CONSTRUCTING COMPARATIVE GENOME MAPS WITH UNRESOLVED MARKER ORDER

DEBRA GOLDBERG
Center for Applied Mathematics, Cornell University, Ithaca NY 14853
E-mail: debra@cam.cornell.edu

SUSAN MCCOUCH
Department of Plant Breeding, Cornell University, Ithaca NY 14853

JON KLEINBERG
Department of Computer Science, Cornell University, Ithaca NY 14853

Comparative genome maps are a powerful tool for interpreting the genomes of related organisms. The species maps which are the input to the process of constructing comparative maps are often themselves constructed from incomplete or inconsistent data, resulting in markers (or genes) whose order is not fully resolved. This incomplete marker order information is often handled by placing markers whose relative order cannot be reliably inferred together in a bin which is mapped to a common location. Previous automated and manual methods have handled such markers in an *ad hoc* or arbitrary way. We present efficient algorithms for comparative map construction that provide a principled method for handling unresolved marker order. The algorithms are based on a technique for efficiently computing a marker order that optimizes a natural parsimony criterion; in this way, they also yield a working hypothesis about the original incomplete data set.

1 Introduction

Comparative mapping is based on the observation that the order of homologous genes along the chromosomes of related species is often conserved. *Colinearity* (conservation of gene order), and to a lesser extent *synteny* (neighborhoods containing a number of homologous gene pairs) in chromosomal regions of different species suggests that these chromosomal segments are likely to be *homeologous* (derived from a common ancestral linkage group). Comparative maps identify colinearity or synteny between genomes of different species, and allow us to exploit the research accumulated for each of the species under consideration to gain new insights into issues including gene characterization, phylogenetic relationships, and principles of chromosome evolution.

Work in comparative mapping dates back as far as studies of Sturtevant and Weinstein in the 1920's[11,12], and it has grown into a very large area of research. We refer the reader to O'Brien et al[7] for a general review of comparative studies in mammals, Paterson et al for plants[8], and Römling and

Tümmler[9] for bacterial genomes. Indeed, comparative genomics has proven so useful in understanding human genetics that it has been termed "the key to understanding the human genome project"[2].

Despite the considerable amount of research in this area, there has been relatively little algorithmic work aimed at formalizing what is meant by a *comparative map* in a mathematical sense, or at providing a precise means for computing such a map from input data. In the absence of such a framework, the maps produced by different labs have been constructed on an individual and largely *ad hoc* basis, making it difficult to reason about these different maps from a common set of principles. Motivated by this state of affairs, Nadeau and Sankoff challenged the community in 1998 "to devise objective methods that reduce the arbitrary nature of comparative map construction"[6].

In recent work[4], we have proposed a formal model of comparative mapping as a *chromosome labeling problem*, in which the goal is to divide chromosomes into contiguous segments for which there is significant evidence of common ancestral linkage groups. We provide background on this model in Section 2.1, where we show a natural labeling criterion for chromosomal segments, based on a trade-off between parsimony and consistency, under which the optimal labeling can be computed efficiently. This approach is distinct from sequence-alignment methods, which work on a much more localized scale, and also distinct from algorithms for inferring chromosomal rearrangement scenarios, which essentially start with a structure like a comparative map, and propose hypotheses about evolutionary history. Our framework is perhaps most similar to work of Sankoff, Ferretti, and Nadeau[10] which seeks to find non-contiguous segments that "cover" all homologous genes in a dataset.

1.1 The present work: Unresolved marker order

In this paper, we propose an algorithmic approach for addressing the ubiquitous problem of *unresolved marker order* in comparative map construction. Genetic maps are constructed from linkage analysis of finite mapping populations, and frequently there are markers which cosegregate in all individuals, so their relative order cannot be determined. Analogously, in physical maps, two markers which are contained in exactly the same set of clones also cannot be ordered precisely. In addition, the raw data may be ambiguous or inconsistent, due to experimental or statistical error, leading to markers whose relative order cannot be determined with a sufficient degree of confidence. Existing computational techniques for comparative map construction, however, have generally relied on the tacit assumption that there is a completely specified linear order on markers; and this assumption is present in our previous work. Since this assumption rarely holds in practice, the resolution of marker order has essentially been dealt with in an *ad hoc* or arbitrary way.

Here we develop a principled method for handling unresolved marker order within our model of comparative mapping. We work with a standard representation for markers whose order cannot be determined: the markers are partitioned into bins, or *megaloci*; markers within the same megalocus are considered to have an unknown relative order, but there is a total order on the megaloci themselves. Thus, the resulting dataset looks linearly ordered, except that in place of individual markers we have a sequence of megaloci.

We provide an efficient algorithm that simultaneously constructs a comparative map and an ordering of the markers in each megalocus. These two tasks are inter-related, in the sense that the megalocus orders are computed so as to optimize a natural parsimony criterion for the map. Our main technical result is to show that these optimal orders can be computed in polynomial time, and, indeed, by an algorithm that performs well in practice. Indeed, we will see that the algorithm can actually run faster in practice on an input with megaloci than on a totally ordered set of markers of the same size; this is essentially because the megaloci serve as a "compressed" representation which the algorithm can manipulate at a high level. We supplement our algorithms with a set of results showing comparative map construction based on this method for mouse-human data. We note that while our optimal orders thus provide a canonical hypothesis about marker order, which can serve as a basis for further lab work, we do not claim that they represent the "correct" or "true" order — essentially, we simply do not have enough information in these settings to identify such a correct order.

A number of studies use representations not based directly on megaloci, and our approach can be adapted to handle several of these as well. We briefly discuss one such extension in Section 3.2.

2 Algorithms

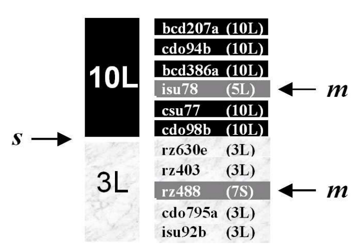
We cast comparative mapping as a *labeling* problem, as in our previous work[4]. We begin with two genomes, the *base* and the *target*, and we wish to *label* segments of the target using names of linkage groups from the base. In this section, we describe our underlying algorithms in detail. First we give some background, including notation and a review of the previously-published algorithms which form the foundation for this work. Then we develop a linear megalocus algorithm, which is extended to a stack megalocus algorithm in the final subsection. In Section 3, we discuss an implementation of this algorithm, and show some results from a comparative analysis of the human and mouse genomes, with human as the base and mouse as the target.

2.1 Chromosome Labeling: An Approach to Comparative Mapping

We fix a chromosome in the target genome, and let $M = \langle 1, 2, \ldots, n \rangle$ denote the sequence of comparatively mapped markers (genes) in order on this chromosome. We divide the base genome into linkage groups (usually chromosome arms) which will serve as labels for the target genome. Thus, we have a label set $L = \{c_1, c_2, \ldots, c_k\}$, where k is the number of linkage groups in the base genome. A comparative map is viewed as an assignment of labels to the markers of the target genome, i.e. a function $f : M \to L$.

We assume each marker i has been comparatively mapped to a single linkage group ℓ_i in the base genome (i.e. each marker i has a single homolog in the base genome, and it is located on ℓ_i). We say marker i has *type* ℓ_i. Markers that have not been comparatively mapped in the base genome are not informative for our purposes. We define a simple distance function $\delta(\cdot, \cdot)$ on pairs of labels as follows: $\delta(a, b) = 0$ if $a = b$; and $\delta(a, b) = 1$ if $a \neq b$. We extend this definition if one of the parameters is a set A, so that $\delta(A, b) = 0$ if $b \in A$; and $\delta(A, b) = 1$ otherwise. In the context of a labeling we say a marker i *matches* its label if $\delta(\ell_i, f(i)) = 0$; we call it a *mismatched marker* otherwise.

In previous work, we cast comparative mapping as the problem of computing an *optimal* labeling of the marker set[4]. In our basic *linear model*, the optimization criterion was based on balancing a *mismatch* penalty m and a *segment boundary* or *segment opening* penalty s. We refer the reader to Figure 1 for details. Only the ratio s/m affects the resulting optimal labelings, so this ratio is essentially the only tunable parameter in the algorithm; intuitively, s/m gives a minimum number of matching markers required to consider opening a new segment. We have found that in practice, the results produced by the algorithm are generally stable over a fairly wide range of parameter values.



A potential labeling is scored by assessing the penalty m for each mismatched marker, and the penalty s for each segment boundary. Formally, the objective function $Q(f)$ is:

$$s\left(|\{i : f(i) \neq f(i+1)\}|\right) + m\left(|\{i : f(i) \neq \ell_i\}|\right).$$

The objective function is minimized in an optimal labeling.

Figure 1. Diagram showing scoring scheme for linear model.

To compute an optimal labeling in this model, we use a dynamic programming formulation in which $S[i, a]$ denotes the optimal cost for labeling the suffix of M beginning at position i subject to the condition that $f(i) = a$. We initialize $S[n, a] = m \cdot \delta(\ell_i, a)$, and compute the optimal solution using a recurrence relation as follows:

$$S[i, a] \quad = \quad m \cdot \delta(\ell_i, a) \quad + \quad \min_{b \in L} \left(S[i{+}1, b] \quad + \quad s \cdot \delta(b, a) \right). \qquad (1)$$

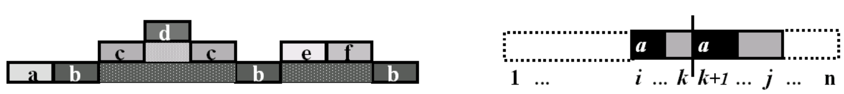
We extended the linear model to a stack model[4] which allows labels to be remembered as though pushed and popped from a stack. In the stack model, a label can still change at a segment boundary by being replaced with another label (with associated penalty s) as in the linear model, but a label can also change by having another label pushed on top of it, which can later be popped off to recall the earlier label. This is demonstrated in Figure 2a. Pushing a label also incurs a penalty of s, but popping is nearly free, incurring only a small penalty ϵ. This corrects the linear algorithm's problem with long-range dependencies, which impedes the labeling of "*aba*" label patterns generated by insertions and other important chromosome rearrangement events.

An optimal labeling in the stack model can also be computed using a dynamic programming algorithm, in which $S[i, j, a]$ denotes the optimal cost of a labeling f of the subsequence of M which starts at position i and ends at position j, subject to the condition that $f(i) = a$. We initialize $S[i, i, a] = m \cdot \delta(\ell_i, a)$, and make use of the following recurrence:

$$S[i, j, a] \quad = \quad \min \begin{pmatrix} m \cdot \delta(\ell_i, a) \; + \; \min_{b \in L} \; (S[i{+}1, j, b] \; + \; s \cdot \delta(b, a) \,), \\ \min_{i<k<j} \; (S[i, k, a] \; + \; S[k{+}1, j, a] \; + \; \epsilon \,) \end{pmatrix} \qquad (2)$$

See Figure 2b for a graphical view of how push/pop is accomplished.

We note that the balance between minimizing mismatches and minimizing stack operations reflects the notion of parsimony discussed in the introduction. While we do not go into the details here, our objective function can be viewed as arising from a maximum *a posteriori* approach with a prior probability term favoring labelings that involve a small number of stack operations. We also note that the process of pushing and popping on a stack is suggestive of the biological process of insertion; this idea is discussed in our earlier work[4].



a. b.

Figure 2. Graphical representation of: a. the stack model, and b. push/pop as implemented in recurrence of stack algorithm.

2.2 Linear Megalocus Model

We now extend our labeling framework to the case in which the input contains *megaloci*, the types of unordered sets of markers discussed in the introduction. The goal is to produce an order for the markers in each megalocus so that the resulting totally ordered set of markers has a labeling with as low a score as possible (under the linear or stack models respectively). The main difficulty here is that the ordering problems in the different megaloci can interact in complex ways, since we must produce a labeling for the full ordered set of markers. Despite this, we show that an order yielding an optimal labeling can be computed efficiently for both the linear and stack models.

We begin with the linear model, since the algorithm for the stack model will build on this. For each megalocus, we consider the set of markers belonging to the megalocus as a *supernode* Z. Within Z, there is an optimal ordering that clusters markers of the same type contiguously. Thus we will search only for solutions of this form: we seek an ordering over these *clusters*, rather than over the markers themselves. Figure 3a depicts an example of a supernode with four clusters, each consisting of markers of the same type.

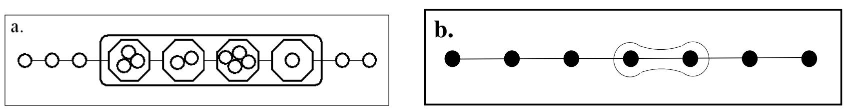


Figure 3. Diagram showing: a. markers (circles) arranged into clusters (octagons) within a supernode (rectangle), and b. High-level view of the map as viewed by the algorithm.

The key idea in the algorithm is that the clusters selected for the beginning and end of the supernode order are the ones that determine how the labeling of the chromosome before and after the supernode will interact with the labeling of the markers in the supernode. Once these two extreme clusters are selected, the remaining clusters can be ordered essentially arbitrarily. In keeping with this idea, we create a representation of the chromosome in which the supernodes and markers outside supernodes are totally ordered, and each supernode is represented by *two* consecutive positions in the order; the first of these positions will be assigned a label for the beginning of the supernode, and the second will be assigned a label for the end of the supernode. (See Figure 3b.) This pair of labels will be enough for us to determine an optimal ordering within the supernode by a post-processing step. Specifically, clusters within the supernode corresponding to the first supernode label (if any) will be placed at the beginning, clusters corresponding to the second label will be placed at the end, and the remainder will be ordered arbitrarily. If the two supernode labels are the same, then the markers matching this label can all be placed at the end, and all other clusters will be considered mismatches.

The full details of the algorithm and its correctness proof are somewhat complex, and due to the space limitations we can only sketch them here. The reader is referred to the Ph.D. thesis of the first author[3] for these details. Let $n' \leq n$ denote the number of positions in the modified map after each supernode has been replaced by a pair of positions. Let S denote the set of indices of supernode start positions, and E denote the set of indices of supernode end positions. If a is a label and $j \in S$ (so $j+1 \in E$), we define $n_j(a)$ to be the number of markers of type a in the supernode associated with start position j, and n_j to be the total number of markers in this supernode. We define $\ell_j = \ell_{j+1}$ to be the set of labels containing a homolog of a marker in the supernode associated with position j; i.e. $\ell_j = \{a \in L | n_j(a) \geq 1\}$.

The optimal labeling is constructed from a dynamic programming recurrence that follows the recurrence used in the basic linear model. As before, $S[i, a]$ denotes the optimal cost for labeling the suffix beginning at position i subject to $f(i) = a$. For markers outside supernodes, and for supernode end positions, this is built from $S[i+1, \cdot]$ as before. To deal with supernode start positions we include a cost for labeling the markers "hidden" inside the supernode by our representation. This cost can be determined from the labels for the supernode start and end positions, by augmenting the recurrence with a *hidden marker penalty* $p(i, \cdot, \cdot)$ defined for supernode start positions $i \in S$.

Thus, with $S[n', a] \ = \ m \cdot \delta(\ell_i, a)$, the recurrence is as follows:

$$S[i,a] \ = \ m \cdot \delta(\ell_i, a) \ + \ \min_{b \in L} \left(S[i+1, b] \ + \ s \cdot \delta(b, a) \ + \ p(i, a, b) \ \right). \quad (3)$$

It remains to define the hidden marker penalty $p(i, a, b)$. To prevent the implicit placement of a marker at both the start and end of a supernode, the case in which the two supernode indices receive the same label must be considered separately from the case in which they receive different labels, resulting in a two-case structure for $p(i, a, b)$:

$$p(i,a,b)) = \begin{cases} \sum_{c \neq a,b} (\min(s, m \cdot n_i(c)) \ + \ m \cdot \Delta_i(a,b) \ : \ \text{for } i \in S, \ a \neq b \\ \sum_{c \neq a} (m \cdot n_i(c)) \ + \ m \cdot \Delta_i(a,a) \ : \ \text{for } i \in S, \ a = b \\ 0 \ : \ i \notin S. \end{cases} \quad (4)$$

Markers which match either of the pair of supernode labels (a or b) will not impart any mismatch penalty, and segment opening penalties associated with these labels are handled explicitly by the recurrence, so we need only consider the "hidden" markers which don't match either of the labels a or b.

For $c \neq a, b$, the first terms in the definition of $p(i, a, b)$ give the total cost attributed to markers of type c due to mismatched markers *or* a segment boundary penalty. The definition for the case $a = b$ does not hinder homeologies from being labeled within a supernode; rather it requires there be distinct labels at the two ends of the supernode whenever markers in the supernode should be labeled with at least two labels.

The function Δ_i used in computing $p(i, a, b)$ adjusts for the effect of assigning mismatch penalties (m) both in the recurrence (for the first and last positions in the supernode) and in the function p. It is defined as follows. For $i \in S$ we define $\mu_i(a, b) = (\delta(\ell_i, a) + \delta(\ell_i, b))$, which is the number of mismatch penalties assessed by the recurrence for a supernode labeled with a and b at its ends. Note that $\mu_i(a, b) \in \{0, 1, 2\}$. The notation $min_{\mu_i(a,b)}$ indicates to sum the $\mu_i(a, b)$ smallest values.

$$\Delta_i(a, b) = \begin{cases} 0 & : \mu_i(a, b) = 0 \\ -2 & : \mu_i(a, b) > 0, a = b \\ 0 & : \ell_i \in \{\{a\}, \{b\}\}, a \neq b \\ -2 & : (*) \\ \min\limits_{\substack{c \neq a, b \\ c \in \ell_i}}{}_{\mu_i(a,b)} \left\{ \begin{array}{rl} 0 : & m \cdot n_i(c) \geq s \\ -1 : & m \cdot n_i(c) < s \end{array} \right\} & : \text{otherwise} \end{cases} \tag{5}$$

Condition $(*)$ is invoked when the first three conditions do not apply, the entire supernode has length $< s$ and consists exclusively of markers of a single type c, and c is not among the labels at the ends of the supernode.

By computing $p(i, a, b)$ for all i, a, and b prior to the recurrence loop and appropriate ordering of operations, this algorithm has running time $O(k^2 n)$, which is the same computational complexity as the original linear model. Since we view the label set as having fixed constant size, this is a running time linear in the number of markers.

2.3 Stack Megalocus Model

We now extend the stack model to also allow rearrangement of markers within megaloci. Since the order of clusters internal to a supernode (i.e. those that don't match the label at either end) is not explicitly determined by the recurrence, we do not allow pushing or popping with internal markers of a supernode; this maintains the stack structure through the megaloci. Given this, the algorithm for the stack megalocus model is based on the function $p(i, a, b)$ defined above for the linear megalocus model, together with a dynamic programming recurrence in which $S[i, j, a]$ has the same meaning as in the basic stack model. We initialize $S[i, i, a] = m \cdot \delta(\ell_i, a)$, and use the following recurrence:

$$S[i,j,a] \;=\; \min \left\{ \begin{array}{l} m \cdot \delta(\ell_i, a) \;+\; \min\limits_{b \in L} \left(S[i{+}1,j,b] + s \cdot \delta(b,a) + p(i,a,b) \,\right) \\ \min\limits_{\substack{i<k<j \\ k \notin S}} \;\left(S[i,k,a] + S[k{+}1,j,a] \right) + \epsilon \end{array} \right\} \tag{6}$$

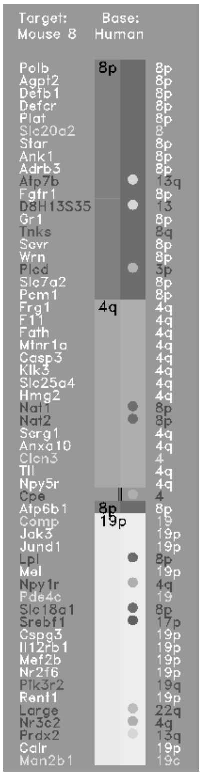
The correctness of the algorithm is established by arguing that an appropriate hidden marker penalty computation $p(\cdot,\cdot,\cdot)$ for a given supernode is included exactly once in a subproblem if and only if the subproblem includes both the supernode start and end positions (and thus includes all the markers of the supernode). The algorithm has running time $O(kn^3)$, which is the same computational complexity as the original stack model. In practice, this algorithm is actually *faster* than the basic stack algorithm for a totally ordered marker set of the same size; for the running time is more precisely $O(k(n')^3)$, and the reduction in the number of elements in the modified map more than makes up for the additional processing for each element.

3 Results and Discussion

3.1 Computational results

The stack megalocus algorithm was implemented in Java and executed on a Sun Ultra-Sparc 10 running Solaris. The implementation was verified using synthetic data. We tested the stack megalocus algorithm with mouse-human data taken from the Mouse Genome Database[1]. The resulting comparative maps compared favorably with the mouse-human maps published by the Human Genome Project[5], despite the fact that they were produced from different input data. Chromosomes with up to 260 markers ran in about 10 seconds. The total processing time for all 19 mouse autosomes is about two minutes. Results were displayed using an OpenDX visualization program, as explained in Figure 4. Due to memory limitations, one mouse chromosome could only be processed after the label set L was manually reduced to only those chromosomes possible in an optimal labeling. We are exploring many space-efficiency options, but are not too concerned since mouse-human is the densest comparative data set, and computing power will improve as more data accumulates.

To provide a sense for the types of analysis one obtains from our stack megalocus algorithm, we have extracted the labeling of three small chromosomal regions in the mouse genome from a full genome analysis. Results from the original stack algorithm and the stack megalocus algorithm are shown and contrasted in each case. In these cases, as in many portions of the genome, the

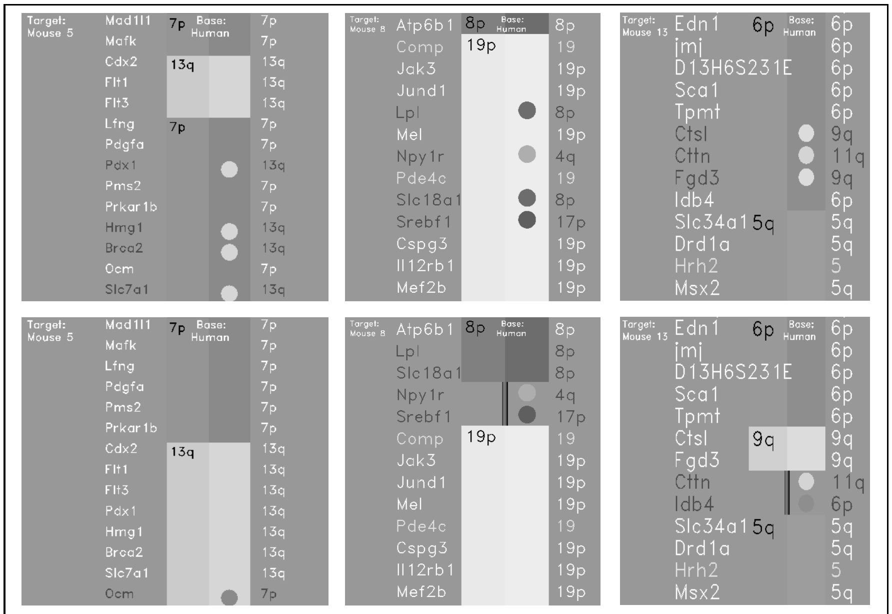


In the accompanying figure, the column of marker names on the left are the mouse chromosome 8 markers which have known homologs (actually orthologs) in human. The shaded rectangles to the right of these marker names show the labeling assigned by one of our comparative mapping methods, colored by chromosome. The actual visualization is in full color for optimally distinguishing among labels; the label name itself is displayed at the top of each rectangle. A translucent band is overlaid over the left half of these rectangles to indicate the arm. The right-most column shows the linkage group of the homolog. Some of the homologs are mapped to a centromeric region, and others are mapped only to a chromosome (the arm is unknown); depending on the precise location of these, they may match the linkage group of either chromosome arm. Marker names and homolog locations of markers which match their assigned label are shown in white, mismatches are shown as black, and those which *may* be matches are shown in gray. A circle color-coded by the chromosome of the homolog is overlaid on the labeling rectangles, providing another way to visualize most mismatches (mismatches involving the two distinct arms of one chromosome are not apparent this way). Certain of the segments (colored rectangles) are connected with an intervening black bar, the result of a post-processing heuristic that indicates portions of the chromosome not considered clearly homeologous to any human linkage group.

Figure 4. Results of the original stack model for a portion of mouse chromosome 8.

rearrangement of markers in a megalocus allows a significantly more parsimonious labeling and provides a hypothesized canonical order for these markers.

Figure 5a shows a region of mouse chromosome 5 where rearrangement of markers in the same megalocus has allowed for a map with fewer mismatches. Figure 5b shows a region of mouse chrosome 8 where parsimonious rearrangement of co-located markers has resulted in a map where mismatched markers can be placed between labeled segments, which is preferable. Figure 5c shows a region of mouse chromosome 13 where rearrangement has also enabled the formation of an additional labeled segment. In this case the proposed segment is a small segment that pushed to a larger segment further down the chromosome, which could be suggestive of a chromosomal rearrangement such as an insertion or inversion. Again, this is a hypothesis that can be tested by further lab investigations (for example, by placing additional markers in this region or sequencing an area around this region in both genomes).



a. Mouse chromosome 5 b. Mouse chromosome 8 c. Mouse chromosome 13

Figure 5. Detail of 3 mouse chromosomes in mouse-human comparisons. The upper figures shows results from the stack algorithm, and the lower figures show results from the stack megalocus algorithm.

3.2 Discussion

This paper seeks to lay a principled foundation for comparative mapping studies in the presence of uncertain marker order. We use marker order and not distance between markers, and have not incorporated species-specific information, so that our algorithms work for a wide variety of species, for genetic and physical as well as high- and low-resolution species maps. We impose no assumptions about evolutionary mechanisms. Results from these algorithms can form the basis of hypotheses to guide further lab studies.

Some maps, such as most versions of the human map, do not use the megalocus representation; instead each marker is assigned an interval where it is likely to be located. The intervals of two markers overlap if and only if their relative order cannot be resolved, and the relative order of markers may be neither completely known nor completely unknown. For these, there is a simple modification to the stack algorithm that assigns a boundary point at the beginning and end of each interval, and assigns a fractional weight to the subinterval between consecutive boundary points for each marker whose interval spans it, such that the weight attributed to each marker sums to one.

We are in the process of putting together a web site where our suite of comparative mapping programs will be made publicly available. DeCAL (Detecting Common Ancestral Linkage segments) will provide access to both the stack algorithm and the stack megalocus algorithm. We are also investigating algorithmic extensions of this work; in particular, it is an interesting open question to extend the approach for pairwise genome comparison discussed here to one that allows for the simultaneous comparison of multiple genomes.

3.3 Acknowledgments

We thank Chris Pelkie and Vic Goldberg for their help with this work. The first author was supported in part by NSF Training Grant DEB-9602229 and the Packard Foundation Fellowship of the third author. The second author was supported in part by USDA National Research Initiative grant 94-37310-0661 and Cooperative State Research Education and Extension Service NYC 149-401. The third author was supported in part by a David and Lucile Packard Foundation Fellowship, an ONR Young Investigator Award, and NSF Faculty Early Career Development Award CCR-9701399.

References

1. JA Blake *et al.* *Nucleic Acids Research* 29(1):91-94, 2001.
2. MS Clark. *Bioessays* 21(2): 121-130, 1999.
3. D Goldberg. Algorithms for the construction of comparative genome maps. Ph.D. thesis, Cornell University, 2001.
4. D Goldberg, S McCouch, J Kleinberg. in *Comparative Genomics*, D Sankoff, JH Nadeau, eds., *Series in Computational Biology* Vol 1, Kluwer Academic Press, 2000.
5. Mouse and Human Genetic Similarities. Human Genome Project, Image Gallery. http://www.ornl.gov/hgmis/graphics/slides/98-075r2jpg.html
6. J Nadeau and D Sankoff. *Trends in Genetics*, 14(12):495-501, 1998.
7. SJ O'Brien *et al.* *Science* 286:458-481, 1999.
8. AH Paterson *et al.* *Plant Cell* 12(9):1523-1539 2000.
9. U Römling and B Tümmler. *J Biotechnology* 33:155-164, 1994.
10. D Sankoff, V Ferretti, JH Nadeau. *J Comp Bio* 4(4):559-565, 1997.
11. AH Sturtevant. *Proc. Nat. Acad. Sci.* 7:235-237, 1921.
12. A Weinstein. *Proc. Nat. Acad. Sci.* 6:625-639, 1920.