

Finding Anchors for Genomic Sequence Comparison

Ross A. Lippert ^{*†} Xiaoyue Zhao [‡] Liliana Florea ^{*†} Clark Mobarry [†] Sorin Istrail [†]

ABSTRACT

Recent sequencing of the human and other mammalian genomes has brought about the necessity to align them, to identify and characterize their commonalities and differences. Programs that align whole genomes generally use a seed-and-extend technique, starting from exact or near-exact matches and selecting a reliable subset of these, called anchors, and then filling in the remaining portions between the anchors using a combination of local and global alignment algorithms, but their choices for the parameters so far have been primarily heuristic. We present a statistical framework and practical methods for selecting a set of matches that is both sensitive and specific and can constitute a reliable set of anchors for a one-to-one mapping of two genomes from which a whole-genome alignment can be built. Starting from exact matches, we introduce a novel per-base repeat annotation, the *Z*-score, from which noise and repeat filtering conditions are explored. Dynamic programming-based chaining algorithms are also evaluated as context-based filters. We apply the methods described here to the comparison of two progressive assemblies of the human genome, NCBI build 28 and build 34 (<http://genome.ucsc.edu>), and show that a significant portion of the two genomes can be found in selected exact matches, with very limited amount of sequence duplication.

Categories and Subject Descriptors

J.3 [Computer Applications]: Life and Medical Sciences-Biology and Genetics

*Corresponding authors: {*Ross.Lippert, Liliana.Florea*}@celera.com.

[†]Informatics Research, Celera/Applied Biosystems, 45 W. Gude Dr., Rockville, MD 20850.

[‡]Department of Statistics, University of California at Berkeley, Berkeley, CA 94720.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

RECOMB '04 March 27–31, 2004, San Diego, California, USA.
Copyright 2004 ACM 1-58113-755-9/04/0003 ...\$5.00.

General Terms

Algorithms, Design

Keywords

Whole-genome alignments, suffix trees, MUMs

1. INTRODUCTION

The availability of the human and other mammalian genome sequences [1, 2, 3, 4] spurred bioinformatics efforts to produce powerful whole-genome alignment tools to identify and characterize their commonalities and differences. Conserved syntenic regions between genomes of more distantly related species may reveal functional regions that were preserved due to negative selective pressure, while differences between genomes of closely related organisms can point to causal factors for phenotypic differences. Whole-genome alignments between different genome assemblies of a same species can be used to identify polymorphisms, validate or point out assembly errors, and to ease annotation efforts by tracking of features such as genes, SNPs and STS markers between different assembly versions.

Programs that align whole genomes generally use a seed-and-extend technique, starting from exact or near-exact matches and selecting a reliable subset of these called anchors, and then filling in the remaining portions between the anchors using local and global alignment algorithms, or combinations of these methods (MUMmer [5]; BLASTZ [6]; AVID [7]; LAGAN [8]), but their choices for the values of parameters so far have been primarily heuristic. MUMmer uses the set of maximal unique matches as its anchors. AVID starts from maximal but not necessarily unique exact matches, filtered to remove those that are shorter than 50% of the length of the longest match. LAGAN generates anchors from short chains of 7-mers separated by a maximum distance. The procedure is recursively applied between previously discovered anchors until the inter-anchor distance is less than some fixed maximum value. BLASTZ first identifies primary anchors as high-scoring local alignments, starting from pairs of spaced 12-mers, with possibly one transition, and extending them in two stages, by allowing substitutions and then gaps. The method is re-applied between consecutive primary matches to find new, secondary matches. Both BLASTZ's and LAGAN's parameters are trained on sets of orthologous regions between several related species. LAGAN is designed to align a priori identified orthologous regions, and therefore is of limited use for aligning whole genomes, where rearrangements may occur. One daunting challenge to alignment

programs is sequence repetitiveness. LAGAN and AVID attempt to avoid repeats by applying the RepeatMasker [9] program on the input sequences and assigning matches between masked regions a much lower priority of selection in their alignment generation steps. Not only is this computationally expensive, but it is not effective for large genomic duplications that are left unmasked by RepeatMasker.

Here we revisit the problem of selecting a reliable set of matches, or anchors, for scaffolding a one-to-one mapping and alignment of whole genomes. We tackle this from the perspective of a minimalist approach to whole-genome comparison and alignment: since later steps are more expensive and their accuracy depends critically on the anchor set selected, we seek a set that is sensitive and specific.

An ideal anchor set should be *comprehensive* and *non-repetitive*, should consist of matches that are *not likely to have occurred by chance* and that are *correctly located* on the genome-to-genome map, and should be one that can be *constructed easily and efficiently*. A good starting set is that of maximal unique exact matches, or MUMs [5]. However, a MUM can contain a pair of repetitive subsequences and thus be misleading. As a means to screen for repetitiveness, we can request that every pair of matched bases be contained in at least one unique submatch of some length k_{Rep} . Moreover, the length of a match is related to its statistical significance. To eliminate matches that are likely to have occurred by chance, we may disallow matches smaller than some k_{Size} in size. Furthermore, spurious matches may divert the alignment into areas of false rearrangements. We use the framework of dynamic programming and chaining to select a subset of ordered and oriented matches, based on the genomic context defined by their neighbors. Lastly, we may demand that each match contain at least one exact unique match of size k_{Seed} , thus ensuring that all matches are exact extensions of unique k_{Seed} -mer matches, where k_{Seed} is chosen such as to detect as large as possible a fraction of the MUMs set while keeping the amount of repetitive sequence low.

The selection of k_{Rep} , k_{Size} , k_{Seed} and chaining parameters is critical to obtaining a one-to-one mapping of good quality. We applied the theories of the length of longest exact match [10] based on the Chen-Stein approximation method [11] to explore the behavior of the longest exact match under a Bernoulli model, and used these to identify sensitive and specific parameters. The real genome may not be so simply modeled as a Bernoulli sequence. We use it strictly as a null model, to assess the departure from randomness of our observations.

In order to substantiate our results empirically, we applied the selection methods described here to the comparison between two different public assemblies of the human genome, the NCBI build 28 and build 34 (<http://genome.ucsc.edu>). Lacking a systematic way to precisely identify false positives and negatives and their true counterparts, we use the amount of non-duplicated sequence found in matches (*single coverage*) as a measure for sensitivity, and the amount of sequence that occurs in multiple matches (*double coverage*) as a measure for specificity. These measures do not account for spurious matches that are incorrectly located, which would require more rigorous testing on control data, such as a pair of simulated genomes whose one-to-one map is known.

Starting from the set of MUMs and appropriately choosing the size k_{Seed} , we find that the selected subset of matches

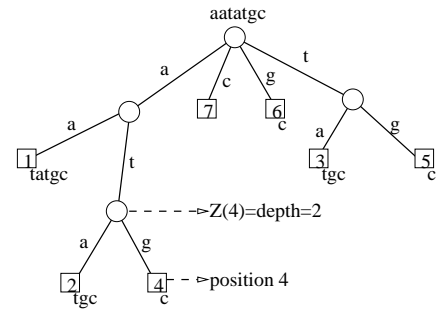


Figure 1: Suffix tree representation and Z -score calculation for the word 'aatatgc'. The Z -score at position i is the depth of the parent of suffix i .

contains comparable signal with slightly less noise than the entire set. This shows that suffix-based approaches, generating MUMs, and hash index-based techniques, generating matches by extending exact unique k -mer pairs, are equally powerful in producing a good starting match set. By appropriately choosing the threshold k_{Rep} for repeat filtering, k_{Size} for filtering short matches, and chaining parameters for screening potentially spurious hits, we determine that the specificity of the match set can be further increased, with little effect on the sensitivity.

The paper is organized as follows. Sections 2 and 3 introduce the data structures and the notation and definitions related to the probability theory of exact matches used by the subsequent sections. Detailed calculation and reasoning about choosing the different parameter values for match selection are given in Section 4. Results from applying the methods to a set of matches resulting from the comparison of the two assemblies, for empirical validation, are presented in Section 5. Lastly, the Chen-Stein method and the theories of longest exact matches are given in Appendix 7.

2. MUMS AND Z -SCORES

Throughout this paper, we will assume that all matches are *exact*, unless specified otherwise. A unique match therefore implies that each of its two (identical) subsequences is unique with respect to both genomes.

In order to evaluate various parameter choices of our anchor selection process we computed two fundamental and computationally expensive kinds of data. The first kind is the MUMs, or maximal unique matches, first specified in the computational biology literature in [5]. The second kind is the Z -score, which we believe to be a novel annotation. The Z -score is a per-base repeat annotation, which records the length of the longest exact match of a subsequence starting at that position elsewhere in the sequence or its reverse complement. From this annotation, the filtering conditions for any k_{Rep} , k_{Seed} can be explored.

Like the MUMs, the Z -scores can be efficiently computed via suffix trees [12, 13]. One can construct a compressed suffix tree on the given sequence and record for each leaf, or suffix position, the depth of its parent (Figure 1). In our case, the Z -scores are computed for the assembly and its reverse complement, for each of the two assemblies compared.

Z -scores serve primarily to allow the efficient application of repeat filtering conditions. Specifically, a score $Z_j \leq l - 1$

signifies that the subsequence of length l starting at position j has a unique occurrence in its genome. For the sake of simplicity, we henceforth assume that the matches are between the forward, *i.e.* non-reverse complemented, sequences of the two genomes. Thus, given a match, to apply the repeat filter one removes any base at position i , and its corresponding base in the other genome, that is not contained in a unique match of length at most k_{Rep} , *i.e.* for which $Z_j \geq k_{Rep}$ for all $j : i - k_{Rep} < j \leq i$.

We can also use Z -scores to answer questions of accessibility: given a maximal unique match, determine the length of the shortest *unique* sub-match from which the match can be re-constructed. Given a unique match with beginning and ending positions b, e in the first subsequence and b', e' in the second subsequence, the length of the shortest unique match it contains is

$$1 + \min\{z : z = \max(Z_i, Z'_{i-b+b'}), z + i \leq e \text{ and } b \leq i \leq e\}.$$

While the computation of the MUMs and the Z -scores requires significant resources, the post-processing required to select an anchor set is inexpensive. Computing the MUMs and Z -scores each used 25GB of memory and took 27 CPU hours on an IBM p690 (1300 MHz Power PC), while the subsequent selection of matches with the methods described in this paper used less than 1GB of space and 1 hour of CPU time. Since neither the MUMs nor the Z -scores depend on our parameter choices, we can defer any decisions about our parameters until after the data structures are computed.

3. PRELIMINARIES

Let $\underline{A} = \{A_i, i = 1, \dots, n\}$, $\underline{B} = \{B_j, j = 1, \dots, n\}$ be independently and identically distributed with letter probabilities p_a, p_c, p_g, p_t . We call this distribution of strings the *Bernoulli model*. When $p_a = p_c = p_g = p_t = \frac{1}{4}$ we call this the *uniform Bernoulli model*. We define two random variables,

$$R_n = \max_m \{A_k = B_{j+k}, k = 1, \dots, m, 0 \leq j \leq n - m\}$$

$$H_n = \max_m \{A_{i+k} = B_{j+k}, k = 1, \dots, m, 0 \leq i, j \leq n - m\}$$

R_n is the length of the longest exact match of a random sequence across another sequence. H_n is the length of the longest exact match when shifts are allowed. Waterman [10] gives the limiting distribution of H_n . Under a uniform Bernoulli model, the limiting distribution of R_n can be similarly obtained. For the non-uniform case, the limiting distribution is estimated by a Monte Carlo method. The details of the theories and the Monte Carlo method are given in Appendix 7.

The Z -score can be formulated mathematically as

$$Z_{i,n} = \max_m \{A_{i+k} = A_{j+k}, k = 0, \dots, m - 1, 1 \leq j \neq i \leq n - m + 1\}$$

(writing just Z_i when n is understood). It is very similar to R_n , except that it allows self-overlap. Under the Bernoulli model, $Z_{1,n}, Z_{2,n}, \dots$, are identically and dependently distributed. They follow the same limiting distribution as R_n , but have a Markov dependence (given Z_i, Z_{i+1}

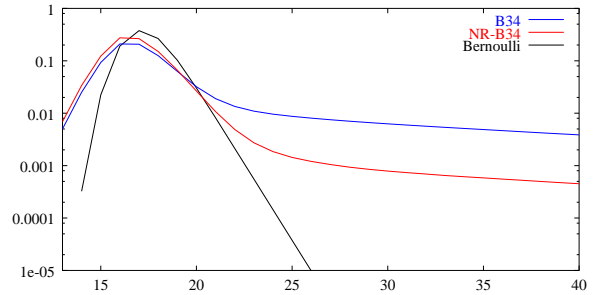


Figure 2: Probability of $Z = k$ for the Bernoulli model and as estimated on NCBI 34 and non-repetitive NCBI 34.

is independent of all the rest of its previous Z s, for any $i = 0, 1, \dots, n - 1$). In Figure 2, we show the distributions of Z -scores at a randomly selected position in the human genome (NCBI release build 34) and at a random *non-masked* position within the same genome, after masking biological repeats with the RepeatMasker program. The mode of the Bernoulli-derived distribution is just slightly to the right of that of the genomic distributions, and the genomic distributions have significant tails.

The base composition of the human genome in a non-repeat region, which is used for H_n and R_n , is $p_a = 0.29, p_c = 0.21, p_g = 0.21, p_t = 0.29$. Let $p \equiv P(A_1 = B_1) = \sum_{i \in \{a,c,g,t\}} p_i^2 = 0.2597$ be the probability of observing a nucleotide match assuming a Bernoulli sequence model. We take $n = 5.74 \times 10^9$, the estimated length of the human genome and its reverse-complement.

4. APPLICATIONS TO ASSEMBLY TO ASSEMBLY COMPARISON

In this section, we present several filtration methods and exploit the properties of the tractable mathematical model introduced earlier to identify and explain our parameter choices, then compare the results to their empirical counterparts. Throughout the analyses, we use the *single coverage (SC)* and *double coverage (DC)* measures defined in the introduction to assess the sensitivity and specificity of a given match set.

4.1 Noise filtering: threshold for match length

In a genomic sequence with high compositional complexity, in which a fraction of the sequence is, however, nearly exactly duplicated, the distribution of match lengths between non-duplicated regions can be approximated well from a Bernoulli sequence model. The length of a longest spurious exact match across two assemblies can then be assumed to follow the same distribution as H_n with non-repetitive base probabilities.

Table 1 gives the probability for the longest exact match to be greater than or equal to k_{Size} under the Bernoulli model, for a selection of relevant k_{Size} values. We can see that, between two random assemblies, it is very unlikely to find a match with length longer than or equal to 40, and that even for a threshold as low as $k_{Size} = 35$ the remaining matches are very likely to be significant. This suggests that we can set $k_{Size} \geq 35$ to reduce the amount of matches

k	32	33	34	35	36	37	38	39	40
$P(H_n \geq k)$	9.88e-01	6.85e-01	2.59e-01	7.48e-02	2.00e-02	5.23e-03	1.36e-03	3.54e-04	9.18e-05

Table 1: Probability that the longest exact match is greater than or equal to k under a Bernoulli model.

k	$P(\cup_{i=1}^k \{Z_i \geq k\})$	$E(\sum_{i=0}^{n-k} I(\cup_{j=i+1}^{i+k} \{Z_j \geq k\}))$
16	1.56e-01	8.98e+08
17	1.83e-04	1.05e+06
18	1.59e-08	9.14e+01
19	1.38e-11	7.93e-02
20	1.73e-13	9.95e-04
21	6.86e-15	3.94e-05

Table 2: Probability and expected number of repeat-filtered bases estimated from the Bernoulli model.

occurring by chance.

This simple notion of match significance is highly effective in screening for short spurious matches, but is limited to same-species comparisons, in which the distribution of lengths for the true and random matches are clearly distinctive.

4.2 Repeat filtering

The uniqueness property is desired of a set of matches, as it indicates specificity. Aside from biological repeats, large genomic duplications also contribute to repetitive MUMs. We can apply the filtering condition given in section 2 parameterized by k_{Rep} to remove all positions i that are not contained in a unique match of length at most k_{Rep} , *i.e.* for which $Z_j \geq k_{Rep}$ for all $j : i - k_{Rep} < j \leq i$. However, too conservative a choice of k_{Rep} results in the fragmentation of a match into smaller, hence less significant matches, which can result in an unnecessary loss of coverage.

Let $P_{rep} = P(\prod_{i=1}^k \{Z_i \geq k\})$ be the probability that $Z \geq k$ consecutively across k positions, which is the condition for the removal of a base by our repeat filtering rule. Table 2 summarizes the probability P_{rep} and expected occurrence of such an event across a Bernoulli genome. We can see that it is very unlikely that $Z \geq k_{Rep}$ across k_{Rep} consecutive positions when $k_{Rep} \geq 18$, with less than 92 false deletions among all the $\sim 5.74 \times 10^9$ possible positions in the genome and its reverse complement.

In our sample comparison, we see that when decreasing k_{Rep} for repeat filtering both the single and double coverage rates decrease, but the latter is drastically reduced, by two orders of magnitude relative to single coverage (Table 3). Note that the amount of sequence lost in the trimmed matches can be easily recovered by extension in the later alignment steps.

4.3 Match accessibility: unique sub-match size

Every unique match can be generated from a shortest unique sub-match, possibly itself, which we call a *seed*. For a given L , the sets of matches of length greater than or equal to L that can be accessed by increasing values of k are included in each other. Therefore, larger seeds can access a larger set of matches than smaller ones, but may be more ambiguous, since any match containing a seed of length $k+1$

k_{Rep}	SC	DC
16	0.59057	6.0016e-06
17	0.65696	1.5178e-05
18	0.69887	2.4944e-05
19	0.71374	3.3517e-05
20	0.73194	4.2853e-05
21	0.74376	5.1184e-05
22	0.75421	5.9241e-05
23	0.76349	6.7472e-05
24	0.77182	7.5851e-05
25	0.77924	8.4866e-05
30	0.80765	1.3257e-04
40	0.83940	2.6511e-04

Table 3: Single and double coverage for varying choices of k_{Rep} .

has the property that any sub-match with length k is exactly repeated somewhere in the match. We call such matches k -repetitive. We want to identify a seed size k_{Seed} that can access a comprehensive set of matches, without significant loss of sensitivity.

From a practical point of view, seed size selection is important for some implementations based on hash tables of k -mer positions in the genome. Indeed, small seed sizes lead to sparse tables, due to the higher multiplicity of k -mers, while large ones cannot be represented in a machine word to allow efficient bit-processing. The uniqueness property of the seed also makes the retrieval and processing of k -mers computationally efficient.

Let P_{k-rep} be the probability that a match is k -repetitive, or equivalently, that its seed is $k+1$ bp or longer. Based on the Monte Carlo results for R_n under the Bernoulli model and the Markov property of Z 's, we have

$$\begin{aligned}
P\left(\prod_{i=1}^{L-k+1} \{Z_i \geq k\}\right) &= P(Z_1 \geq k) \prod_{i=2}^{L-k+1} P(Z_i \geq k | Z_{i-1} \geq k) \\
&= P(R_n \geq k) (P(Z_2 \geq k | Z_1 \geq k))^{L-k},
\end{aligned}$$

and

$$\begin{aligned}
P_k &= P(Z_2 \geq k | Z_1 \geq k) \\
&= P(Z_2 \geq k, Z_1 > k | Z_1 \geq k) \\
&\quad + P(Z_2 \geq k, Z_1 = k | Z_1 \geq k) \\
&= P(Z_2 \geq k | Z_1 > k) P(Z_1 > k | Z_1 \geq k) \\
&\quad + P(Z_2 \geq k | Z_1 = k) P(Z_1 = k | Z_1 \geq k) \\
&= \frac{P(Z_1 > k)}{P(Z_1 \geq k)} + P(Z_2 \geq k) \frac{P(Z_1 = k)}{P(Z_1 \geq k)} \\
&= \frac{P(R_n > k)}{P(R_n \geq k)} + P(R_n = k)
\end{aligned}$$

Table 4 shows P_{k-rep} for different values of k , assuming a minimum match length $L = 40$ bp. Our experiments comparing different assemblies of the human genome show that

k	$P(\cup_{i=1}^{L-k+1}\{Z_i \geq k\})$	$E(\sum I(\cup_{i=1}^{L-k+1}\{Z_i \geq k\}))$
15	8.75e-01	8.75e+05
16	5.93e-02	5.93e+04
17	6.24e-06	6.24e+00
18	1.43e-10	1.43e-04
19	3.64e-13	3.64e-07
20	4.68e-14	4.68e-08

Table 4: Probability and expected occurrence of exact matches $L = 40$ bp or longer within which any sub-word with length k is repetitive, or equivalently, which are generated from a seed of size $\geq k + 1$.

a typical number of MUMs is $\sim 10^6$. Multiplying the second column by 10^6 therefore gives an estimate of the expected number of k -repetitive MUMs in the genome. Note that k -repetitive matches, or equivalently, matches that require a seed of size $k + 1$ or longer, are very unlikely to occur for $k \geq 17$, with less than 7 expected occurrences in the genome. Therefore increasing the seed size k_{Seed} above 18 is not likely to contribute to a significant increase in sensitivity, while it may increase repetitiveness.

In our empirical evaluation, Figure 3 shows the fraction of MUMs accessed as extensions of k_{Seed} unique matches for relevant values of k_{Seed} , as well as the fraction of bases from the NCBI build 34 genome which are contained in the accessed MUMs. Table 5 gives the single coverage and double coverage rates by MUMs for the same genome for different values of k_{Seed} . Extra columns have been added to show the incremental gain in the coverage rates for k_{Seed} over $k_{Seed} - 1$. Increasing k_{Seed} beyond 17 brings diminishing returns, in that equally many repetitive (ΔDC) and non-repetitive (ΔSC) bases are recruited with each increment, while single coverage increases by only $\sim 0.2\%$ of the coverage of the full set of MUMs.

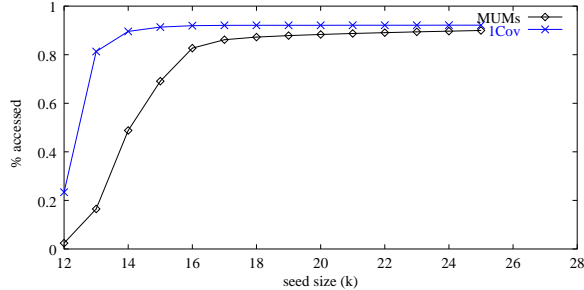


Figure 3: Sequence coverage (Cov) and fraction of MUMs ($MUMs$) accessed when varying the seed size k .

4.4 Match filtering by context: chaining

The noise and repeat filtering techniques presented fail in cases where slight differences between otherwise duplicated regions of the genome generate a deceptively unique match. In addition, our simple noise filter cannot be applied to cross-species comparisons, where the expected match length is close to that of the Bernoulli model. The resulting spurious matches can confound whole-genome alignment algorithms, by creating false rearrangement events.

Under the long-standing and accepted assumption that matches are more likely to be preserved as a block, one can use the genomic context to identify off-diagonal events that are likely to be spurious. Chaining entails selecting a subset of the matches, ordered and oriented, that will guide the construction of the nucleotide-level alignment. Since the statistical bases of groups of matches are considerably less developed than those for matches alone, we approach this from a more empirical perspective.

Chaining algorithms have been successfully and efficiently used to select a subset of matches to anchor the alignment between two typically orthologous sequences [14]. We introduce the concept of chaining graph $G = (V, E)$, which has as vertices the set of matches $V = (f_1, \dots, f_m)$ connected by directed edges $f_i \rightarrow f_j$ that establish precedence relationships, *i.e.* what match can follow another in a chain consistent with the target alignment. Each match f_i has an associated score $score(f_i)$ equal to its alignment score, and each edge has a connection penalty $connect(f_i, f_j)$ that approximates the cost of the unmatched sequence fragments between f_i and f_j in the overall alignment. A maximum scoring path, or chain, through the set of matches can then be computed with dynamic programming.

The whole-genome alignment problem introduces new elements into this formulation, such as the need to account for rearrangement of genomic sequence (translocations, reversals). We consider a scoring scheme in which every nucleotide match is assigned some positive score M and each type of rearrangement event receives a penalty. The following is a general framework for a linear scoring dynamic programming-based chaining model. We consider only exact matches here, but generalizing to other types is immediate.

A match f is completely described by its projections (b^1, e^1) and (b^2, e^2) and its orientation ori : $+1 =$ forward (diagonal) or $-1 =$ reverse complement (anti-diagonal). We denote $f_i \rightarrow f_j$ if f_i 'can precede' f_j in the alignment, *i.e.* there is an edge between f_i and f_j . For simplicity, we only allow an edge between f_i and f_j if $b_i^1 < b_j^1$ and either $b_j^2 \geq e_i^2$ or $e_j^2 \leq b_i^2$ (Figure 4.4). (In practice, we may relax the condition by allowing at most $L < k_{Size}$ overlapping bases between consecutive matches.)

The score of a path $P = f_1 \rightarrow f_2 \rightarrow \dots \rightarrow f_m$ then is:

$$\begin{aligned}
score(P) &= \sum_{i=1,m} score(f_i) \\
&\quad - \sum_{i=2,m} connect(f_{i-1}, f_i) \\
&= M * \sum_{i=1,m} length(f_i) \\
&\quad - O * \sum_{i=2,m} reversal(f_{i-1}, f_i) \\
&\quad - R * \sum_{i=2,m} translocation(f_{i-1}, f_i) \\
&\quad - D * \sum_{i=2,m} diag_diff(f_{i-1}, f_i)
\end{aligned}$$

where:

$$reversal(f_{i-1}, f_i) = 1 \text{ if } ori_i \neq ori_{i-1}; 0 \text{ otherwise}$$

k_{Seed}	SC	DC	ΔSC	ΔDC	$\frac{\Delta DC}{\Delta SC + \Delta DC}$
14	0.895980	0.001255	0.082975	0.000670	0.008
15	0.913541	0.001994	0.017561	0.000740	0.040
16	0.919603	0.003044	0.006062	0.001050	0.148
17	0.920696	0.003694	0.001092	0.000650	0.373
18	0.920940	0.003919	0.000245	0.000225	0.479
19	0.921047	0.004046	0.000106	0.000127	0.545
20	0.921168	0.004144	0.000121	0.000098	0.447
21	0.921257	0.004228	0.000089	0.000083	0.483
22	0.921335	0.004293	0.000078	0.000065	0.455
23	0.921401	0.004354	0.000066	0.000061	0.482
24	0.921463	0.004410	0.000062	0.000055	0.473
25	0.921495	0.004476	0.000032	0.000066	0.675

Table 5: Single coverage and double coverage increase with the choice of seed size.

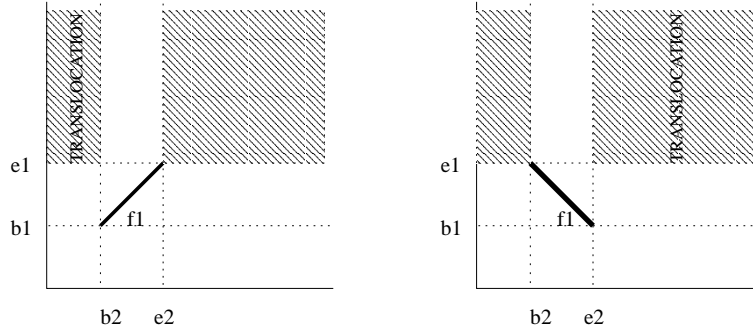


Figure 4: Areas of extension for a chain ending in f_1 .

$$translocation(f_{i-1}, f_i) = \begin{cases} 1, & \text{if } (b_i^2 < b_{i-1}^2 \text{ and } ori_{i-1} = 1) \\ & \text{or } (b_i^2 > b_{i-1}^2 \text{ and } ori_{i-1} = -1) \\ 0, & \text{otherwise} \end{cases}$$

$$diag_diff(f_{i-1}, f_i) = \begin{cases} \min(C, |b_i^2 - b_i^1 - e_{i-1}^2 + e_{i-1}^1|), & \text{if } ori_{i-1} = 1 \\ \min(C, |e_i^2 + b_i^1 + b_{i-1}^2 - e_{i-1}^1|), & \text{if } ori_{i-1} = -1 \end{cases}$$

where the diagonal difference penalty is capped at $C = 100,000$ to allow for distant but significant genomic rearrangements.

Then determining a maximal path entails recursively computing, for every match f_i , the score and components of a maximum scoring path ending in it:

$$Score(f_i) = \max(0, score(f_i) + \max_{f_{i-1} \rightarrow f_i} (Score(f_{i-1}) - connect(f_{i-1}, f_i)))$$

Different chaining methods can be obtained by varying the constraints on constructing the graph and the M , D , O and R parameters. In practice, we implemented and explored the parameter space for two such methods.

- *Gchain* allows translocations and reversals and produces one chain of matches tiling the entire reference sequence (sequence 1). It is a global method.
- *Lchain* iteratively builds maximal scoring local chains, then removes the used matches from the original set,

in the end retaining only those chains that score above a threshold T . To maintain locality, it disallows edges between matches located farther than $C = 100,000$ bp apart in either sequence, as well as reversals and translocations. This approach is local.

The global method is more stringent in its selection of matches, but less computationally demanding, since the dynamic programming calculation is only invoked once.

We empirically determined a set of parameters that produce a sensitive and specific set of matches, as follows. The overall contribution of R and O to the score should be small, the two parameters acting primarily as noise filters. Indeed, larger translocation and reversal effects are reflected in the diagonal difference penalty. Therefore we can assume $R = O = 0$. This reduces the problem of finding a good set of parameters to finding a ratio M/D , or a pair $(M, 0)$, for each of the two methods *Lchain* and *Gchain*, and, in the case of *Lchain*, an additional score cutoff T .

We calibrated the parameters on the set of MUMs with minimum length $k_{Size} = 40$ between chromosomes 17 of NCBI build 28 and build 34. To explore the parameter space for M , D and T in a finite manner we measured the sensitivity and specificity of the subset of matches retained when we ask for the fraction f of matches in the top scoring chains, for progressively lower values of f , and for different M, D parameter choices. Table 6 shows the variation in the single and double coverage for a range of relevant (M, D) values when f varies from 0.9 to 0.3 in coarse 0.1 increments. We note that all methods perform similarly for a given cutoff with only small variations, therefore supporting the robust-

		<i>Lchain</i>								<i>Gchain</i>
f			0.9	0.8	0.7	0.6	0.5	0.4	0.3	-
D=1,	SC	μ	0.88253	0.84428	0.79364	0.73083	0.65266	0.57406	0.44933	0.86776
		σ	0.00091	0.00279	0.00497	0.00521	0.01032	0.00460	0.00763	0.04368
M=1,10, ..100	DC	μ	0.00566	0.00318	0.00162	0.00060	0.00034	0.00020	0.00008	0.00024
		σ	0.00017	0.00016	0.00012	0.00026 (bimodal)	2.8e-06	7.2e-06	2.3e-06	2.2e-05
D=0,	SC	μ	0.88100	0.84335	0.79198	0.72808	0.64942	0.57206	0.44693	0.87834
		σ	0	0	0	0	0	0	0	0
M=1,10, ..100	DC	μ	0.00544	0.00313	0.00159	0.00039	0.00034	0.00021	0.00008	0.00025
		σ	0	0	0	0	0	0	0	0

Table 6: Ranges of variation (μ, σ) for single and double coverage for various combinations of parameter values (M, D), when selecting a fraction f of MUMs from top-scoring chains. When $D = 0$, the score is completely determined by the match size, and all methods retrieve the same sets of matches, with scores scaled by M . Hence the variance is 0.

ness of the linear-score chaining model. This suggests the following calibration strategy for *Lchain*: first select a trade-off between sensitivity and specificity, from those sampled by the fraction f , and then select the set of parameters M, D and $T = T(f, M, D)$ that perform the best in that category. For *Gchain*, which does not depend on T , the parameters are chosen from among the sampled M, D combinations. For instance, for *Lchain*, $f = 0.8$ and $f = 0.6$ appear as good choices when sensitivity or specificity are sought, respectively, and $M = 20, D = 1$ and $M = 1, D = 0$ provide the best methods in their categories (data not shown). For *Gchain*, $M = 30, D = 1$ is the optimal combination.

The interplay between sensitivity and specificity may change when different types of matches and different sequence patterns with respect to repetitiveness, granularity and frequency of rearrangements are considered, as well as between cross- and same-species comparisons. Therefore, whenever the conditions change substantially from those above, a recalibration on a subset of matches that is representative for the entire comparison is needed.

Using the technique in [14], the dynamic programming calculation becomes $\mathcal{O}(n \log n)$. The run time for *Gchain* for a problem at the scale of the entire human genome is only a few minutes. For *Lchain*, where the procedure is applied iteratively, the run time depends on the score threshold selected, and varies from a few minutes to up to 1 CPU day.

5. RESULTS

We applied our methods to the set of MUMs obtained from the comparison of the NCBI build 34 and build 28 assemblies of the human genome, containing 2,865,069,167 and 2,852,918,546 ACGT-bases, respectively. Table 7 summarizes the results. Raising the length threshold from 20 bp to 40 bp has only small effects on the sensitivity and specificity, since the weak uniqueness of the MUMs is sufficient to eliminate most of the short spurious matches. By restricting the MUMs set to only those obtained by extending a unique k_{Seed} -mer seed the amount of duplicated sequence in the matches decreases by almost a half, with only a slight effect on sensitivity. This suggests that suffix-tree and hash indexed approaches, the former generating MUMs and the latter producing matches starting from unique k -mer seeds, are equally powerful in creating a good starting set of matches. Quantitatively, the choice of k_{Seed} within the 18 – 25 inter-

val has only marginal effects. Applying the repeat filtering rule considerably increases the specificity, by two orders of magnitude, however at a loss of sensitivity. Nevertheless, a large portion of the lost matches can be regained in a future alignment stage. Lastly, chaining algorithms can be very effective in reducing noise and repetitiveness while maintaining a high coverage of the genome with matches. In particular, applying *Gchain* in two stages, with each of the two sequences in turn used as reference, has the same potential to eliminate repeats as the Z -score based repeat filter, while the coverage achieved is substantially higher. Chaining algorithms are good filter alternatives for cross-species comparisons, where noise filters cannot be applied. In addition, local chaining methods like *Lchain* can be highly effective in identifying large segmental duplications in the genomes.

We addressed considerations of designing a reliable anchor set to guide the construction of a whole-genome alignment, and proposed a suite of efficient match filtration methods inspired by a novel per-base repeat annotation of the genome, the Z -score, and from the dynamic programming chaining techniques. Selecting the appropriate parameters for filtering is computationally inexpensive once the main data structures are computed. This allows deferring the calibration until after the calculation of MUMs and Z -scores, when a large parameter space can be explored with relatively little resources.

We tested the effectiveness of our techniques on the comparison of two different public assemblies of the human genome, and showed that a large portion (up to 89%) of the genomes can be aligned reliably in exact matches, with a very low amount of duplication. From this set, we estimate that a good approximation of the whole-genome alignment of same-species sequences can be constructed relatively easily. It was our intention to present the reader with a collection of methods which, when used alone or in combination, and observant to the type of comparison and alignment strategy, provide a good foundation from which a reliable one-to-one mapping and alignment of whole genomes can be accurately and efficiently built.

6. ACKNOWLEDGMENTS

We thank Mike Waterman for fruitful discussions on the statistical modeling of exact matches.

Parameters	# of matches	SC	DC
$k_{Size} = 20$	657809	0.92362	0.00623
$k_{Size} = 40$	535602	0.92280	0.00607
$k_{Size} = 40, k_{Seed} = 18$	465326	0.92094	0.00392
$k_{Size} = 40, k_{Seed} = 20$	471060	0.92116	0.00414
$k_{Size} = 40, k_{Seed} = 25$	479873	0.92149	0.00447
$k_{Size} = 40, k_{Rep} = 18$	4674558	0.69887	2.49e-05
$k_{Size} = 40, k_{Rep} = 20$	4502206	0.73194	4.28e-05
$k_{Size} = 40, k_{Rep} = 25$	4011643	0.77924	8.49e-05
$k_{Size} = 40, k_{Rep} = 30$	3293565	0.80765	1.33e-04
$k_{Size} = 40, k_{Rep} = 40$	2100126	0.83940	2.65e-04
$k_{Size} = 40, Gchain, M = 30, D = 1$	451719	0.88753	1.06e-04
$k_{Size} = 40, Lchain, M = 1, D = 0$	492325	0.83528	4.72e-04
$k_{Size} = 40, Lchain, M = 20, D = 1$	494130	0.89155	0.00190

Table 7: Designing anchors for the comparison of NCBI 28 and NCBI 34.

7. APPENDIX

7.1 The Poisson approximations using Chen-Stein method

By using the Chen-Stein method [11] and a de-clumping technique, Waterman [10] gives the limiting distribution of H_n with very small error bounds.

THEOREM 1. For $t = \log_{1/p}(n^2(1-p)) + c$ (some constant),

$$|P(H_n < t) - e^{-\lambda(n,t)}| \leq (b_1 + b_2)(1 - e^{-\lambda(n,t)})/\lambda(n,t),$$

where $\lambda(n,t) = [(2n - 2t + 1) + (n - t)^2(1-p)]p^t$, and

$$b_1 + b_2 = O\left(\frac{\log(n)}{n}\right) + O\left(\frac{\log(n)}{n^{2\gamma}}\right) + O\left(\frac{\log(n)}{n^{6\delta}}\right),$$

where $\gamma \in (0, \frac{1}{2} - \frac{1}{2t+1}]$, and $\delta \in (0, 1/6]$.

Similarly, the Poisson approximation for R_n under uniform Bernoulli model can be obtained.

THEOREM 2. For $t = \log_{\frac{1}{p}} n + c$ (some constant),

$$|P(R_n < t) - e^{-\lambda(n,t)}| \leq (b_1 + b_2)(1 - e^{-\lambda(n,t)})/\lambda(n,t),$$

where $\lambda(n,t) = (n - t + 1)p^t$, and $b_1 + b_2 = O(\frac{\log(n)}{n})$.

7.2 The Monte Carlo method

For non-uniform Bernoulli model, the limiting distribution of R_n is estimated by using Monte Carlo method. Let χ be the set of $\{A, C, G, T\}$. Let $w = w_1 w_2 \cdots w_t$ be a word of length t from set χ . Let $Y_i(w)$ be the random indicator

$$Y_i(w) = I(w \text{ starts at position } i \text{ in } \underline{A}),$$

$$\tilde{Y}_i(w) = Y_i(w)(1 - Y_{i-1}(w)) \cdots (1 - Y_{i-t+1}(w)),$$

and

$$\tilde{N}(w) = \sum \tilde{Y}_i(w), \quad \tilde{\mu}(w) = E\tilde{Y}_i(w).$$

Reinert *et al.* [15] prove that $\tilde{N}(w)$ can be approximated by a Poisson distribution with mean $\tilde{\lambda}_t(w) \equiv (n - t + 1)\tilde{\mu}(w)$ using the Chen-Stein method. Based on that result, we can estimate

$$P(R_n < t) \doteq \sum_{w \in \chi^t} P(\tilde{N}(w) = 0 | A_1 A_2 \cdots A_t = w) \\ \times P(A_1 A_2 \cdots A_t = w)$$

t	$\hat{\mu}(t)$	$\sqrt{\hat{V}ar(\hat{\mu}(t))}$
15	9.77e-01	9.46e-05
16	7.87e-01	3.14e-04
17	4.10e-01	3.60e-04
18	1.44e-01	1.78e-04
19	4.14e-02	6.29e-05
20	1.11e-02	1.76e-05
21	2.89e-03	4.86e-06
22	7.53e-04	1.34e-06

Table 8: Estimation of the variation of the Monte Carlo method.

It is computationally unfeasible to evaluate the above summation and we use the Monte Carlo method to empirically estimate $P(R_n < t)$. We generate n_1 independent Bernoulli random strings S_1, S_2, \dots, S_{n_1} , each with length t and get the estimation

$$\hat{P}(R_n < t) = \sum_{i=1}^{n_1} P(R_n < t | A_1 A_2 \cdots A_t = S_i)$$

The above procedure is repeated B times and thus we have $\hat{P}_1(R_n < t), \hat{P}_2(R_n < t), \dots, \hat{P}_B(R_n < t)$. Let $\mu(t) = E(\hat{P}(R_n < t))$ and $\sigma^2(t) = Var(\hat{P}(R_n < t))$ which can

be estimated by $\hat{\mu}(t) = \frac{1}{B} \sum_{j=1}^B \hat{P}_j(R_n < t)$ and $\hat{\sigma}^2(t) =$

$\frac{1}{B-1} \sum_{j=1}^B (\hat{P}_j(R_n < t) - \hat{\mu}(t))^2$ respectively. $\hat{\mu}(t)$ is used at

the end for estimating $P(R_n < t)$ and $Var(\hat{\mu}(t))$ can be estimated by $\hat{V}ar(\hat{\mu}(t)) = \frac{\hat{\sigma}^2(t)}{B}$. Table 8 gives the $\hat{\mu}(t)$ and

$\sqrt{\hat{V}ar(\hat{\mu}(t))}$ based on $n_1 = 2000$ and $B = 200$ samples.

8. REFERENCES

- [1] Venter, J.C., Adams, M.D., Myers, E.W., Li, P.W., Mural, R.J., Sutton, G.G. *et al.* (2001). The sequence of the human genome. *Science*, 291(5507): 1304-51.
- [2] Lander, E.S., Linton, L.M., Birren, B., Nusbaum, C., Zody, M.C., Baldwin, J. *et al.* (2001). Initial sequencing and analysis of the human genome. *Nature*, 409(6822): 860-921.

- [3] Mural,R.J., Adams,M.D., Myers,E.W., Smith,H.O., Miklos,G.L., Wides,R. *et al.* A comparison of whole-genome shotgun-derived mouse chromosome 16 and the human genome. *Science*, 296(5573): 1661-71.
- [4] Waterston,R.H., Lindblad-Toh,K., Birney,E., Rogers,J., Abril,J.F., Agarwal,P. *et al.* (2002). Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420(6915): 520-62.
- [5] Delcher,A.L., Kasif,S., Fleischmann,R.D., Peterson,H., White,O. and Salzberg,S.L. (1999). Alignment of whole genomes. *Nucleic Acids Research*, 27(11): 2369-2376.
- [6] Schwartz,S., Kent,W.J., Smit,A., Zhang,Z., Baertsch,R., Hardison,R.C., Haussler,D. and Miller,W. (2003). Human-mouse alignments with BLASTZ. *Genome Res.*, 13(1): 103-7.
- [7] Bray,N., Dubchak,I. and Pachter,L. (2003). AVID: A global alignment program. *Genome Res.*, 13(1): 97-102.
- [8] Brudno,M., Do,C.B., Cooper,G.M., Kim,M.F., Davydov,E., Green,E.D., Sidow,A., Batzoglou,S.; NISC Comparative Sequencing Program. (2003). LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.*, 13(4): 721-31.
- [9] Smit,A.F. (1999). Interspersed repeats and other mementos of transposable elements in mammalian genomes. *Curr. Opin. Genet. Dev.*, 9: 657-663.
- [10] Waterman,M.S. (1995). Introduction to computational biology. *Chapman & Hall*.
- [11] Chen,L.H.Y. (1975). Poisson approximation for dependent trials. *Ann. Prob.*, 3:534-545.
- [12] McCreight,E.M. (1976). A space-economical suffix tree construction algorithm. *Journal of the ACM*, 23(2): 262-272.
- [13] Ukkonen,E. (1992). Constructing suffix-trees on-line in linear time. *Proceedings of the IFIP 12th World Computer Congress*: 484-492.
- [14] Eppstein,D., Galil,Z., Giancarlo,R. and Italiano, G. (1992). Sparse dynamic programming I: Linear cost functions. *Journal of the ACM*, 39(3): 519-545.
- [15] Reinert,G., Schbath,S. and Waterman,M.S. (2000). Probabilistic and statistical properties of words: an overview. *Journal of Computational Biology*, 7(1-2): 1-46.