BMC Bioinformatics

BioMed Central

Methodology article

Open Access

A fast algorithm for determining the best combination of local alignments to a query sequence

Gavin C Conant* and Andreas Wagner

Address: Department of Biology, The University of New Mexico, Albuquerque, NM, USA

Email: Gavin C Conant* - gconant@unm.edu; Andreas Wagner - wagnera@unm.edu

* Corresponding author

Published: 18 May 2004

BMC Bioinformatics 2004, **5**:62

Received: 15 November 2003

Accepted: 18 May 2004

This article is available from: http://www.biomedcentral.com/1471-2105/5/62

Abstract

Background: Existing sequence alignment algorithms assume that similarities between DNA or amino acid sequences are linearly ordered. That is, stretches of similar nucleotides or amino acids are in the same order in both sequences. Recombination perturbs this order. An algorithm that can reconstruct sequence similarity despite rearrangement would be helpful for reconstructing the evolutionary history of recombined sequences.

Results: We propose a graph-based algorithm for combining multiple local alignments to a query sequence into the single combination of alignments that either covers the maximal portion of the query or results in the single highest alignment score to the query. This algorithm can help study the process of genome rearrangement, improve functional gene annotation, and reconstruct the evolutionary history of recombined proteins. The algorithm takes $O(n^2)$ time, where n is the number of local alignments considered.

Conclusions: We discuss two example applications of the algorithm. The algorithm is able to provide useful reconstructions of the metazoan mitochondrial genome. It is also able to increase the percentage of a query sequence's amino acid residues for which similar stretches of amino acids can be found in sequence databases.

Background

The introduction of the Smith-Waterman local alignment algorithm [1] and the subsequent development of the FASTA [2] and BLAST [3,4] database search tools has revolutionized comparative sequence studies. In the age of completely sequenced genomes, these algorithms are often used to compare a DNA or protein sequence of unknown function – a *query* sequence – with one or more *reference* sequences. Such reference sequences are often contained in databases of many thousand DNA or protein sequences. If the query sequence is similar to one or more reference sequences with known function, an informed guess about the function of the query sequence is possible. Sequence alignment algorithms, by their nature, assume that similarities between pairs of sequences are linearly ordered (homologous residues occur in the same order in both sequences). Thus, these algorithms are not well-suited to compare sequences that have undergone rearrangements through recombination. However, during evolution, rearrangements of genomic DNA occur frequently and on all scales, from individual genes to entire genomes. To give but a few examples: the mouse

chromosome 16 shows substantial regions of synteny (blocks of genes with conserved order) to six different human chromosomes [5]; a comparison of metazoan mitochondrial genomes demonstrated that the gene orders between the major metazoan phyla are often essentially random with respect to each other [6]; the genomes of three yeast species closely related to the baker's yeast *S. cerevisiae* show identifiable inversions and translocations in comparison to that yeast [7]; and Seoighe and Wolfe estimate that baker's yeast itself has undergone roughly 84 reciprocal translocations since a whole-genome duplication event roughly 100 Mya [8].

The question of how to deduce the number and order of rearrangements that have occurred between any two genomes has been the subject of intense research because such rearrangements can be used for phylogenetic inference (see for instance [9,10]). Here we address an even simpler question: How can one extend existing sequence comparison algorithms such that they are no longer sensitive to the linear ordering of sequence similarity? Doing so would permit an automatic comparison of two or more sequences for which rearrangements have occurred and an identification of the rearranged sequence fragments. The algorithm we propose achieves this goal. It combines local alignments between a query and one or more reference sequences without regard to the alignments' order in the reference sequence(s). The algorithm is equally applicable to short and long (genome-scale) sequences and to nucleotide and amino-acid sequences. The algorithm is also agnostic about how local alignments are generated (i.e. using BLAST, [3,4]; FASTA, [2]; or dynamic programming, [1]).

Aside from being a starting point for reconstructing the rearrangement history of genomes [10], the algorithm also has other uses. First, it can give clues about the evolutionary origins of proteins: Protein-coding genes often contain multiple functional modules or domains, and similar domains occur in various combinations in different proteins [11-15]. Such combinations of domains have come about through recombination, and the algorithm can easily and automatically identify recombined proteins in large databases of protein sequences.

Second, the algorithm may help infer a query sequence's function from database searches in cases where the query sequence shows similarity to short regions of several database sequences, but where there is no single database sequence with similarity extending over most of the query's length. For such query sequences, it may sometimes be possible to infer function by combining information from multiple partial matches. Current sequence database search tools are less than ideal for this purpose. They return lists of reference sequences ordered by the statistical significance of their similarity to the query. Such lists of matches, however, cannot be readily converted into a single combination of alignments for use in functional inference. Our algorithm solves this problem.

Below, we first discuss how to optimally combine multiple local matches to a query sequence into one alignment combination. Then, we evaluate the statistical significance of these alignment combinations. Specifically, we propose two distinct optimality criteria for use when combining alignments. The algorithm takes a very similar form for both criteria. The first criterion is to select the one combination of alignments (out of all possible such combinations) that covers the maximum number of sites in the query sequence. Figure 1A illustrates this idea. The second criterion uses alignment scores: the sum of match, mismatch, and gap scores for an alignment, usually based on a cost matrix such as PAM [16,17] or BLOSUM [18]. One might ask why we have chosen raw alignment scores rather than a measure of alignment significance such as E or P value. The reason is that E-values are not additive across alignment combinations (see equation 2) and hence cannot be evaluated by our algorithm.

We refer to the combination of alignments selected under either criterion as the "Optimal Alignment Combination" (OAC). Unfortunately, as the number of local alignments to a query sequence grows, the number of possible combinations of local alignments increases very quickly. In the worst case when no two local alignments overlap, 2^n such combinations are possible (although in this special case finding the OAC is trivial). Note that n can be very large in realistic applications: Programs such as BLAST may return many hundreds of database matches when performing whole genome comparisons. Clearly, an exhaustive search for the OAC can be computationally demanding. However, our algorithm can determine an OAC in $O(n^2)$ time, where n is the number of initial local alignments.

Algorithm description

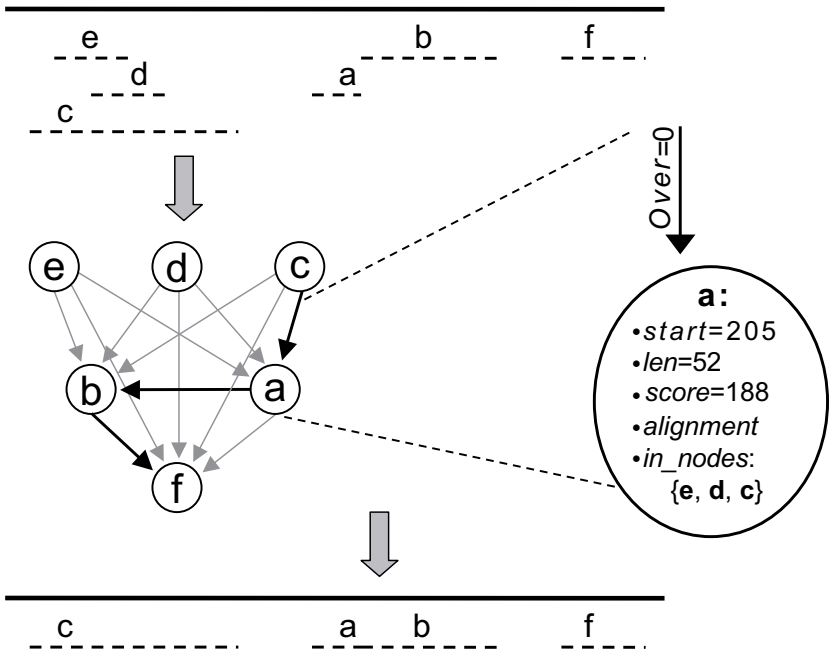
We represent each local alignment as a node in a directed graph (see Figure 1A). Each node m is given a name, as well as a starting position m_{start}, a length m_{len} and the score m_{score} of the alignment corresponding to it.

We sequentially process each node m, checking all other nodes n to see if they meet the following criterion

$$m_{start} + m_{len} - 1 \leq n_{start + overlap}$$

Note that $m_{start} + m_{len} - 1$ gives the end position for the alignment represented by m. A directed edge is added from m to n in any case where the above criterion is met.

A)



B)

```
1)   Sub recurse_search (node) {
2)        Foreach (i ∈ node.in_nodes) {
3)             If (i.Done = FALSE)
4)                  recurse_search(i)
5)        }
6)        best_so_far=0
7)        Foreach (i ∈ node.in_nodes) {
8)             If (i.best+node.len-edges{node,
                   i}.over > best_so_far) {
9)                  node.best=i,
10)                 best_so_far= i.best+node.len
                         -edges{node,i}.over
11)            }
12)       }
13)       node.Done=TRUE;
14)  }
```

Figure 1

The maximal coverage alignment problem: For a query sequence to which multiple local alignments (through standard alignment methods or database search) have been generated, we wish to pick the combination of local alignments that covers the maximum proportion of the query sequence. **A)** Representing a series of alignments as a graph. Each lettered alignment is shown as a node. Letters indicate the order of these sequences in the reference sequence (i.e. the fourth alignment is letter "a" because it occurs first in the reference sequence). Edges join nodes with permissible overlap. The path through this graph corresponding to the OAC is shown with darkened arrows. At right is a representation of some of the data stored by each node and edge. The five most important pieces of information stored in the nodes are the starting position and length of the alignment (*start* and *len*), the alignment score (if that is used as an objective function), the sequence alignment for the node in question (needed when overlap is permitted and alignment scores are the objective function) and the list of "in nodes:" those nodes with a directed edge leading to the current node. Edges store the extent of overlap between the nodes they connect (which is also easily calculated from the two nodes' values of *start* and *len*. **B)** Pseudo-code implementation of the last step in our algorithm, the depth-first search (see text). The dot (.) operator represents access to data structure members.

Overlap is an integer greater than or equal to zero. Its use in the above expression reflects the fact that we allow local alignments to overlap by a limited amount, because alignments may not end exactly where sequence similarity does. Nodes representing alignments with overlapping regions longer than *overlap* are not connected by edges: they cannot co-occur in any alignment combination. A list is kept of every node with an out-degree of 0. These nodes are potential end points for the OAC, since there are no other alignments which end *after* them relative to the query.

The final step in our algorithm is a variant of a depth-first search [19] of the graph, starting from each of the nodes of out-degree 0. The pseudo-code shown in Figure 1B describes this search. The code is shown with the longest combination as the optimality criterion. We discuss the algorithm with this optimality criterion first and then discuss the minor changes required to use alignment scores as the optimality criterion.

Our algorithm is a divide-and-conquer approach based on the observation that membership in alignment combinations that can be part of an OAC is associative. This associatively means that if nodes *m* and *n* can occur in an optimal alignment combination, and if *n* and *o* can occur in the combination, then *m* and *o* can also occur in the optimal alignment combination. Practically, this means that it is possible to recurse through the graph, picking for every node the combination of ancestors – nodes representing alignments starting before the current node's alignment on the query sequence – that gives the longest combination up to and including that node. The depth-first ordering of the nodes in the search guarantees that all of a node's ancestors will have been processed already and that their best combination of alignments will be known by the time the node is visited. In other words, as nodes are processed in the search, the OAC problem is solved up to the currently visited node.

Figure 1B describes this search, showing a recursive routine *recurse_search* that analyzes a node "*node*". The routine first steps through each node *i* in the list *in_nodes*, which contains all nodes who have outgoing edges pointed at the current node. Each node *i* is checked to see if it has already been processed (lines 2–5 in Figure 1B), in which case *i.Done* has the value TRUE. For any *i* where *i.Done* is FALSE, *recurse_search* is called recursively for that node (line 4).

Once this recursion is complete, the algorithm finds the best alignment combination up to the current node *node* as follows. First, note that the length of the best combination for a node is stored in its variable *best*. To determine *node.best*, each member *i* of *in_nodes* is examined to see if it forms the best combination when combined with *node* (lines 7–11). The best combination for each node *i* (*i.best*) is already known as a result of the depth-first search ordering. To create a combination including the current node, the algorithm adds this *i.best* value to the length of the alignment corresponding to the current node (*node.len*), subtracting any overlap (*over*). The value of *over* is obtained from the data structure *edges*{*node, i*}, which returns this value in *O(1)* time (lines 8 and 10).

For each node *i*, we compare *i.best+node.len – over* (line 8) to the best alignment combination found so far (*best_so_far*). If the new combination is better, it replaces *best_so_far*. When this loop over all nodes *i* has completed, *best_so_far* must contain the best combination; that length is assigned to *node.best* (line 10).

After the depth-first search from each node of out-degree 0 (see above) is complete, one simply has to examine each of these nodes with out-degree 0 to find the one associated with the combination of alignments with the largest number of residues in it. This, by definition, is the OAC.

When looking for the largest alignment score rather than the longest alignment combination, two modifications to the above routine must be made. In the first place, *node.score* replaces *node.len* in lines 8 and 10 of figure 1B. This has the effect of selecting combinations with high scores rather than long combinations. The second modification comes in calculating the value of *over*. If *over* were not computed, the score of any overlapped residues would be counted twice: once in *node.score* and once in *i.score*. Thus, *over* must be determined using the two alignments, which are held in the data structures *node.alignment* and *i.alignment*. Knowledge of the alignment score scheme used (for instance BLOSUM62 with a gap opening penalty of -12 and an extension penalty of -2) then allows one to calculate the score of the two combined alignments with the overlapping residues counted only once. All other details remain the same for this optimality criterion.

Statistical significance of OACs

To evaluate the statistical significance of an alignment, programs such as BLAST typically determine expectation values (E-values). An E-value gives the expected number of chance hits having an alignment score at least as high as that observed, given the size of the database used in the search. Analytic descriptions of the statistics of ungapped alignments and their expected scores are known [20]. Although it has not been formally shown that gapped alignments obey the same distributions as do ungapped alignments, there is considerable empirical evidence that this is the case [21]. Karlin and Altschul have given a formula for computing a P-value – a measure of significance closely related to an E-value – for the sum of the scores of

an alignment combination, which we use here to compute the significance of our combinations [22] (but see also [21]). Their analysis allows the calculation of a P-value for the combination of r different alignments, given the scores of those alignments (S_i) and two parameters (K and λ) that characterize the alignment scoring matrix used (for example BLOSUM [18]). Specifically, the probability of seeing a combination of alignments where the sum of the normalized alignment scores (given by $T = \sum_{i=1}^{r} \lambda S_i - \ln(K_{mn})$) is at least as large as some critical value t is given by:

$$P(T \geq t) = \int_{t}^{\infty} \frac{e^{-t}}{r!(r-2)!} \cdot \int_{0}^{\infty} y^{r-2} \exp\left(-e^{(y-t)/r}\right) \cdot dy \qquad (2)$$

If we assume that the number of cases in a random database where T exceeds the T_{obs} from our real data follows a Poisson distribution, we can use this P-value to obtain E (see [20] for details). For alignments of practical importance, P is small ($<10^{-2}$) and in that case P and E-values are essentially identical.

We note that combinations of alignments will always have a lower net E-value than a single alignment of the same alignment score, since E-values decrease exponentially with increasing alignment score. We illustrate this point with a simple example involving the duplicate *S. cerevisiae* genes SSA1 and SSB1. In the first comparison, we aligned SSA1 against the complete sequence of SSB1. This resulted in an alignment of length 582, a (non-normalized) score of 1814, and an E-value of 7.0×10^{-189}. We next split the SSB1 sequence into two equally sized pieces and aligned those pieces to SSA1. These two non-overlapping alignments were input into our algorithm for finding OACs. The result was an alignment combination of 582 residues with an E-value of 1.8×10^{-182}. When considered separately, the alignment of SSA1 to the first half of SSB1 has a score of 1043 and an E-value of 3.5×10^{-107}. The alignment of SSA1 to the second half of SSB1 has a score of 771 and an E-value of 2.3×10^{-78}.

Note that the E-values of the global alignment and the OAC are different (7.0×10^{-189} versus 1.8×10^{-182}) even though the sum of the independent alignment scores (1043 + 771) was equal to the alignment score when the full-length genes were aligned (1814).

Performance

An arbitrary graph of n nodes can have at most $O(n^2)$ edges (one edge between every possible pair of nodes). The above algorithm visits every edge in the graph three times: once initialising the graph, and twice in the above search procedure. Thus, the worst case of the algorithm, in running time and memory, is $O(n^2)$. In other words, the running time of the algorithm has an upper bound proportional to the square of the number of alignments. Real-world performance is also acceptable: a list of 580 local alignments can be processed in less than 70 ms on an 800 MHz Pentium III, while 3100 local alignments take only 3 seconds on the same platform.

Example data

We have evaluated the performance of our algorithm for two different problems. The first regards recombination in metazoan mitochondrial genomes. We compared the human mitochondial genome (GenBank accession number NC_001807 [23]; our query sequence) to three other mitochondrial reference genomes: the hagfish *Myxine glutinosa* (GenBank accession number NC_002639, [24]), the fruit fly *Drosophila melanogaster* (Genbank accession number NC_001709), and the nematode *Caenorhabditis elegans* (GenBank accession number NC_001328). We used the LALIGN package [25] to find all local alignments between each pair of genomes. Only alignments with E-values of 10^{-5} or less were included in our analysis. We input the resulting alignments into the above algorithm, searching for the combination of alignments that resulted in the highest alignment score. We allowed an overlap of 50 nucleotides between alignments.

Figure 2 shows the results for the three comparisons. The fragments of the reference genomes with partial matches to the query genome are indicated by numbers corresponding to their starting and ending positions in the reference genome. Figure 2A shows that the human and hagfish genomes have identical order except for a short region of similarity near the beginning of the human genome which is located near the end of the hagfish genome. Since the mitochondrial genome is circular, this constitutes a minor difference and does not indicate a history of many rearrangements. A similar situation exists in the human/fruit fly comparison, except that an inversion appears to have occurred between the two species (Figure 2B; indicated in grey). In contrast, the evolutionarily more distant nematode sequence shows only short regions of similarity to the human sequence, with two or three possible inversion events indicated (Figure 2C). Collectively, these results suggest that information about the relative ordering of circular genomes can be conserved for a long time (such as that separating *Drosophila* and humans). Whether this method is sensitive enough to produce distance measures of sufficient accuracy for problems such as phylogenetic inference remains to be seen. A potential problem in applying the algorithm to phylogenetic inference is that sequences not only get rearranged but they also diverge through point mutations. Thus, a loss of detectable homology due to sequence divergence may bias the inference of the number of rearrangements that

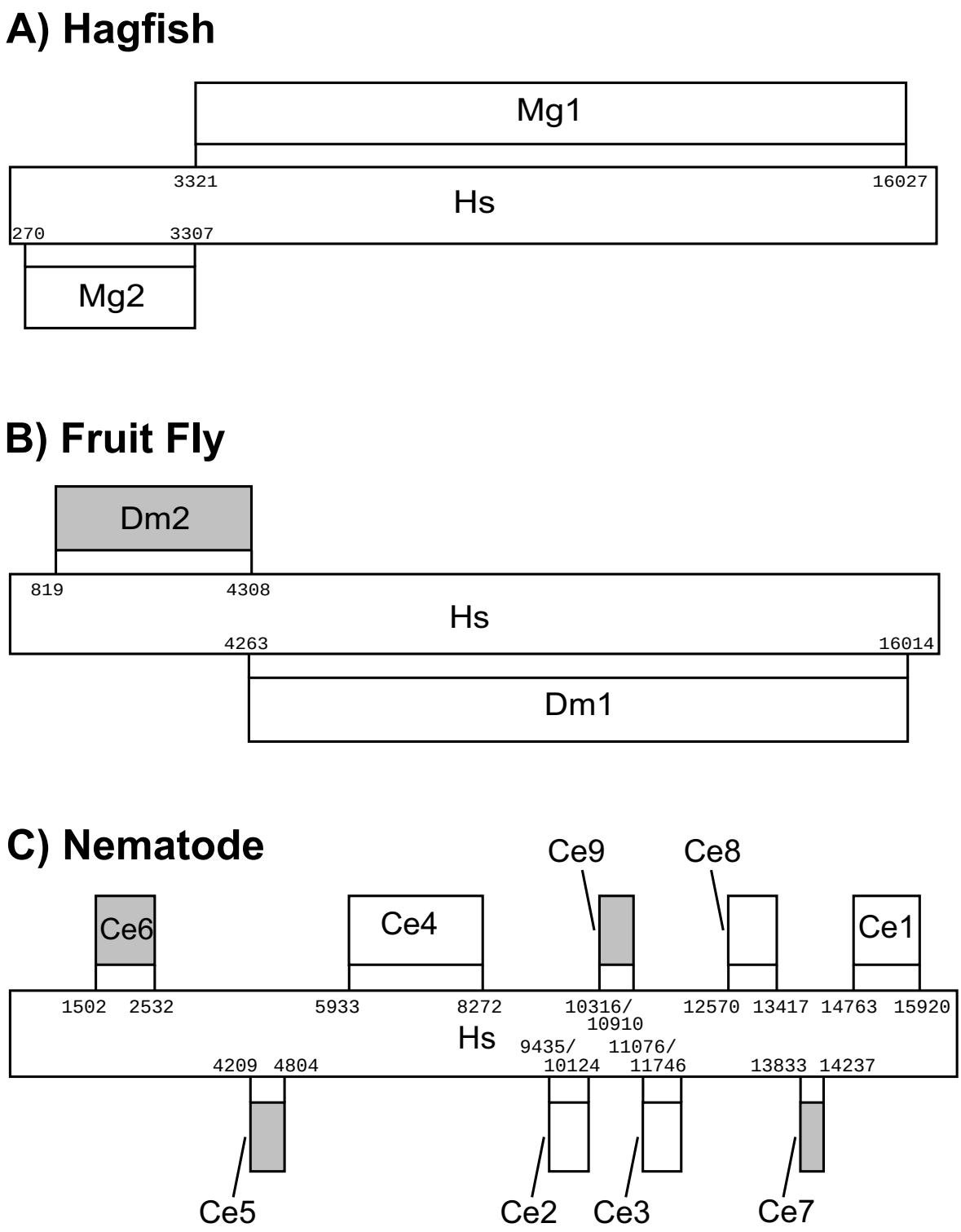


Figure 2

Comparison of the human mitochondrial genome to three other mitochondrial genomes: A) hagfish *Myxine glutinosa (Mg)*; **B)** fruit fly *Drosophilia melanogaster (Dm)*; **C)** nematode *Caenorhabditis elegans (Ce)*. Blocks indicate regions of local similarity between two genomes. Numbers in these blocks (Gm1 etc.) indicate the order of the blocks in the reference genome. Grey blocks indicate inversions relative to the human genome (*i.e.* these sequences are found on the opposite strand of the DNA helix and in the reverse direction)

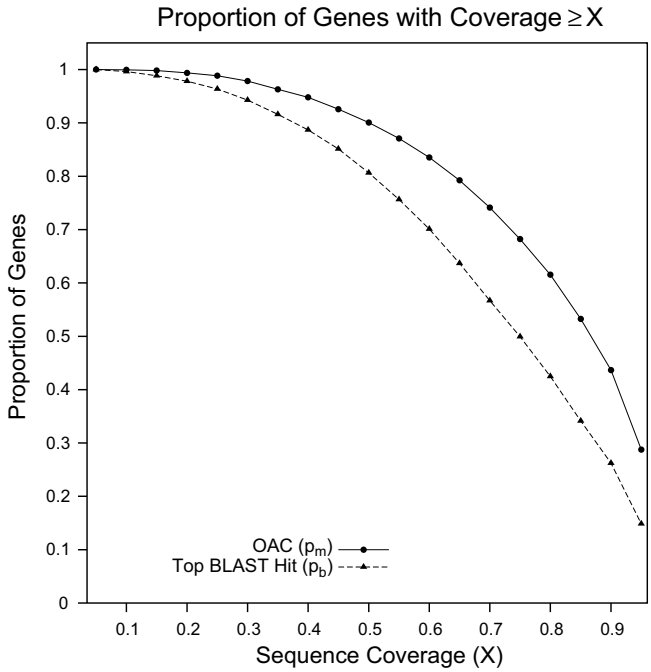


Figure 3
Combining alignments increases the percentage of aligned residues in queries. Proportion of *S. cerevisiae* genes with *D. melanogaster* hits spanning at least X% of their sequence when only the top BLAST hit (p_b) is considered and when the OAC is used (p_m). Only yeast genes showing BLAST hits with E-values of 1×10^{-5} or less were considered.

have occurred between two sequences. However, the method is attractive in that it can be applied automatically to many genomes: there is no need for explicit manual declarations of homology between sequences, which is often required with other approaches [10].

With our second analysis, we sought to demonstrate the increase in sequence coverage provided by the OAC. To do so, we ran BLASTP on every protein-coding gene in the *Saccharomyces cerevisiae* genome [26], using the protein-coding genes of the *Drosophila melanogaster* genome [27] as the database. We have used the Washington University implementation of BLAST [28] rather than the NCBI version because this first version is more conservative in attempting to extend short regions of similarity into longer alignments. We considered only hits with E-values smaller than 1×10^{-5} in this analysis. In calculating the OAC, we allowed a maximum overlap of 10 amino acid residues between alignments: matches with overlap below this threshold were considered as connected nodes in the alignment graph of Figure 1A. For each gene with at least one significant hit we calculated the proportion of that gene covered by the top BLAST hit (p_b) and by the OAC (p_m). The average number of residues in the query sequences covered by only the top BLAST hit was 71%. This average increased to 80% when the OAC was used. Note that our inclusion of queries with only a single hit will underestimate the increase in the number of residues covered in a typical application of the algorithm, because for such queries the single hit is the OAC and thus $p_b = p_m$. Figure 3 shows the proportion of genes with $p_b \geq X$ and $p_m \geq X$, where X is the proportion of the total gene covered by the alignments. It is clear from this figure that a higher proportion of query residues are aligned to the database when the OAC is used.

As we have discussed, this algorithm to combine multiple local alignments to a query sequence can serve to improve functional gene annotation, the reconstruction of the evolutionary history of shuffled proteins, and the discrimination of rapidly from slowly evolving gene regions. The algorithm is thus a tool to further increase leverage in inferring functional and evolutionary information from sequences.

Abbreviations
OAC: Optimal Alignment Combination

Authors' contributions
The algorithm described here was jointly developed by GCC and AW. GCC implemented the algorithm in c++ and ran the example analyses. This manuscript was jointly written by GCC and AW.

Algorithm implementation
Source code for our c++ implementation of this algorithm is available from our website: http://www.unm.edu/~compbio/software/find_max_cover.

Acknowledgements
We would like to especially thank Michael Fuller for help with figure preparation. We would also like to thank Michael Gilchrist and Annette Evangelisti for helpful discussions during the preparation of this manuscript. GCC is supported by the Department of Energy's Computational Sciences Graduate Fellowship program, administered by the Krell Institute. AW would like to thank the NIH for its support through NIH grant GM063882-01.

References
1. Smith TF, Waterman MS: **Identification of common molecular subsequences.** *Journal of Molecular Biology* 1981, **147:**195-197.
2. Pearson WR, Lipman DJ: **Improved tools for biological sequence comparison.** *Proceedings of the National Academy of Sciences, USA* 1988, **85:**2444-2448.
3. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ: **Basic Local Alignment Search Tool.** *Journal of Molecular Biology* 1990, **215:**403-410.
4. Altschul SF, Madden TL, Schaffer AA, Zhang JH, Zhang Z, Miller W, Lipman DJ: **Gapped Blast and Psi-Blast : A new-generation of protein database search programs.** *Nucleic Acids Research* 1997, **25:**3389-3402.

5. Mural RJ, Adams MD, Myers EW, Smith HO, Gabor Miklos GL, Wides R, Halpern A, Li PW, Sutton GG, Nadeau J, Salzberg SL, Holt RA, Kodira CD, Lu F, Chen L, Deng Z, Evangelista CC, Gan W, Heiman TJ, Li J, Li Z, Merkulov GV, Milshina NV, Naik AK, Qi R, Chris Shue B, Wang A, Wang J, Wang X, Yan X, Ye J, Yooseph S, Zhao Q, Zheng L, Zhu SC, Biddick K, Bolanos R, Delcher AL, Dew IM, Fasulo D, Flanigan MJ, Huson DH, Kravitz SA, Miller JR, Mobarry CM, Reinert K, Remington KA, Zhang Q, Zheng XH, Nusskern DR, Lai Z, Lei Y, Zhong W, Yao A, Guan P, Ji R-R, Gu Z, Wang Z-Y, Zhong F, Xiao C, Chiang C-C, Yandell M, Wortman JR, Amanatides PG, Hladun SL, Pratts EC, Johnson JE, Dodson KL, Woodford KJ, Evans CA, Gropman B, Rusch DB, Venter E, Wang M, Smith TJ, Houck JT, Tompkins DE, Haynes C, Jacob D, Chin SH, Allen DR, Dahlke CE, Sanders R, Li K, Liu X, Levitsky AA, Majoros WH, Chen Q, Xia AC, Lopez JR, Donnelly MT, Newman MH, Glodek A, Kraft CL, Nodell M, Ali F, An H-J, Baldwin-Pitts D, Beeson KY, Cai S, Carnes M, Carver A, Caulk PM, Center A, Chen Y-H, Cheng M-L, Coyne MD, Crowder M, Danaher S, Davenport LB, Desilets R, Dietz SM, Doup L, Dullaghan P, Ferriera S, Fosler CR, Gire HC, Gluecksmann A, Gocayne JD, Gray J, Hart B, Haynes J, Hoover J, Howland T, Ibegwam C, Jalali M, Johns D, Kline L, Ma DS, MacCawley S, Magoon A, Mann F, May D, McIntosh TC, Mehta S, Moy L, Moy MC, Murphy BJ, Murphy SD, Nelson KA, Nuri Z, Parker KA, Prudhomme AC, Puri VN, Qureshi H, Raley JC, Reardon MS, Regier MA, Rogers Y-HC, Romblad DL, Schutz J, Scott JL, Scott R, Sitter CD, Smallwood M, Sprague AC, Stewart E, Strong RV, Suh E, Sylvester K, Thomas R, Ni Ni N, Tsonis C, Wang G, Wang G, Williams MS, Williams SM, Windsor SM, Wolfe K, Wu MM, Zaveri J, Chaturvedi K, Gabrielian AE, Ke Z, Sun J, Subramanian G, Venter JC: **A comparision of whole-genome shotgun-derived mouse chromosome 16 and the human genome.** *Science* 2002, **296:**1661-1671.
6. Blanchette M, Kunisawa T, Sankoff D: **Gene order breakpoint evidene in animal mitochondrial phylogeny.** *Journal of Molecular Evolution* 1999, **49:**193-203.
7. Kellis M, Patterson N, Endrizzi M, Birren B, Lander ES: **Sequencing and comparison of yeast species to identify genes and regulatory elements.** *Nature* 2003, **423:**241-254.
8. Seoighe C, Wolfe KH: **Extent of genomic rearrangement after genome duplication in yeast.** *Proceedings of the National Academy of Sciences, USA* 1998, **95:**4447-4452.
9. Sankoff D, Blanchette M: **Multiple genome rearrangement and breakpoint phylogeny.** *Journal of Computational Biology* 1998, **5:**555-570.
10. Sankoff D, El-Mabrouk N: **Genome Rearrangement.** *Topics in Computational Biology* Edited by: Jiang T, Xu Y and Zhang M. Boston, MIT Press; 2001.
11. Teichmann SA, Park J, Chothia C: **Structural assignments to the Mycoplasma genitalium proteins show extensive gene duplications and domain rearrangements.** *Proceedings of the National Academy of Sciences, USA* 1998, **95:**14658-14663.
12. Apic G, Gough J, Teichmann SA: **Domain combinations in archaeal, eubacterial and eukaryotic proteomes.** *Journal of Molecular Biology* 2001, **310:**311-325.
13. Bashton M, Chothia C: **The geometry of domain combinations in proteins.** *Journal of Molecular Biology* 2002, **315:**927-939.
14. Li W-H, Gu Z, Wang H, Nekrutenko A: **Evolutionary analyses of the human genome.** *Nature* 2001, **409:**847-849.
15. Müller A, MacCallum RM, Sternberg MJE: **Structural characterization of the human proteome.** *Genome Research* 2002, **12:**1625-1641.
16. Dayhoff MO, Eck RV, Park CM: **A model of evolutionary change in proteins.** *Atlas of protein sequence and structure Volume 5*. Edited by: Dayhoff M O. Washington, D.C., National Biomedical Research Foundation; 1972:89-99.
17. Dayhoff MO, Schwartz RM, Orcutt BC: **A model of evolutionary change in proteins.** *Atlas of protein sequence and structure Volume 5, suppl.* 2. Edited by: Dayhoff M O. Washington, D.C., National Biomedical Research Foundation; 1978:345-352.
18. Henikoff S, Henikoff JG: **Amino-acid substitution matrices from protein blocks.** *Proceedings of the National Academy of Sciences, USA* 1992, **89:**10915-10919.
19. Moret BME, Shapiro HD: **Algorithms from P to NP: Volume I: Design and Efficiency.** Redwood City, CA, The Benjamin/Cummings Publishing Company, Inc.; 1991.
20. Karlin S, Altschul SF: **Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes.** *Proceedings of the National Academy of Sciences, USA* 1990, **87:**2264-2268.
21. Altschul SF, Gish W: **Local alignment statistics.** *Methods in Enzymology* 1996, **266:**460-480.
22. Karlin S, Altschul SF: **Applications and statistics for multiple high-scoring segments in molecular sequences.** *Proceedings of the National Academy of Sciences, USA* 1993, **90:**5873-5877.
23. Ingman M, Kaessmann H, Paabo S, Gyllensten U: **Mitochondrial genome variation and the origin of modern humans.** *Nature* 2000, **408:**708-713.
24. Delarbre C, Rasmussen AS, Arnason U, Gachelin G: **The complete mitochondrial genome of the hagfish Myxine glutinosa: Unique features of the control region.** *Journal of Molecular Evolution* 2001, **53:**634-641.
25. Huang XQ, Miller W: **A time-efficient, linear space local similarity algorithm.** *Advances in Applied Mathematics* 1991, **12:**337-357.
26. Goffeau A, Barrell BG, Bussey H, Davis RW, Dujon B, Feldmann H, Galibert F, Hoheisel JD, Jacq C, Johnston M, Louis EJ, Mewes HW, Murakami Y, Philippsen P, Tettelin H, Oliver SG: **Life with 6000 genes.** *Science* 1996, **274:**546-567.
27. Adams MD, Celniker SE, Holt RA, Evans CA, Gocayne JD, Amanatides PG, Scherer SE, Li PW, Hoskins RA, Galle RF, George RA, Lewis SE, Richards S, Ashburner M, Henderson SN, Sutton GG, Wortman JR, Yandell MD, Zhang Q, Chen LX, Brandon RC, Rogers YH, Blazej RG, Champe M, Pfeiffer BD, Wan KH, Doyle C, Baxter EG, Helt G, Nelson CR, Gabor GL, Abril JF, Agbayani A, An HJ, Pfannkoch C, Baldwin D, Ballew RM, Basu A, Baxendale J, Bayraktaroglu L, Beasley EM, Beeson KY, Benos PV, Berman BP, Bhandari D, Bolshakov S, Borkova D, Botchan MR, Bouck J, Brokstein P, Brottier P, Burtis KC, Busam DA, Butler H, Cadieu E, Center A, Chandra I, Cherry JM, Cawley S, Dahlke C, Davenport LB, Davies P, de Pablos B, Delcher A, Deng Z, Mays AD, Dew I, Dietz SM, Dodson K, Doup LE, Downes M, Rocha S, Dunkov BC, Dunn P, Durbin KJ, Evangelista CC, Ferraz C, Ferriera S, Fleischmann W, Fosler C, Gabrielian AE, Garg NS, Gelbart WM, Glasser K, Glodek A, Gong F, Gorrell JH, Gu Z, Guan P, Harris M, Harris NL, Harvey D, Heiman TJ, Hernandez JR, Houck J, Hostin D, Houston KA, Howland TJ, Wei MH, Ibegwam C, Jalali M, Kalush F, Karpen GH, Ke Z, Kennison JA, Ketchum KA, Kimmel BE, Kodira CD, Kraft C, Kravitz S, Kulp D, Lai Z, Lasko P, Lei Y, Levitsky AA, Li J, Li Z, Liang Y, Lin X, Liu X, Mattei B, McIntosh TC, McLeod MP, McPherson D, Merkulov G, Milshina NV, Mobarry C, Morris J, Moshrefi A, Mount SM, Moy M, Murphy B, Murphy L, Muzny DM, Nelson DL, Nelson DR, Nelson KA, Nixon K, Nusskern DR, Pacleb JM, Palazzolo M, Pittman GS, Pan S, Pollard J, Puri V, Reese MG, Reinert K, Remington K, Saunders RD, Scheeler F, Shen H, Shue BC, Kiamos I, Simpson M, Skupski MP, Smith T, Spier E, Spradling AC, Stapleton M, Strong R, Sun E, Svirskas R, Tector C, Turner R, Venter E, Wang AH, Wang X, Wang ZY, Wassarman DA, Weinstock GM, Weissenbach J, Williams SM, Woodage T, Worley KC, Wu D, Yang S, Yao QA, Ye J, Yeh RF, Zaveri JS, Zhan M, Zhang G, Zhao Q, Zheng L, Zheng XH, Zhong FN, Zhong W, Zhou X, Zhu S, Zhu X, Smith HO, Gibbs RA, Myers EW, Rubin GM, Venter JC: **The genome sequence of Drosophila melanogaster.** *Science* 2000, **287:**2185-2195.
28. Gish W: **WU-BLAST.** . 1996-2003 http://blast.wustl.edu