COMBAT: Search Rapidly for Highly Similar Protein-Coding Sequences Using Bipartite Graph Matching

Bing Sun, Jacob T. Schwartz, Ofer H. Gill, and Bud Mishra

Courant Institute of Mathematical Sciences, New York University
251 Mercer Street, New York, NY 10012, USA
bingsun@cs.nyu.edu, jack@brainlink.com, gill@cs.nyu.edu, mishra@nyu.edu

Abstract. Comparing vertebrate genomes requires efficient cross-species sequence alignment programs. We describe COMBAT, a new mer-based method which can search rapidly for highly similar translated genomic sequences, with the stable-marriage algorithm with incomplete lists (SMI) as a filter scheme. We apply the COMBAT program to the comparative analysis of the human with the most recent bovine genome assemblies, and 84%∼95% of the homologous blocks identified by this program are confirmed by BLASTZ.

1 Introduction

In the past decade many genome projects have produced complete genomes for increasingly many organisms. Since 1999 many tools have proven effective in aligning large genomic sequences of two closely related organisms. These include MUMmer [4], GLASS [1], AVID [2], DIALIGN [8], LAGAN [3], BLASTZ [9], BLAT [7], and etc. Common characteristics in many of these programs are: i) they assume the conserved regions of the sequences being aligned appear in the same order and orientation, as is particularly likely for closely related organisms; ii) they build tables of scores for matches and mismatches between amino acids or nucleotides incorporating penalties for insertions or deletions, and from these constructs obtain mathematically 'optimal' alignments; iii) many local alignment programs search for exact or spaced exact matches, and then extend the local similarities in both directions in passes directed by specified scoring functions.

However, certain shortcomings limit the use of many of these programs. First, genomic order and orientation need not be conserved between species of interest. Secondly, the scoring matrix (eg. a PAM or a BLOSUM matrix) most appropriate for aligning a set of sequences should be determined by the level of relatedness of sequences. Hence the percentage of similarity between two genomes has to be preestimated to choose a proper scoring matrix. Also, the fact that the rate of evolution varies across the genome makes it impractical to pick a universal scoring matrix or a set of gap costs [5]. Finally, by using the "match and extend" strategy many local alignment algorithms pay a steep cost in extending short matches in both directions.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part II, LNCS 3992, pp. 654–661, 2006.

This paper describes a novel local alignment algorithm, called COMBAT (Clean Ordered Mer-Based Alignment Tool), which addresses the above challenges by implementing the following two critical stages: i) generating an index of all overlapping K-mers in translated genomic sequences, where the index represents the blocks to which a K-mer belongs and is used to search efficiently for homologous blocks. ii) using the *SMI* algorithm to find the optimal one-to-one mapping from a list of multiple local mappings and so form a global matching map. COMBAT makes no assumption of gene order and orientation, does not utilize any sophisticated scoring matrix, and does not have the expensive "extend" stage of many local alignment programs. The COMBAT algorithm is fully described in the next section.

2 Method for Pairwise Genomic Comparison

The goal of COMBAT is to identify protein-encoding regions in genomic sequences using a genome comparison approach. Let us suppose the two genomes being compared are called genome A and genome B. We define some of the terms used below, and present the parameters involved in Table 1.

J-interval: a continuous genomic sequence of length J. Adjacent J-intervals are spaced $J/2$ bases apart. The J-interval index is simply called J-index.

Partner Interval Pair (PIP): an instance of PIP (a, b) consists of a J-interval a in genome A and a J-interval b in genome B if there are more than T K-mers shared by both a and b.

Table 1. The involved parameters in COMBAT program

J	The length of a J-interval
K	The K-mer size
T	The minimum number of common K-mers required in any PIP
S	The actual number of common K-mers in a PIP
E, F	The chaining filtering criterion requires that there must be at least F PIPs, each no further than E intervals from each other.

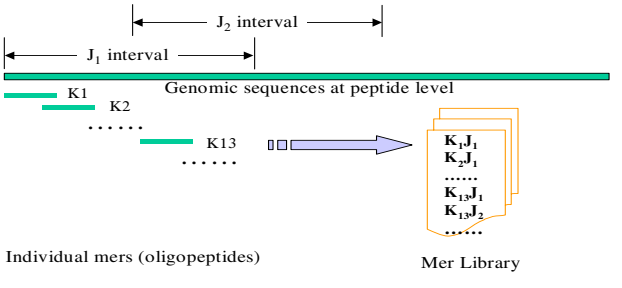

Fig. 1. How mer library for one genome is built. K_i denotes the i^{th} K mer. J_j denotes the index of the j^{th} J-interval. Most mers (like K_{13}) occur in the region covered by two adjacent J-intervals, so they may appear twice in the mer library.

The COMBAT algorithm comprises the following steps:

Step 1: Build Clean Ordered Mer Libraries
First, we translate genomic sequences of genome A and B in all three frames over both forward and reverse orientations. After choosing a mer-size K we generate overlapping K-mers starting at every base position, ignoring mers in repeats annotated by RepeatMasker. We cover the considered genome with J-intervals. The "representation of position" that we attach to each K-mer is the index of each J-interval to which it belongs, shown in Figure 1. We keep only one copy of duplicate K-mers in each J-interval from the mer library. This makes all the K-mers unique in every interval. Next we sort the mer library by the mer sequences. Such mer libraries for genome A and genome B are built separately.

Step 2: Search For Common Mers
Next we scan the clean ordered mer libraries prepared in the manner just described to compute offsets between the pairs of matching mers found. When the offset d_{ij} = A_i - B_j exists, it is the J-index difference between the i-th mer occurring in genome A and the matching mer in genome B. It is easy to recover the J-index of genome B using d_{ij}. We then sort this list of mers/offset pairs by their offsets and their J-indexes on genome A. For each J-index of genome A in this list we count the number of K-mers that have the same offsets. We keep only those intervals as PIPs whose number of common K-mers is beyond the threshold T for the next step.

Step 3: Find One-to-One Correspondence
As a natural result of genome duplications one region of one genome might match several regions of another genome. Usually the single best, orthologous match for each conserved region gains most biologists's attention. BLASTZ uses the axtBest program to produce the best alignments [9]. In this paper we first present the application of the stable marriage (SM) problem in large-scale genome comparison as an alignment filter[1]. The well-know SM problem was introduced by Gale and Shapley [6]. The problem can be stated as below: given two finite equal-sized sets of players, called *men* and *women* ($m_i \in men, w_j \in women$), where each m_i/w_j ranks w_j/m_i in strict order forming his/her preference list, find a one-to-one stable matching M between the two sexes. M is "stable" if there is no two couples (m, w) and (m', w') in M such that m prefers w' to w and w' prefers m to m'. The so-called "proposal algorithm" solves this problem.

Let $P = \{(a, b)\}$ denote the set of all PIPs (a, b) found in step 2, $X = \{a \mid \exists b : (a, b) \in P\}$, and $Y = \{b \mid \exists a : (a, b) \in P\}$. P can be viewed as a bipartite graph which is a multiple mapping between two sets of J-indexes. We wish to find M, a one-to-one stable matching, from P. Since normally some a in X matches to a true subset of Y, this SM problem becomes the relaxed version — the Stable Marriage Problem with Incomplete Lists (SMI). We form a preference list for each J-interval in P as follows:

[1] One might borrow the idea of maximum weight matching (MWM) for this task. The MWM solver maximizes the cumulative similarity, thus might not give single best matches for individual regions.

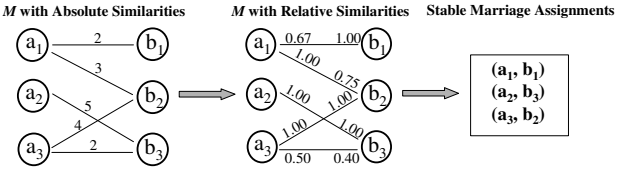

Fig. 2. An example of the stable marriage problem procedure. First compute relative similarities from absolute similarities in the bipartite graph, then follow the SMI algorithm to find the stable marriage assignments. $a_1 \sim a_4/b_1 \sim b_4$ denote the J-indexes on genome A/B. In multiple mapping M with absolute similarities the numbers on the edges show the number of K-mers shared by the partner intervals. The numbers associated with an edge in the middle panel are the relative similarities for a pair of partner intervals.

1. A measure of absolute interval similarity S is calculated, and $S = \{S_{(a,b)} \mid (a,b) \in P\}$, where $S_{(a,b)}$ denotes the number of K-mers shared by a PIP (a,b).
2. Relative similarities are computed subsequently as fractions of the absolute similarities of the best match partner for any J-interval in P. Then each J-interval j ranks its match partners in strict order of their relative similarities to j, forming j's preference list.

In the example in Figure 2, b_2 is the best match for a_1, so we set $R_{(a_1,b_2)} = 1.00$. The relative similarity for the other match partner of a_1 is computed as a fraction of $S_{(a_1,b_2)}$. Thus, $R_{(a_1,b_1)} = \frac{S_{(a_1,b_1)}}{S_{(a_1,b_2)}} = \frac{2}{3} \approx 0.67$. Relative similarities are asymmetric. Under the marriage interpretation, this means that any two match partners like each other to the different extent. We modify the proposal algorithm and explain the SMI algorithm used by COMBAT as follows:

```
1. X={a},Y={b},M={}. Every a and b has an ordered preference list.
2. WHILE X is not empty, LOOP
3.   choose an interval a from X
4.   b=the first interval on a's list(If have ties, randomly choose one)
5.   IF a is not on b's preference list, THEN
6.       delete b from a's list;
7.       IF a's preference list is empty, THEN
8.          delete a from X; goto line 2
9.       ELSE goto line 4
10.  ELSE
11.      IF (x, b) is in M for some x in X, THEN
12.         remove (x, b) from M; add x to X;
13.      add (a, b) to M
14.      FOR each successor x (x ranks after a) in b's list, LOOP
15.         delete x from b's list, and b from x's list;
16.      END LOOP
17. END LOOP
18. RETURN M
```

This SMI algorithm's complexity is $O(n^2)$ in time, and is linear in space (n is the number of PIPs). The result returned by this algorithm is a list of incomplete one-to-one mapping, which means J-intervals in genome A map to at most one partner in genome B, and vice versa. Lastly, in order to remove randomly matching PIPs we perform a chaining procedure which requires that there must be at least F partner intervals, each no further than E intervals from each other. This step is not necessary if we choose strict values of J and K.

3 Results for the Human/Cow Genome Comparison

We have applied COMBAT to Human Assembly (hg17, May 2004) and Cow Assembly (bosTau1, Sep. 2004, BCM HGSC Btau_1.0), both from the UCSC Genome Bioinformatics Site. As an example illustrating our results, we take chromosome I from hg17 and the first 33,000 cow scaffolds, and align them by COMBAT. These two sequences are approximately 250MB in size. Let us call the first sequence *chr1*, and the second sequence *cow1*. The resulting alignment maps using different configurations are shown for positive strands in Figure 3. Figures 3-(1),(2),(4), and (5) are the results produced by COMBAT, with each plus sign representing the index coordinates of a pair of matching intervals found by COMBAT. Figures 3-(3) and (6) are the matches produced by BLASTZ, filtered by the axtBest program [9] (downloaded from the UCSC Genome Bioinformatics Site and transformed to fit our J-intervals context), with each dot representing the index coordinates of the starting positions of two matched regions. The BLASTZ result is transformed twice according to two values of J used.

The chaining criterion used by COMBAT turns out to be relatively insensitive to the value of E used (see Figure 3-(1) and 3-(2)). To evaluate COMBAT, we have tested the appearance of every matching pair of intervals found by COMBAT in the BLASTZ result (transformed by the same J used by COMBAT)[2]. In Figure 3-(1), 95% of the 625 partner interval pairs found by COMBAT are true positives. In the other direction, out of 8,389 matching regions in the BLASTZ result, 7% are confirmed by COMBAT. In Figure 3-(4), there are 84% true positives out of 1235 PIPs, and they cover 11% of the BLASTZ result. In Figure 3-(5), there are 85% true positives out of 971 PIPs, and they cover 9% of the BLASTZ result. This high specificity indicates a promising wide use of COMBAT. The low coverage is not surprising because only highly similar protein-coding regions are expected to be found.

The computational core of the COMBAT algorithm was implemented as a C++ program and all experiments were performed on NYU Bioinformatics Group's cluster of Pentium IV machines with 3 GB memory running RedHat

[2] Consider a pair of matching J-interval (a, b) in COMBAT result as a true positive case if there exists a pair of matching regions (x, y) (a and x in genome A, b and y in genome B) in BLASTZ result and one of the following conditions is satisfied: 1) a is contained in x and b is contained in y; 2) x is contained in a and y is contained in b; 3) the starting positions of a/b is within J bases of those of x/y, respectively; 4) the ending positions of a/b is within J bases of those of x/y, respectively.

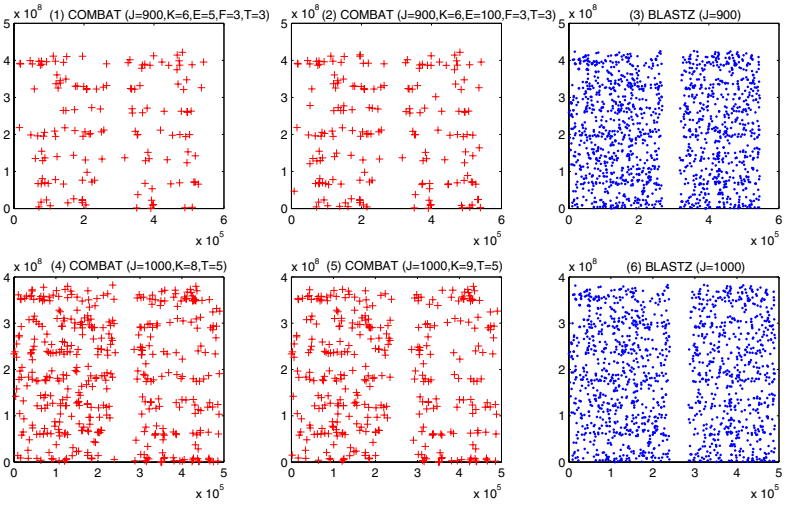

Fig. 3. Alignment maps on positive strands between *chr1* and *cow1*, with the X-axis showing the J-indexes along the *chr1* sequence, and the Y-axis showing those along the *cow1* sequence. (1),(2),(4), and (5) are the results produced by COMBAT; (3) and (6) are the transformed results produced by the BLASTZ. (4) and (5) are done without using the chaining procedure.

Linux 7.3. To compare 0.25 Gb of human sequence against 0.24 Gb of cow sequence ($\sim$ 1/10 of total genomes) and produce the one-to-one mapping list of highly similar regions, it took 23 CPU hours under the configuration shown in Figure 3-(1), and took 2 CPU hours under the configuration shown in Figure 3-(4). For the sake of performance comparison, we point to the published report of BLASTZ taking 481 days of CPU time to align 2.8 Gb of human sequence against 2.5 Gb of mouse sequence on a cluster of 1024 833-Mhz Pentium III [9].

4 Error Estimation

Consider two random J-intervals a in genome A and b in genome B (each of length J over an alphabet of 20 amino acids and 1 stop codon) . For the sake of simplicity, we will consider these intervals in one orientation only. Let P_k denote the probability that there is a common K-mer at any position. Assuming that letters occur at any given position with equal probability and independency, we get $P_k = 1/(21)^K$. Let the positive-valued random variable w denote the number of common K-mers in a and b. We can show that w follows a Poisson distribution with parameter $\lambda_w = J^2 P_k$. The expectation of a new random variable $\binom{w}{i}$ can be estimated by considering all possible $\binom{J}{i}$ subsets of K-mers from a and counting the probability of each such subset having exact matches with i K-mers in b.

$$E\left[\binom{w}{i}\right] = \binom{J}{i}(JP_k)((J-1)P_k)\cdots((J-i+1)P_k) \approx \frac{J^{(i)}}{i!}\left(\frac{J}{21^K}\right)^i \approx \frac{\left(J^2/21^K\right)^i}{i!} \tag{1}$$

Using Brun's sieve, the probability that two randomly selected J-intervals from genome A and genome B have exactly m K-mers in common is:

$$Pr[w = m] = e^{-(J^2/21^K)} \frac{\left(J^2/21^K\right)^m}{m!} \tag{2}$$

Using parameters of this Poisson distribution, we can choose a lower threshold such that two random J-intervals are unlikely (with probability $> 1 - \epsilon$) to have more than θ_w K-mers in common. Using Chebychev's inequality, we see that a conservative choice would be:

$$\theta_w = \mu_w + \frac{\sigma_w}{\sqrt{\epsilon}}, \qquad \text{where} \quad \mu_w = \frac{J^2}{21^K},\ \sigma_w = \frac{J}{21^{K/2}} \tag{3}$$

As argued earlier, by using the one-tailed Chebychev bound, we have:

$$Pr(w > \theta_w) = Pr(w - \mu_w > \frac{\sigma_w}{\sqrt{\epsilon}}) < \epsilon \tag{4}$$

By choosing a very small value of ϵ (for example, $\epsilon \approx O(1/G)$, where G is the genome size), we could make the probability of false positive adequately small.

Table 2. Exemplary choices of parameters given G and s when $\epsilon = 1/G$. The θ here has the same meaning of the T parameter in Table 1. Since ϵ is extremely small here, the suggested range of θ is very conservative.

$s = 0.8$		$s = 0.6$	
$G = 10^9$	$G = 10^6$	$G = 10^9$	$G = 10^6$
$J = 1000, K = 8$	$J = 1000, K = 6$	$J = 1000, K = 9$	$J = 1000, K = 6$
$162 < \theta < 200$	$108 < \theta < 200$	$35 < \theta < 150$	$108 < \theta < 150$

In the other direction, let s be a desired similarity value, in the sense that COMBAT must almost always find pairs a and b, whenever they have a similarity value of s or higher. The number of observed K-mers shared by a and b can be viewed as a random variable v: $B(|a \cap b|, s)$ which has a Binomial distribution with mean $\mu = |a \cap b|s$ and variance $\sigma^2 = |a \cap b|s(1 - s)$. Using the Chernoff bound, we can choose an upper threshold of $|a \cap b|s/2 > Js/4$ to guarantee a probability of success larger than $(1 - \epsilon)$, if J is sufficiently large, i.e., $Js > 16 \ln(1/\epsilon)$. Assuming $\epsilon = 1/G$, and $16 \ln(G)/s < J \ll G$, we will need to satisfy the following inequality:

$$\frac{J^2}{21^K} + J\sqrt{\frac{G}{21^K}} < \theta < Js/4 \quad \text{or} \quad \frac{J}{21^K} + \sqrt{\frac{G}{21^K}} < \theta' < s/4 \tag{5}$$

Since G and s are determined by the genomes, we need only to choose K and J. Table 2 shows some exemplary choices of parameters. Note that since our estimations are rather conservative, we found that, in practice, COMBAT performs quite well even for suboptimal choices of parameters.

5 Summary and Acknowledgements

To get adequate speed when performing comparison at the scale of whole genomes many high-speed alignment programs have a fast search stage that

uses a heuristic to identify regions likely to be homologous. Providing a way of indexing sequences is a key to an efficient search stage. COMBAT indexes both genomic sequences. By using the index of intervals instead of genomic positions we have been able to decrease by J-fold the size of the index for a vertebrate genome, and make it practical to run on a single CPU machine. We show that COMBAT is capable of rapidly finding matching regions across vertebrate species working in translated mode. Then a detailed alignment can be easily retrieved by using the standard alignment algorithms [Smith-Waterman,1970; Needleman-Wunsch,1981]. Therefore, the complex large-scale genome comparison problem is simplified by COMBAT. We also solve the problem of finding a one-to-one mapping in a multiple mapping list by using the SMI algorithm.

Since COMBAT looks for exact K-mers matches, it cannot find regions of relatively low similarity. However, the basic COMBAT scheme can be varied to increase its sensitivity. For example, we can generate K-mers consisting of n exactly matching submers $K_1 \sim K_n$ with g number of bases between them ($g \in [0, \alpha]$, where α is a threshold). This scheme makes it possible to find inexact K-mer matches with gaps or mismatches, and will be experimented in the future.

This project was sponsored by the Department of the Army Award Number W81XWH-04-1-0307.

References

1. Pachter, L., Mesirov, J.P., Berger, B., Batzoglou, S. and Lander., E.S. Human and mouse gene structure: Comparative analysis and application to exon prediction. *Genome Res.*, pages 950–958, 2000.
2. Bray, N., Dubchak, I., and Pachter, L. Avid: A global alignment program. *Genome Res.*, pages 97–102, 2003.
3. Brudno, M., Do, C.B., Cooper, G.M., Kim, M.F., and Davydov, E.D. Lagan and multi-lagan: Efficient tools for large-scale multiple alignment of genomic dna. *Genome Res.*, pages 721–731, 2003.
4. Delcher, A.L., Kasif, S., Fleischmann, R.D., Peterson, J., White, O., and Salzberg, S.L. Alignment of whole genomes. *Nucleic Acids Res.*, pages 2369–2376, 1999.
5. Frazer, K.A., Elnitski, L., Church, D.M., Dubchak, I., and Hardison, R.C. Cross-species sequence comparisons: A review of methods and available resources. *Genome Res.*, pages 1–12, 2003.
6. Gale, D., and Shapley, L.S. College admissions and the stability of marriage. *Am. Math. Monthly*, pages 9–15, 1962.
7. Kent, W.J. Blat - the blast-like alignment tool. *Genome Res.*, (4):656–664, 2002.
8. Morgenstern, B., Rinner, O., AbdeddaÏm, S., Haase, D., Mayer, K.F.X., Dress, A.W.M., and Mewes, H.W. Exon discovery by genomic sequence alignment. *Bioinformatics*, (6):777–787, 2002.
9. Schwartz, S., Kent, W.J., Smit, A., Zhang, Z., et al. Human-mouse alignments with blastz. *Genome Res.*, page 103–107, 2003.