

Multiple Genome Alignment by Clustering Pairwise Matches

Jeong-Hyeon Choi^{1,3*}, Kwangmin Choi¹, Hwan-Gue Cho³, and Sun Kim^{1,2*}

¹ School of Informatics, Indiana University, IN 47408, USA,
{jeochoi,kwchoi,sunkim}@bio.informatics.indiana.edu

² Center for Genomics and Bioinformatics, Indiana University, IN 47405, USA

³ Department of Computer Science and Engineering, Pusan National University,
Korea, {jhchoi,hgcho}@pusan.ac.kr

Abstract. We have developed a multiple genome alignment algorithm by using a sequence clustering algorithm to combine local pairwise genome sequence matches produced by pairwise genome alignments, e.g. BLASTZ. Sequence clustering algorithms often generate clusters of sequences such that there exists a common shared region among all sequences in each cluster. To use a sequence clustering algorithm for genome alignment, it is necessary to handle numerous local alignments between a pair of genomes. We propose a multiple genome alignment method that converts the multiple genome alignment problem to the sequence clustering problem. This method does not need to make a guide tree to determine the order of multiple alignment, and it accurately detects multiple homologous regions. As a result, our multiple genome alignment algorithm performs competitively over existing algorithms. This is shown using an experiment which compares the performance of TBA, MultiPipMaker (MPM) and our algorithm in aligning 12 groups of 56 microbial genomes and by evaluating the number of common COGs detected.

1 Introduction

Recent advances in both sequencing technology and algorithm development for genome sequence software have made it possible to determine the sequence of a whole genome. As a consequence, the number of completely sequenced genomes is increasing rapidly. However, algorithm development for genome annotation has been relatively slow, and the annotation of completely sequenced genomes inevitably depends on human expert knowledge. The most effective method to understand genome content is to compare multiple genomes, especially when they are close enough to share common subsequences. One important computational method is to align whole genome. Aligning whole genomes is useful in several ways. For example, sequencing a genome can be much easier and more accurate by aligning its *contigs* to completely sequenced genomes that are close to the one being sequenced. Another example of use of whole genome alignment is to identify conserved regions among multiple genomes which include not

* Corresponding author

only common genes but regulatory regions and non-coding RNA sequences. A recent paper demonstrated that intergenic functional regions in multiple Yeast strains can be detected by comparing the whole genomic sequences [1]. Work on comparing human and mouse sequences has also demonstrated the possibility of predicting functions and structures of human genes by genome alignment methods, e.g., [2,3].

The dynamic programming algorithms, such as Needleman-Wunsch [4] and Smith-Waterman [5], can be extended to align optimal alignment of multiple sequences, e.g., MSA [6]. However, computing optimal alignments for long genomic sequences is not practical in terms of computation time and memory requirement; assuming N sequences of equal length L , time complexity is $O(2^N L^N)$ and the space complexity is $O(L^N)$. Thus heuristic approaches, such as progressive and iterative alignments, are widely used.

The progressive alignment (1) aligns all pairs of sequences by a fast anchor-based alignment approach or a slow full dynamic programming method like Needleman-Wunsch algorithm, (2) produces a guide tree via distance matrix using the pairwise alignment scores, and (3) aligns the sequences sequentially guided by the phylogenetic relationships indicated by the tree. The major problem with progressive alignment programs such as CLUSTALW [7] and PILEUP in GCG Wisconsin Package is that the order of progressive alignment is determined and fixed by the initial pairwise alignments. To overcome this problem, some iterative alignment methods make initial alignments for groups of sequences and then revise the alignments to compute a more accurate result by various approaches [8,9,10,11].

Unlike the progressive and iterative alignment strategy, chaining alignment strategy aligns multiple sequences without making guide tree. This method is based on chaining all pairwise local alignments or multiple local alignments. MGA [12] first finds multiMEMs, maximal exact matches occurring in multiple genomes, as anchors and then calculates the optimal chain for multiMEMs by using graph or range tree. DIALIGN [13] aligns pairs of sequences to locate aligned regions that do not include gaps, that is, continuous diagonals in a dot matrix plot. A consistent collection of weighted diagonals is then computed, and diagonals with maximal sum of weights are generated as alignments.

As more whole genome sequences become available, there has been growing need for computational methods for aligning multiple whole genomes. MLAGAN [14] aligns genomic sequences in progressive alignment phases with LAGAN and optimal iterative improvement phases. MAVID [15] uses a progressive alignment approach to incorporate the following ideas: maximum likelihood inference of ancestral sequences, automatic guide tree construction, protein-based anchoring of ab-initio gene predictions, and constraints derived from a global homology map of the sequences. TBA [16] builds a threaded blockset under the assumption that all matching segments occur in the same order and orientation in given sequences; inversions and duplications are not addressed. TBA is shown to generate very high quality multiple genome alignments that verified with their rigorous column-by-column comparing evaluation method. MultiPipMaker [17]

aligns a reference sequence individually with each secondary sequence, prepares a crude multiple alignment from the pairwise alignments, removes overlaps in the local pairwise alignments, and refines the crude multiple alignment to generate a true multiple alignment using rigorously defined multiple alignment scores.

1.1 Motivation

As we discussed, a heuristic alignment approach is the most widely used technique for multiple sequence alignment. However, the following reasons make it difficult to use heuristic alignment strategy for the multiple genome alignment problem.

1. It is hard to utilize the guide tree since there are many local regions where their phylogenetic relationships are different from those of the “entire” genome. Even duplications of a gene within a genome have their own phylogenetic relationship.
2. The greedy progressive alignment works due to the guide tree. Given that the utilization of a guide tree is not trivial for genome alignment as discussed above, the greedy progressive alignment strategy should be avoided.
3. The iterative alignment requires generation of profiles – alignment of multiple sequences – while combining pairwise matches. Generating and aligning profiles for long genome sequences is not practical.

An alternative to the progressive alignment method is to compute subsequences common to all genomes being aligned and chain them together to generate multiple sequence alignment. For example, MGA [12] computes multiMEMs common in all genomes being aligned. MEMs in multiple genomes are naturally short in length and thus it is necessary to chain them to generate multiple sequence alignments. Although MGA was successful in generating “global” alignments of closely related genomes, it is not effective to compute alignments for relatively distant genomes. One way to circumvent the difficulty is to use directly pairwise genome alignments which are already extended matches between a genome pair. Thus some algorithms such as TBA [16] and MultiPipMaker [17] are designed to combine pairwise matches for the multiple genome alignment problem.

We have developed a multiple genome alignment algorithm by using a sequence clustering algorithm [18] to combine pairwise matches. Sequence clustering algorithms generate clusters of sequences that are candidates for sequence families. These clusters are often generated to have common shared regions among all sequences in each cluster, and these shared regions are used to predict sequence domains. We argue that sequence clustering algorithms can be used to generate multiple genome alignments by combining pairwise genome matches since sequence clustering algorithms often require all pairwise alignments of the input sequence set and combine pairwise matches to generate sequence clusters. The genome alignment problem, however, is different from the sequence clustering problem in terms of the number of input sequences and the number of

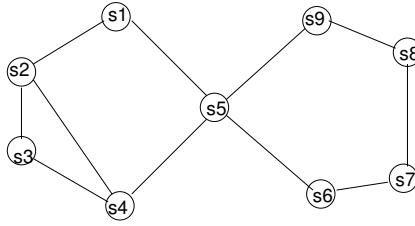


Fig. 1. Biconnected components and articulation point. There are two biconnected components, $\{s_1, s_2, s_3, s_4, s_5\}$ and $\{s_5, s_6, s_7, s_8, s_9\}$. The vertex s_5 is an articulation point since removing the vertex results in splitting the graph.

alignments between sequences. The number of genomes to be aligned is much smaller than that of sequences to be clustered. In addition, there are numerous local alignments between a pair of genomes while one or only a few aligned regions exist between a pair of (protein) sequences. The basic idea of this paper is to transform the genome alignment problem to the sequence clustering problem, which will be tackled by our sequence clustering algorithm BAG [18]. Thus we explain briefly how BAG works in the next section and we describe how to transform the genome alignment problem to the sequence clustering problem by generating subsequences in Section 2.2.

2 A Sequence Clustering Algorithm and the Generation of Input Data from Pairwise Matches

2.1 BAG sequence clustering algorithm

Let us first review some definition of graph. A *connected component* of a graph G is a subgraph where any two vertices in the subgraph are reachable from each other. An *articulation point* of G is a vertex whose removal disconnects G . For example, in Fig. 1, the removal of a vertex s_5 disconnects G . A *biconnected graph* is a graph where there are at least two disjoint edge paths for any pair of vertices. A *biconnected component* of G is a maximal biconnected subgraph. In Fig. 1, a subgraph G_1 induced by vertices $\{s_2, s_3, s_4\}$ is a biconnected graph but it is not a biconnected component since another subgraph G_2 induced by vertices $\{s_1, s_2, s_3, s_4, s_5\}$ is biconnected and G_1 is a subgraph of G_2 . There are two biconnected components, $\{s_1, s_2, s_3, s_4, s_5\}$ and $\{s_5, s_6, s_7, s_8, s_9\}$ of G in Fig. 1.

For a given set of (protein) sequences $\{s_1, s_2, \dots, s_n\}$, a weighted graph G can be built from all pairwise comparison results by using FASTA and BLAST. A node is created for each sequence s_i and an edge between two sequences, s_i and s_j , is created when the pairwise alignment score of s_i and s_j is more significant than a preset threshold. The alignment score is associated with the edge as weight so that clusters can be refined while increasing cutoff for edges. Then

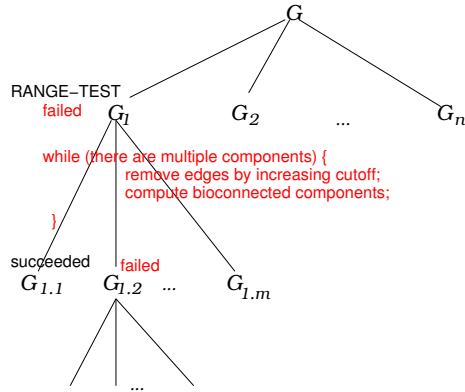


Fig. 2. Illustration of the splitting step. For any biconnected component that failed for RANGE-TEST, a cutoff score is increased until it can be split into multiple biconnected components.

our sequence clustering algorithm BAG will be used for multiple genome alignment later in this paper. BAG explicitly uses two graph properties: biconnected components and articulation points (see Fig. 1). We argue that a biconnected component (*BCC* in short) is a candidate for a family of sequences because biconnected graph requires at least two disjoint edge paths between *every pair of nodes* in the graph and requires a much stronger condition than single linkage. We also argue that an articulation point is a candidate for a multidomain protein since it is the only vertex that connects two or more biconnected components, i.e., multiple families. According to the graph properties mentioned previously, we named our algorithm as BAG which stands for Biconnected components and Articulation point based Grouping of sequences.

For a given set of sequences $\{s_1, s_2, \dots, s_n\}$, BAG recursively computes the biconnected components with an increased cutoff score at each iteration. Our algorithm BAG works as follows:

1. Build a weighted graph G from all pairwise comparison results where a node is created for each sequence s_i and an edge between two sequences, s_i and s_j , is created when the pairwise alignment score of s_i and s_j is more significant than a preset threshold. The alignment score is associated with the edge as weight so that clusters can be refined while increasing cutoff for edges.
2. Select a cutoff score T_c , remove edges with weight smaller than T_c , and generate biconnected components, G_1, G_2, \dots, G_n , with a set of articulation points $\{a_1, a_2, \dots, a_m\}$. See [18] for the detail how to select this cutoff score.
3. Perform RANGE-TEST for each biconnected component G_i to determine whether all sequences in G_i have common shared regions. If G_i fails RANGE-TEST, split G_i iteratively and recursively into multiple clusters $\{G_{i_1}, \dots, G_{i_l}\}$ until every biconnected component G_{i_j} passes the RANGE-TEST as increas-

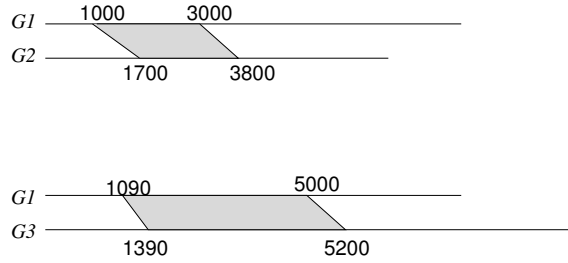


Fig. 3. Difficulty in generating subsequences from pairwise alignments. In G_1 , there are two alignments at similar positions, one with G_2 and another with G_3 . One straightforward way to name subsequences is to assign a name with starting positions of alignment in each genome. For G_1 , we would generate two subsequences $G_1:1000$ and $G_1:1090$. However, these two subsequences correspond to the same homologous region and it is difficult to match them. In addition, this result in generating numerous subsequences when there are multiple genomes; in the worst case, the number of subsequences of a genome can be the length of the genome.

ing the cutoff score by δ .⁴ Note that increasing the cutoff score will remove some edges and the resulting graph can be split into multiple biconnected components at a certain cutoff. This step is illustrated in Fig. 2.

4. After iterative splitting is done (step 2 and 3), clusters are considered for merging. Each articulation point is tested for having common shared regions with its neighbors in different clusters; we call this the AP-TEST. Then a hypergraph is built as follows. Clusters from the previous step become vertices and articulation points that fail AP-TEST become edges in the hypergraph. A set of biconnected components of the hypergraph is iteratively merged into one until there is no candidate component for further merging as relaxing the cutoff score T_c .

2.2 Generation of the input data to BAG

BAG is originally designed to handle protein sequences which are typically 1,000 amino acid characters or less in length. So it cannot directly handle all pairwise alignments of genomes since there are numerous edges (pairwise matches) between a pair of nodes (genomes). This issue is handled by generating subsequences with their own identifiers. However, the generation of subsequences is not straightforward as illustrated in Fig. 3 where two different identifiers are gener-

⁴ While increasing the cutoff score, we have several competing cutoff values that could generate different clustering result. In the past, we selected the first cutoff that splits G_i into multiple clusters. We recently developed a better method, called *Cluster Utility* to select the best one among different cutoff choices. See [19] for more detail.

Table 1. The bag input for local alignments, (497087, 499555, 698686, 701075) for NC_000908 vs. NC_000912 pair, (497227, 498276, 2682403, 2683463) for NC_000908 vs. NC_000913 pair, and (698823, 699860, 2682400, 2683452) for NC_000912 vs. NC_000913 pair to detect common COG0112.

seq1	seq2	score	pos1	pos2
NC_000908:497000	NC_000912:698000	60562	87,2555	686,3075
NC_000908:498000	NC_000912:699000	60562	0,1555	0,2075
NC_000908:499000	NC_000912:700000	60562	0,555	0,1075
NC_000908:497000	NC_000913:2682000	19888	227,1276	403,1463
NC_000908:498000	NC_000913:2683000	19888	0,276	0,463
NC_000912:698000	NC_000913:2682000	22113	823,1860	400,1452
NC_000912:699000	NC_000913:2683000	22113	0,860	0,452

ated for a single homologous region. Our approach is to convert local alignments to subsequence identifiers which start at one of evenly spaced break positions.

Let a break position be denoted by b and a local alignment α be represented by (s_i, e_i, s_j, e_j) where s_i and e_i (s_j and e_j) are the starting and end positions of α in genome G_i (G_j). Then α generates the subsequence identifiers $(G_i:p_i k, G_j:p_j k), (G_i:p_i(k+1), G_j:p_j(k+1)), \dots, (G_i:q_i, G_j:q_j)$ where $p_i = b\lfloor s_i/b \rfloor$, $p_j = b\lfloor s_j/b \rfloor$, $q_i = b\lfloor e_i/b \rfloor$, $q_j = b\lfloor e_j/b \rfloor$. Table 1 shows the BAG input for local alignment regions (497087, 499555, 698686, 701075) for NC_000908 vs. NC_000912 pair, (497227, 498276, 2682403, 2683463) for NC_000908 vs. NC_000913 pair, and (698823, 699860, 2682400, 2683452) for NC_000912 vs. NC_000913 pair. As discussed, the alignments start at slight different positions, for example, positions 497087 and 497227 for NC_000908, which is not trivial to handle given that there are numerous local matches for whole genomes. By generating subsequences that start at a 1000 bp break position ($b = 1000$), the two different starting positions, 497087 and 497227 for NC_000908, have the same identifier, NC_000908:497000, with their aligned positions adjusted as shown in Table 1. Now BAG can combine matching regions using subsequence identifiers into two clusters, $\{\text{NC_000908:497000}, \text{NC_000912:698000}, \text{NC_000913:2682000}\}$ and $\{\text{NC_000908:498000}, \text{NC_000912:699000}, \text{NC_000913:2683000}\}$.

3 Strategy for Combining Pairwise Matches

Let us summarize the whole procedure for combining pairwise matches in order to generate multiple sequence alignments. For a set of genomes, $\{G_1, G_2, \dots, G_n\}$,

1. Compute pairwise alignments using BLASTZ for each pair of genomes, G_i and G_j , $1 \leq i < j \leq n$.
2. Generate subsequence identifiers at each breakpoint as described in Section 2.2 so that all pairwise genome data can be used for BAG. This is necessary since each pairwise alignment for a local homologous region can start at different position. For example, consider three local pairwise matches for a

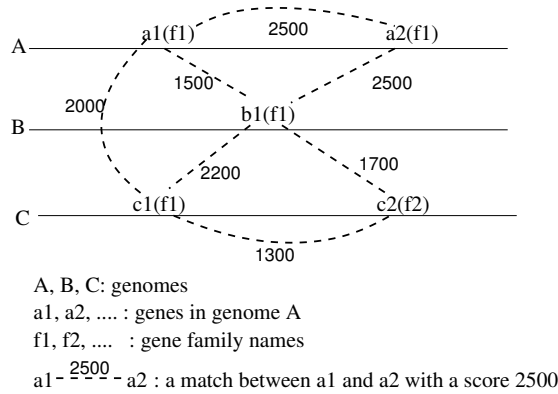


Fig. 4. Co-occurrences of genes in three genomes.

- gene a_1 in Fig. 4, $a_1 - a_2$, $a_1 - b_1$, and $a_1 - c_1$. Positions for a_1 in three pairwise matches start at slightly different positions in most cases and generation of subsequence identifiers can handle this easily (see Table 1 for a real example)
- Once subsequence identifiers are generated, BAG enumerates biconnected components with a common shared region. For example, all five genes in Fig. 4 form a biconnected component, but they do not share a common shared region (we assume this since two different families are clustered together). Increasing a cutoff to 1500 eliminates an edge between c_1 and c_2 from the graph and recomputation will successfully separate the current graph into two biconnected components.
 - Among final biconnected components, i.e., those that successfully passed RANGE-TEST, we select ones that cover at least k genomes. We call this *support value*. By default, k is the number of genomes being aligned. For the example in Fig. 4, $k = 3$. There are two biconnected components, $\{a_1, a_2, b_1, c_1\}$ and $\{b_1, c_2\}$, but $\{b_1, c_2\}$ is not considered as an alignment since $\text{SUPPORT}(\{b_1, c_2\}) < k$.

4 Experimental Results

To evaluate the performance of our method, we have used completely sequenced microbial genomes published in NCBI, grouped them into sets of genomes based on taxonomy, and selected 12 groups as shown in Table 2.

The multiple alignment programs can be evaluated in terms of the number of common COGs detected by alignment results; the common COGs are extracted from protein table (PTT) files downloaded from GenBank at NCBI. In this experiment, “common” COGs are those that are present in “all” genomes being aligned. This criterion excludes any COGs present only in a proper subset of genomes being aligned. As an alternative to our evaluation method, it is worth noting the rigorous column-by-column comparing evaluation method that used

Table 2. The 12 groups selected from microbial genomes for the experiments.

<i>Actinobacteridae</i> (5):	NC_003450, NC_002677, NC_002755, NC_000962, NC_004572
<i>Alphaproteobacteria</i> (5):	NC_004463, NC_003317, NC_002678, NC_003103, NC_000963
<i>Bacillales</i> (4):	NC_002570, NC_000964, NC_003212, NC_002745
<i>Betaproteobacteria</i> (3):	NC_003295, NC_003112, NC_003116
<i>delta-epsilon</i> (3):	NC_000915, NC_000921, NC_002163
<i>Ecoli</i> (3):	NC_000913, NC_002655, NC_002695
<i>Enterobacteriaceae</i> (4):	NC_000913, NC_003197, NC_003143, NC_002528
<i>Euryarchaeota</i> (10):	NC_002607, NC_000917, NC_000909, NC_003551, NC_003552, NC_000916, NC_002578, NC_002689, NC_000868, NC_000961
<i>Firmicutes</i> (7):	NC_003030, NC_002570, NC_000964, NC_003212, NC_002745, NC_003028, NC_002737
<i>Mollicutes</i> (5):	NC_000908, NC_004432, NC_000912, NC_002771, NC_002162
<i>Mycoplasma</i> (4):	NC_000908, NC_004432, NC_000912, NC_002771
<i>Thermoprotei</i> (3):	NC_000854, NC_003364, NC_002754

by TBA. Our method confirms only whether common homologous regions can be detected or not, while the TBA method try to evaluate how accurate the alignment result is. In the current implementation of our algorithm, focus is just on how accurately pairwise alignments can be combined and the final alignments are generated using existing multiple sequence alignment algorithms such as CLUSTALW [7].

Given the goal of detecting common COGs present in all genomes, we can evaluate the performance of multiple genome alignment algorithms in two ways. The first evaluation criterion is that a common COG is detected only when the region aligned by the algorithm is present in all genomes. The second criterion is that a common COG is detected if the region aligned by the algorithm is present at least in the half of the all genomes, i.e, the support value is more than half of the number of genomes. The first criterion will be called as ALL and the second as HALF.

We compared performances of TBA, MultiPipMaker (MPM), and different versions of our algorithm BAG, BAG-BCC (Fw), BAG-BCC (Fw+Bw), BAG-CC (Fw), and BAG-CC (Fw+Bw); Fw means the case that only forward local alignments are used as TBA works, Fw+Bw means the case that both forward and backward local alignments are used, CC means that connectedness is used for clustering criterion, and BCC means that biconnectedness is used for clustering criterion. With the ALL criterion, BAG (Fw+Bw) and MPM consistently outperformed TBA. This is because TBA aligned forward strands only. To make comparison with TBA fair, the alignment result of BAG (Fw) is shown in the table. BAG (Fw) outperformed TBA except for *Actinobacteridae*, *delta-epsilon*, and *Enterobacteriaceae*. With the relaxed HALF criterion, both BAG and TBA

Table 3. The alignment results of TBA, MPM, and BAG in terms of the number of common COGs detected with the ALL and HALF criterion.

Group	#COGs	ALL				HALF			
		TBA	MPM	BAG		TBA	MPM	BAG	
				Fw	Fw+Bw			Fw	Fw+Bw
Actinobacteridae	447	11	47	9	66	450	535	227	671
Alphaproteobacteria	468	0	17	1	62	17	357	87	487
Bacillales	935	85	192	91	398	756	1032	330	884
Betaproteobacteria	979	0	255	7	102	704	633	18	150
delta-epsilon	759	52	237	9	111	755	807	12	128
<i>E. coli</i>	1972		1897	1004	1487		2006	1032	1555
Enterobacteriaceae	520	29	117	20	210	1694	1801	389	1123
Euryarchaeota	330	0	1	0	4	0	90	11	177
Firmicutes	573	0	12	2	39	87	300	189	514
Mollicutes	258	1	21	1	37	27	231	48	200
Mycoplasma	291	2	40	2	49	297	356	16	155
Thermoprotei	671	0	74	5	175	37	347	6	211

are competitive. Detailed summary of the alignment results are given in Table 3. In the tables, the second column represents the number of common COGs among all genomes in each group, the third and the seventh columns represent the number of common COGs detected by TBA with ALL and HALF condition respectively, the fourth and ninth columns by MPM, the fifth and the tenth columns by BAG-BCC (Fw), and the sixth and the eleventh columns by BAG-BCC (Fw+Bw). TBA could not align *E. coli* for a unknown system related reason. Note that BCC (biconnectedness) produced better results than CC (single linkage), which demonstrates clearly the effectiveness of biconnectedness though we omit it for page limit. It is also worth looking at how many detected alignments correspond to the common COGs. Table 4 shows the result with the ALL criterion. In general, higher ratios of alignments by BAG correspond to common COGs than those by TBA and MPM. However, this does not necessarily imply that BAG generates more accurate alignment results since there are many regions shared among genomes that are outside common COGs. The primary role of genome alignment algorithms is probably to generate common shared regions in multiple genomes so that these regions can be investigated further biologically or computationally, thus our experiment suggests that BAG is complementary to existing algorithms such as TBA and MPM.

5 Conclusion

In this paper, we proposed a new strategy for multiple genome alignment. It uses a sequence clustering algorithm, BAG, to combine pairwise alignments. Our strategy accurately detected multiple homologous regions and performed competitively over existing algorithms as shown in the experiments with 12 groups of 56 microbial genomes in terms of the number of common COGs detected. This

Table 4. The alignment results of TBA, MultiPipMaker (MPM) and BAG in terms of the number of clusters that detect common COGs (with the ALL criterion) and the number of clusters generated by the algorithm. The number pair n/m denote that m alignments were generated by the corresponding algorithm and, among them, n alignments correspond to the common COGs. Note that the number of clusters are significantly smaller than the number of COGs detected in Table 3 since an alignment covers multiple common COGs that appear in tandem.

Group	TBA	MPM	BAG			
			BCC		CC	
			Fw	Fw+Bw	Fw	Fw+Bw
Actinobacteridae	28 / 1731	45 / 3010	13 / 361	82 / 1632	1 / 70	9 / 20
Alphaproteobacteria	0 / 871	12 / 14426	2 / 130	68 / 1413	0 / 107	2 / 121
Bacillales	99 / 1514	169 / 6368	115 / 455	480 / 1411	42 / 211	46 / 176
Betaproteobacteria	0 / 328	244 / 1604	8 / 26	117 / 182	8 / 30	27 / 41
delta-epsilon	20 / 250	190 / 1433	11 / 12	94 / 127	12 / 13	36 / 44
Ecoli	/	379 / 1303	1043 / 1060	1957 / 2039	988 / 992	648 / 676
Enterobacteriaceae	40 / 1453	83 / 5489	26 / 523	231 / 1626	3 / 325	35 / 360
Euryarchaeota	0 / 736	1 / 3954	0 / 240	8 / 1264	0 / 152	0 / 63
Firmicutes	0 / 2000	20 / 10769	3 / 720	64 / 1985	2 / 294	1 / 147
Mollicutes	2 / 443	20 / 1311	2 / 61	51 / 343	1 / 47	6 / 33
Mycoplasma	2 / 298	36 / 1025	3 / 36	61 / 240	3 / 31	9 / 31
Thermoprotei	0 / 51	86 / 1414	9 / 12	274 / 338	9 / 11	112 / 136

evaluation method is not sufficient in a sense because COGs are protein coding genes and there are many interesting regions common in multiple genomes such as non-coding RNAs and intergenic regions. Alignments of these regions together with biological interpretation will be reported in a forthcoming paper. However, the most primary use of genome alignment tools is to detect coding regions as mentioned in [20], and we believe that our evaluation method using common COGs is effective, especially for microbial genomes where intergenic regions are quite small.

Acknowledgments

This work partially supported by INGEN (Indiana Genomics Initiatives), NSF CAREER Award DBI-0237901, and KOSEF (Korea Science and Engineering Foundation) F01-2004-000-10016-0. We thank Haixu Tang of Indiana University for his suggestions on the presentation of this work.

References

1. Kellis, M., Patterson, N., Endrizzi, M., Birren, B., Lander, E.: Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature* **423** (2003) 241–254

2. Pevzner, P., Tesler, G.: Human and mouse genomic sequences reveal extensive breakpoint reuse in mammalian evolution. *Proc. Natl. Acad. Sci. U.S.A.* **100** (2003) 7672–7677
3. Schwartz, S., Kent, W.J., Smit, A., Zhang, Z., Baertsch, R., Hardison, R.C., Haussler, D., Miller, W.: Human-mouse alignments with BLASTZ. *Genome Res.* **13** (2003) 103–107
4. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* **48** (1970) 443–453
5. Smith, T.F., Waterman, M.S.: Identification of common molecular sequences. *J. Mol. Biol.* **147** (1981) 195–197
6. Lipman, D.J., Altschul, S.F., Kececioglu, J.D.: A tool for multiple sequence alignment. *Proc. Natl. Acad. Sci. U.S.A.* **86** (1989) 4412–4415
7. Thompson, J., Higgins, D., Gibson, T.: CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22** (1994) 4673–4680
8. Corpet, F.: Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Res.* **16** (1988) 10881–10890
9. Gotoh, O.: Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. *J. Mol. Biol.* **264** (1996) 823–838
10. Notredame, C., Higgins, D.: SAGA: sequence alignment by genetic algorithm. *Nucleic Acids Res.* **24** (1996) 1515–1524
11. Kim, J., Pramanik, S., Chung, M.: Multiple sequence alignment using simulated annealing. *Comput. Appl. Biosci.* **10** (1994) 419–426
12. Höhl, M., Kurtz, S., Ohlebusch, E.: Efficient multiple genome alignment. *Bioinformatics* **18** (2002) S312–S320
13. Morgenstern, B., Frech, K., Dress, A., Werner, T.: DIALIGN: Finding local similarities by multiple sequence alignment. *Bioinformatics* **14** (1998) 290–294
14. Brudno, M., Do, C.B., Cooper, G.M., Kim, M.F., Davydov, E., NISC Comparative Sequencing Program, Green, E.D., Sidow, A., Batzoglou, S.: LAGAN and Multi-LAGAN: Efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.* **13** (2003) 721–731
15. Bray, N., Pachter, L.: MAVID: Constrained ancestral alignment of multiple sequences. *Genome Res.* **14** (2004) 693–699
16. Blanchette, M., Kent, W.J., Riemer, C., Elnitski, L., Smit, A.F., Roskin, K.M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E.D., Haussler, D., Miller, W.: Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res.* **14** (2004) 708–715
17. Schwartz, S., Elnitski, L., Li, M., Weirauch, M., Riemer, C., Smit, A., Program, N.C.S., Green, E.D., Hardison, R.C., Miller, W.: MultiPipMaker and supporting tools: alignments and analysis of multiple genomic DNA sequences. *Nucleic Acids Res.* **31** (2003) 3518–3524
18. Kim, S.: Graph theoretic sequence clustering algorithms and their applications to genome comparison. In Wu, C.H., Wang, P., Wang, J.T.L., eds.: *Computational Biology and Genome Informatics*. World Scientific (2003)
19. Kim, S., Gopu, A.: Cluster utility: A new metric to guide sequence clustering. Technical report, School of Informatics, Indiana University (2004)
20. Miller, W.: Comparison of genomic DNA sequences: Solved and unsolved problems. *Bioinformatics* **17** (2001) 391–397