSCAN) [16] apply a density criterion to locate the dense regions with high connectivity between the cluster objects and then split them by low density regions. Hierarchical clustering [65] on the other hand, uses a similarity matrix to group objects in a hierarchical structure.

Two approaches are common in hierarchical clustering: agglomeration and divisive. In agglomeration, each object is considered as a singleton cluster then pairs of clusters are merged, recursively, based on a similarity measure to form one big cluster that contains all objects. In the divisive approach, all objects are considered to be in one cluster, and then a splitting method is used to split clusters recursively until singleton clusters are reached. Nevertheless, there is the common belief that clustering a noisy dataset of objects using hierarchical clustering is more effective than other clustering algorithms in terms of clustering quality [40, 65]. Since data from RH mapping experiments often contain noise, in this dissertation we use hierarchical clustering.

Divide-and-conquer is an important strategy for solving sorting problems [8, 36]. This strategy works by recursively breaking down a problem into several sub-problems, where these generated sub-problems are simple enough to be solved directly. Then, the solutions of the sub-problems are combined to give a solution to the original problem. This technique reduces a problem into several smaller independent sub-problems. Parallel processing of these sub-problems can be used to further decrease the overall run time for the solution. In this dissertation, we propose a new approach for solving Radiation Hybrid Mapping computational challenges and constructing reliable maps. The proposed approach uses the divide-and-conquer strategy to reduce the mapping complexity problem. Hierarchical clustering is used to break the problem into sub-problems and to merge the solutions of the sub-problems.

### 1.2. Radiation Hybrid Mapping

Genome mapping [29, 47] is important for finding the order of markers within chromosomes. Map information can also be used in genome sequencing projects [12, 26]. Radiation Hybrid (RH) mapping [21, 29, 34] 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). RH mapping is widely used to map the human genome [61], animals [29] and most recently plants [28, 34]. RH mapping 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.

The process of constructing RH 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, where each fragment contains the attached markers. The presence or absence of each marker in the chromosome fragments is screened and assigned

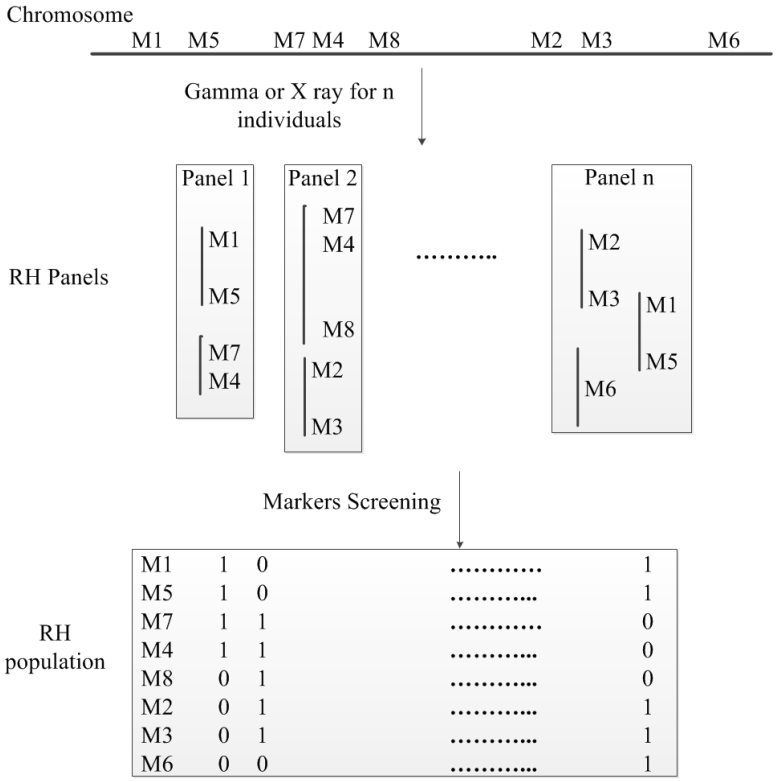


Figure 1. The process of generating radiation hybrid population.

the value 1 if a marker is present, 0 otherwise, "?" is used when the reading is ambiguous. 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 $M \times N$ 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 a RHvector {C1, C2, C3, ..., Cn}. The value of each attribute in the RHvector Ci indicates if that marker is present (1), absent (0) or ambiguous (?). The basic principle of RH mapping is that markers with similar RHvectors are likely to 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 a marker may be present in one and absent in the other. 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 an RH population for 3 markers on 5 panels 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 [6, 24, 63]. The computational complexity is correspondingly high, since for $n$ markers, there are $n!/2$ marker orders.

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** |

Two main types of maps can be constructed using RH mapping: framework maps and comprehensive maps. In framework maps, the subset of the most useful markers is ordered with high confidence, while the comprehensive maps provide the ordering of most markers in the dataset, typically using heuristic algorithms.

### 1.3. Motivation and Problem Statement

In the modern high-resolution RH mapping approaches that motivated the proposed algorithms, the number of markers can become very large, making the traditional algorithms that scale exponentially unsuitable for building RH maps. Moreover, when mapping all

markers together, including unreliable markers, the accuracy of the constructed maps may be reduced [1, 53].

Traveling-salesman-based algorithms have been used for the RH mapping problem for many years [4, 9, 60]. Traditional approaches for constructing framework maps [43, 51] depend on resampling analysis [20] to iteratively filter out markers that cannot be mapped consistently. Such approaches require repeating the already computationally expensive mapping task for each resampled data set. An alternative approach is presented in [11], 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. Both approaches are time consuming and do not scale well with the dataset size.

Many algorithms have been implemented to build comprehensive maps [3, 11, 41, 57]. For a large number of markers, the run time for heuristic approaches typically scales exponentially, which makes these algorithms unsuitable. Moreover, the mapping process fundamentally relies on the distances between immediate neighbors, and an unreliable marker can disrupt the overall order. If unreliable markers are placed on the map in the early stages of map construction, other markers will likely be mapped incorrectly. Therefore, for a noisy dataset, limiting mapping to reliable markers first is not only beneficial from a performance perspective but also in the interest of map quality.

To overcome these challenges, we propose a clustering-based approach that constructs framework and comprehensive maps in a short time taking into account the existence of unreliable markers. The proposed approach uses a divide-and-conquer strategy to construct reliable maps based on clusters. The clustering helps break the mapping problem into several small sub-problems and associate markers with their most highly linked neighbors. Similarities to other markers, which could otherwise result in unstable maps, are ignored. The proposed approach uses the LOD (Logarithms of odds -base 10) scores to measure similarities between markers [46]. Figure 2 shows an example of markers that are far apart

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

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).

The main motivation of this research is to investigate the role of the divide-and-conquer strategy based on clustering for building reliable and scalable algorithms to solve markers ordering problems. Basically, we propose efficient algorithms that shift the mapping process from simultaneously ordering a large number of markers to mapping few neighbors markers at a time, and reduce the effect of the noisy markers. The traditional mapping algorithms take a long time to construct maps and do not handle noisy data. However, the proposed algorithms apply clustering to reduce the complexity of the mapping process to build solid maps and filter out unreliable markers in a short time.

### 1.4. Contributions

This dissertation investigates the problem of building high quality radiation hybrid maps in a short time. The proposed approaches incorporate clustering analysis to filter out unreliable markers and build radiation hybrid maps for large numbers of markers in short time. The contributions are:

1. A clustering-based approach for constructing radiation hybrid maps is presented. We present three algorithms that explore the trade-off between the number of markers included in the framework map and placement accuracy. The general pipeline for the three proposed algorithms consists of three steps:

   1.1 The filtering and mapping step: three variant methods are proposed to construct maps for each subset of markers.

   1.2 The merging step: a method for merging the constructed framework maps of all groups to form the whole chromosome framework is proposed.

   1.3 The polishing step: a local improvement method is proposed for the final constructed framework maps.

2. An iterative framework radiation hybrid mapping algorithm is presented. This proposed algorithm maps most of the markers in two phases taking into account the stability of markers. The first phase maps only the most stable markers creating an initial framework. The second phase iteratively maps the less stable markers on the constructed framework map when the overall ordering can no longer be disrupted.

3. A noise-aware algorithm for building radiation hybrid maps is presented. The proposed algorithm uses both the exhaustive and heuristics algorithms to build solid maps by dividing the mapping process into two levels. The first level constructs several small groups of markers and then uses the exhaustive search to build a map for each group. Then the heuristics algorithms are used to merge these maps to construct a skeleton map. The second level uses the skeleton map to map the remaining markers to form comprehensive maps.

4. All proposed algorithms have been evaluated on several human chromosomes using radiation hybrid datasets with varying sizes. The proposed constructed maps have been compared with the physical maps of the corresponding chromosomes, and also compared with maps generated using state-of-the-art mapping softwares.

### 1.5. Organization of the Dissertation

This dissertation is a paper-based version, where each chapter has been derived from papers published during the PhD work. This dissertation is organized into five chapters:

1. In **Chapter 2**, a clustering based approach for radiation hybrid mapping is introduced. We present three algorithms to construct framework maps based on clusters that exclude unreliable markers. This chapter is derived from the publication, which is accepted in *Proceedings of the 12th International Workshop on Data Mining in Bioinformatics. ACM, 2013* which is titled as *"A fast and scalable clustering-based approach for constructing reliable radiation hybrid maps"*. Furthermore, the extended version of the paper is published in *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 2014*.

2. In **Chapter 3**, an approach for building comprehensive maps is described. The presented approach maps most markers by first mapping the most reliable markers, creating a framework map, and then incrementally adding the remaining markers. This work has been published in *Proceedings SIAM International Conference on Data Mining (SDM14)* which is titled *"Iterative Framework Radiation Hybrid Mapping"*.

3. In **Chapter 4**, an enhanced mapping approach is proposed that handles noise markers and builds solid comprehensive maps. The proposed approach divides the mapping process into two-layers, where in the first layer, 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. This work has been published in *"ACM Conference on Bioinformatics, Computational Biology and Health Informatics (ACM BCB 2014)"* which is titled *"A Noise-Aware Method for Building Radiation Hybrid Maps"*.

4. In **Chapter 5**, we conclude the dissertation work and provide some future work.

# 2. RELIABLE RADIATION HYBRID MAPS: AN EFFICIENT SCALABLE CLUSTERING-BASED APPROACH

In Chapter 1, we introduced the radiation hybrid mapping process and its computational complexity challenges for mapping large numbers of markers. In Chapter 2, a clustering-based approach is discussed and applied for filtering and mapping a subset of markers to construct reliable framework maps.

The remainder of this chapter is organized as follows: Section 2.1 briefly introduces the problem of constructing framework maps. Section 2.2 presents the related work in the area of constructing framework maps. In Sections 2.3, 2.4, 2.5 and 2.6 our proposed approach is introduced in detail. Section 2.7 presents the experimental evaluation of the proposed approach, and Section 2.8 concludes the chapter.

## 2.1. Introduction

Radiation Hybrid (RH) mapping provides a means of ordering markers on a chromosome [21]. The process of mapping markers is equivalent to the traveling salesman problem, and thereby has combinatorial complexity. Harnessing this computational complexity for a large number of markers is one problem addressed in this chapter. A related problem is that the quality of mapping information differs across markers. If unreliable markers are included in the mapping process, the overall accuracy of the map is decreased.

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 [20] to iteratively filter out markers that cannot be mapped consistently [42, 43, 45, 51]. Such approaches require repeating the already computationally expensive mapping task for each resampled data set. An alternative approach is presented in [11], 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 [46] 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 [1]. We use agglomerative hierarchical clustering [55], 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 [38, 64]. 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.

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.

### 2.2. Related Work

Conventional approaches for building skeleton maps by filtering out unreliable markers [43, 51] depend mainly on resampling analysis [20]. 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 [11, 41, 57]. RHMAPPER [57] is one of the commonly used packages, and it uses a hidden Markov model (HMM) [7] 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 [41] 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 the Multimap reduces the mapping time compared with the step-by-step approach that serves as comparison method in [41], 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 [11], which provides several algorithms for ordering a set of markers, i.e. simulated annealing [49], taboo search [10] and LinKernighan heuristic [27]. Carthagene also uses an incremental insertion procedure (Buildfw command) to construct framework maps. The Buildfw command tries to build framework maps as follows: Firstly, 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. Secondly, 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 that are 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 [11, 41, 57] 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 [2]. Framework maps that are constructed using these techniques have the additional shortcoming that they may not cover the whole chromosome.

### 2.3. 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

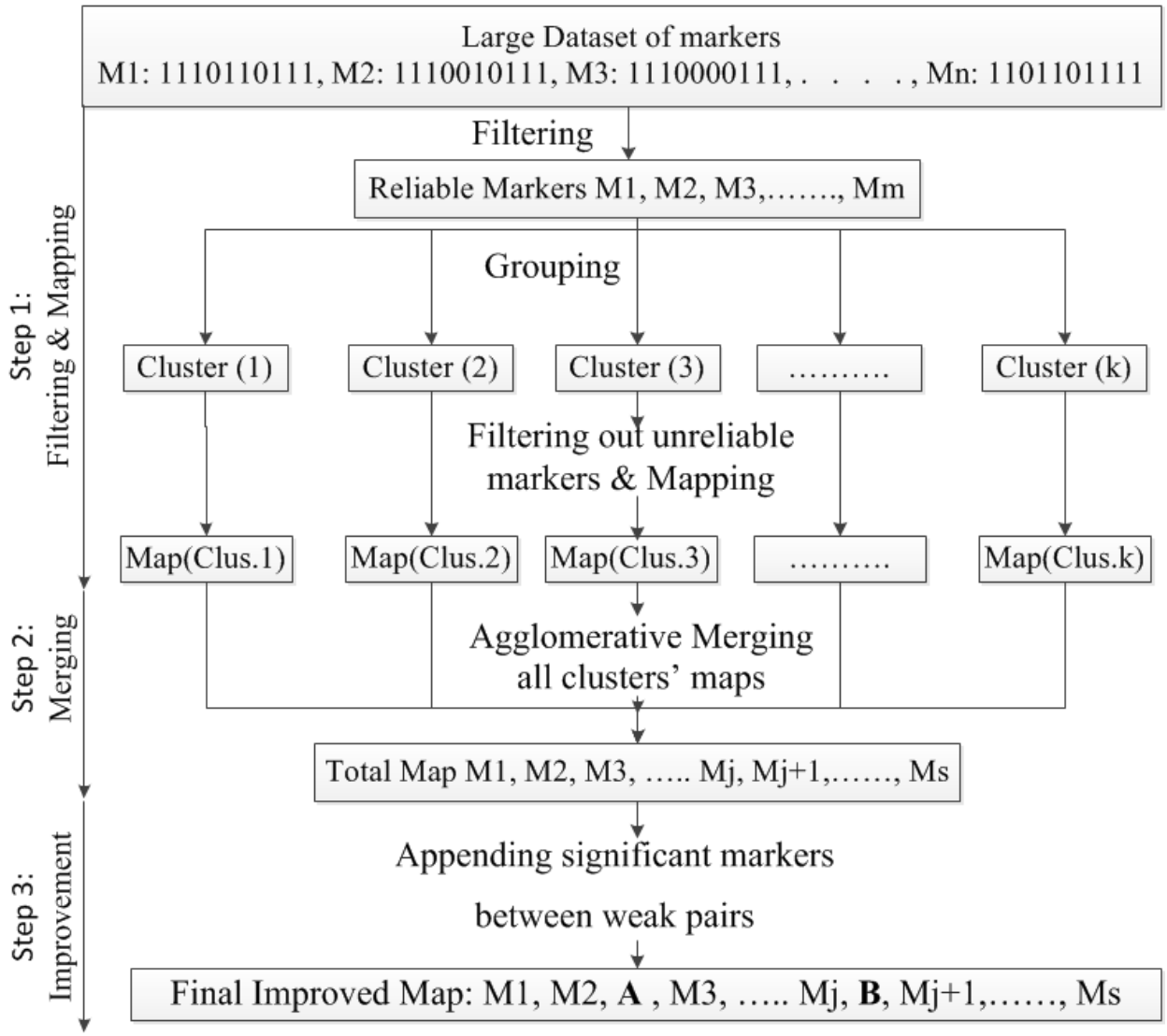


Figure 3. Systematic proposed approach.

pairs of markers to enhance the overall map. Figure 3 shows the systematic work flow for the proposed approach.

### 2.4. 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

**Algorithm 1** Step 1: Filtering and Mapping

```
Data : RHData /* NoOfMrk by NoOfIndv matrix */
Data : 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
        end
        BestMap = GetMapUniquePositions(BestMap)
        BestMap ∈ BestMaps
    end
end
return BestMaps
```

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.

### 2.4.1. 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 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.

### 2.4.2. Mapping Strategy

We use the Carthagene tool [11] 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, 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 check if a better map can be achieved. Then we remove all inconsistently ordered markers

**Algorithm 2** Step 2: Merging Maps

**Data** : $RHData$ /* NoOfMrk by NoOfIndv matrix */
**Data** : $T$ /* LOD threshold */
**Data** : $BestMaps$ /* 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) ∈ FWMap
                RemoveConnectedBoundaries(MapsBoundaries, E1, E2)
                BoundariesLabels(M1)=BoundariesLabels(M2)
            end
        end
    end
end
return FWMap
```

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.

### 2.4.3. 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 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

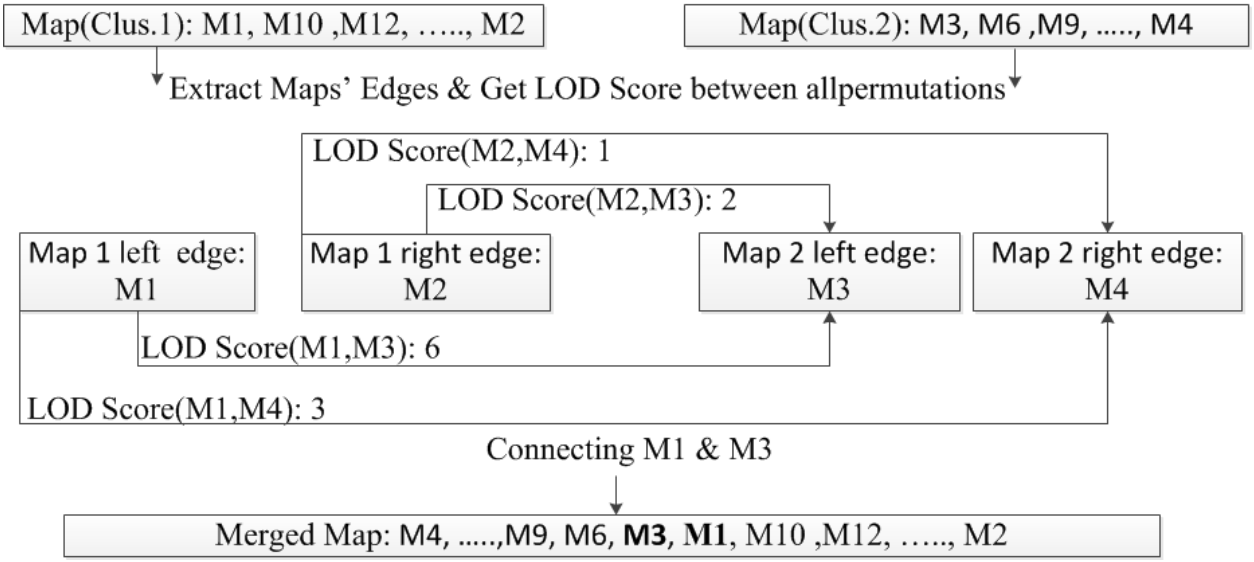

Figure 4. Merging two clusters.

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 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.

### 2.4.4. 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 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. 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 the marker will be placed in the middle of the gap.

Figure 5 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 $M_{k+1}$.

### 2.5. 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 as for Method 1, as discussed in Sections 2.4.3 and 2.4.4, respectively.

#### 2.5.1. 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 triplet markers, we use Carthagene. The Buildfw command in Carthagene finds triplets of markers such that all alternatives orders

**Algorithm 3** Step 3: Improvement Method

**Data** : $RHData$ /* NoOfMrk by NoOfIndv matrix */
**Data** : $S$ /* LOD threshold */
**Data** : $BestMap$ /* 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**
**end**
**return** $ImpMap$.

have a loglikelihood not within a given threshold of the best order. The 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 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.

### 2.6. 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 [54] 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

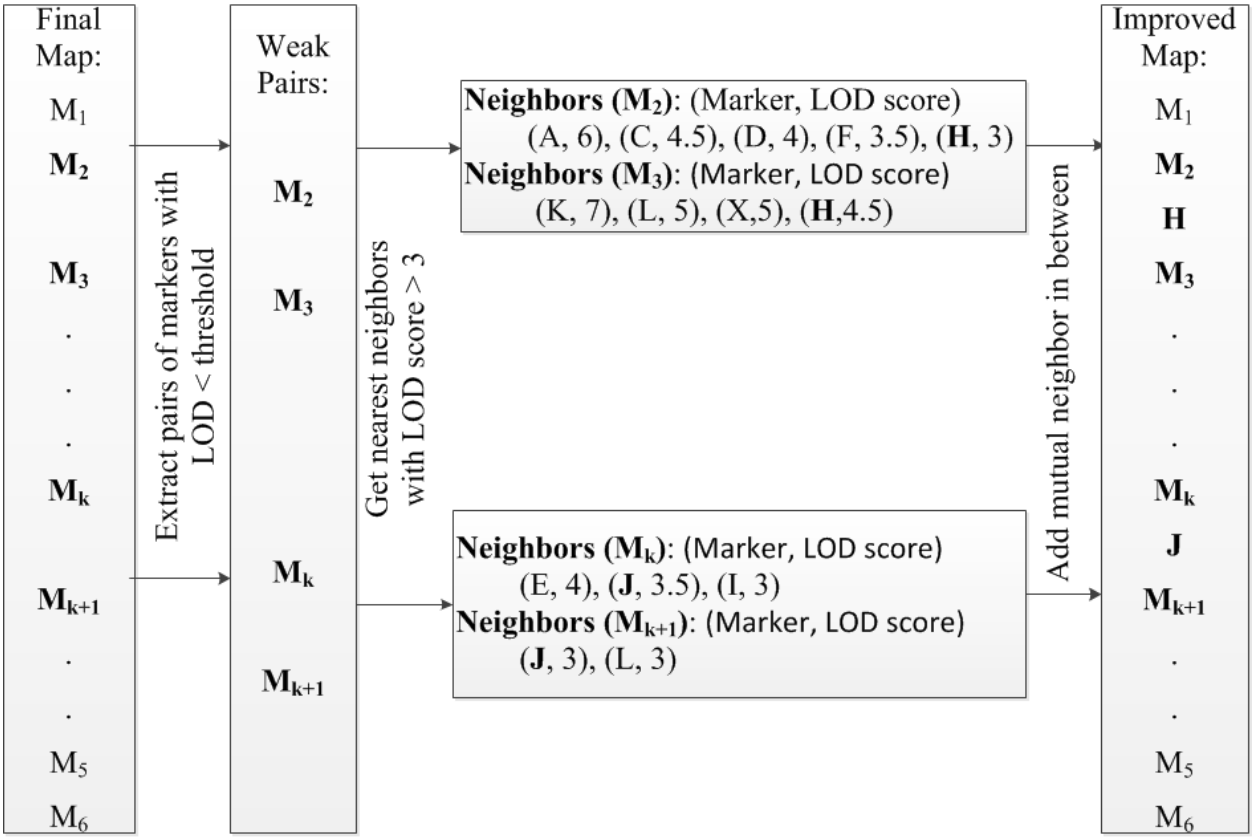

Figure 5. Local improvement step.

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 2.4.3 and 2.4.4, respectively.

### 2.6.1. Iterative Framework Constructing

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

**Algorithm 4** Step 1: Filtering and Framework Constructing

```
Data : RHData /* NoOfMrk by NoOfIndv matrix */
Data : 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
        BestMap = GetMapUniquePositions(BestMap)
    end
    BestMap ∈ BestMaps
end
return BestMaps
```

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.

## 2.7. Experiments and Results

### 2.7.1. 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 [58] and TNG [39] panels produced by Stanford University and the Genebridge 4 [23] 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 seven different chromosomes with varying number of markers, showing the performance pattern over the increasing number of markers. 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 Ensemble site [30], and the RH data set are downloaded from EMBL-EBI site [3].

**Algorithm 5** Step 1: Iterative Framework Constructing

```
Data : RHData /* NoOfMrk by NoOfIndv matrix */
Data : T /* LOD threshold */
Data : AddThres, KeepThres /* Map growing thresholds */
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=Delta_likelihood(BestMap,SecondBestMap)
                if DL < KeepThres then
                    RemoveMarker(BestMap, Mrk)
                end
            end
        end
        (E1, E2) = GetFirstandLastMarkers(BestMap)
        if BestMap is ∅ then
            (E1,E2) = FindtheWeakestPair(C)
            BestMap = (E1, E2)
        end
        BestMap = GetMapUniquePositions(BestMap)
    end
    BestMap ∈ BestMaps
end
return BestMaps
```

### 2.7.2. Evaluation of the Approach

Before we start discussing the proposed method and compare it with a state-of-the-art model, Figure 7 shows the results of mapping a few markers using both the proposed approach and a state-of-the-art model. In this figure three maps are drawn for a segment of human Chromosome 1. The Autograph tool [14] is used to visualize the maps. The middle map is the reference map, showing the true physical locations of markers on the chromosome. The left map in the figure represents the map generated using the Carthagene tool, where

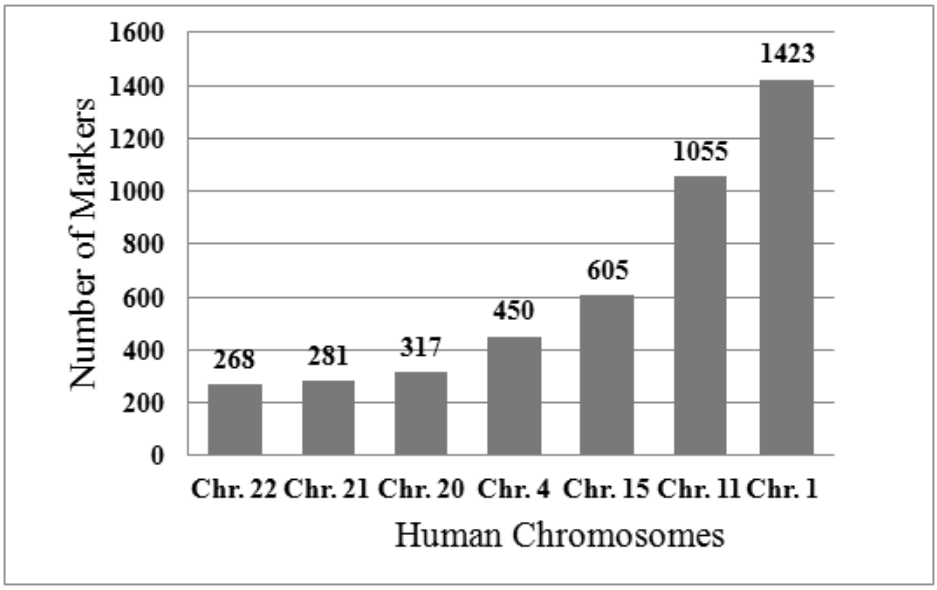


Figure 6. Experiments datasets: human chromosomes attached number of markers.

all markers are mapped without any filtering. The right map is the map that is constructed for only the set of reliable markers after applying our clustering-based approach.

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 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.

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 in black cells) instead a sequence of consecutive forward steps as shown in Figure 8(b).

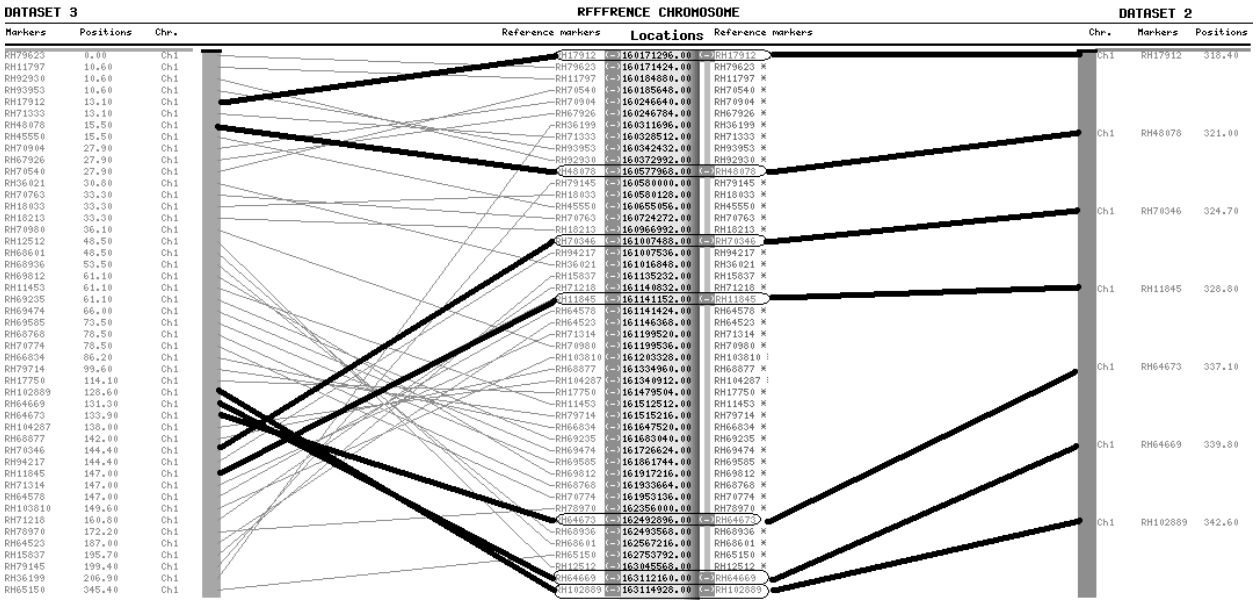


Figure 7. Comparison between three maps created for the same segment of human chromosome 1. Left: map created for all markers, the reliable and unreliable markers. Middle: the physical map for all markers. Right: framework map created for only the reliable markers. Good markers order conservation is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

Two metrics are 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 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 15 show how well the proposed framework map orders agree with the physical maps for the seven chromosomes. For each chromosome, four plots are drawn: (a) framework map constructed using the

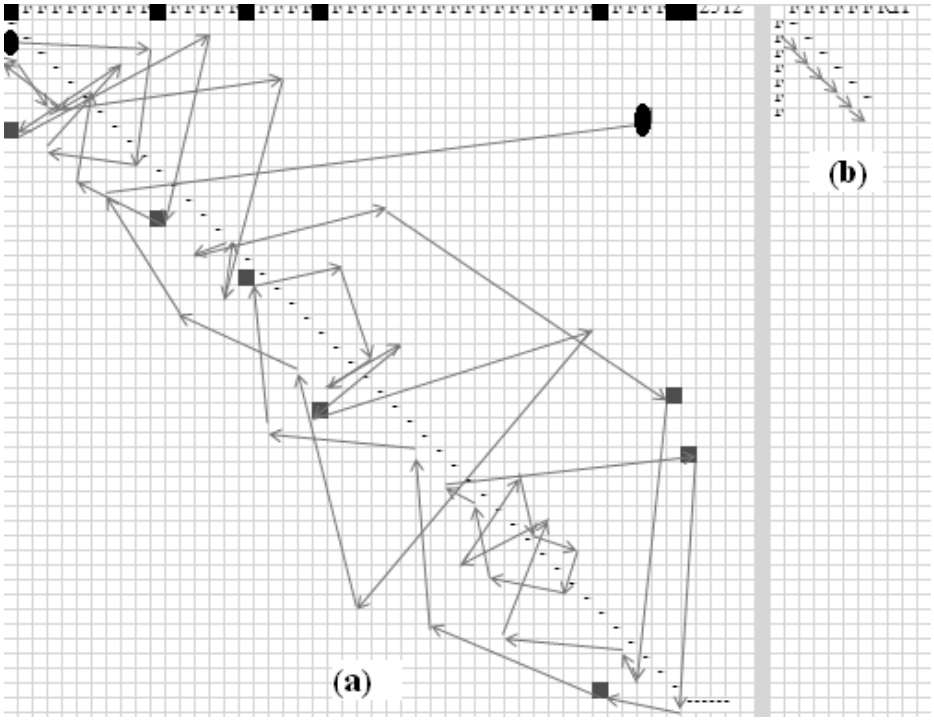


Figure 8. Neighborhood LOD score matrix for a segment of human Chromosome 1 shows step by step process of constructing the map, (a) for all markers, (b) for the reliable markers. Markers are listed in the matrix according to their true physical order. The first and last markers in the constructed map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

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 [11]. 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. Chromosomes 22, 21 and 20 can be seen in Figures 9, 10 and 11, respectively.

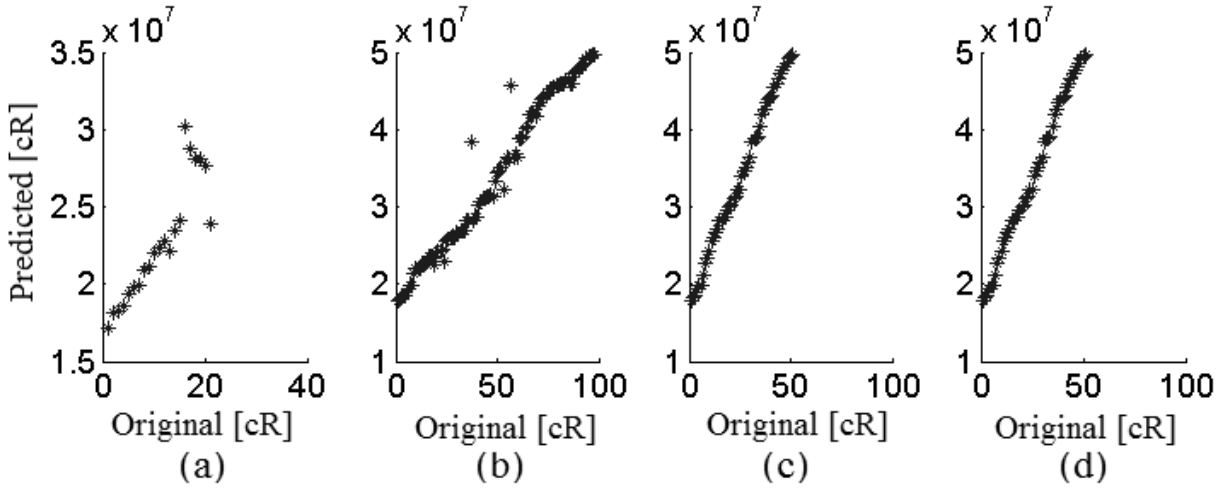


Figure 9. Original permutation of the markers along the x-axis; predicted marker locations (centiRay) along the y-axis. The four maps are constructed for Chromosome 22 using (a) Carthagene Method, (b) Method 1, (c) Method 2 and (d) Method 3.

For Chromosome 20, Carthagene is only able to map 6 markers, while each of our three methods map more than 10 times as many. Figure 11 illustrates this difference. 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. Figures 9(a) and 10(a) show a visual representation of the Carthagene framework maps for Chromosomes 22 and 21, respectively, and Figure 11(a) for Chromosome 20, where the partial coverage with the physical maps can be seen clearly along the y-axis in plot 11(a) $2.3 \times 10^7$ to $2.65 \times 10^7$ compared to our proposed maps in plots 11(b, c and d) 0 to $6.4 \times 10^7$.

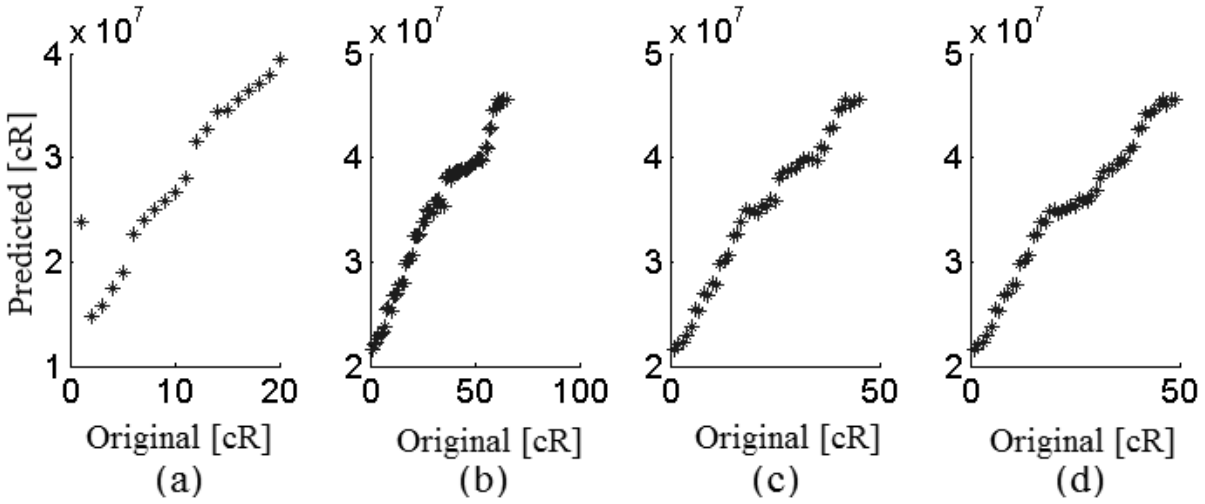


Figure 10. Original permutation of the markers along the x-axis; predicted marker locations (centiRay) along the y-axis. The four maps are constructed for Chromosome 21 using (a) Carthagene Method, (b) Method 1, (c) Method 2 and (d) Method 3.

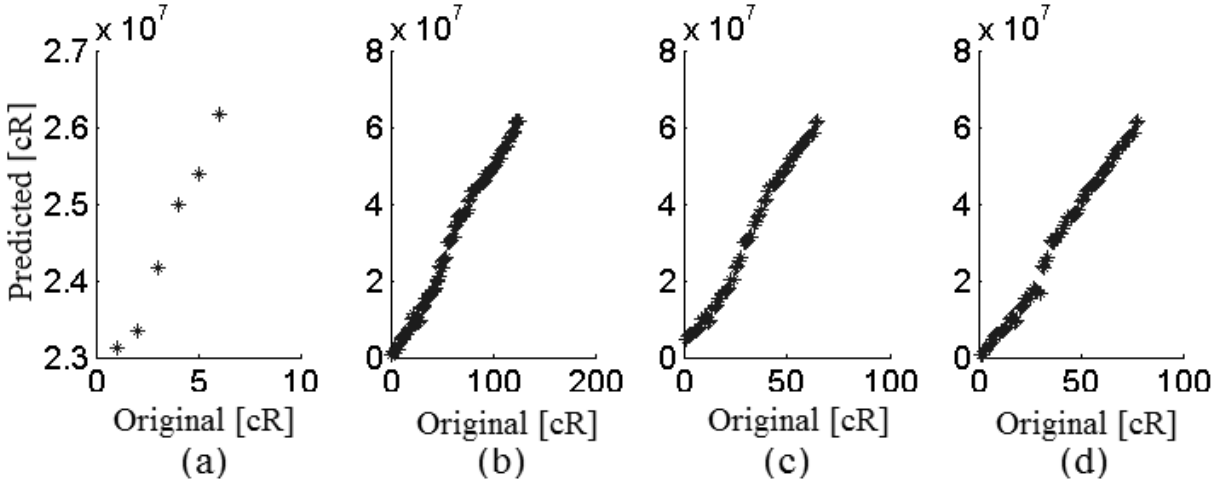


Figure 11. Original permutation of the markers along the x-axis; predicted marker locations (centiRay) along the y-axis. The four maps are constructed for Chromosome 20 using (a) Carthagene Method, (b) Method 1, (c) Method 2 and (d) Method 3.

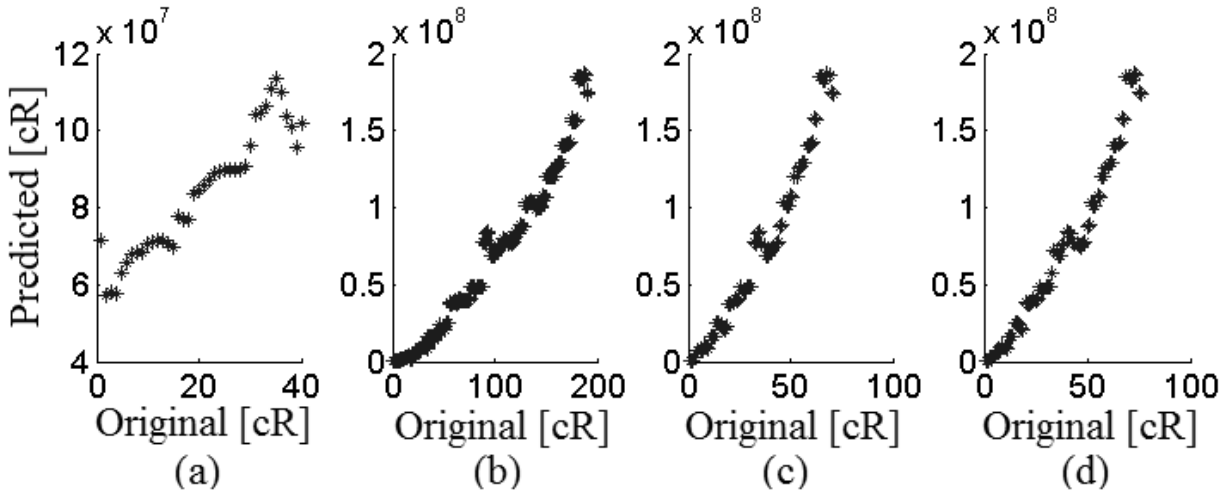


Figure 12. Original permutation of the markers along the x-axis; predicted marker locations (centiRay) along the y-axis. The four maps are constructed for Chromosome 4 using (a) Carthagene Method, (b) Method 1, (c) Method 2 and (d) Method 3.

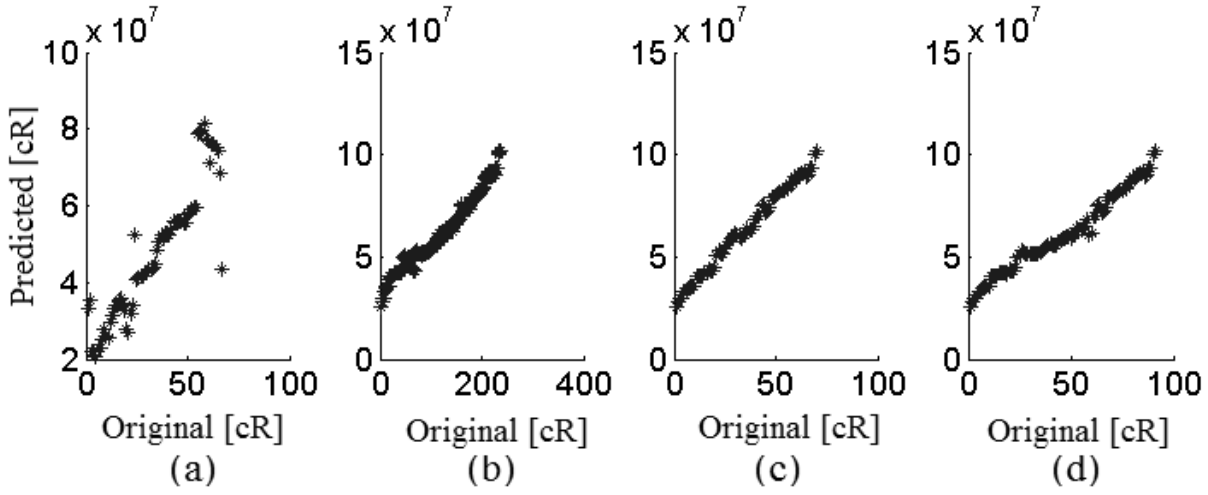


Figure 13. Original permutation of the markers along the x-axis; predicted marker locations (centiRay) along the y-axis. The four maps are constructed for Chromosome 15 using (a) Carthagene Method, (b) Method 1, (c) Method 2 and (d) Method 3.

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%. Figures 12 and 13 show the agreement and coverage of the framework maps constructed for Chromosomes 4 and 15 applying the Carthagene 12(a) and 13(a) and proposed Method 1 (b), Method 2 (c) and Method 3 (d). In Figure 13, 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 13(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 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%. Figure 14 provides a visual representation of the constructed framework maps.

The plots in Figure 15 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 15(a) shows that the Carthagene map has several misordered markers at the beginning of the map, where some markers are

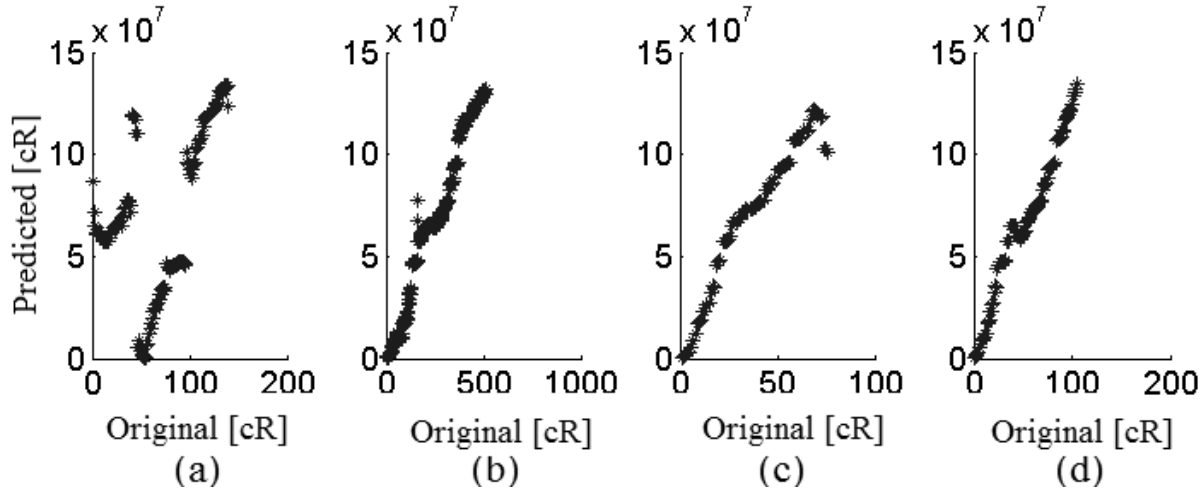


Figure 14. Original permutation of the markers along the x-axis; predicted marker locations (centiRay) along the y-axis. The four maps are constructed for Chromosome 11 using (a) Carthagene Method, (b) Method 1, (c) Method 2 and (d) Method 3.

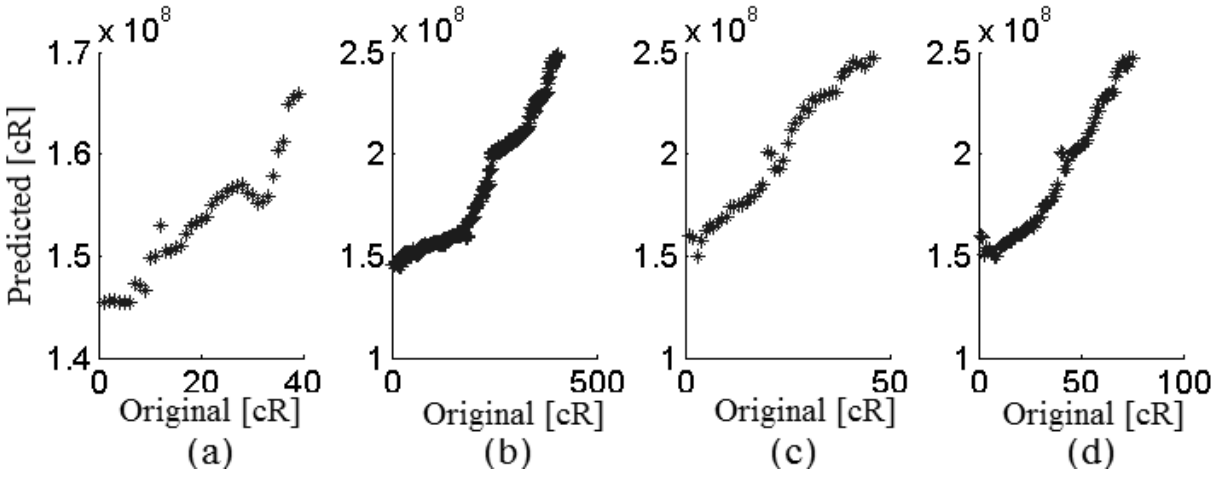


Figure 15. Original permutation of the markers along the x-axis; predicted marker locations (centiRay) along the y-axis. The four maps are constructed for Chromosome 1 using (a) Carthagene Method, (b) Method 1, (c) Method 2 and (d) Method 3.

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

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

| | | Carthagene_Method | | | | First_Method | | | | Second_Method | | | | Third_Method | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Chomosome | Markers | Mapped Markers | Run Time | Agreement Percentage | Coverage Percentage | Mapped Markers | Run Time | Agreement Percentage | Coverage Percentage | Mapped Markers | Run Time | Agreement Percentage | Coverage Percentage | Mapped Markers | Run Time | Agreement Percentage | Coverage Percentage |
| **22** | 268 | 21 | 0:04:27 | 0.66 | 0.38 | 98 | 0:00:21 | 0.79 | 0.94 | 51 | 0:00:09 | 0.98 | 0.94 | 51 | 0:00:09 | 0.98 | 0.93 |
| **21** | 281 | 20 | 0:04:31 | 0.95 | 0.74 | 65 | 0:00:13 | 0.76 | 0.72 | 45 | 0:00:11 | 0.75 | 0.72 | 49 | 0:00:11 | 0.88 | 0.72 |
| **20** | 317 | 6 | 0:08:25 | 100 | 0.05 | 125 | 0:00:45 | 0.83 | 0.98 | 65 | 0:00:12 | 0.98 | 0.91 | 78 | 0:00:15 | 0.87 | 0.98 |
| **4** | 450 | 40 | 0:23:12 | 0.67 | 0.16 | 191 | 0:00:49 | 0.59 | 0.91 | 71 | 0:00:05 | 0.63 | 0.91 | 76 | 0:00:06 | 0.57 | 0.91 |
| **15** | 605 | 67 | 1:28:40 | 0.53 | 0.13 | 236 | 0:02:20 | 0.58 | 0.94 | 70 | 0:00:12 | 0.78 | 0.94 | 91 | 0:00:12 | 0.77 | 0.94 |
| **11** | 1055 | 140 | 14:52:53 | 0.45 | 0.28 | 513 | 0:10:37 | 0.56 | 0.75 | 80 | 0:01:16 | 0.77 | 0.95 | 105 | 0:01:19 | 0.8 | 0.75 |
| **1** | 1423 | 39 | 10:30:53 | 0.66 | 0.08 | 407 | 0:44:52 | 0.58 | 0.41 | 62 | 0:04:53 | 0.76 | 0.35 | 75 | 0:04:53 | 0.76 | 0.35 |

### 2.7.3. 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 clusters is small in comparison with the overall number of markers. Figure 16 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.

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, Chr. 22, which has 268 markers, and ending with a long Chromosome, Chr. 1, which has 1423 markers.

Table 2 shows the run time for mapping different chromosomes with 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

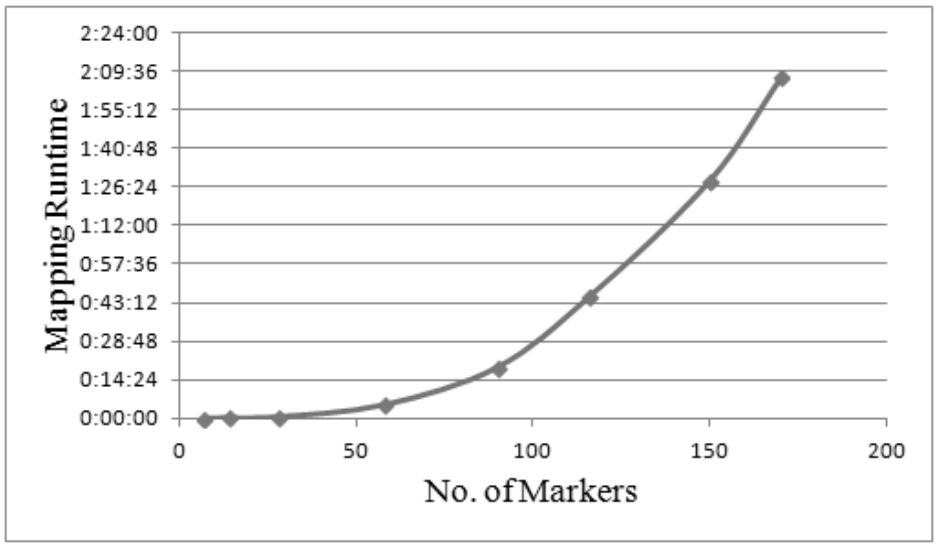


Figure 16. Relationship between No. of markers and mapping complexity in Carthagene.

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 [43, 51] 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 prohibitively a 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 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. 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 [57]. 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 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.

## 2.8. Conclusion

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 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 applied them to human radiation hybrid data and compared the framework maps with published physical maps. As a 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 for chromosomes with 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 chapter, 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.

# 3. ITERATIVE FRAMEWORK RADIATION HYBRID MAPPING

In Chapter 2, a clustering-based approach is discussed and applied for constructing framework maps with only the reliable subset of markers. In Chapter 3, an iterative framework mapping approach is presented for mapping most of the markers by first applying the clustering-based approach that is discussed in Chapter 2 creating a framework map and then incrementally adding the remaining markers.

This chapter is organized as follows: Section 3.1 briefly introduces the comprehensive mapping problem. Section 3.2 presents the related work in the area of constructing comprehensive maps. In Sections 3.3, 3.4 and 3.5 our proposed approach is introduced in detail. Section 3.6 presents the experimental results of the proposed approach, and Section 3.7 concludes the chapter.

## 3.1. Introduction

Genome maps show the order of markers on the chromosomes of a species and the estimated distances between markers [29]. Radiation Hybrid (RH) mapping [21] is a widely used mapping technique in which a chromosome is broken into random fragments using radiation. Then, the presence or absence of a marker is screened to generate a radiation hybrid population. Markers can be genes or any simple short fragments of DeoxyriboNucleic Acid (DNA) sequence that appear only once in the genome.

The mapping problem can be viewed in analogy to the traveling salesman problem. If the distances between each pair of markers were known exactly, the shortest path that covers all markers would identify the order of markers on the chromosome. Traveling-salesman-based algorithms have been used for this problem for many years [4, 9, 60]. These algorithms assume that the distance information is equally trustworthy for each marker, and that only the distances among a small number of nearest neighbors are meaningful because there would only be a few markers on each fragment. Neither of these assumptions holds well for current experiments that use a large number of markers to produce high-resolution maps. The use

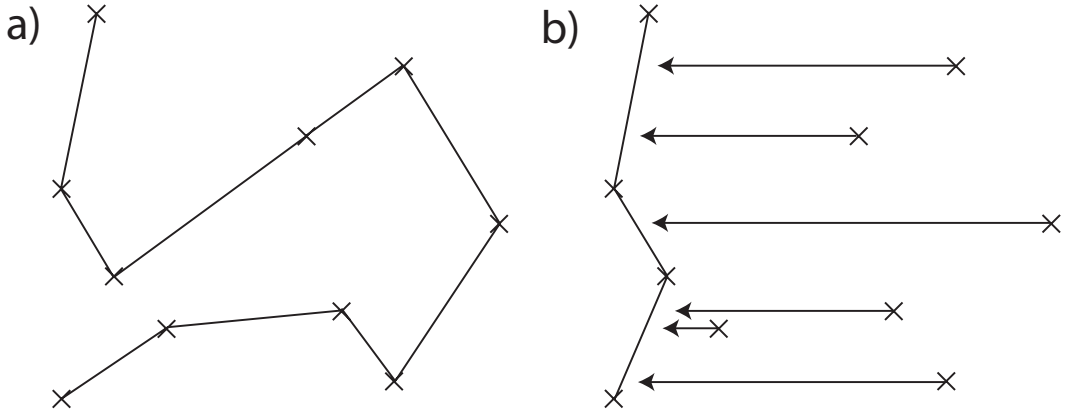

Figure 17. Toy example illustrating the process of mapping a marker as a travel salesman problem (left) and using the proposed approach (right).

of high-throughput experiments that have higher rates of errors further affects algorithm design.

We present an approach that only uses the most stable markers for creating an initial framework. This framework is then iteratively filled in by less stable markers. The presence of the framework prevents unstable markers from disrupting the overall order of markers.

Figure 17 shows how markers that have a large distance to all others can disrupt the overall ordering as determined using a traveling-salesman-based algorithm (left). Assuming that the correct order is from top to bottom, there are many markers that are covered in reverse order. A better approach, in which only the most reliable markers are considered in the first iteration is shown on the right. In that approach, less reliable markers are added later, when the overall ordering can no longer be disrupted. A further advantage of this approach is that the original mapping is done on a much smaller set of markers and is, thereby, computationally much more efficient. Considering the combinatorial complexity of the traveling salesman problem, this advantage is very important for the construction of high-resolution maps.

Mathematically, the radiation hybrid mapping problem can be described by a $M \times N$ matrix, where $M$ is the number of markers and $N$ is the number of individual organisms in

the mapping population. The presence or absence of each marker in each individual can be seen as a binary attribute that is 1 when the marker is present and 0 when it is absent. The assumption in RH mapping is that deleted and retained chromosome sections, respectively, are substantially longer than the average distance between markers. When that is the case, neighboring markers have higher probability to be retained or lost together than markers that are distant on the chromosome. Physical distances between markers can be calculated on that basis.

Two main types of maps can be constructed using RH mapping: framework maps and comprehensive maps. In framework maps, the subset of the most useful markers is ordered with high confidence. Traditional approaches for constructing framework maps [43, 51] depend on resampling analysis [20]. An alternative approach uses an incremental insertion procedure [11], both approaches are time consuming and do not scale well with the dataset size. A clustering-based approach proposed in [54] constructs framework maps in a short time.

Comprehensive maps identify the order of all markers in a dataset and are typically constructed using heuristic algorithms. Since there are $n!/2$ possible marker orderings, even heuristic approaches typically scale exponentially. In the modern high-resolution RH mapping approaches that motivated the proposed algorithm, the number of markers can become very large, making algorithms that scale exponentially unsuitable. In this chapter we address the problem of iteratively constructing comprehensive maps based on a framework map to address the problem of scaling to a large datasets of markers.

A related problem is that when mapping all markers together, including unreliable markers, the accuracy of the constructed maps may be reduced [1]. The traveling salesman problem fundamentally relies on the distances between immediate neighbors, and an unreliable marker can disrupt the overall order. If unreliable markers are placed on the map in the early stages of map construction, other markers will likely be mapped incorrectly. Therefore, for a noisy dataset, limiting mapping to reliable markers first is not only beneficial from a

performance perspective but also in the interest of map quality. Once framework information is known and used, insertion of additional markers no longer depends on nearest-neighbor distances alone, but rather includes global information about the markers that have already been placed.

The Carthagene tool [11] offers a single-iteration approach for constructing a comprehensive map from a framework map by placing unmapped markers in their best-matching interval. One problem with the Carthagene tool is that the constructed maps only have few markers in the final constructed map and that coverage of the chromosome is partial. In addition, the reported best position of the unmapped markers commonly has multiple markers in the same interval, so there is a need for further steps to map the markers in their intervals to build the comprehensive map.

The proposed approach shifts the mapping process from simultaneously ordering a large number of markers to mapping few markers at a time. There are two main tasks: 1) the grouping process, where a large number of markers is divided into small groups, and the second task, 2) the local mapping process, where markers are distributed on the framework map's intervals, and the markers in each interval are mapped separately. Mapping only few markers in each interval can be done in a short time. Figure 17 shows the concept behind using a framework map to map a large number of markers in an efficient way.

Markers are placed in groups based on their LOD (logarithms of odds -base 10) scores [46]. Mapping these groups of markers and merging the constructed maps to form a solid framework map with only reliable markers is fast and was described in [54]. The first step is also parallelizable. The remaining markers are iteratively mapped based on the framework map constructed in the first step. The stability of the remaining markers is checked twice. Only markers that can be mapped reliably, given the framework map, are considered to be added to the map. This further improves the overall stability of the map.

### 3.2. Related Work

Several software packages have been proposed for constructing radiation hybrid maps [11, 41, 52, 57]. RHMapper package [57] constructs framework maps in two steps: 1) extracting all possible strongly connected triplet of marker, and 2) combining the triplet of markers based on the overlapping between triples. The algorithm of the RHMapper package does not scale well with the number of markers. For instance, extracting marker triplets from one hundred markers takes several hours, in addition to the time needed for the overlapping step to assemble all marker triplets.

Another common package is Multimap [41]. This package constructs framework maps by starting with the strongest connected pair of markers and then iteratively adding one marker at a time. This process does not guarantee construction of a solid framework that covers the whole chromosome.

The Carthagene tool [11] provides multiple approaches for constructing comprehensive and framework maps and several algorithms have been implemented to order markers, such as simulated annealing [49], taboo search [10] and LinKernighan heuristic [27]. Carthagene also provides a nearest neighbor stepwise marker insertion method (pattern expansion algorithm) for ordering markers from scratch.

The Buildfw command is an incremental insertion procedure provided by Carthagene that tries to build a good framework map and may also be adjusted to do some post-processing steps to build a comprehensive map. This command works as follows: First, it builds a framework map starting by selecting a triplet of markers where the difference between the likelihood of the best map and the second best map that can be built with these three triplet of markers is the greatest. Secondly, the command inserts the remaining markers incrementally at each possible position. 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 can 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 map and will be used for the next steps.

The Buildfw command uses a flag to indicate if post-processing is applied to the markers that cannot be inserted on the framework map, the values of the flag are: 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.

The main drawback of the Buildfw command is that the number of markers on the constructed framework map is too small with an incomplete coverage of the chromosome. Adjusting the Buildfw command to build comprehensive maps using the constructed framework maps (with partial coverage and few markers) does not help building good comprehensive maps. Moreover, a large number of markers will be just reported with their best intervals on the framework map without mapping on their intervals. The results in [54] show that the constructed framework maps discard too many markers from the final maps and cover the chromosomes partially. Using these poor framework maps with only few intervals between framework markers, and a large number of remaining markers to be mapped, results in reporting large numbers of markers in each interval, and thus is slow when mapping the large number of markers in the intervals.

### 3.3. Proposed Approach

The proposed approach constructs comprehensive maps in two steps: 1) the first step constructs a framework map, where only the most reliable subset of markers is extracted and grouped together into several groups; then a map is constructed for each group of markers; and finally the groups' maps are combined to form a solid framework map. Figure 18 shows

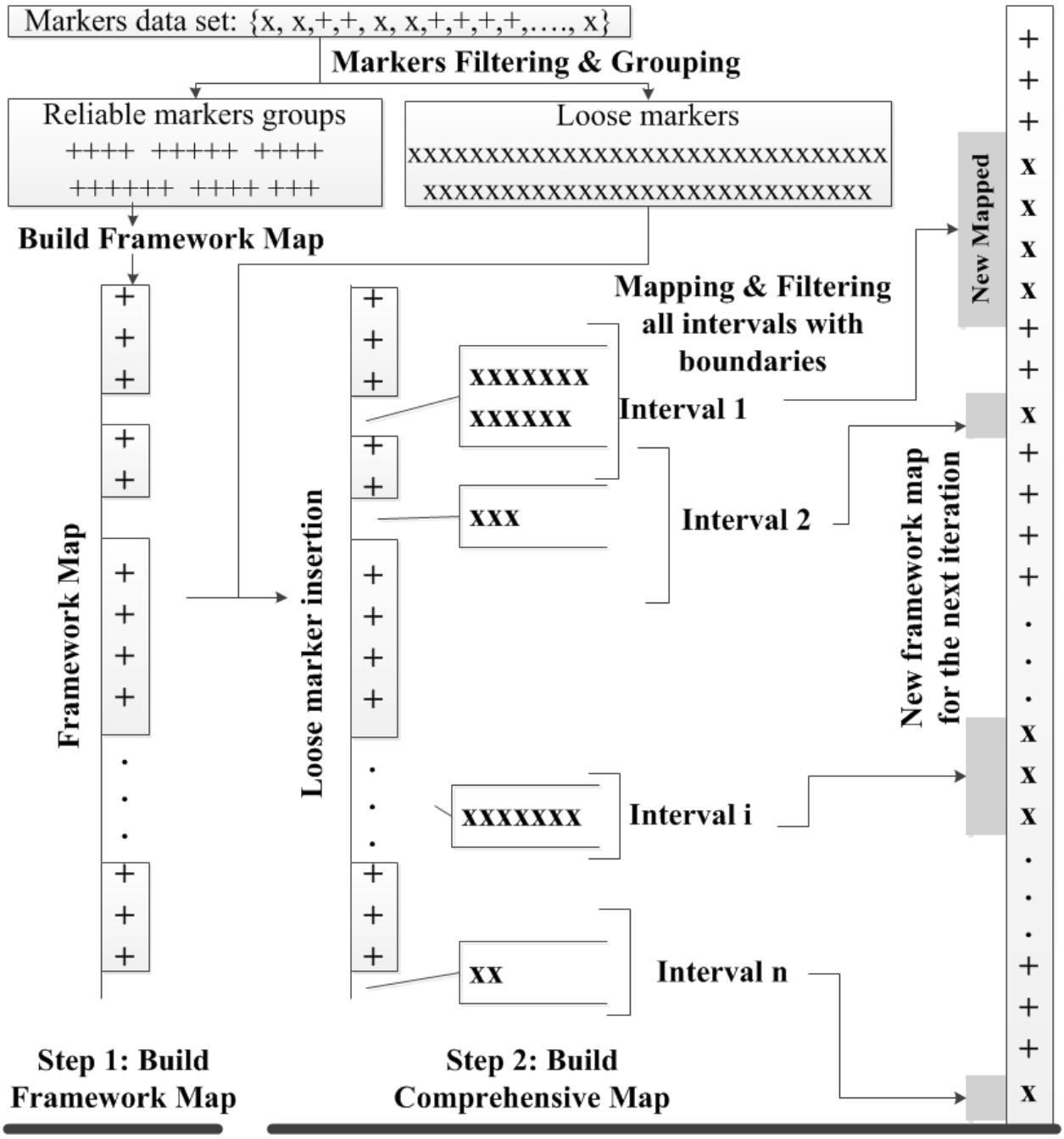


Figure 18. Illustrates the proposed approach, first step constructing a framework map; second step iteratively adding the remaining markers. The resulting framework map of an iteration will be used for insertion of the remaining loose markers in the next iteration.

the systematic workflow for the first step in our proposed approach. Mapping only the reliable set of markers in this step helps reduce the effect of the noisy markers. Moreover,

building a framework map divides the whole chromosome map into small fragments so the remaining markers can be mapped locally in these fragments. 2) The second step builds a comprehensive map for all the markers based on the constructed framework map in the first step. Building the comprehensive map starts by extracting the remaining markers, i.e. the markers not on the framework, and then finding the best interval for each marker on the framework map. After adding the markers in their best intervals, a local mapping process for these intervals is applied to place the reliable markers in their best positions. The local mapping process extends the current framework map for the next iterations, and filters out unstable markers from their intervals. The second step is repeated until all markers are mapped on the framework map. Figure 18 shows the systematic workflow for one iteration in the second step, where the resulting framework in an iteration will be used to map the remaining markers in the next iteration. Dividing the chromosome's map into intervals and then mapping markers in each interval helps to reduce the computational complexity and mapping time

### 3.4. Framework Map Method

To produce a solid framework map in a short time, we used the clustering approach proposed in [54]. The clustering approach constructs reliable framework maps in three sequential steps: First, the mapping process starts with filtering out unreliable markers to find the subset of the most reliable markers that can contribute on the framework map; then these reliable markers are grouped into smaller subsets. A framework map is constructed for each small subset. The second step combines the constructed frameworks to form the whole framework map. The third step polishes the framework by adding significant markers to the framework to strengthen the final framework map. Algorithm 6 shows the workflow for the clustering approach.

#### 3.4.1. Filtering and Mapping Task

This task is divided into two tasks: the first task filters out unreliable markers and the second task maps the reliable markers. Initially, the whole set of markers is grouped into

several clusters. The single linkage clustering method is used to group close markers together in separate clusters. The grouping process helps filter out unreliable markers where only the reliable markers that belong to large clusters i.e. with 3 markers at least, are considered for the framework construction process.

The mapping task is done for each cluster of markers separately. The Carthagene tool is used to extract the most solid combinations of three ordered markers in each cluster to form the cluster's framework. The Buildfw command is used to find the triples of markers such that all alternative orders have a loglikelihood not within a given threshold of the best order. The recommended LOD threshold is three. However, if such a triplet does not exist in a cluster, then the cluster edges form the framework for that cluster. Constructing the framework maps for all clusters can be done in a short time as clusters can be mapped on multiple computing nodes in parallel.

### 3.4.2. Merging the Maps of Clusters

After constructing a framework map for each cluster, the merging procedure aggregates all framework maps to form the final framework map that covers the whole chromosome. The merging process starts by extracting the edges of each framework map and then grouping them incrementally. In each group of edges, the strongest framework maps' edges are connected and then the next strongest edges and so on until all framework maps' edges in that cluster are connected.

### 3.4.3. Local Improvement

Merging all framework maps may cause some large gaps between markers from two different clusters. The goal of this step is to strengthen the final map by filling these gaps with markers. The final map is scanned to identify the large gaps i.e. pairs of markers with LOD score less than a threshold, and then a selected mutual neighbor marker for both marker pairs is appended to the framework. If we get multiple mutual neighbors, we pick the one for which the LOD score difference is the smallest, so that marker will be placed in the middle of the gap.

**Algorithm 6** Step 1: Framework Mapping Construction

**Data** : $RHData$ /* NoOfMrk by NoOfindv matrix */
**Data** : $T$ /* LOD threshold */
**Result** : $FW$ /* Framework Map */
**/* Filtering and Mapping Task */**
Clusters = Group $(RHData, T)$
FWs = $\emptyset$
**for** ***each*** $C$ *in* $Clusters$ **do**
  **if** $Count\_Markers(C) > 2$ **then**
    FW = FindBestTripleMrksinOrder($C$)
    **if** $FW$ *is* $\emptyset$ **then**
      FW = ExtractClusterEnds($C$)
    **end**
    FWs.append(FW)
  **end**
**end**
**/* Merging clusters Maps Task */**
FW_Ends = ExtractFWsEnds($FWs$)
**while** $Count(FW\_Ends) > 2$ **do**
  Ends_Clusters = Group $(FW\_Ends, T)$
  **for** ***each*** $EC$ *in* $Ends\_Clusters$ **do**
    incrementallyConnectAllEdges(EC)
  **end**
  FW_Ends= ExtractFWsEnds(FWs)
**end**
**/* Local Improvement Task */**
Gaps=FindGaps($FW$)
**for** ***each*** $G$ *in* $Gaps$ **do**
  mrks = ExtractGapEnds($G$)
  mrk = FindMutualNeighbor($mrks$)
  FW.append($G, mrk$)
**end**
**return** $FW$

### 3.5. Comprehensive Map Method

After mapping the most reliable markers in the framework step, we incrementally map the remaining markers of the dataset on the framework map to build a solid comprehensive map. The proposed iterative method works as follows: first, we find the best interval on the framework map for each unmapped marker. Second, we find the best position for each

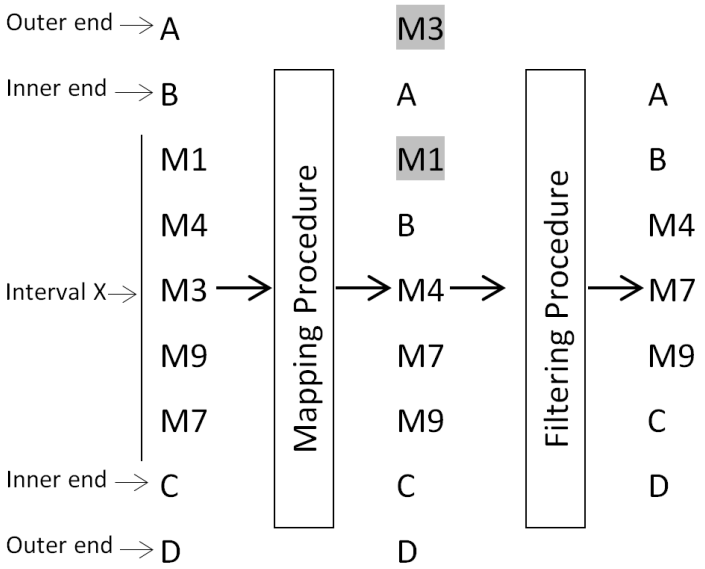


Figure 19. Toy example illustrating the process of filtering unstable markers, i.e. {M3 and M1}.

marker in its pre-defined interval on the framework. Third, we filter out unreliable markers and map them in the next iterations. Algorithm 7 shows the pseudocode for this procedure.

### 3.5.1. Identifying Markers Intervals

Initially, the set of unmapped markers is extracted from the whole dataset. Then each of these unmapped markers is inserted in all possible intervals on the framework map. The best possible insertion interval is reported for each marker based on the relations between the inserted marker with its neighboring markers. The process of finding the best intervals is available in the Carthagene tool. Two inputs are used to run the Carthagene tool: 1) the pre-defined framework map and 2) the unmapped set of markers. The Carthagene output file is parsed to extract the best interval for each marker. Running this process for all unmapped markers results in placing one or multiple markers in some interval.

### 3.5.2. Placing Markers

This step identifies the best position for each marker inside its best interval on the framework map. Placing multiple markers on a framework interval needs further steps to identify the exact location for each marker. The Carthagene tool has some heuristic algo-

rithms for building maps, and some other improvement methods to enhance the constructed maps. To find the best possible position for each marker, a map is constructed for the markers inside each interval in addition to the two outer and two inner ends of that interval, using the Carthagene tool. All markers that are mapped inside the two outer and two inner interval ends are appended to the framework map interval in their best positions. However, the markers that are mapped out of the two inner and two outer interval ends are filtered out and are considered unstable markers. For the intervals with only one marker inside, that marker is appended to the framework directly. For instance if some markers are assigned on the interval x in the framework map FW:<a, b, {interval x}, c, d>, where b and c are the inner interval ends, and a and d are the outer ends then after mapping the markers on interval x, we only accept the list of ordered marker that satisfy the condition: a, b, <amarkers list>, c, d, any other marker that does not satisfy the condition will be filtered out. Figure 19 shows the process of filtering unreliable markers.

Ordering markers for each framework map interval is done following the mapping procedure used in [32, 33, 54]. The mapping procedure starts by merging markers with identical mapping information, and then building an initial map applying the pattern expansion algorithm. After that the taboo and simulated annealing search algorithms are applied for enhancing the constructed map. A fixed-length sliding window is applied to check all possible permutations of markers.

### 3.5.3. Filtering Unreliable Markers

Finally, after appending all stable markers in their best positions, the markers on the first and last interval of the framework are double-checked and trimmed in case the markers do not have good connections with neighboring markers. Markers inserted and mapped at the edges of the framework are checked with only the inner or outer interval ends and that makes the double-check task at the end of this process necessary. The current resulting framework map is used to map the remaining markers in the next iteration. This

**Algorithm 7** Step 2: Iterative Framework Mapping

```
Data : RHData /* NoOfMrk by NoOfIndv matrix */
Data : FW_Map /* Framework Map */
Result : Ch_Map /* Comprehensive map */
L_mks=GetUnmappedMrks(FW_Map, RHData) while L_mks >0 do
    /* Identifying Markers Intervals */
    for each mk in L_mks do
        Mrks_Int=FindBestInterval(mk, FW_Map)
    end
    /* Placing Markers */
    for each Interval in Mrks_Int do
        mrks=GetMarkersIn(Interval)
        if Count_Mrks(mrks) >1 then
            InnerEnds=GetInnerEnds(Interval)
            OuterEnds=GetOuterEnds(Interval)
            MrksBestPositions=MapMrksInterval (InnerEnds, mrks, OuterEnds)
            InnerEndsPOSs=GetPositions(InnerEnds)
            OuterEndsPOSs=GetPositions(OuterEnds)
            for each mk in mrks do
                MrkPOS=GetBestPosition(mk)
                if MrkPOS between InnerEndsPOSs and OuterEndsPOSs then
                    FW_Map= Append(Interval, MrkPOS, mk)
                end
            end
        end
        if CountNoofMrks(mrks)==1 then
            FW_Map=Append(Interval, MrkPOS, mk)
        end
    end
    /* Filtering Unreliable Markers */
    FW_Map=TrimUnreliableMrksEdges(FW_Map)
    L_mks=GetUnmappedMrks(FW_Map, RHData)
end
Ch_Map=FW_Map.
return Ch_Map.
```

iterative process continues until all markers are inserted into the previous framework and a comprehensive map is constructed.

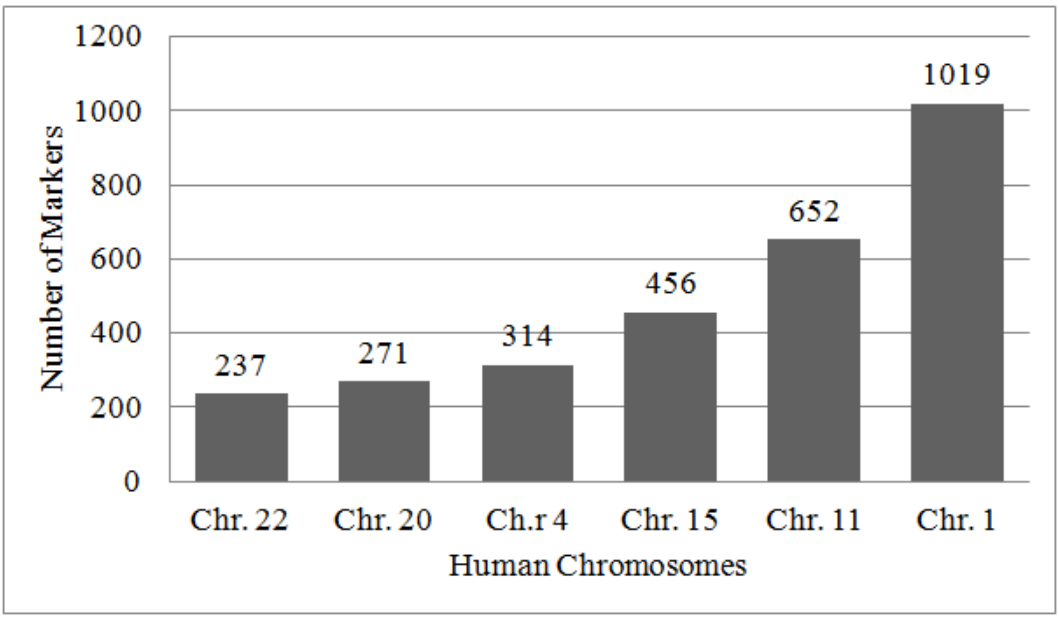


Figure 20. Experiments datasets: human chromosomes attached number of markers. The shortest and longest chromosomes are involved in the data set, Chr. 22 and Chr. 1, respectively. While the others are random medium length chromosomes.

### 3.6. Experiments and Results

#### 3.6.1. Datasets

The proposed approach is tested on real RH datasets from two common standard panels of human radiation hybrids data: Genebridge 4 panel by the Sanger center and the G3 panels produced by Stanford University. To evaluate the performance of the proposed approach, we picked six different chromosomes with varying number of markers, showing the performance pattern over the increasing of markers number. Figure 20 shows the selected chromosomes and the number of markers in each chromosome. The choice of markers in each chromosome is determined by the availability of markers in both the radiation hybrid dataset and NCBI_RH map. The physical marker locations are extracted from the Ensemble site [30], and the RH data set are downloaded from EMBL-EBI site [3].

#### 3.6.2. Results and Discussion

To evaluate the performance of the proposed approach, we compared the run time of the proposed approach with the run time of using Carthagene. We test the scaling of

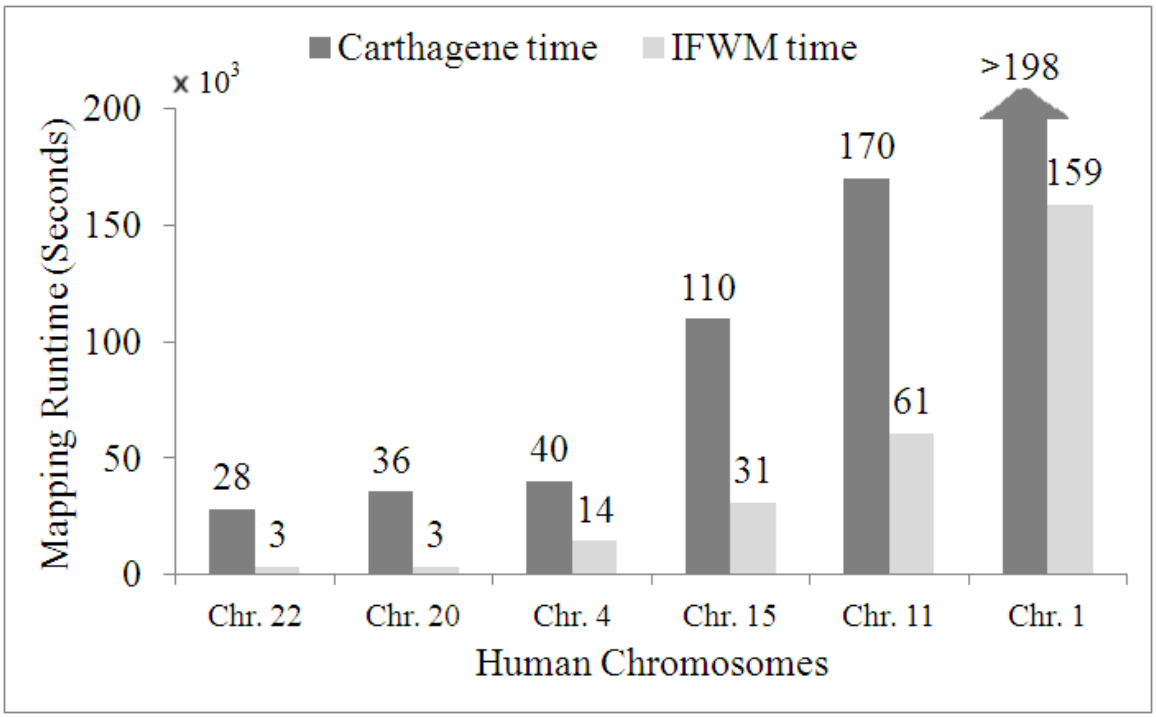


Figure 21. The proposed approach mapping running time vs the traditional Carthagene approach.

the algorithms by mapping different Chromosomes with different numbers of markers. We mapped six chromosomes starting with the shortest chromosome, Chr. 22, which has 237 markers, and ending with the longest Chromosome Chr. 1, which has 1019 markers.

Figure 21 compares the run time for mapping different chromosomes using our proposed approach (IFWM) against the traditional Carthagene approach. In the Carthagene run we map all markers together to get a comprehensive map without the intermediate step of a framework map. Carthagene framework maps have insufficient coverage of the chromosome, as demonstrated in [54], and there is no direct mechanism for iteratively augmenting them to comprehensive maps, so we did not consider those in the comparison.

To be fair for the comparison, we restricted the input for the Carthagene tool to the markers that are mapped by our proposed approach. The results show that our approach outperforms the Carthagene approach regarding the mapping run time for all the chromosomes. Mapping markers on Chromosome 1 (1019 markers) took around 44 hours to get

the final comprehensive map. We tried to build a comprehensive map for that chromosome using the Carthagene tool installed in our high computing clusters. The job was assigned the maximum allowed running time which is two weeks. After two weeks of running the job had been aborted for exceeding the allowed running time. Our proposed approach is much faster, as expected, since both the framework map construction, and the iterative mapping of additional markers are done on much smaller sets of markers than for Carthagene.

The results of the proposed approach maps along with the maps generated using the Carthagene tool are shown graphically in Figures 22-27. The plots show the original permutation of the markers along the x-axis, while the predicted map permutation is shown on the y-axis. For each chromosome except Chromosome 1, two plots are drawn: the traditional Carthagene approach on the left side, and the proposed approach (IFWM) on the right.

For both Chromosomes 20 and 22, the proposed approach outperforms the Carthagene approach. Figure 22 shows that the proposed comprehensive map for Chromosome 20 agrees with the physical map for almost all markers, while the agreement of the traditional Carthagene map does not fit well for those markers ordered on the boundaries of the map. For Chromosome 22, Figure 23, the agreement of our proposed map with the physical map is high, while the traditional Carthagene map does not agree with the physical map for those markers on the tail of the map. Furthermore, some markers that do not belong to the same positions are assigned with the same positions and that results in a map with smaller number of positions.

For Chromosomes 4 and 15, which have more markers, the constructed maps show better agreement with the corresponding physical map comparing to the Carthagene maps. Figure 24 shows the maps for Chromosome 4, mapping markers on the first map fragment of both approaches is flipped. However, the length of the flipped fragment of the traditional Carthagene map is much longer comparing with the length of the flipped fragment of our proposed approach. Figure 25 shows the maps for Chromosome 15, the agreement of both

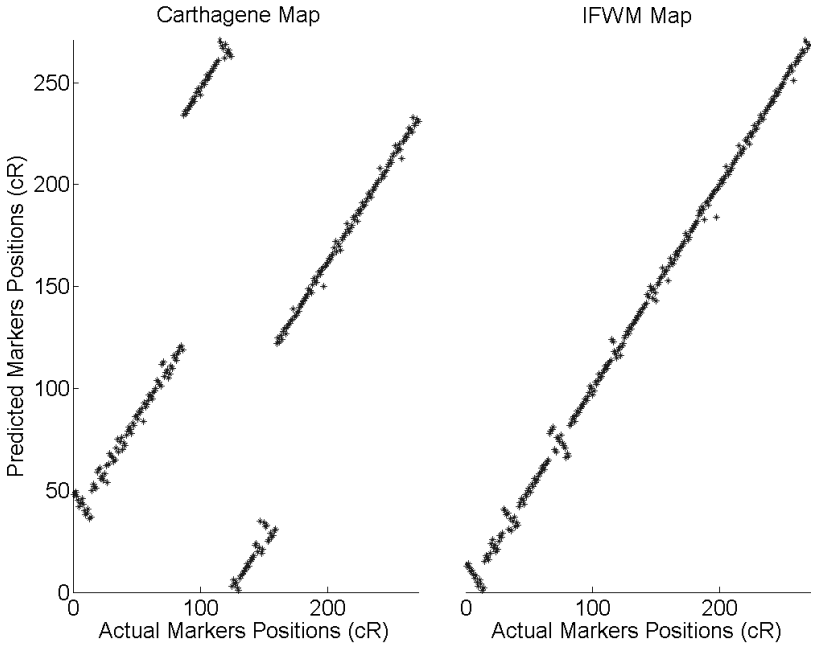


Figure 22. Constructed maps for Chromosome 20.

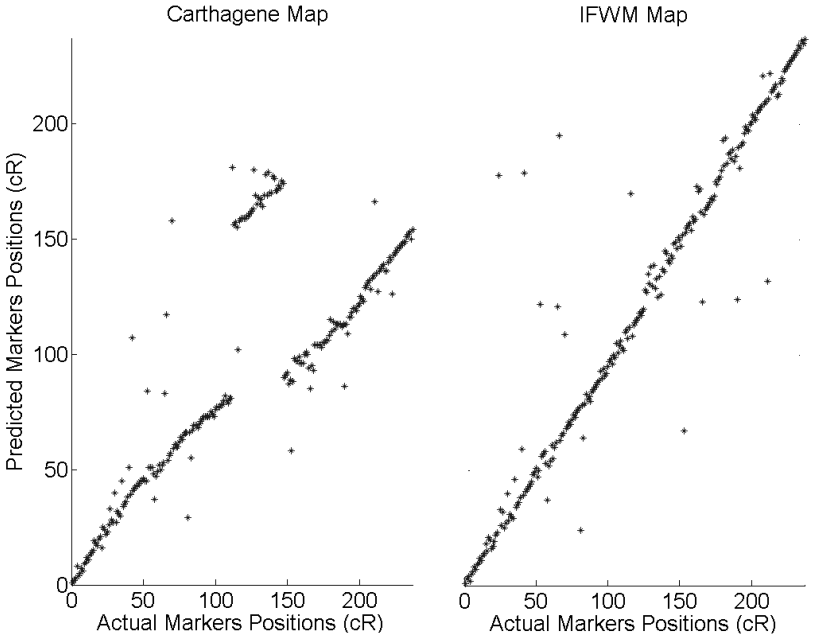


Figure 23. Constructed maps for Chromosome 22.

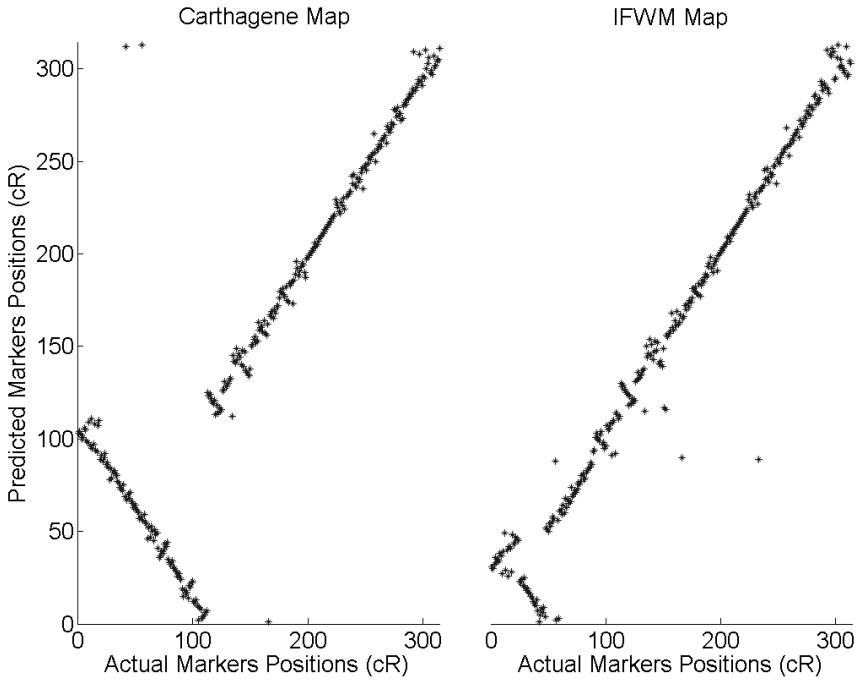

Figure 24. Constructed maps for Chromosome 4.

the traditional Carthagene method and the proposed method with the physical map is comparable. However, the number of positions of the traditional Carthagene map is smaller than the number of markers.

For the longer Chromosomes, 11 and 1, our proposed approach outperforms the Carthagene approach. Figure 26 shows the traditional Carthagene map for Chromosome 11 with 400 different positions, while the physical map and our proposed map have 650 different positions and that means the Carthagene assigned several markers with the same positions. We tried to build a comprehensive map for Chromosome 1 using the Carthagene tool, but the job was aborted because it exceeded the allowed run time. Figure 27 shows the agreement of the proposed map for Chromosome 1 using our proposed approach with the physical map.

The traditional Carthagene approach relies on nearest neighbors and local search strategies to build maps and some noisy markers may disrupt the overall order. On the other hand, the proposed approach takes advantage of the marker information that have

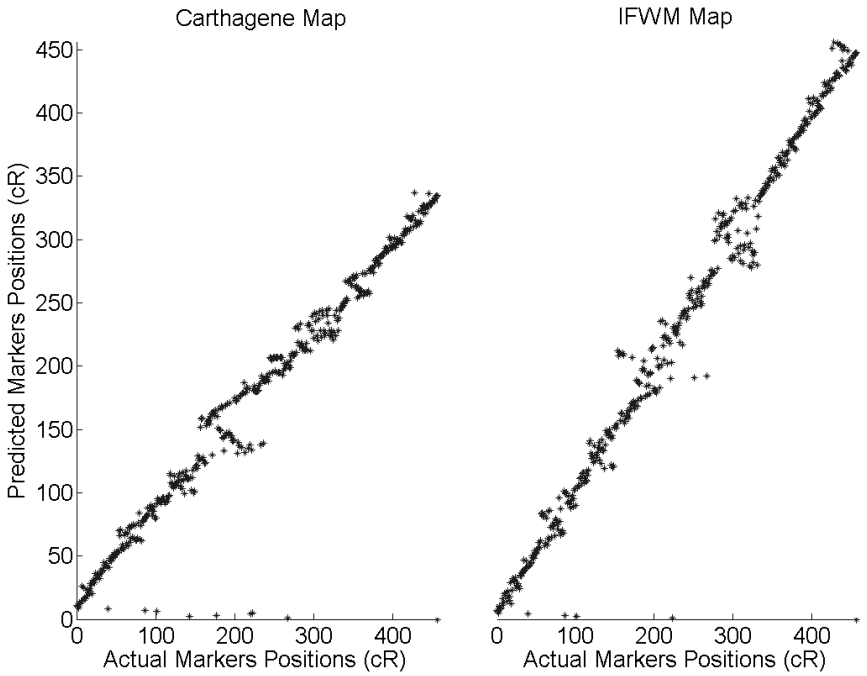


Figure 25. Constructed maps for Chromosome 15.

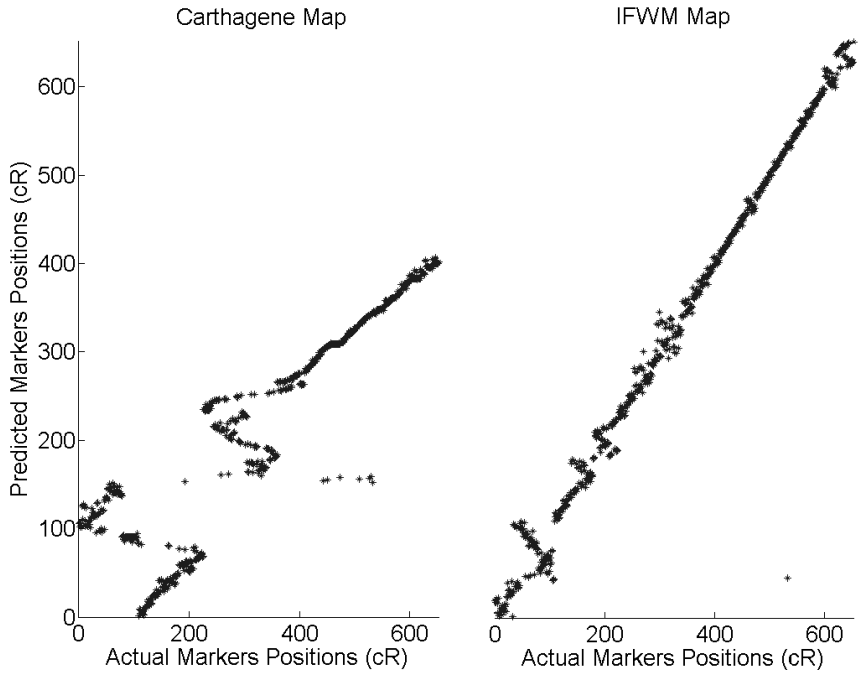


Figure 26. Constructed maps for Chromosome 11.

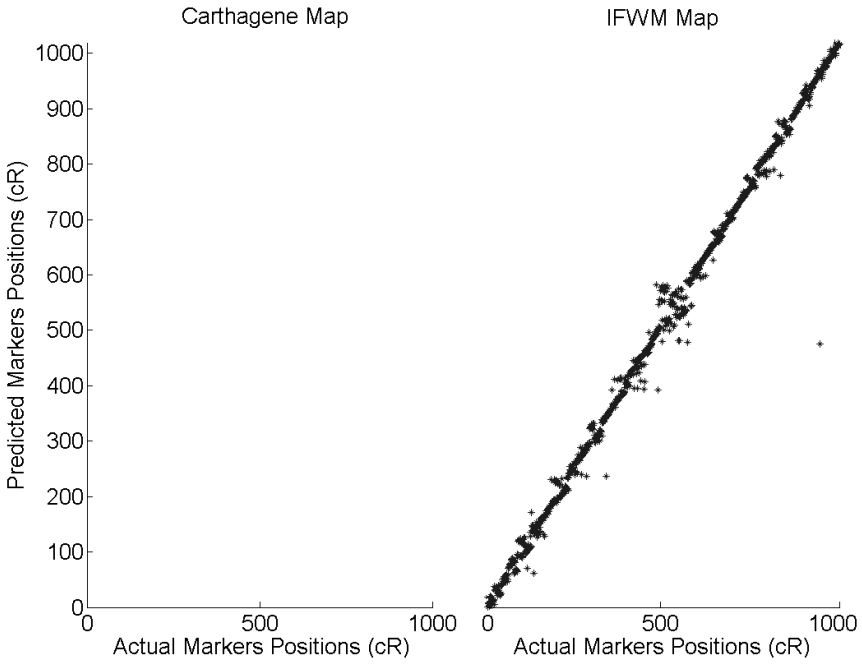


Figure 27. Constructed map for Chromosome 1; we were unable to construct the Carthagene map because of the running time.

already been mapped to build more accurate maps than the traditional approach maps. Overall, Figures 22-27 show that our maps are more accurate than the maps constructed using Carthagene.

The accuracy of the maps generated using the proposed approach depends mainly on the accuracy of the constructed framework in the first step. Using the clustering approach, the most reliable markers are selected first to form a solid framework map. All the remaining markers are mapped based on the intervals on the framework map, applying interval boundaries constraints to check the stability of markers to filter out unreliable markers, and keeping the reliable markers for the next iterations. Also the iterative step of mapping the remaining markers helps keep the most unstable markers to be mapped in the last iterations, so these markers do not affect the mapping of other markers, thus reducing the probability of cumulative errors.

## 3.7. Conclusion

A fast method for constructing comprehensive maps has been proposed in this chapter. Given a large dataset of markers, the proposed method works in two steps: The first step builds a solid framework map using the most reliable subset of markers. This step is done by grouping markers into several smaller groups, then building a framework for each small group, and finally combining these frameworks to form the whole framework map. A polishing method is applied to strengthen the final constructed framework map.

The second step iteratively appends other markers in their best positions to form a comprehensive map. This process starts by finding the best interval on the framework map for each unmapped marker, then finding the exact best position for the marker inside its interval, and finally removing loosely inserted markers in the final map. The process stops when all markers are inserted on the framework to form a comprehensive map.

To validate the proposed method, we applied it with several human chromosomes radiation hybrid dataset. The constructed maps showed high agreement between maps generated by our approach and the physical maps for the corresponding chromosomes. We also compared our approach maps with Carthagene maps. The comparison showed that our approach can produce comprehensive maps with better agreement than the Carthagene approach. Moreover, the run time for our proposed approach is much lower than the run time for generating maps using the Carthagene.

# 4. A NOISE-AWARE METHOD FOR BUILDING RADIATION HYBRID MAPS

In Chapter 3, we discussed the iterative framework mapping approach for constructing comprehensive maps for most of the markers. The maps were constructed by building a framework and then iteratively adding the remaining markers. In this chapter, we propose a new approach that recognizes that the focus on nearest-neighbor distances that characterizes the traveling-saleman model, is no longer appropriate for the large number of markers in modern high-resolution mapping experiments. The proposed approach splits the mapping process into two levels, where the higher level only operates on the most stable markers of the lower level. A divide and conquer strategy, which is applied at the lower level, removes much of the impact of noise. Because of the high density of markers, only the most stable representatives from the lower level are then used at the higher level. The groupings within the lower level are so small that exhaustive search can be used. Markers are then mapped iteratively, while excluding problematic markers. The results for RH mapping dataset of the human genome show that the proposed approach can construct high-resolution maps with high agreement with the physical maps in a comparatively very short time.

The remainder of this chapter is organized as follows: Section 4.1 introduces comprehensive mapping problems. Section 4.2 presents the related work in the area of constructing comprehensive maps. In Section 4.3 our proposed approach is discussed in detail. Section 4.4 presents the experimental results of the proposed approach, and Section 4.5 concludes the chapter.

## 4.1. Introduction

Genome maps show the linear order of genes or unique short fragments of Deoxyribo-Nucleic Acid (DNA) sequence, known as markers, and the distance between them along a chromosome. Radiation Hybrid (RH) mapping [21] is a widely used mapping technique for generating physical maps. The basic hypothesis in RH mapping is that radiation fails to separate close markers in a chromosome. Thus, the closer the markers are on a chromosome,

the more likely they are to be retained or lost together. The LOD (logarithms of odds -base 10) scores [46] is widely used to measure similarity between a pair of markers [11, 22, 54].

The construction of robust high-resolution maps is important in genome sequencing of large genomes, and especially those with a large fraction of repetitive DNA, as is the case for plants [29]. For the large number of markers that are used in high-resolution mapping, many markers can be expected to be within any one deleted portion of the chromosome. That means the overall ordering can be reconstructed based on a small fraction of the original markers. On the other hand, the total number of markers that have to be mapped is large. There are $n!/2$ possible marker orderings to evaluate in order to find the best order. Even heuristic approaches are too inefficient for the large number of markers under consideration. We use a divide-and-conquer approach where the set of markers is broken into several small groups of related markers, to which exhaustive search can be applied. Performance is not the only motivation for using a divide-and-conquer strategy. Quality is actually improved by using the collective information of the marker groups rather than relying on distance information at the level of individual markers.

Traditionally, the mapping problem has been viewed in analogy to the symmetric unidimensional wandering salesman problem [11, 43]. Traveling-salesman-based algorithms have been used for this problem for many years [4, 9, 60]. Typically, building maps for a large number of markers is done by applying heuristic algorithms. However, even the runtime of heuristic approaches typically scales exponentially with the number of markers. In the modern high-resolution RH mapping approaches that motivated the proposed approach, the number of markers can become very large, making these heuristic algorithms unsuitable to be applied directly.

Even more importantly, when there are many markers within the range of any one deletion, but some of them are unreliable, the true ordering may not minimize the nearest neighbor distances. Excluding unstable markers is one approach for dealing with this problem and has been used across the framework mapping literature [11, 20, 51]. Using clustering as

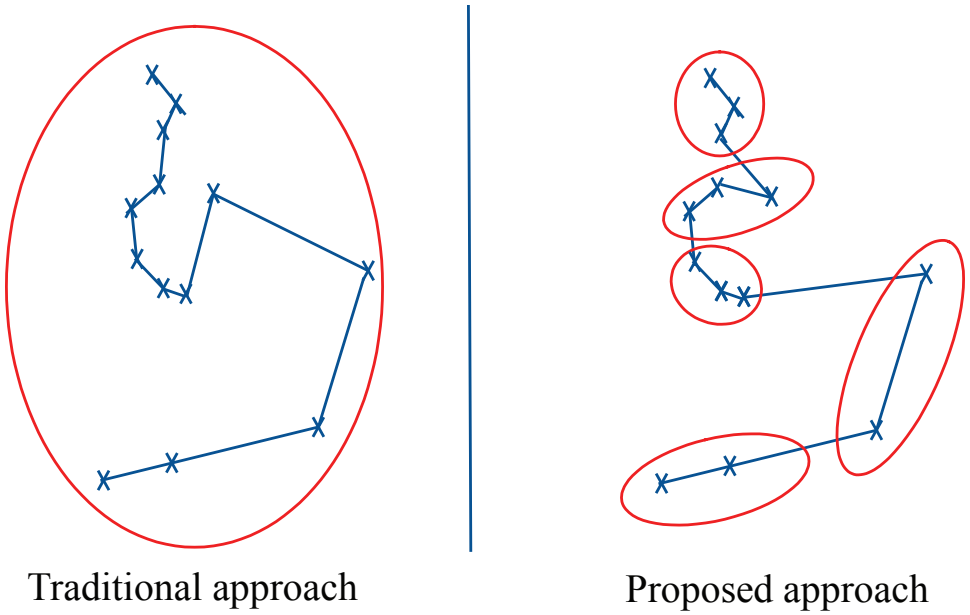


Figure 28. Toy example illustrating the mapping process as a travel salesman problem (left) and using the proposed approach (right).

a way of maximizing with-group similarities while minimizing between-group similarities is another approach that we have used in the past [54, 53].

The proposed method constitutes a unified approach, in which markers are clustered into small linkage groups. Each linkage group consists of strongly connected markers. Because of the use of collective neighborhood information in the clustering process, the effect of noisy markers is reduced.

Figure 28 shows how a traveling-salesman-based algorithm (left) generates a map when some markers with large distances to all others are involved in the mapping process. Assuming that the correct order is from top to bottom, there are some markers that disrupt the overall ordering as they are covered in reverse order. Our proposed approach is shown on the right, in which only neighboring markers with short distances among them are considered in each cluster. While the noisy connections between markers from different clusters are removed; mapping these clusters separately, and then connecting the maps of the clusters

results in higher-quality maps. Not only is the map quality is improved, but performance is also increased by at least 7 orders of magnitude.

### 4.2. Related Work

In the literature, several software packages have been used for constructing framework radiation hybrid maps [11, 41, 57]. RHMapper package [57] constructs framework maps in two steps: 1) dividing the markers into triplets and finding all the strongly connected triplets. 2) Based on the overlapping between the triplets, combining these triplets and report the fully connected framework map. This package can be used to map a small set of markers. However, it does not scale well with the number of markers. For instance, mapping a hundred markers takes several hours, and the runtime increases dramatically with the increasing number of markers [57].

Another common package for constructing framework maps is Multimap [41]. This package constructs framework maps as follows: 1) find the strongest connected pair of markers as an initial map. 2) Iteratively place one marker at a time in its best position, then keep that marker in its position if some predefined conditions are satisfied. This process does not guarantee construction of a solid framework that covers the whole chromosome. Also, the process of mapping one marker at a time does not work well for large numbers of markers.

The Carthagene tool [11] provides methods for constructing framework maps and comprehensive maps, as well. A framework map can be constructed using the incremental insertion procedure, the Buildfw command. This procedure works as follows: First, divide the markers into triplets and find a triplet of markers such that the difference between the likelihood of the best map and the second best map that can be built with this triplet of markers is the greatest. Second, insert each of the remaining markers at each possible position and check the quality of that marker in its position and the quality of the map including that marker, if both conditions are satisfied then, keep the marker in its position as a part of the framework map. The results in [54] show that the constructed framework

maps using this procedure discard too many markers from the final maps and cover the chromosomes partially.

The incremental insertion procedure, Buildfw, can also be used to build comprehensive maps by setting the post processing flag to the value 2. The constructed comprehensive map relies on the framework map. Adjusting the Buildfw command to build comprehensive maps using the constructed framework maps (with partial coverage and few markers) does not help building good comprehensive maps. Moreover, a large number of markers will be just reported with their best intervals on the framework map without the exact position of each marker on its interval.

Carthagene implements some heuristics building procedures for constructing initial comprehensive maps by connecting strongly neighboring markers together. Then, several heuristic algorithms are used to validate and improve the initial maps. The simulated annealing [49], taboo search [10] and LinKernighan heuristic [27] are used to find a map with a better loglikelihood over an initial map. These algorithms have been used to construct several maps in [31, 33, 53].

All the previous packages [11, 41, 57] map markers in a single-level. In [53], we developed the Iterative Framework Mapping approach for building markers in two levels. In the higher level, the most stable markers are used to construct a solid framework map. Then, in the lower level, the less stable markers are iteratively filled in the framework.

The constructed maps using the two-level approach depend mainly on the higher level where a framework map is constructed. Finding solid three markers from each group of markers to build a solid framework map may result in mapping only a small portion of the markers in the framework map. Also, using that framework map with a limited number of intervals to map the remaining markers results in discarding many markers from the constructed comprehensive map. Constructing a framework map with larger number of markers results in mapping more markers on the final comprehensive map, where a larger number of intervals will be available to map the remaining markers.

## 4.3. Proposed Approach

The proposed approach divides the mapping process into two levels, and each level consists of several steps. The proposed approach uses exhaustive search in the first level, which results in building several solid short fragments of maps. Then heuristics methods are applied to merge these fragments to build a framework map. In the second level, the proposed approach extends the framework map to build a comprehensive map. The major steps of the proposed approach shown in Figure 29 include:

### 4.3.1. Markers Filtering

In this step, we remove the markers that do not contribute with the mapping process, the flat markers. The filtering process is started with calculating pairwise distance for all pairs of markers. This calculation can be done upon uploading the data to Carthagene. Then, all of these groups of identical markers with a distance zero are identified. For each identified group, a random marker is taken as a delegate marker while all the other markers are removed from the dataset. Mapping only a few delegate markers instead of a large number of identical markers helps decrease the runtime and the resources needed for the mapping tasks.

After filtering out all the identical markers, we group the remaining markers. We remove all those singleton groups because those singleton markers are far away from the other markers and therefore do not contribute to the constructed map. Figure 30 shows a toy data before and after applying the filtering process.

### 4.3.2. Markers Grouping

In this step, markers are divided into groups based on their LOD scores. We use agglomerative hierarchical clustering [55], which starts by considering each marker as a singleton cluster, and then pairs of clusters are merged, recursively until all markers are merged into one cluster. We use the single linkage metric to merge the neighboring clusters to build the connectivity dendrogram, where clusters are merged by the shortest distance between two markers from two different clusters. Tracing the dendrogram in a bottom

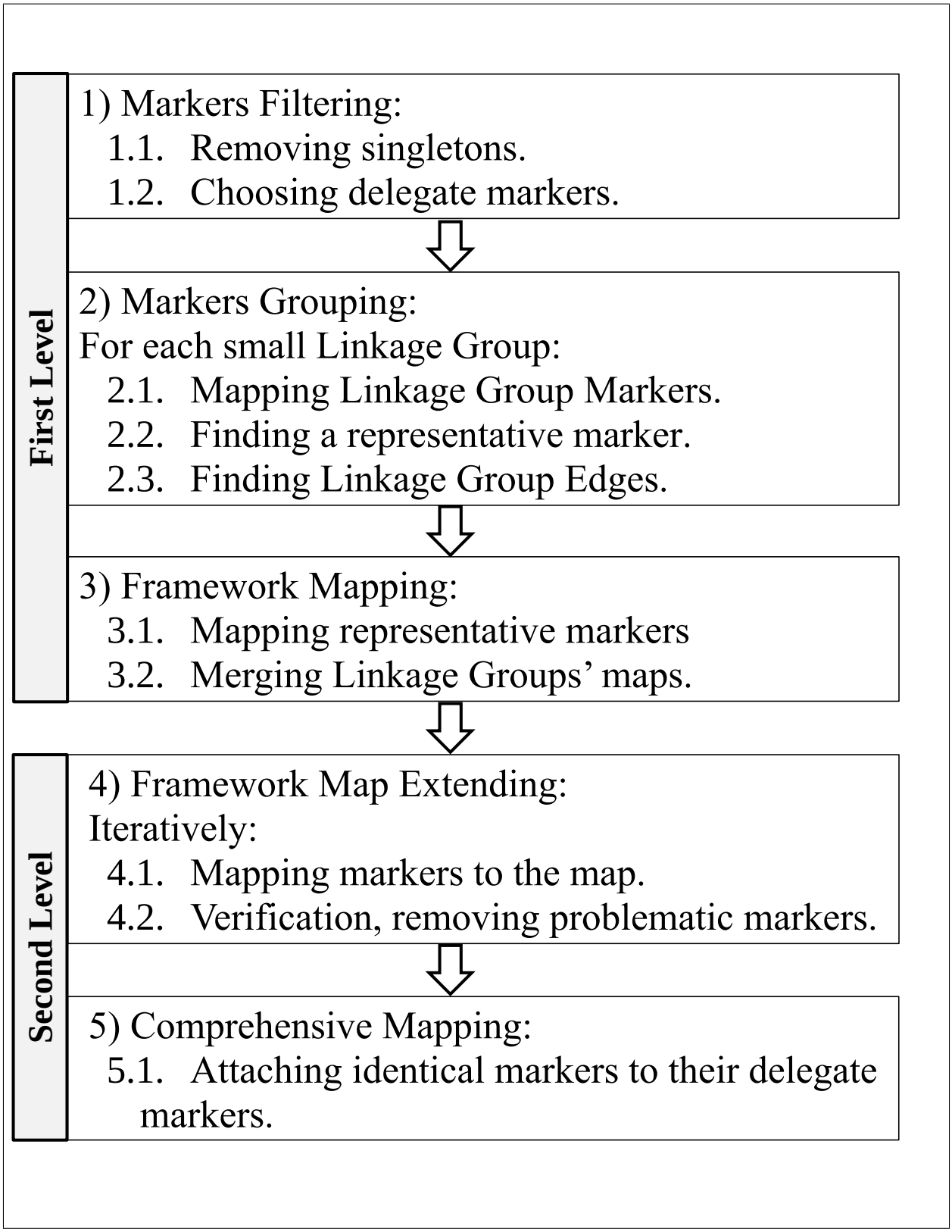


Figure 29. Illustrates the proposed approach.

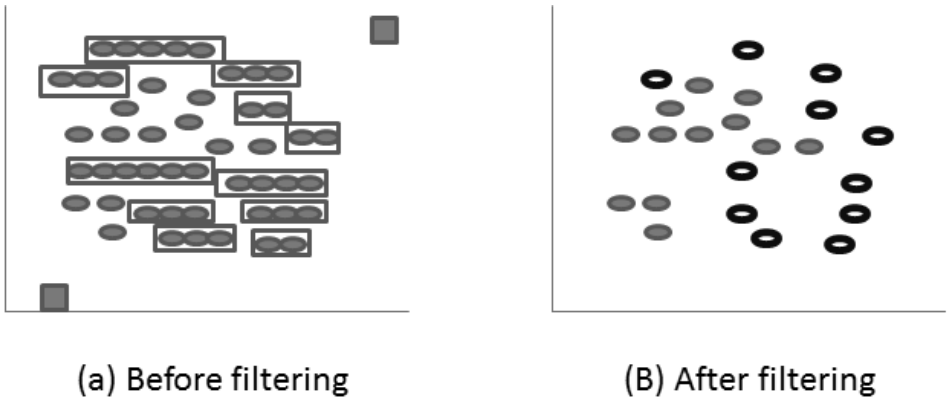


Figure 30. Toy example shows dataset before and after applying the filtering process: a) data before filtering, where the singleton markers are shown in squares while the identical groups are grouped in rectangle frames. b) Data after filtering, delegate markers are the empty circles.

up way shows the similarity between markers, where the closest markers will be grouped together in the leaf nodes. Dividing the markers into smaller groups and working on each group separately, with only a few markers, will further reduce the complexity of the mapping and will ignore the noisy similarities to other markers.

We find all the strongly linked groups of markers within a range of 5 to 9 markers by tracing the dendrogram from the leaf nodes and go up one level at a time until we find up to 9 markers grouped together. Then, for each of these groups we construct a solid map, define the representative marker and find the edges; these steps are explained in detail in this section.

- Mapping Linkage Groups Markers:

  To map an identified group of markers, we perform an exhaustive search of all possible marker orders for the best map. In each permutation the constructed map is checked with the best known map, if the permutation's map has a better loglikelihood then that permutation's map will be saved as the best map. At the end of this step, the proposed approach will find the best map for each identified group of markers.

- Finding Representative Markers:

  In this step, after finding the best map for each group of markers, we connect these maps to build a framework map. To merge these maps, we simplify the mapping process further by finding one representative marker for each short map. Then, we map these representative markers to construct a framework map. To find a representative marker in a group, we measure the reliability of each marker in that group, and then we take the most reliable marker as a representative marker for that group.

  We build the $m \times m$ neighborhood matrix for each linkage group of markers, where m is the number of markers. We use the best k maps, e.g. k=9, that are generated as discussed in Section 4.3.2 to construct the neighborhood matrix. We initialize all elements in the neighborhood matrix to zero. Then, we go over all the k maps and check the neighborhood relationship between paired ordered markers (Mi, Mj) for the best map, and we increment both indexes (i, j) and (j, i) in the neighborhood matrix by 1 each time the pair (Mi, Mj) appear next to each other. After parsing all the k maps, we normalize the neighborhood matrix values by dividing each value by the number of maps, k.

  The resulting matrix will indicate the stability of neighbors in the best 9 maps. Looking at the constructed neighborhood matrix, the representative marker will be the marker Mi, where the ordered pair (Mi, Mj) has the maximum value in the neighborhood matrix. If there are many pairs of markers having the same maximum value, we choose the pair that is mapped in the middle of the best map. This set of representative markers from all groups will be used later to build a framework map. A toy example is shown in Figure 31.

- Finding Linkage Group Edges:

  In this step, we find the most reliable two edge markers for each group's map. We use the same neighborhood matrix as in Section 4.3.2 to find these edges. Using the

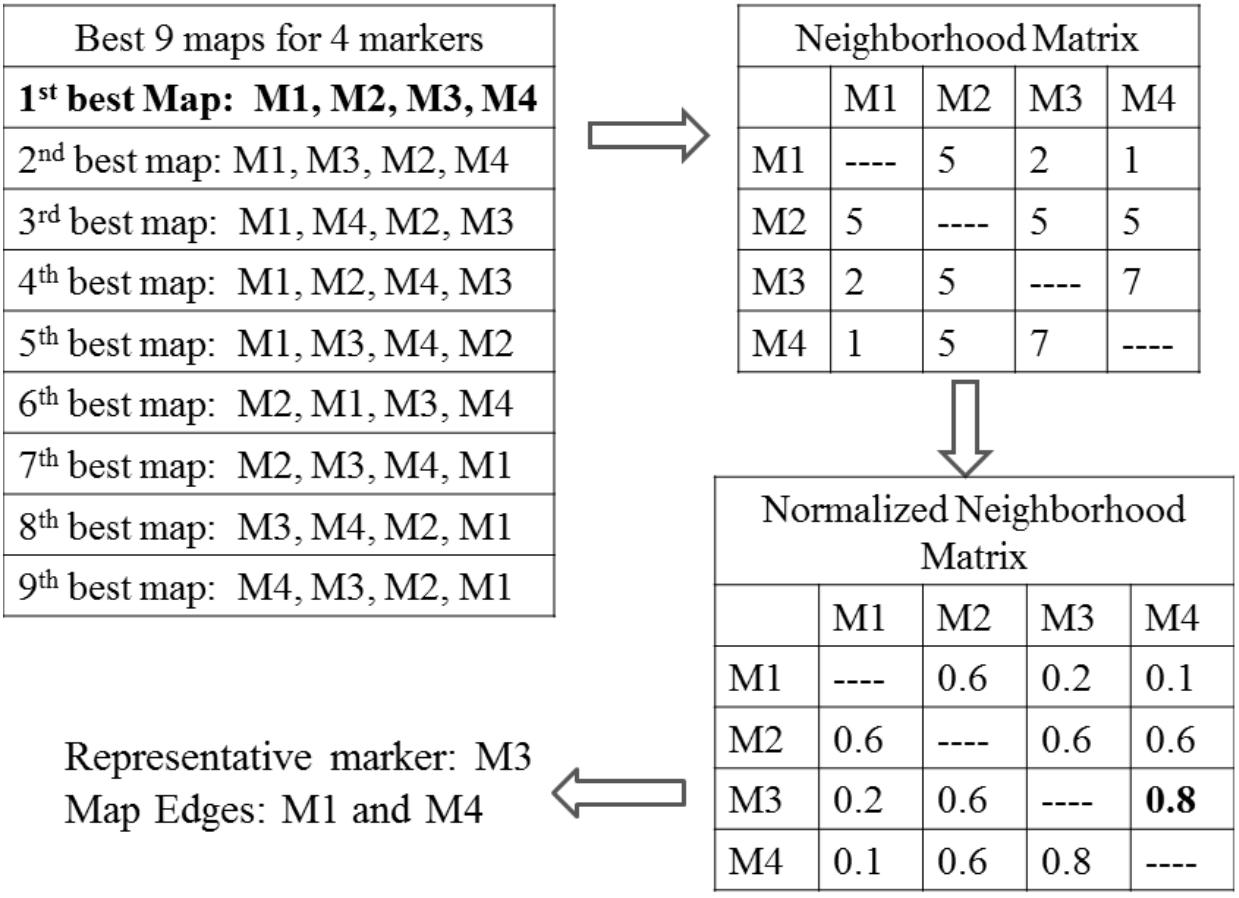


Figure 31. Toy example shows the process of finding representative markers and maps' edges. For a four markers (m=4) we have 4!/2= 12 maps, we keep the best 9 maps (k= 9).

best map as a reference, we consider the reliable edges to be those markers that have values greater than 0.5 with their next closest neighbors. We start checking one end of the map and go in the opposite direction until we find a reliable marker with good relationship with its next neighbor greater than 0.5. We do the same for the other end. Figure 31 shows a toy example, where the connectivity between the two ends is 0.6 for the pair M1, M2 and 0.8 for the pair M4, M3, so we choose M1 and M4 as the map's edges. We will use these two edges to merge the consecutive groups' maps after knowing the best place for each map.

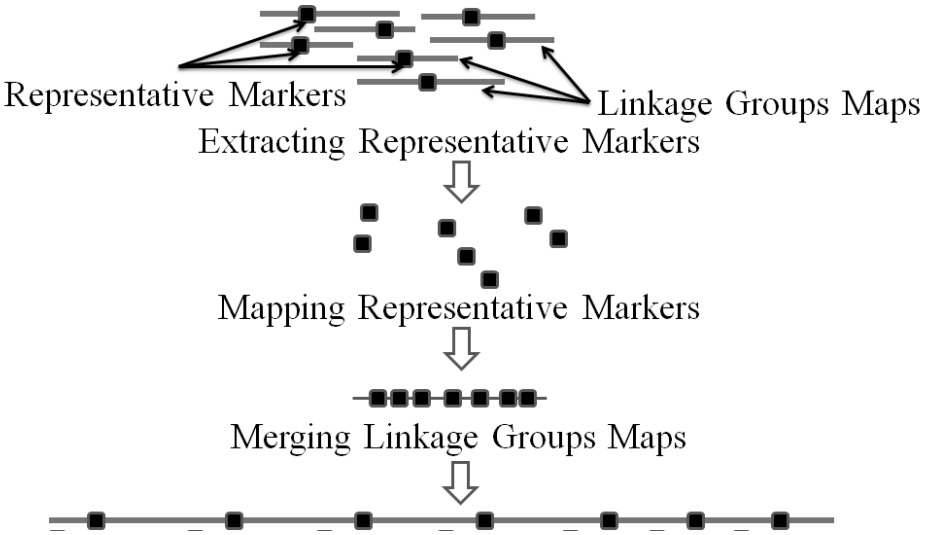

Figure 32. Shows the process of constructing a framework map.

### 4.3.3. Framework Mapping

Until this point, we have constructed several separate short maps. To build a map for the whole dataset, we need to integrate all these maps together and form a framework map. To generate a framework map, we map the set of representative markers for all the constructed groups as follows: first, we build an initial map using the pattern expansion algorithm. Second, we use the local improvement algorithms: 1) the taboo search and 2) simulated annealing algorithm to enhance the constructed map. Third, we use a fixed sliding window to check all permutations within the window size and enhance the map if a better map is achieved. Fourth, we try to place each marker in all possible intervals and check map improvement. All these algorithms are implemented in the Carthagene tool [11] and have been used in several studies [1, 13, 15, 56].

After constructing a map from those representative markers, the linkage groups markers are integrated to the map. The integration process is processed as follows: 1) connect the strongest edges of consecutive linkage group maps based on the similarity between the edges. 2) Place the markers in their positions based on the maps of the linkage groups.

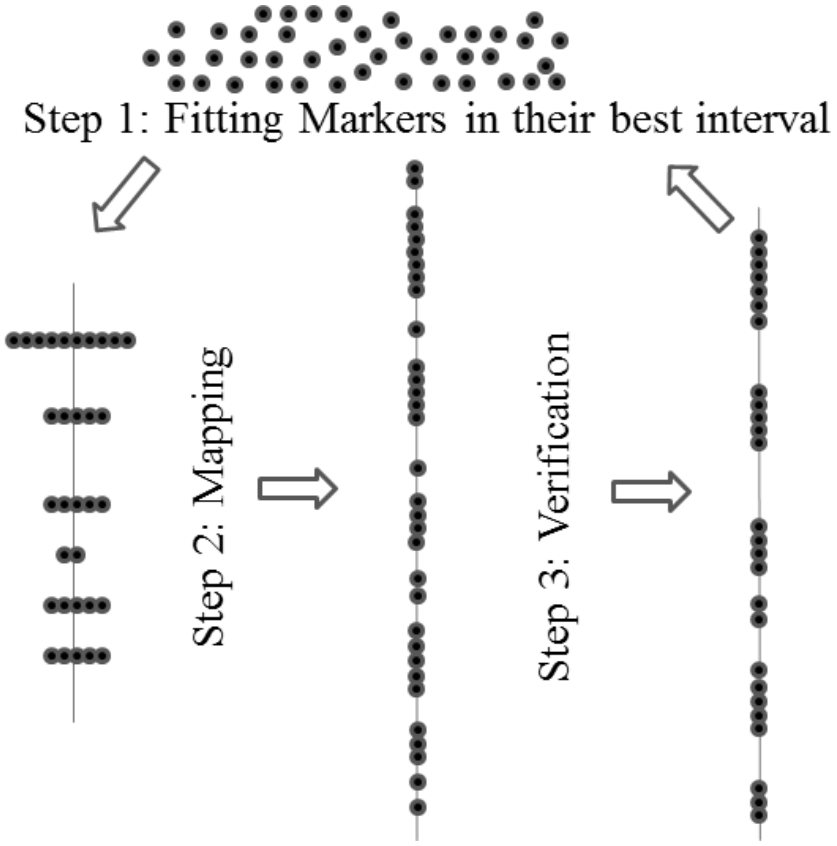


Figure 33. Shows the process of extending a framework map.

### 4.3.4. Framework Map Extending

In this step, we try to map the remaining markers using the framework map that is constructed in Section 4.3.3. The mapping process is done iteratively. In each iteration, we find the best interval for each marker on the framework map, and then add each marker in its best interval. After that, we use the mapping strategy that is explained in Section 4.3.3 to map each interval separately; all the mapped markers inside the boundaries of their intervals will be placed on the map, while the remaining unstable markers will be removed from the map. This process is repeated until all markers are mapped on the framework map. The detailed process is explained in Chapter 3.

### 4.3.5. Comprehensive Mapping

The final step in the proposed approach is to include all the flat markers identified in Section 4.3.1. We go through the map and find the delegate markers. Then, the position of delegate is assigned to each flat marker associated with the delegate.

## 4.4. Experiments and Results

The human genome radiation hybrid data set is used in this study. The three public standard panels of human radiation hybrids that are commonly used are the G3 [58] and TNG [39] panels produced by Stanford University and the Genebridge 4 [23] panel by the Sanger Center. In this study, we use the Stanford G3 panel where the number of individuals is 93. To evaluate the performance of the proposed approach, we have selected 8 different chromosomes with varying number of markers, showing the performance pattern over the increasing number of markers. 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 from the Ensemble site [30], and the RH data set are downloaded from the EMBL-EBI site [3].

### 4.4.1. Datasets

### 4.4.2. Results and Discussion

To evaluate the performance of the proposed approach in terms of runtime, we compared the runtime of the proposed approach with the runtime of the Carthagene approach. In order to check the scaling of our proposed approach compared to the Carthagene approach, we mapped eight chromosomes, starting with the shortest chromosome, Chr. 21, which has 173 markers, and ending with the longest Chromosome, Chr. 1, which has 1423 markers.

Our two-level mapping approach can generate maps for large numbers of markers in a short time compared to the single-level approach, i.e. Carthagene approach. To be fair in the runtime comparison, for each chromosome, we mapped the whole set of markers, and we also mapped only the set of markers that are mapped by our proposed approach using the

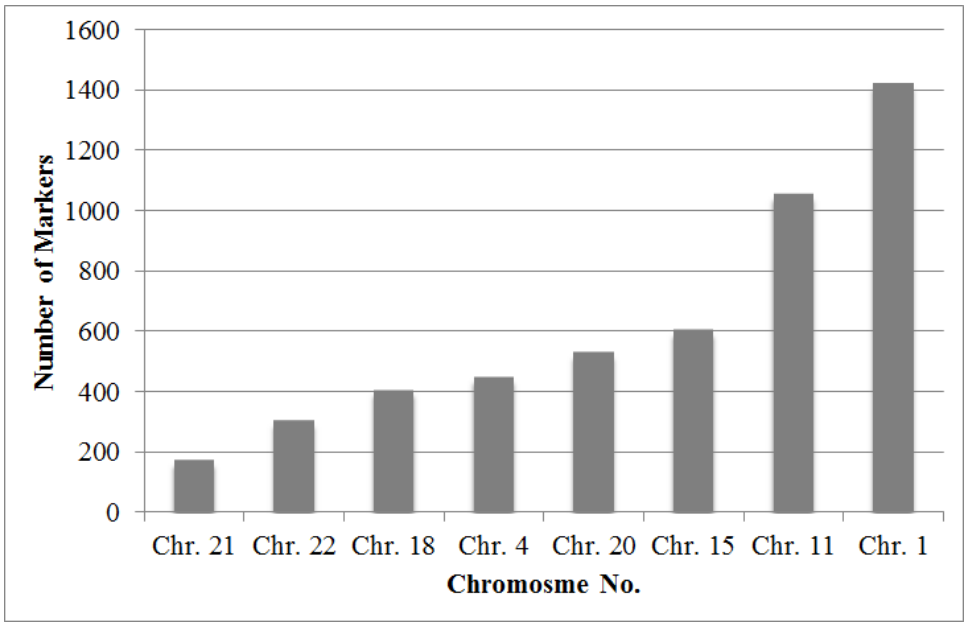


Figure 34. Experiments datasets: human chromosomes with varying number of markers. The shortest and longest chromosomes are involved in the data set, Chr. 21 and Chr. 1, respectively. While the others are random medium length chromosomes.

Carthagene tool. In both cases, Figure 35 shows a big difference between the runtime for our proposed approach compared to the traditional approach for all chromosomes.

For the shortest chromosome, Chr. 21 with 173 markers, our proposed approach mapped 142 markers in 15 minutes, while the Carthagene tool mapped these 142 markers in 120 minutes, and took 225 minutes to map the 173 markers. For the longest chromosome, Chr. 1 with 1423 markers, the assigned job for mapping the 1423 markers was aborted after two weeks of running for exceeding maximum allowed runtime in our high computing clusters. While the job for mapping the 1218 markers took around 242 hours. Our proposed approach mapped these 1218 markers in around 17 hours. For all mapped chromosomes, the ratio between the Carthagene runtime and that of the proposed approach is never smaller than 7 fold improvement. Moreover, the overall mapping runtime can be decreased further in our proposed approach as the process of mapping the groups can be done in parallel.

The improved performance is due to the divide-and-conquer nature of the proposed algorithm. Since the number of markers that are mapped at any one time in our algorithm is

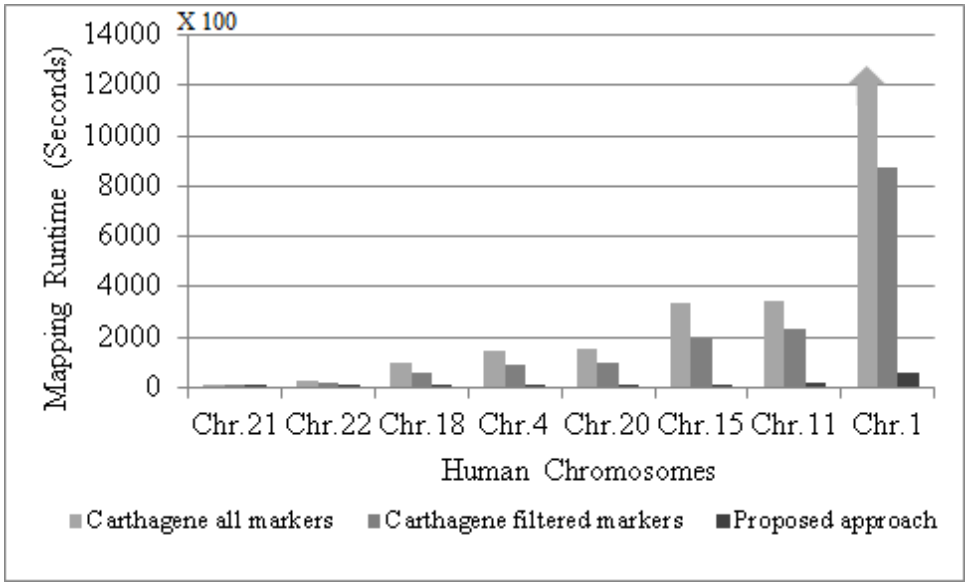


Figure 35. The proposed approach mapping running time vs the traditional Carthagene approach.

small in comparison with the overall number of markers. For typical mapping algorithms in Carthagene, the computation time is approximately doubled for each additional 30 markers. The single-level Carthagene approach maps all markers together, which requires more time to construct maps. However, the proposed two-level approach divides the set of markers into several small groups, and then maps each group separately. Thus, constructs maps in a short time.

In terms of wrongly mapped markers, Figures 36-43 show the agreement of the constructed maps with the corresponding chromosomes' physical maps. Two plots are drawn for each chromosome, where in each plot the x-axis shows the original permutation of the markers, while the y-axis shows the predicted map permutation. Overall, the maps that are generated by our proposed approach have higher agreement with the physical maps than the maps that are constructed by the Carthagene approach. Our proposed approach filters out the noisy markers and maps most of the markers to get high quality maps, while the Carthagene approach maps all the markers together including the noisy ones which decreases the maps' quality.

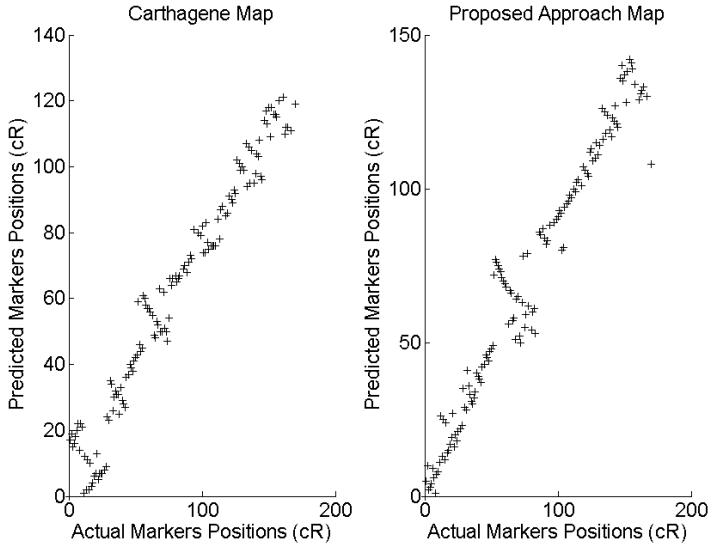


Figure 36. The constructed maps for Chromosome 21: left map (173 markers): Carthagene map for all chromosome's markers, right map (142 markers): our proposed approach map for the reliable markers.

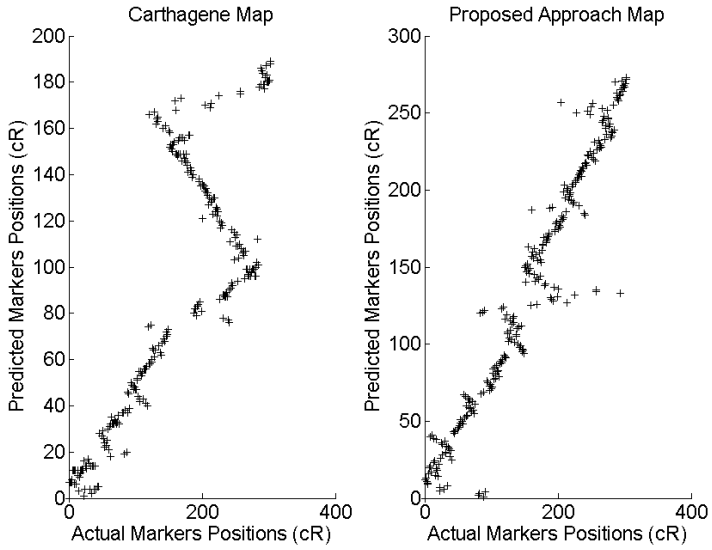


Figure 37. The constructed maps for Chromosome 22: left map (303 markers): Carthagene map for all chromosome's markers, right map (273 markers): our proposed approach map for the reliable markers.

For short chromosomes, e.g., 21 and 22, the proposed approach outperforms the Carthagene approach. Figure 36 shows that the proposed map for Chromosome 21 agrees with the physical map for almost all markers, except for two short reversed fragments. However, the agreement of the traditional Carthagene map does not fit well with the physical map especially at the beginning of the chromosome. Also, the figure shows several mis-orderings, where several markers are flipped close their correct ordering. Figure 36 shows that some markers are assigned to the same position in the Carthagene map, where 130 unique centiRay (cR) positions are identified out of 173. However, our proposed map maps the markers to their original positions, where all the identical cR positions are identified. Figure 37 shows the maps for Chromosome 22. The traditional Carthagene map does not align well with the physical for most of the markers. A long reversed fragment of the map is shown in the second half of the map. Also, the markers at the end of the mapped are mapped in the wrong positions. The agreement of our proposed map with the physical map is higher, where only a few markers are mapped into wrong positions but remain close to their correct positions in the middle of the map.

Also, our proposed approach outperforms the Carthagene approach for the medium-length Chromosomes 18, 4, 20 and 15, which have more markers. Figure 38 shows the maps for Chromosome 18, our constructed map aligned well with the physical map. On the other hand, the Carthagene map shows a long reversed fragment in the middle of the chromosome. Also, several markers are mapped in the wrong positions in the beginning of the chromosome in the Carthagene map, while only a few markers are mapped in the wrong position in the beginning of our proposed approach map. For Chromosome 4, Figure 39 shows the constructed maps using our proposed approach and the Carthagene approach; most of the markers are aligned well with the physical map in the Carthagene map and our proposed map, as well. However, one long reversed fragment is shown at the beginning of the maps.

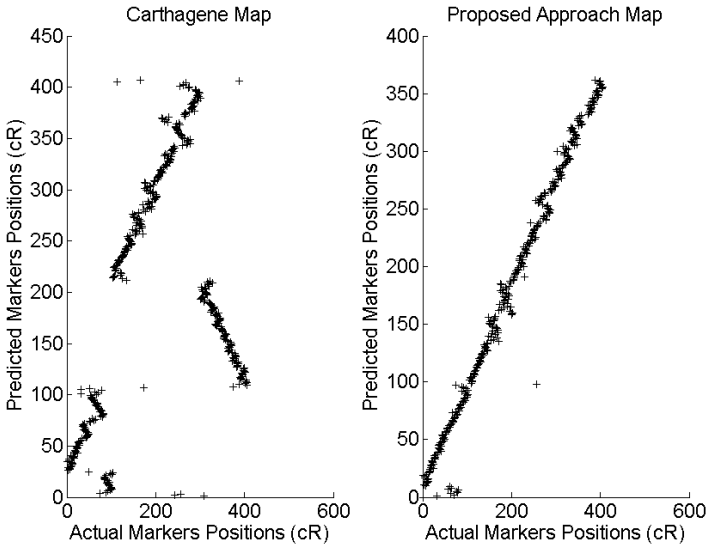


Figure 38. The constructed maps for Chromosome 18: left map (407 markers): Carthagene map for all chromosome's markers, right map (362 markers): our proposed approach map for the reliable markers.

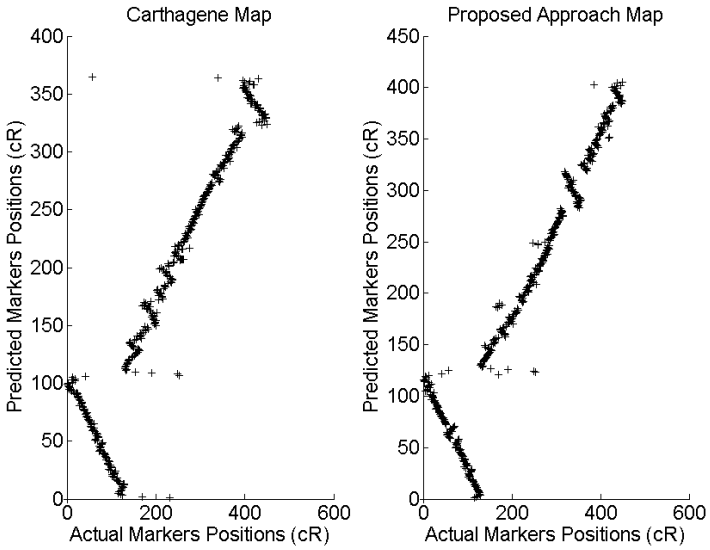


Figure 39. The constructed maps for Chromosome 4: left map (450 markers): Carthagene map for all chromosome's markers, right map (405 markers): our proposed approach map for the reliable markers.

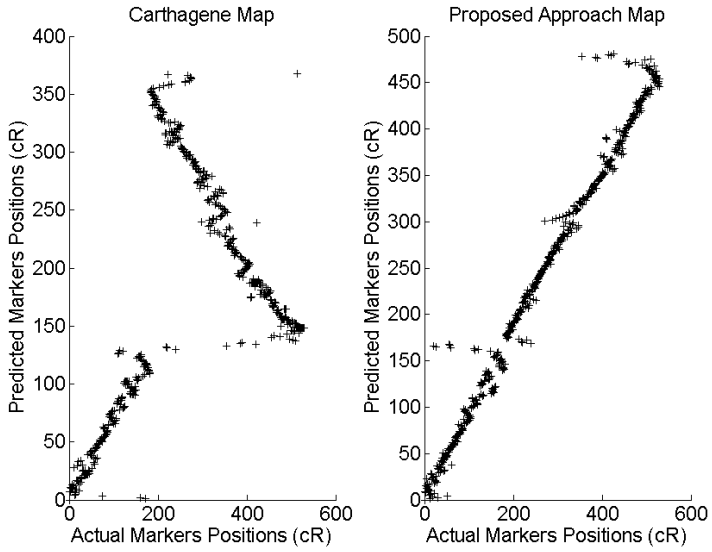


Figure 40. The constructed maps for Chromosome 20: left map (317 markers): Carthagene map for all chromosome's markers, right map (285 markers): our proposed approach map for the reliable markers.

Figure 40 shows the maps for Chromosome 20, our proposed map shows a complete agreement with the physical map except for a few markers at the ends and in the middle of the map. However, the Carthagene map shows several short flipped mapped fragments along the chromosome, and many markers in the middle are mapped in the wrong positions. Moreover, the Carthagene map shows a long reversed fragment in the second half of the map, while this long fragment is mapped correctly in our proposed approach map. For Chromosome 15, Figure 41 shows that long chromosome fragments in the beginning and end of the Carthagene map are mapped in reverse order, while our proposed approach shows better alignment for these fragments except for a few markers flipped close to their correct ordering.

For the longer chromosomes, Chr. 11 and Chr. 1, our proposed approach outperforms the Carthagene approach. Figure 42 shows the Carthagene map for Chromosome 11, where markers do not align well with the physical map, the whole chromosome is mapped into reversed fragments, as well as several markers are mapped in the wrong positions. In contrast,

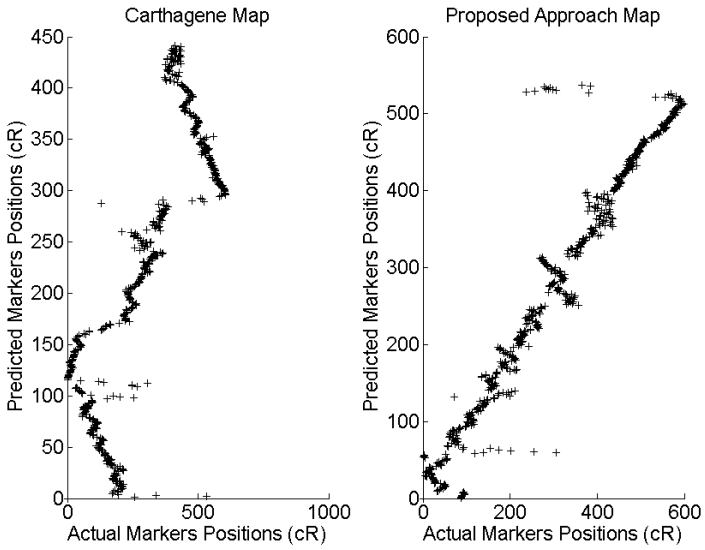


Figure 41. The constructed maps for Chromosome 15: left map (605 markers): Carthagene map for all chromosome's markers, right map (537 markers): our proposed approach map for the reliable markers.

a few markers do not place correctly in our map, while most of the markers are mapped in their correct positions.

For Chromosome 1, our proposed map agrees with the physical map for almost all the markers, except for a few markers that are mapped in the wrong position. On the other hand, the Carthagene map has several markers placed in the wrong positions, and many reversed fragments along the whole chromosome. Overall, Figures 36-43 show that our maps are more accurate than the maps constructed by Carthagene.

As a two-level comparison approach, we consider the Iterative FrameWork Mapping (IFWM) approach that we proposed in [53]. The IFWM and the proposed approach construct maps in two levels: first, construct a framework map. Second, iteratively extend that framework map with the remaining markers to build a comprehensive map. The two methods differ in the first step while they share the second step. Two approaches are proposed in [54] to construct framework maps, the complete-clusters approach, which considers all the markers between clusters' boundaries, and the second approach, which considers a solid

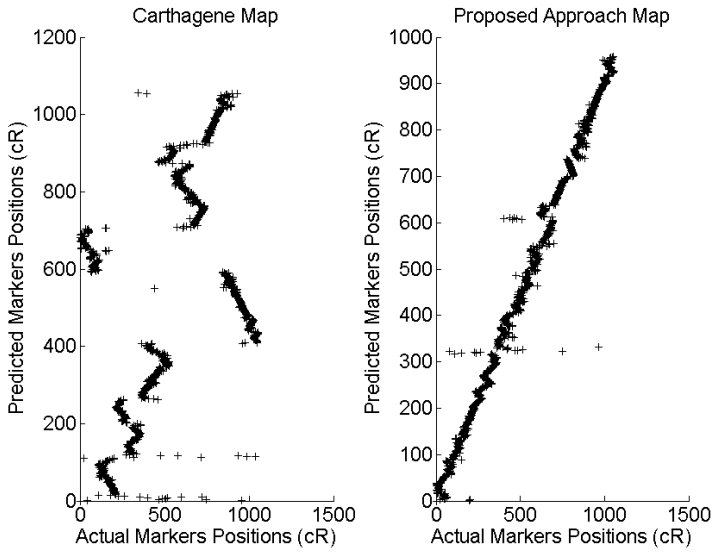


Figure 42. The constructed maps for Chromosome 11: left map (1056 markers): Carthagene map for all chromosome's markers, right map (959 markers): our proposed approach map for the reliable markers.

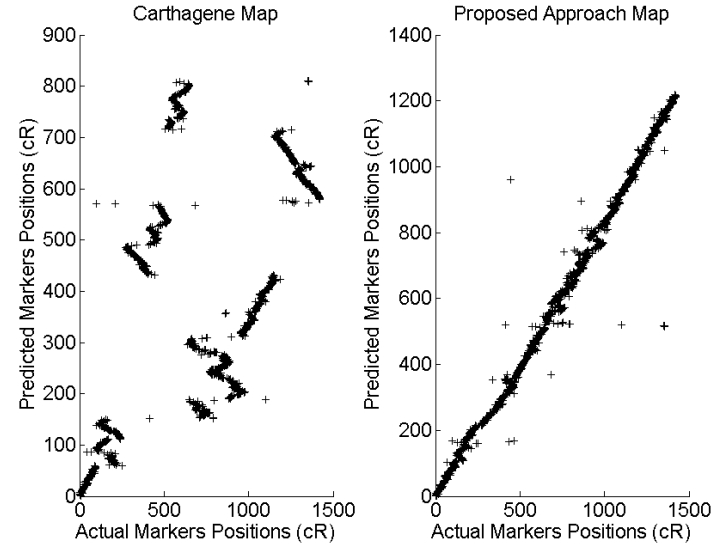


Figure 43. The constructed maps for Chromosome 1: left map (1218 markers): Carthagene map for all chromosome's markers, right map (1218 markers): our proposed approach map for the reliable markers.

triplet of markers from each constructed cluster. In this comparison, we consider the IFWM using the complete-clusters approach where a high-resolution framework map is constructed first, then the framework map is extended iteratively to map the remaining markers.

Using the proposed approach, we expect to increase the quality of the constructed maps comparing to the IFWM approach. The IFWM approach builds several short maps, and then connects them using the maps' ends. However, markers at the ends are not always trustworthy to be used for the merging step. The proposed approach identifies the reliable edges of each short map, and then uses these reliable edges to connect the maps. Thus, decreases the flipping in the final constructed maps. Moreover, the integration of the exhaustive and heuristics algorithms improves the accuracy of the constructed maps. Figure 44 shows a fragment of Chromosome 11, where the order of the same set of markers on the left map using the proposed approach is aligned better than the alignment of the IFWM approach on the right map. The proposed approach maps markers close to their correct ordering. However, the IFWM map shows relatively long flipped fragments.

To compare the accuracy of both methods, we used the proposed approach to map the six chromosomes that are mapped using the IFWM in [53]. Then, we calculated the Spearman's rank correlation coefficient value between the physical maps and the proposed approach maps, and that between the physical maps and the IFWM maps. The experimental results in Table 3 show that that the accuracy between the proposed approach maps and the accuracy of the IFWM approach. For the six chromosomes, the proposed approach outperforms the IFWM approach for Chromosomes 22, 20, 4, 11 and 1. While the IFWM approach outperforms the proposed approach for Chromosome 15. Overall, the integration of exhaustive and heuristics algorithms to construct framework maps improves the quality of the constructed maps compared to the IFWM approach. Moreover, the maps that are constructed using the proposed approach have more mapped markers. Thus, they provide information about a larger number of markers.

Table 3. Comparison of the number of mapped markers, runtime and correlation between the proposed approach maps and the physical maps and that between IFWM maps and the physical maps.

| **Chromosome 22** | **Input Markers** | **IFWM Method** | **Proposed Method** |
|---|---|---|---|
| No. of markers in map | 303 | 230 | 266 |
| Spearman's rank correlation | - | 0.85 | 0.96 |
| Mapping Runtime (Seconds) | - | 471 | 394 |
| **Chromosome 20** | | | |
| No. of markers in map | 530 | 275 | 292 |
| Spearman's rank correlation | - | 0.89 | 0.98 |
| Mapping Runtime (Seconds) | - | 756 | 571 |
| **Chromosome 4** | | | |
| No. of markers in map | 450 | 312 | 405 |
| Spearman's rank correlation | - | 0.91 | 0.94 |
| Mapping Runtime (Seconds) | - | 3076 | 1427 |
| **Chromosome 15** | | | |
| No. of markers in map | 605 | 525 | 537 |
| Spearman's rank correlation | - | 0.96 | 0.95 |
| Mapping Runtime (Seconds) | - | 7639 | 6280 |
| **Chromosome 11** | | | |
| No. of markers in map | 1056 | 795 | 959 |
| Spearman's rank correlation | - | 0.94 | 0.99 |
| Mapping Runtime (Seconds) | - | 34471 | 16859 |
| **Chromosome 1** | | | |
| No. of markers in map | 1423 | 1082 | 1218 |
| Spearman's rank correlation | - | 0.58 | 0.99 |
| Mapping Runtime (Seconds) | - | 266191 | 60297 |

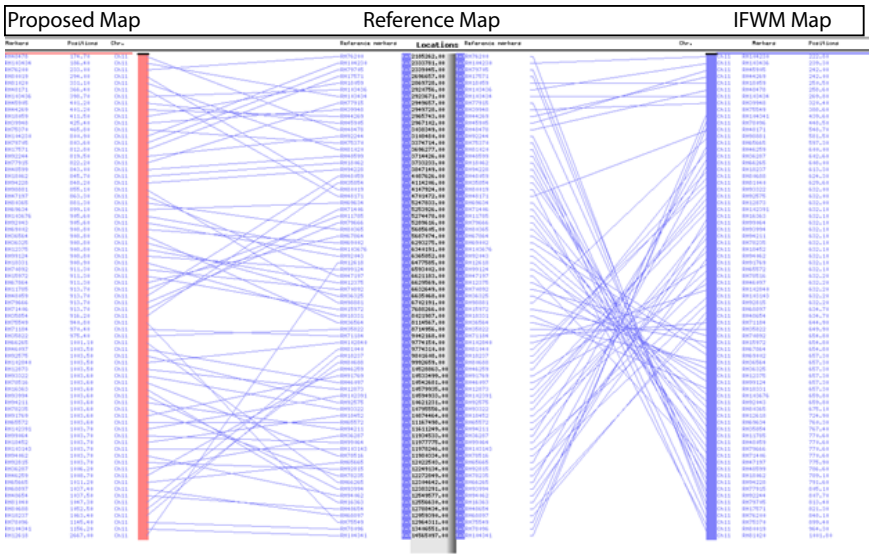


Figure 44. Comparison between three maps created for the same segment of human Chromosome 11. Left: our proposed map. Middle: the physical map. Right: IFWM map. Good markers order conservation is indicated by parallel lines.

The proposed approach can map more markers compared to the IFWM approach, because the proposed approach divides the markers into relatively small groups, where each group includes 5 to 9 markers. Then, each group is mapped using exhaustive searching algorithm. Connecting all groups' maps results in providing a solid framework map. Using that solid framework map to map the remaining markers results in assigning markers to their correct intervals. Thus, only a small number of markers will be filtered out from each interval. However, the IFWM using the complete-clusters divides the markers into groups with no constraints on the number of markers, i.e. a group of markers can include 18 markers, and then applies heuristic algorithms to map these relatively large groups. Constructing framework map from such large groups using heuristic algorithms may result in a poor map, and using that map to map the remaining markers may result in assigning markers in the wrong intervals. Thus, large numbers of markers will be filtered out from these intervals.

Moreover, the IFWM using the complete-clusters only considers the mapped markers between the cluster boundaries, and these boundaries are not always reliable to be used for the filtering, and using them may result in filtering large numbers of markers. To demonstrate

our assumption about the number of mapped markers, we apply our proposed approach on the 6 chromosomes that were used in the evaluation of the IFWM approach. Table 3 shows the number of mapped markers for each chromosome using the IFWM approach against the proposed approach. For all chromosomes, the numbers of mapped markers using the proposed approach are greater than the numbers of mapped markers using the IFWM approach. Thus, the proposed approach provides maps with more information about a larger number of markers.

In addition, the proposed approach outperforms the IFWM regarding the mapping runtime. Although the proposed approach constructs maps with more markers compared to the IFWM approach, the mapping runtime for the proposed approach is less than the IFWM runtime for all the 6 chromosomes. Table 3 shows the mapping runtime for each chromosome using the IFWM approach against the proposed approach.

The runtime for both methods mainly depends on the clusters sizes and the number of remaining markers to the framework. The proposed approach clusters' sizes are smaller than the IFWM clusters' sizes, so the required runtime to map the IFWM clusters is longer than the runtime for mapping the proposed approach clusters. Regarding to the number of remaining markers, the number of remaining markers to be mapped in the second step for the IFWM approach is larger than the number of remaining markers in the proposed approach. Having more markers to be mapped in the second step requires more time to complete the mapping process. Thus, the mapping runtime for the proposed approach is decreased compared to the IFWM runtime.

### 4.5. Conclusion

We have proposed a new approach for constructing high-resolution maps for large datasets of radiation hybrid populations using both the exhaustive and heuristics algorithms. The proposed approach splits the mapping process into two levels: the higher level maps the most reliable markers using an exhaustive searching algorithm. The lower level uses heuristics algorithms to map the remaining markers and excludes the unreliable markers.

Applying a divide and conquer strategy helps increase the performance of the mapping and improve the quality of the constructed maps.

In our experimental comparison, we used the radiation hybrid dataset of several human chromosomes and compared the maps generated using our proposed approach against the Carthagene tool's maps. The constructed maps showed high agreement between maps generated by our approach and the physical maps for the corresponding chromosomes. Also, the comparison showed that our approach can produce high-resolution maps with better agreement than the Carthagene approach. Moreover, the runtime for our proposed approach is much lower than the runtime for generating maps using the Carthagene tool.

# 5. CONCLUSION AND FUTURE WORK

In this dissertation, we proposed several algorithms for mapping large radiation hybrid populations. The radiation hybrid mapping process is often viewed as the traveling salesman problem. Therefore, the mapping complexity increases by increasing the number of markers to map. In this study, we showed that the radiation hybrid mapping problem can be treated through a divide and conquer approach. The one dimension structure of the RH mapping problem and the employment of divide and conquer approach can reduce the computational complexity of the RH mapping problem into smaller sub problems. Also, the mapping process is susceptible to noise, and as a result, it is often beneficial to remove unreliable markers. In this study, we addressed these problems by providing a clustering-based approach for constructing high quality framework and comprehensive maps in a short time.

We proposed three algorithms for constructing framework maps in Chapter 2. The proposed algorithms uses a divide-and-conquer strategy to construct framework maps based on clusters that exclude unreliable markers. Clusters of markers are ordered using parallel processing and are then combined to form the complete framework map. We compared the performance of our algorithms to the traditional mapping techniques. The presented results showed that our framework mapping algorithms can produce high quality maps with good coverage compared to the traditional approaches map.

The resulting framework maps are typically more reliable but do not provide information about as many markers. Thus, we used the clustering-based approach to present a new approach for constructing comprehensive maps in Chapter 3. The iterative framework mapping approach constructs comprehensive maps by applying the clustering-based approach to build a framework map and then iteratively extending that framework map to include most of the markers. The proposed approach is intended to reduce the effect of the noisy markers and use the global information about the markers that have already been mapped on the framework map to map the remaining markers (the less reliable markers). The results showed that quality of maps can be increased by first mapping the most reliable

markers, and second incrementally adding the remaining markers. Also the run time for our proposed approach is much lower than the run time for generating maps using the traditional comprehensive mapping approaches.

Moreover, we proposed an enhanced approach for constructing comprehensive maps in Chapter 4. The proposed approach divides the mapping process into two-levels: the first level uses the exhaustive search to construct several high quality short maps, and to find representative markers for these maps. The second level, uses heuristics algorithms to map these representative markers and then merges the short maps to form a framework map. The framework map will be used to incrementally map the remaining markers. Applying both the exhaustive and heuristic algorithms results in building high quality maps, also the exhaustive search helps mapping a large number of markers in the first level which decreases the total mapping running time. The constructed maps showed that large numbers of markers can be mapped in our approach in a high quality and short time compared to the traditional single-level approach.

All the proposed algorithm were evaluated using data from radiation hybrid (RH) populations of the human genome. We considered several chromosomes of the human genome, for which the correct ordering is known, and compared the performance of our proposed algorithm with the traditional radiation hybrid mapping softwares. We compared our constructed maps with the corresponding physical maps to check the accuracy of our results; also, we compared our maps with other approach maps to check the performance of our algorithms. For framework mapping approach, the results showed that 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. Also for the comprehensive mapping approach, the results showed that our two-stage approach is not only much faster than mapping the complete genome in one step, but that the quality of the resulting maps is also much higher.

In summary, several algorithms have been proposed for constructing high quality framework and comprehensive RH maps from large RH populations. The constructed maps using the proposed algorithms showed a high agreement with the corresponding physical maps. Also, the computational cost of the proposed algorithms is much lower than for the traditional approaches. As for future work, we plan to apply the proposed algorithms to map other species, where the genomes sizes are too large compared to the human genome. For example, Wheat has three genomes and each genome is almost twice of the human genome. Moreover, the proposed algorithms are developed to construct solid maps from a single species. However, considering related species with known genomes' maps can be integrated to this work. We plan to validate the linkage groups of markers with other related species maps to exclude unstable markers which may result in providing more accurate maps.

# REFERENCES


[1] Omar Al-Azzam, Loai Alnemer, Charith Chitraranjan, Anne M Denton, Ajay Kumar, Filippo Bassi, Muhammad J Iqbal, and Shahryar F Kianian, *Network-based filtering of unreliable markers in genome mapping*, Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on, vol. 1, IEEE, 2011, pp. 19–24.

[2] Omar Ghazi Al-Azzam and Anne M Adviser-Denton, *Mining for significant information from unstructured and structured biological data and its applications*, (2012).

[3] Clara Amid, Ewan Birney, Lawrence Bower, Ana Cerdeño-Tárraga, Ying Cheng, Iain Cleland, Nadeem Faruque, Richard Gibson, Neil Goodgame, Christopher Hunter, et al., *Major submissions tool developments at the european nucleotide archive*, Nucleic acids research **40** (2012), no. D1, D43–D47.

[4] David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook, *The traveling salesman problem: a computational study*, Princeton University Press, 2011.

[5] Andrea Baraldi and Palma Blonda, *A survey of fuzzy clustering algorithms for pattern recognition. i*, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on **29** (1999), no. 6, 778–785.

[6] Amir Ben-Dor, Benny Chor, and Dan Pelleg, *Rhoradiation hybrid ordering*, Genome Research **10** (2000), no. 3, 365–378.

[7] Michael Boehnke, Kenneth Lange, and David R Cox, *Statistical methods for multipoint radiation hybrid mapping.*, American journal of human genetics **49** (1991), no. 6, 1174.

[8] Gilles Brassard and Paul Bratley, *Fundamentals of algorithmics*, vol. 524, Prentice Hall Englewood Cliffs, 1996.

[9] Jitender Cheema and Jo Dicks, *Computational approaches and software tools for genetic linkage map estimation in plants*, Briefings in bioinformatics **10** (2009), no. 6, 595–608.

[10] Djurdje Cvijovic and Jacek Klinowski, *Taboo search: an approach to the multiple minima problem*, Science **267** (1995), no. 5198, 664–666.

[11] Simon De Givry, Martin Bouchez, Patrick Chabrier, Denis Milan, and Thomas Schiex, *Carh ta gene: multipopulation integrated genetic and radiation hybrid mapping*, Bioinformatics **21** (2005), no. 8, 1703–1704.

[12] Paul H Dear, *Genome mapping*, eLS (2001).

[13] O Demeure, D Pomp, D Milan, MF Rothschild, and CK Tuggle, *Mapping of 443 porcine est improves the comparative maps for ssc1 and ssc7 with the human genome*, Animal genetics **36** (2005), no. 5, 381–389.

[14] Thomas Derrien, Catherine André, Francis Galibert, and Christophe Hitte, *Autograph: an interactive web server for automating and visualizing comparative genome maps*, Bioinformatics **23** (2007), no. 4, 498–499.

[15] A Doligez, AF Adam-Blondon, G Cipriani, G Di Gaspero, V Laucou, D Merdinoglu, CP Meredith, S Riaz, C Roux, and P This, *An integrated ssr map of grapevine based on five mapping populations*, Theoretical and applied genetics **113** (2006), no. 3, 369–382.

[16] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise.*, KDD, vol. 96, 1996, pp. 226–231.

[17] Michael R Gary and David S Johnson, *Computers and intractability: A guide to the theory of np-completeness*, 1979.

[18] Fred Glover and Gary A Kochenberger, *Handbook of metaheuristics*, Springer, 2003.

[19] Michael Goebel and Le Gruenwald, *A survey of data mining and knowledge discovery software tools*, ACM SIGKDD Explorations Newsletter **1** (1999), no. 1, 20–33.

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

[21] Stephen J Goss and Henry Harris, *New method for mapping genes in human chromosomes*, Nature **255** (1975), 680–684.

[22] David A Greenberg, Paula Abreu, and Susan E Hodge, *The power to detect linkage in complex disease by means of simple lod-score analyses*, The American Journal of Human Genetics **63** (1998), no. 3, 870–879.

[23] Gabor Gyapay, Karin Schmitt, Cécile Fizames, Hywel Jones, Nathalie Vega-Czarny, Dominique Spillett, Delphine Muselet, Jean-François Prud'Homme, Colette Dib, Charles Auffray, et al., *A radiation hybrid map of the human genome*, Human Molecular Genetics **5** (1996), no. 3, 339–346.

[24] David Hall, Suchendra M Bhandarkar, Jonathan Arnold, and Tongzhang Jiang, *Physical mapping with automatic capture of hybridization data*, Bioinformatics **17** (2001), no. 3, 205–213.

[25] Jiawei Han, Micheline Kamber, and Jian Pei, *Data mining: concepts and techniques*, Morgan kaufmann, 2006.

[26] Paul Havlak, Rui Chen, K James Durbin, Amy Egan, Yanru Ren, Xing-Zhi Song, George M Weinstock, and Richard A Gibbs, *The atlas genome assembly system*, Genome Research **14** (2004), no. 4, 721–732.

[27] Keld Helsgaun, *An effective implementation of the lin–kernighan traveling salesman heuristic*, European Journal of Operational Research **126** (2000), no. 1, 106–130.

[28] Venu Kalavacharla, Khwaja Hossain, Yong Gu, Oscar Riera-Lizarazu, M Isabel Vales, Suresh Bhamidimarri, Jose L Gonzalez-Hernandez, Shivcharan S Maan, and Shahryar F Kianian, *High-resolution radiation hybrid map of wheat chromosome 1d*, Genetics **173** (2006), no. 2, 1089–1099.

[29] Venu Kalavacharla, Khwaja Hossain, Oscar Riera-Lizarazu, Yong Gu, Shivcharan S Maan, and Shahryar F Kianian, *Radiation hybrid mapping in crop plants*, Advances in Agronomy **102** (2009), 201–222.

[30] Rhoda J Kinsella, Andreas Kähäri, Syed Haider, Jorge Zamora, Glenn Proctor, Giulietta Spudich, Jeff Almeida-King, Daniel Staines, Paul Derwent, Arnaud Kerhornou, et al., *Ensembl biomarts: a hub for data retrieval across taxonomic space*, Database: the journal of biological databases and curation **2011** (2011).

[31] Ajay Kumar, *Combining radiation hybrids and 45k nimblegen array to develop high density marker scaffold for d-genome of chinese spring and¡ em¿ aegilops tauschii¡/em¿ accession al8/78*, Plant and Animal Genome XXI Conference, Plant and Animal Genome, 2013.

[32] Ajay Kumar, Filippo Bassi, Etienne Paux, Omar Al-Azzam, Monika de Jimenez, Anne Denton, Yong Gu, Eric Huttner, Andrzej Kilian, Sachin Kumar, et al., *Dna repair and crossing over favor similar chromosome regions as discovered in radiation hybrid of triticum*, BMC genomics **13** (2012), no. 1, 339.

[33] Ajay Kumar, Kristin Simons, Muhammad J Iqbal, Monika M de Jiménez, Filippo M Bassi, Farhad Ghavami, Omar Al-Azzam, Thomas Drader, Yi Wang, Ming-Cheng Luo, et al., *Physical mapping resources for large plant genomes: radiation hybrids for wheat d-genome progenitor aegilops tauschii*, BMC genomics **13** (2012), no. 1, 597.

[34] Ralf G Kynast, Ron J Okagaki, Mark W Galatowitsch, Shannon R Granath, Morrison S Jacobs, Adrian O Stec, Howard W Rines, and Ronald L Phillips, *Dissecting the maize genome by using chromosome addition and radiation hybrid lines*, Proceedings of the National Academy of Sciences of the United States of America **101** (2004), no. 26, 9921–9926.

[35] Daniel T Larose, *An introduction to data mining*, Traduction et adaptation de Thierry Vallaud (2005).

[36] Richard Char-Tung Lee, RC Chang, SS Tseng, and YT Tsai, *Introduction to the design and analysis of algorithms*, Unalis, 1999.

[37] Charles E Leiserson, Ronald L Rivest, Clifford Stein, and Thomas H Cormen, *Introduction to algorithms*, The MIT press, 2001.

[38] Anton V Leouski and W Bruce Croft, *An evaluation of techniques for clustering search results*, Tech. report, DTIC Document, 2005.

[39] Kathryn L Lunetta, Michael Boehnke, Kenneth Lange, and David R Cox, *Selected locus and multiple panel models for radiation hybrid mapping.*, American journal of human genetics **59** (1996), no. 3, 717.

[40] James MacQueen et al., *Some methods for classification and analysis of multivariate observations*, Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, vol. 1, California, USA, 1967, p. 14.

[41] Tara Cox Matise, Mark Perlin, and Aravinda Chakravarti, *Automated construction of genetic linkage maps using an expert system (multimap): a human genome linkage map*, Nature genetics **6** (1994), no. 4, 384–390.

[42] D Mester, Y Ronin, D Minkov, E Nevo, and A Korol, *Constructing large-scale genetic maps using an evolutionary strategy algorithm*, Genetics **165** (2003), no. 4, 2269–2282.

[43] David I Mester, Yefim I Ronin, MA Korostishevsky, VL Pikus, AE Glazman, and Abraham B Korol, *Multilocus consensus genetic maps (MCGM): formulation, algorithms, and results*, Computational biology and chemistry **30** (2006), no. 1, 12–20.

[44] David I Mester, Yefim I Ronin, Eviatar Nevo, and Abraham B Korol, *Fast and high precision algorithms for optimization in large-scale genomic problems*, Computational Biology and Chemistry **28** (2004), no. 4, 281–290.

[45] DI Mester, YI Ronin, Y Hu, J Peng, E Nevo, and AB Korol, *Efficient multipoint mapping: making use of dominant repulsion-phase markers*, Theoretical and Applied Genetics **107** (2003), no. 6, 1102–1112.

[46] Newton E Morton, *Sequential tests for the detection of linkage*, American journal of human genetics **7** (1955), no. 3, 277.

[47] David W Mount, *Sequence and genome analysis*, Bioinformatics: Cold Spring Harbour Laboratory Press: Cold Spring Harbour **2** (2004).

[48] Christos H Papadimitriou and Umesh V Vazirani, *On two geometric problems related to the travelling salesman problem*, Journal of Algorithms **5** (1984), no. 2, 231–246.

[49] William H Press, *Numerical recipes in fortran 77: the art of scientific computing*, vol. 1, Cambridge university press, 1992.

[50] Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang, *On clustering heterogeneous social media objects with outlier links*, Proceedings of the fifth ACM international conference on Web search and data mining, ACM, 2012, pp. 553–562.

[51] Yefim Ronin, David Mester, Dina Minkov, and AB Korol, *Building reliable genetic maps: different mapping strategies may result in different maps*, Nat. Science **2** (2010), 576–589.

[52] Alejandro A Schäffer, Edward Stallknecht Rice, William Cook, and Richa Agarwala, *rh_tsp_map 3.0: end-to-end radiation hybrid mapping with improved speed and quality control*, Bioinformatics **23** (2007), no. 9, 1156–1158.

[53] Raed Seetan, Anne Denton, Omar Al-Azzam, Ajay Kumar, Jazarai Sturdivan, and Shahryar Kianian, *Iterative framework radiation hybrid mapping*, Society for Industrial and Applied Mathematics.

[54] Raed I Seetan, Anne M Denton, Ajay Kumar, M Javed Iqbal, Omar Al-Azzam, and Shahryar F Kianian, *A fast and scalable clustering-based approach for constructing reliable radiation hybrid maps*, Proceedings of the 12th International Workshop on Data Mining in Bioinformatics, ACM, 2013, pp. 34–41.

[55] Peter HA Sneath, Robert R Sokal, et al., *Numerical taxonomy. the principles and practice of numerical classification.*, 1973.

[56] Warren M Snelling, Readman Chiu, Jacqueline E Schein, Matthew Hobbs, Colette A Abbey, David L Adelson, Jan Aerts, Gary L Bennett, Ian E Bosdet, Mekki Boussaha, et al., *A physical map of the bovine genome*, Genome Biol **8** (2007), no. 8, R165.

[57] Lincoln Stein, Leonid Kruglyak, Donna Slonim, and Eric 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).

[58] Elizabeth A Stewart, Kathleen B McKusick, Amita Aggarwal, Ewa Bajorek, Shannon Brady, Angela Chu, Nicole Fang, David Hadley, Mark Harris, Sami Hussain, et al., *An sts-based radiation hybrid map of the human genome*, Genome research **7** (1997), no. 5, 422–433.

[59] DK Tasoulis, VP Plagianakos, and MN Vrahatis, *Unsupervised clustering of bioinformatics data*, European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems, Eunite, 2004, pp. 47–53.

[60] Vasudha Vashisht, *Open loop travelling salesman problem using genetic algorithm*, population **1** (2013), no. 1.

[61] Michael A Walter, Dominique J Spillett, Philip Thomas, Jean Weissenbach, and Peter N Goodfellow, *A method for constructing radiation hybrid maps of whole genomes*, Nature genetics **7** (1994), no. 1, 22–28.

[62] Jason TL Wang, Mohammed J Zaki, Hannu TT Toivonen, and Dennis Shasha, *Introduction to data mining in bioinformatics*, Springer, 2005.

[63] Daniel E Weeks and Kenneth Lange, *Preliminary ranking procedures for multilocus ordering*, Genomics **1** (1987), no. 3, 236–242.

[64] Peter Willett, *Recent trends in hierarchic document clustering: a critical review*, Information Processing & Management **24** (1988), no. 5, 577–597.

[65] Ying Zhao, George Karypis, and Usama Fayyad, *Hierarchical clustering algorithms for document datasets*, Data Mining and Knowledge Discovery **10** (2005), no. 2, 141–168.