

## ***A hierarchical approach to aligning collinear regions of genomes***

*Mikhail A. Roytberg<sup>1</sup>, Aleksey Y. Ogurtsov<sup>2</sup>,  
Svetlana A. Shabalina<sup>2</sup> and Alexey S.  
Kondrashov<sup>2,\*</sup>*

<sup>1</sup>*Institute of Mathematical Problems in Biology, Pushchino, Moscow  
Region 142290, Russia*

<sup>2</sup>*National Center for Biotechnology Information, NIH, 45 Center Drive,  
Bethesda, MD 20892-6510, USA*

Received on February 4, 2002; revised on May 9, 2002; accepted on May 21, 2002.

\*To whom correspondence should be addressed.



GO BACK

CLOSE FILE

# Abstract

**Motivation:** *As a first approximation, similarity between two long orthologous regions of genomes can be represented by a chain of local similarities. Within such a chain, pairs of successive similarities are collinear (non-conflicting), i.e. segments involved in the  $n$ th similarity precede in both sequences segments involved in the  $(n + 1)$ th similarity. However, when all similarities between two long sequences are considered, usually there are many conflicts between them. Although some conflicts can be avoided by masking transposons or low-complexity sequences, selecting only those similarities that reflect orthology and, thus, belong to the evolutionarily true chain is not trivial.*

**Results:** *We propose a simple, hierarchical algorithm of finding the true chain of local similarities. Starting from similarities with low  $P$ -values, we resolve each pairwise conflict by deleting a similarity with a higher  $P$ -value. This greedy approach constructs a chain of similarities faster than when a chain optimal with respect to some global criterion is sought, and makes more sense biologically.*

**Availability:** *A software tool OWEN based on the proposed approach is described in the accompanying note and is freely available at <ftp://ftp.ncbi.nih.gov/pub/kondrashov/owen>.*

**Contact:** [kondrashov@ncbi.nlm.nih.gov](mailto:kondrashov@ncbi.nlm.nih.gov)

Abstract

Introduction

Informal overview...

Formal background

Algorithms

Examples

Discussion

Acknowledgements

References



GO BACK

CLOSE FILE

**Supplementary information:** *Algorithm Chain and examples of chains of local similarities are available at <ftp://ftp.ncbi.nih.gov/pub/kondrashov/owen/extra>.*

**Abstract**

**Introduction**

**Informal overview...**

**Formal background**

**Algorithms**

**Examples**

**Discussion**

**Acknowledgements**

**References**



**GO BACK**

**CLOSE FILE**

# Introduction

Since all modern cells originated from the common ancestor ([Doolittle, 2000](#)), similarity can be found between every two genomes. However, the nature of this similarity depends strongly on the evolutionary distance. Gene order is poorly conserved between phylogenetically remote genomes, despite strong conservation of some orthologous genes ([Wolf \*et al.\*, 2001](#)). Comparing such genomes mostly means comparing unordered sets of protein sequences they encode.

In contrast, the order of orthologous genes is partially preserved between less distant genomes. In particular, regions of large-scale collinearity exist within all vertebrates ([Venkatesh \*et al.\*, 2000](#)) and all flowering plants ([Eckardt, 2001](#)). Thus, and since protein-coding exons constitute only a minority of the genomes of multicellular eukaryotes, these genomes must be compared by aligning their long, collinear regions ([Miller, 2001](#)). Finding such regions is an important problem ([Hannenhalli and Pevzner, 1999](#); [Zafar \*et al.\*, 2001](#))), which is not addressed here.

The degree of similarity between collinear regions of not-too-similar genomes is highly variable. Nearly-identical segments alternate with those possessing no meaningful similarity ([Jareborg \*et al.\*, 1999](#); [Shabalina and Kondrashov, 1999](#)). Thus, gene order is much more conservative than many nucleotide sites. Comparison of such genomes (e.g. of human and mouse) is better done in terms of sets of local similarities, and some regions should remain unaligned ([Schwartz \*et al.\*, 2000](#); [Miller, 2001](#)). Of course, for pairs of

Abstract

Introduction

Informal overview...

Formal background

Algorithms

Examples

Discussion

Acknowledgements

References



GO BACK

CLOSE FILE

very similar genomes, such as human and chimpanzee, global alignment makes perfect sense (e.g. [Kent and Zahler, 2000](#)).

Local similarities between orthologous segments of genome regions with large-scale collinearity are also mostly collinear (successive, non-conflicting), i.e. follow in the same order in both genomes ([Schwartz \*et al.\*, 2000](#)). In other words, macrocollinearity usually implies microcollinearity ([Rossberg \*et al.\*, 2001](#)) because the rate of divergence of rapidly evolving segments of genomes exceeds the rates of processes that disrupt microcollinearity, such as evolution due to duplications, inversions, transpositions, and convergence. Biologically, orthologous local similarities correspond either to units of function and selective constraint or, perhaps, to cold spots of mutation ([Shabalina \*et al.\*, 2001](#)). Still, when all local similarities between two long sequences are considered, usually there are numerous conflicts between them, although many conflicts involve rogue similarities between transposons and microsatellites, which can be recognized and masked ([Miller, 2001](#)).

Thus, although overall similarity between two macrocollinear genome regions can be mostly represented by the ‘evolutionarily true’ chain of microcollinear local similarities between their orthologous segments ([Schwartz \*et al.\*, 2000](#); [Shabalina \*et al.\*, 2001](#)), finding local similarities that belong to this chain is not trivial. We will concentrate on this task, and treat the procedure of finding individual local similarities (e.g. [Smith and Waterman, 1981](#); [Lipman and Pearson, 1985](#); [Altschul \*et al.\*, 1997](#); [Zhang \*et al.\*, 1998](#); [Arslan \*et al.\*, 2001](#)) as a parameter.

**Abstract**

**Introduction**

**Informal overview...**

**Formal background**

**Algorithms**

**Examples**

**Discussion**

**Acknowledgements**

**References**



**GO BACK**

**CLOSE FILE**

In order to find a chain of local similarities, one must resolve all conflicts by erasing some conflicting local similarities completely or, if this is enough to resolve a conflict, by trimming them. [Schwartz \*et al.\* \(2000\)](#) described two methods of finding the true chain, both of which seek the chain which is optimal as a whole, i.e. maximizes some global score. In this paper, we propose another approach, which does not seek the optimal chain. Instead, we resolve each pairwise conflict that needs to be resolved in favor of the stronger local similarity. The presentation will be in terms of pairwise comparison, and our software tool OWEN ([Ogurtsov \*et al.\*, 2002](#)) currently handles only two genomes, although multiple genomes can be compared in the same way.

**Abstract**

**Introduction**

**Informal overview...**

**Formal background**

**Algorithms**

**Examples**

**Discussion**

**Acknowledgements**

**References**



**GO BACK**

**CLOSE FILE**

# Informal overview of the approach

Our simple, hierarchical, greedy approach is motivated by an observation that the pattern of similarity between long collinear regions of moderately similar genomes is rather different from that between moderately similar relatively short DNA or protein sequences. In the second case, the degree of similarity is often rather uniform, and parts of the global alignment which are significant *per se* cover only a small fraction of sequences. Thus, we cannot proceed greedily and need to optimize some global scoring function of the whole alignment. Building blocks of alignments of uniformly similar sequences are matches of individual letters, and conflicts between them are ubiquitous.

In contrast, moderate genome-level similarity is patchy, and building blocks of genome alignments are local similarities of some lengths, many of them individually statistically significant. When a pair of significant local similarities is in conflict, it indicates that a microcollinearity-disrupting event did happen during evolution of the compared genomes from the common ancestor (Fig. 1). If so, two possibilities emerge.

First, microcollinearity could have been violated due to local convergent evolution or to insertion, into one genome, of a repetitive sequence that is also present, at a different location, in the other genome. In this case one of the conflicting similarities does not involve orthologous sequences, and the pattern of orthology can still be presented by a chain. Second, microcollinearity could have been violated due to local reshuffling of segments of one or both sequences (Fig. 1a) or to small-scale duplication(s) (Fig. 1b). In this case, similarities

Abstract

Introduction

Informal overview ...

Formal background

Algorithms

Examples

Discussion

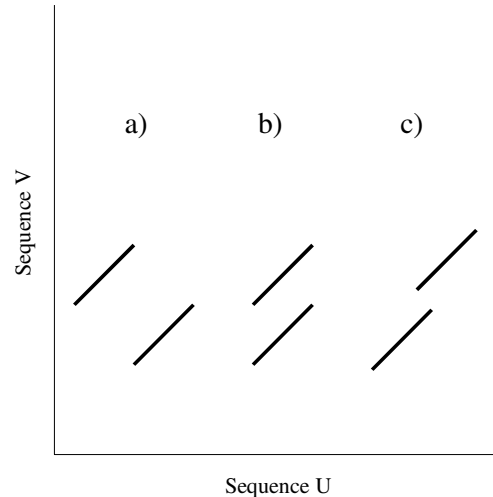
Acknowledgements

References



GO BACK

CLOSE FILE



**Fig. 1.** Types of conflicts between local similarities, shown as diagonals on the dot-matrix. (a) A severe conflict due to similar segments appearing in the opposite orders in the two sequences. (b) A severe conflict due to complete overlap of similar segments. (c) A mild conflict due to partial overlap of similar segments. Here, the conflicting similarities can be reconciled, by trimming their ends.

between all orthologous segments do not form a chain.

In the first case, convergent evolution of sequences rarely makes them profoundly similar, so that orthology after such evolution is probably reflected by the strongest of the conflicting similarities. Insertion of a repeat can lead to



a strong non-orthologous similarity, so it is better to mask repeats. However, even in this case the orthologous similarity can be stronger than any of several collinear non-orthologous similarities that conflict with it. If so, only individual resolution of conflicts between similarities will find the true chain (Fig. 2).

In the second case, it is hard to say which of the conflicting similarities between orthologs must be kept in the chain, and which are to be erased (and, perhaps, recorded as ‘footnotes’ to the chain). Keeping the strongest similarity obviously makes sense.

Thus, the simplest rule of always keeping a stronger similarity is justified. In this paper, ‘stronger’ will mean ‘having a lower  $P$ -value’, but OWEN also allows human intervention in resolving individual conflicts (Ogurtsov *et al.*, 2002).

We can now formulate two basic principles of our approach to finding the true chain of local similarities:

(1) All conflicts between a statistically significant similarity and any number of weaker similarities are resolved in favor of the former. Thus, a similarity that does not conflict with any stronger similarity is always included into the chain.

(2) Principle 1 holds both for the whole sequences to be compared and for any pair of their orthologous subsequences (‘fractality’). This is important because a similarity that is not significant when we compare two sequences of length  $10^7$  may become significant when the lengths are reduced (after other, stronger similarities have been found) to  $10^3$ . Thus, stronger similarities provide statistical support for those weaker similarities that do not conflict with them.

Abstract

Introduction

Informal overview ...

Formal background

Algorithms

Examples

Discussion

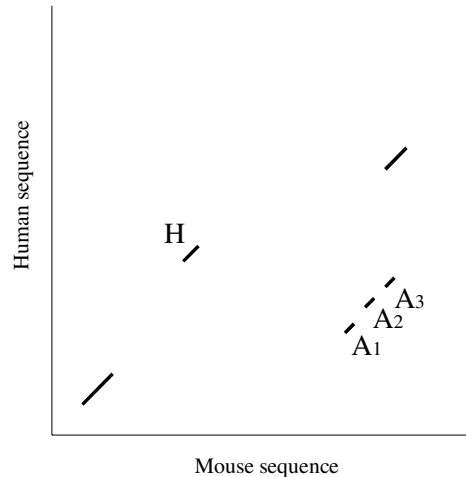
Acknowledgements

References



GO BACK

CLOSE FILE



**Fig. 2.** Comparison of mouse (AF139987) and human (AF045555) sequences. The strongest, orthologous similarity  $H$  corresponds to exon 10 at locus *Rfc2* (nucleotides 104514–104583 in the mouse sequence). Since non-orthologous similarities  $A_1$ ,  $A_2$ , and  $A_3$  constitute a chain with a score higher than that of a chain consisting of  $H$  alone, seeking the optimal as a whole chain using PipMaker (Schwartz *et al.*, 2000) or option Optimal in OWEN (Ogurtsov *et al.*, 2002) erases  $H$ . In contrast, resolving conflicts individually (option Greedy in OWEN) keeps  $H$  and erases  $A_1$ ,  $A_2$ , and  $A_3$ . Of course, if  $H$  were of secondary origin, and  $A_1$ ,  $A_2$ , and  $A_3$  were orthologous, only seeking the optimal as a whole chain would produce the correct result.

Abstract

Introduction

Informal overview ...

Formal background

Algorithms

Examples

Discussion

Acknowledgements

References



GO BACK

CLOSE FILE

We call the chain of local similarities that is found by applying these principles backbone chain and hope that it is close to the evolutionarily true chain that reflects orthology. We start assembling the backbone chain from the strongest similarity, then add to it the strongest similarity that does not conflict with the first one, etc. Algorithmically, resolving conflicts individually is a stone that kills two birds.

First, we can use a greedy algorithm to select the backbone chain from any set of conflicting similarities. Second, we can create this set hierarchically, i.e. start from finding only very strong similarities and resolve all conflicts between them, then independently screen gaps between successive strong similarities for weaker similarities, etc. Thus, time-consuming screening of the whole dot-matrix for all weak similarities can be avoided.

*Abstract*

*Introduction*

*Informal overview ...*

*Formal background*

*Algorithms*

*Examples*

*Discussion*

*Acknowledgements*

*References*



**GO BACK**

**CLOSE FILE**

# Formal background

Here we introduce terminology that is necessary to define the backbone chain of local similarities between sequences  $U$  and  $V$  and to describe algorithms that find it. The segment of  $U(V)$  starting at the position  $b$  and ending at the position  $e$  is denoted  $U[b, e](V[b, e])$ .

## Local similarities

A *similarity*  $H$  between  $U$  and  $V$  is a pair of segments  $U[b_1, e_1]$  and  $V[b_2, e_2]$  together with their *alignment*  $Al(H)$  and its *score*  $Score(H)$ . These segments are referred to as *U-domain* and *V-domain* of the similarity, denoted  $Domain(H, U)$  ( $Domain(H, V)$ ). The beginning and the end of the *U-domain* of  $H$  are  $Beg(H, U)$  and  $End(H, U)$ , respectively, and the analogous notations are used for the *V-domain*. We do not specify an algorithm of finding  $H$  and the corresponding alignment, or how a score is assigned to the alignment. We only assume that the score increases with the number of matches, and decreases with the number of mismatches and gaps within the alignment, so that alignments with higher scores are ‘better’. Later, some restrictions on alignments and their scores will be introduced.

Let  $H$  be a similarity between segments  $U[b_1, e_1]$  and  $V[b_2, e_2]$  and  $G$  be a similarity between  $U[c_1, f_1]$  and  $V[c_2, f_2]$ , where  $[c_1, f_1]$  is a subfragment of  $[b_1, e_1]$  and  $[c_2, f_2]$  is a subfragment of  $[b_2, e_2]$ . The similarity  $G$  is a *subsimilarity* of  $H$ , if  $Al(G)$  is a *subalignment* of  $Al(H)$ , i.e. if  $Al(G)$

Abstract

Introduction

Informal overview...

Formal background

Algorithms

Examples

Discussion

Acknowledgements

References



GO BACK

CLOSE FILE

establishes the same correspondence between letters from  $U[c_1, f_1]$  and  $V[c_2, f_2]$  as does  $Al(H)$ .

## Chains of local similarities

Similarity  $H_1$  *precedes* similarity  $H_2$  (notation:  $H_1 < H_2$ ) if the  $U$ -domain of  $H_1$  precedes the  $U$ -domain of  $H_2$  and the  $V$ -domain of  $H_1$  precedes the  $V$ -domain of  $H_2$ , i.e. if  $End(U, H_1) < Beg(U, H_2)$  and  $End(V, H_1) < Beg(V, H_2)$ . Similarities  $H_1$  and  $H_2$  are in *conflict*, if neither  $H_1$  precedes  $H_2$ , nor  $H_2$  precedes  $H_1$ . Two similarities that are not in conflict are *collinear*. A *chain of similarities* is a set of similarities  $\{H_1, H_2, \dots, H_N\}$  in which every two similarities are collinear, ordered according to the relation of precedence.

A similarity  $H$  is *collinear to the chain of similarities*  $B = \{H_1, H_2, \dots, H_N\}$ , if it is collinear to all members of  $B$ . If  $H$  is collinear to  $B$ ,  $H$  *follows*  $k$ th similarity of  $B$  (or  $H$  can be included between  $k$ th and  $(k + 1)$ th similarity of  $B$ ), where  $1 \leq k \leq N - 1$ , if  $H_k < H < H_{k+1}$ . For  $k = N$  this means that  $H_N < H$ , and for  $k = 0$  this means that  $H < H_1$ .

## Quality of a local similarity

The quality of a similarity can be characterized by its  $P$ -value (Durbin *et al.*, 1998; Mott, 2000). Informally,  $P$ -value of a similarity with score  $S$  within  $U$  and  $V$  is the probability that a pair of sequences with the same lengths and

Abstract

Introduction

Informal overview...

Formal background

Algorithms

Examples

Discussion

Acknowledgements

References



GO BACK

CLOSE FILE

statistical properties as  $U$  and  $V$  contains at least one similarity of score  $S$  or higher. Thus, highly significant similarities have low  $P$ -values.

Let  $P(S, L_1, L_2)$  be  $P$ -value of a similarity between sequences of lengths  $L_1$  and  $L_2$  with score  $S$ . For our purposes, the only important things are that  $0 \leq P \leq 1$  and that  $P(S, L_1, L_2)$  decreases with the increase of  $S$  and increases with  $L_1$  and  $L_2$  (informally,  $P$  ‘normalizes’ the score  $S$  by  $L_1$  and  $L_2$ ).

### *Reliability of similarities and their chains*

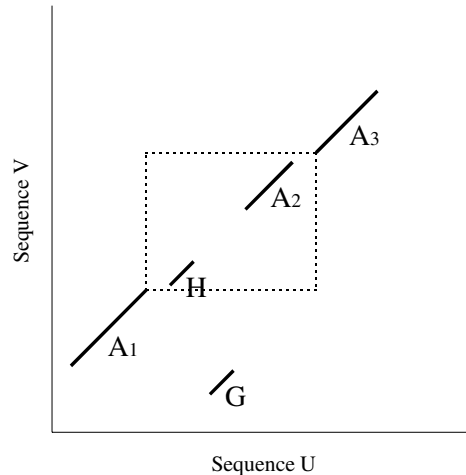
Let  $H$  be a similarity between the fragments  $U[c_1, f_1]$  and  $V[c_2, f_2]$  and  $\varepsilon$  be a number between 0 and 1. Consider fragments  $U[b_1, e_1]$  and  $V[b_2, e_2]$  containing  $U[c_1, f_1]$  and  $V[c_2, f_2]$  respectively, i.e.  $b_1 < c_1 < f_1 < e_1$  and  $b_2 < c_2 < f_2 < e_2$ . Similarity  $H$  is  $\varepsilon$ -reliable within  $U[b_1, e_1]$  and  $V[b_2, e_2]$  if  $P(\text{Score}(H), e_1 - b_1 + 1, e_2 - b_2 + 1) < \varepsilon$ .

Let  $F$  be a chain of similarities and  $H$  be a similarity from  $F$ . Consider a sub-chain  $F_H$  of  $F$  consisting of all similarities having scores higher than  $\text{Score}(H)$ . The similarities from  $F_H$  divide the compared sequences into a series of pairs of segments and the similarity  $H$  belongs to one of the pairs of segments, say,  $U[c_1, f_1]$  and  $V[c_2, f_2]$ .  $H$  is  $\varepsilon$ -reliable in  $F$  if  $H$  is  $\varepsilon$ -reliable within  $U[c_1, f_1]$  and  $V[c_2, f_2]$  (Fig. 3). A chain of similarities  $F$  is  $\varepsilon$ -reliable, if every similarity from  $F$  is  $\varepsilon$ -reliable in  $F$ .

- Abstract
- Introduction
- Informal overview...
- Formal background
- Algorithms
- Examples
- Discussion
- Acknowledgements
- References



⏪	⏩
◀	▶
GO BACK	
CLOSE FILE	



**Fig. 3.** The concept of  $\varepsilon$ -reliability of a similarity within a chain. Consider the chain  $F = \{A_1, H, A_2, A_3\}$  and suppose that  $\text{Score}(H) < \text{Score}(A_2) < \text{Score}(A_3) < \text{Score}(A_1)$ , so that  $F_H = \{A_1, A_2, A_3\}$ ,  $F_{A_2} = \{A_1, A_3\}$ ,  $F_{A_3} = \{A_1\}$ , and  $F_{A_1}$  is empty. Then, for example,  $A_2$  is  $\varepsilon$ -reliable in  $F$ , if it is  $\varepsilon$ -reliable in the marked region  $U[\text{End}(A_1, U) + 1, \text{Beg}(A_3, U) - 1] * V[\text{End}(A_1, V) + 1, \text{Beg}(A_3, V) - 1]$  between the similarities  $A_1$  and  $A_3$ .  $A_1$  is  $\varepsilon$ -reliable in  $F$ , if it is  $\varepsilon$ -reliable within the whole sequences  $U$  and  $V$ .

### Comparing sets of similarities

Let  $R$  be a set of similarities and  $\langle R \rangle$  be the vector of scores of all similarities from  $R$ , in decreasing order. Let  $\mathbf{r} = \langle r_1, r_2, \dots \rangle$  and  $\mathbf{s} = \langle s_1, s_2, \dots \rangle$  be

- Abstract
- Introduction
- Informal overview...
- Formal background**
- Algorithms
- Examples
- Discussion
- Acknowledgements
- References



⏪	⏩
◀	▶
GO BACK	
CLOSE FILE	

different vectors of scores (possibly, of different lengths). We will use the following lexicographic procedure to compare  $\mathbf{r}$  and  $\mathbf{s}$ . If for some  $k$   $r_k$  and  $s_k$  are different,  $\mathbf{r} > \mathbf{s}$  ( $\mathbf{r} < \mathbf{s}$ ) if  $r_k > s_k$  ( $r_k < s_k$ ) for the smallest  $k$  for which  $r_k \neq s_k$ . Otherwise, the longer vector is greater. Among two sets of similarities  $R$  and  $Q$ ,  $R$  is *stronger* than  $Q$  if  $\langle R \rangle > \langle Q \rangle$ .

## *Backbone chain*

The chain of similarities  $F$  is a *backbone chain* of a set of similarities  $R$  (with a given  $P$ -value cutoff  $\varepsilon$ ), if

- (a)  $F$  is  $\varepsilon$ -reliable,
- (b) every similarity from  $F$  belongs to  $R$  or is a subsimilarity of an element from  $R$ ,
- (c) no other chain of similarities that satisfies (a) and (b) is better than  $F$ .

Abstract

Introduction

Informal overview...

Formal background

Algorithms

Examples

Discussion

Acknowledgements

References



GO BACK

CLOSE FILE

# Algorithms

Here we describe algorithms that find the backbone chain of similarities for sequences  $U$  and  $V$  at the level of reliability  $\varepsilon$ . The backbone chain is assembled from a given set  $R$  of  $N$  similarities. We assume that we never have to choose between two conflicting similarities of exactly the same score ('different scores condition'). This assumption allows us to avoid algorithmically clumsy and biologically unimportant situations that can be resolved by a heuristic. We also assume that we possess an algorithm SetLocSim (**int** Beg $U$ , Beg $V$ , End $U$ , End $V$ , **real**  $\varepsilon$ ) that finds the set of all similarities with domains within  $U$ [Beg $U$ , End $U$ ] and  $V$ [Beg $V$ , End $V$ ], which are  $\varepsilon$ -reliable within this region.

We can proceed in two ways. First, we can find the backbone chain under assumption that all similarities from  $R$  have already been found. This is done by algorithm Chain. In the 'basic' case (the backbone chain contains only the initial similarities, and not their subsimilarities, Fig. 3) run-time of Chain is  $\sim N \cdot \log(T)$ , where  $T$  is a number of similarities in the backbone chain. In the general case when some similarities overlap (their  $U$ -domains and/or  $V$ -domains have non-empty intersection) the run-time depends on the number of overlapping similarities, but usually is  $\sim N \cdot \log(N)$ , if conflicts between overlapping similarities can be resolved by constructing their subsimilarities.

Alternatively, we can avoid finding all elements of  $R$  and proceed hierarchically. This is done by algorithm Fractal which generates (using SetLocSim) the necessary subsets of  $R$  and extracts, using a modification of Chain, from each of them the corresponding part of the backbone chain. In

Abstract

Introduction

Informal overview...

Formal background

Algorithms

Examples

Discussion

Acknowledgements

References



GO BACK

CLOSE FILE

the worst (and extremely improbable) case the run-time of Fractal is the same as that of Chain. Normally, Fractal finds the backbone chain in time  $c(\varepsilon) \cdot T$ , where  $c(\varepsilon)$  depends only on  $\varepsilon$ .

We start from describing Fractal, which is implemented in OWEN. After this, Chain will be described, first for the basic case and, finally, for the general case.

### *Algorithm fractal*

Let us define *S-restriction* of a set of similarities as its subset consisting of all similarities with scores  $S$  or higher. One can find  $S$ -restriction  $K_S$  of a backbone chain  $K$ , knowing only  $S$ -restriction  $R_S$  of  $R$ . Indeed, let  $K$  be the backbone chain of  $R$  with a  $P$ -value cut-off  $\varepsilon$ . Then, for an arbitrary  $S$ ,  $K_S$  coincides with the  $S$ -restriction of the backbone chain of  $R_S$  with the same  $\varepsilon$  ('greedy statement').

Fractal utilizes this statement in the following way. Let  $P(S, \text{length}(U), \text{length}(V)) = \varepsilon$ , so that  $S$  is the minimal score corresponding the  $P$ -values  $\varepsilon$  or lower within the whole sequences  $U$  and  $V$ . Fractal creates (using SetLocSim) the subset  $R_S$  of all similarities with the scores  $S$  or higher and constructs (using a modification of Chain) the  $S$ -restriction  $K_S$  of the backbone chain of  $R_S$ . Then, all elements of  $K_S$  belong to the final backbone chain for  $U$  and  $V$ . Thus, it is enough to process independently pairs of segments of  $U$  and  $V$  ('boxes') between successive similarities from  $K_S$ .

Fractal uses greedy paradigm twice. First, it hierarchically implements the greedy statement. Second, extraction of the backbone chain from the current

Abstract

Introduction

Informal overview...

Formal background

Algorithms

Examples

Discussion

Acknowledgements

References



GO BACK

CLOSE FILE

set of similarities is performed by greedy algorithm Chain. Let us define *box*  $\langle b_1, e_1, b_2, e_2 \rangle$  as a pair of segments  $U[b_1, e_1]$  and  $V[b_2, e_2]$  of sequences  $U$  and  $V$ . Fractal (Fig. 4) operates with two global objects:

- (i) the list of similarities BackboneChain (which finally contains the desired backbone chain) and
- (ii) the list of non-intersecting boxes WorkBoxList, which consists of all boxes to be processed.

We start with empty BackboneChain and with WorkBoxList, containing only the box corresponding to the whole initial sequences. To process the current box we first create the corresponding set of similarities CurrentLocSim (line 4, Fig. 4b) and then extract the  $S$ -restricted backbone chain CurrentBackboneChain from the set (line 7). The procedure Chain\_R (line 7) is described below. CurrentBackboneChain is not empty because the set CurrentLocSim is not empty. Then we include similarities from the CurrentBackboneChain into the BackboneChain (step 8) and add the boxes corresponding to the intervals between the elements of the CurrentBackboneChain to the WorkBoxList (lines 9 and 10). If CurrentBackboneChain is empty (there are no  $\varepsilon$ -reliable similarities within the current box), a global alignment for the box (lines 12–17) may be of interest.

The number of boxes to be processed is no more than  $2T + 1$ , where  $T$  is the number of similarities in the final backbone chain. Therefore, ProceedBox is called (line 8, Fig. 4a) no more than  $2T + 1$  times, and the run-time of all lines in Fig. 4a, except the line 8, is proportional to  $T$ .

Abstract

Introduction

Informal overview...

Formal background

Algorithms

Examples

Discussion

Acknowledgements

References



GO BACK

CLOSE FILE

```

(a)
ALGORITHM Fractal (sequence U, sequence V, real epsilon)
BEGIN
    // Prologue
1.  Mask transposones and low complexity regions in the sequences U and V;
2.  BackboneChain := empty;
3.  MainBox := <1, length(U), 1, length(V)>;
4.  WorkBoxList := {MainBox};
    // Main Loop
5.  WHILE WorkBoxList is not empty DO BEGIN
6.      CurrentBox := first (WorkBoxList);
7.      delete CurrentBox from WorkBoxList;
8.      ProceedBox (CurrentBox, epsilon);
9.  END WHILE
    // Epilogue
10. RETURN BackboneChain;
END ALGORITHM

(b)
ALGORITHM ProceedBox (box CurrentBox, real epsilon)
BEGIN
    // Prologue
    // Let CurrentBox = <b1, e1, b2, e2>, i. e. it
    // corresponds to the segments U[b1, e1] and V[b2, e2].
    // We create temporary sequences U_Temp and V_Temp to be
    // compared and calculate their lengths L_U and L_V.
1.  U_Temp = U[b1, e1];
    L_U = length(U_Temp);
2.  V_Temp = V[b1, e1];
    L_V = length(V_Temp);
    // Find the Score cut-off S, corresponding to the given P-value cut-off epsilon.
3.  S = min{Score_P (Score, L_U, L_V) ≤ epsilon};
    // Main Part
    // Construct a set CurrentSim of all local similarities within
    // U_Temp and V_Temp, having P-value less than a cut-off epsilon
4.  CurrentSim = SetLocalSim(b1, e1, b2, e2, epsilon);
5.  IF (CurrentSim is NOT empty) // the similarities do exist
6.  THEN
    // Extract the S-restricted backbone chain
    // CurrentBackboneChain for the set CurrentSim and
    // the P-value cut-off epsilon
7.  CurrentBackboneChain= Chain_R(CurrentSim, epsilon, S);
8.  include CurrentBackboneChain into Backbone_Chain;
9.  Construct the series of boxes within the CurrentBox, which are separated with
    the local similarities from the CurrentBackboneChain;
10. include the boxes into WorkBoxList;
11. ELSE
12. Construct a set AllWeakSim of all local similarities ;
    within U_Temp and V_Temp, having P-value less than a cut-off P_Weak,
    which is "weaker" (i.e. larger) than epsilon;
13. Create an optimal (i.e. having maximal total score) chain of local similarities
    from AllWeakSim;
14. IF the optimal chain is significant as a whole
15. THEN
16. include the chain into the WeakChainSet;
17. END IF
18. END IF
END ALGORITHM

```

**Fig. 4.** Algorithm Fractal (a) and its function ProceedBox (b).

Abstract

Introduction

Informal overview ...

Formal background

Algorithms

Examples

Discussion

Acknowledgements

References



GO BACK

CLOSE FILE



random  $\varepsilon$ -reliable similarities. As long as  $\varepsilon$  is small,  $M_r$  is low, and  $M_b$ , since orthologous similarities are collinear, cannot be very high and declines with  $\varepsilon$ . Thus, the value of  $M$  and, therefore, the run-time of ProceedBox is a function  $c(\varepsilon)$  and the total run-time of Fractal is  $c(\varepsilon) \cdot T$ .

## Overview of algorithms Chain\_Basic and Chain

Chain\_Basic and Chain find the backbone set of similarities by processing similarities from  $R$  one by one in the order of their decreasing scores. For a current similarity  $H$  they answer two questions: (1) should  $H$  be added to the already built part  $B = \{A_1, \dots, A_s\}$  of the desired backbone chain? and (2) if yes, after which member of  $B$  should  $H$  be included?

Chain\_Basic, applicable if similarities do not overlap, uses two global lists of similarities: (a) CurrentLocSim, which initially contains the provided set of similarities  $R$ ; and (b) BackboneChain, which is initially empty, and at the end contains the desired backbone chain. To perform the search efficiently, we also support on BackboneChain the structure of 2–3-tree (Aho *et al.*, 1974).

When  $\varepsilon$  is small, overlaps of similarities should be rare. Still, they do occur. Algorithm Chain that finds the backbone chain when overlaps may be present is a straightforward generalization of Chain\_Basic. Both algorithms can be seen at <ftp://ftp.ncbi.nih.gov/pub/kondrashov/owen/extra>.

Abstract

Introduction

Informal overview...

Formal background

Algorithms

Examples

Discussion

Acknowledgements

References



GO BACK

CLOSE FILE



# Discussion

We proposed a hierarchical, greedy approach to constructing chains of local similarities that describe overall correspondence between long, orthologous regions of moderately similar genomes. This approach is simple, efficient, and makes sense biologically.

Conceptually, resolving each essential pairwise conflict in favor of the better similarity is the simplest option. In contrast, the rationale behind the only reasonable alternative, seeking the optimal as a whole chain of similarities (Zhang *et al.*, 1994; Schwartz *et al.*, 2000), is obscure. Also, determining which chain is optimal requires assigning more or less arbitrary penalties for gaps between similarities.

Our algorithm Fractal is very efficient, due to two reasons. First, the run-time of creating the backbone chain for a set of  $N$  similarities is determined by the run-time of sorting it by scores. This can be done rapidly, in time  $\sim N \cdot \log(N)$  or, under some natural conditions on the range of the scores (Aho *et al.*, 1974), even in time  $\sim N \cdot \log \log(K)$  (where  $K$  is the highest score), by using priority queues (Johnson, 1982), stratified trees (van Emde Boas, 1977), or bounded ordered dictionaries (Melhorn and Nahler, 1990).

To find the optimal chain one has to use dynamic programming. Run-time of currently the most effective sparse dynamic programming (Eppstein *et al.*, 1992) depends on the data structure used to store the candidate points and can be  $\sim N \cdot \log(N)$  (Chao *et al.*, 1995) or even  $\sim N \cdot \log \log(L)$ , where  $L$  is the length of the shorter sequence (Eppstein *et al.*, 1992). However, the multiplicative

Abstract

Introduction

Informal overview...

Formal background

Algorithms

Examples

Discussion

Acknowledgements

References



GO BACK

CLOSE FILE

constant for sorting is smaller than for dynamic programming, since only one tree, instead of two is used and there is no need for extra operations (such as processing of intersections of boundaries between the candidates zones). In practice, both the backbone chain and the optimal chain can be found, from a provided set of similarities, very rapidly. OWEN (Ogurtsov *et al.*, 2002) supports both these options.

Second, and more importantly, there is no need to construct all similarities when conflicts are resolved individually. Indeed, if we compare two sequences of length  $10^7$  (typical length of fragments of collinearity preserved between human and mouse genomes), finding all similarities requires a prohibitively high run-time  $\sim 10^{14}$  with a high constant. However, strong similarities can be found rapidly, as long as we assume that they contain even relatively short runs of matches. Thus, we can start from using ‘core-based’ (BLAST-like, Altschul *et al.*, 1997) methods of finding local similarities, and perform exhaustive searches for weak similarities only within rectangles defined by strong similarities within the original  $10^7 \times 10^7$  dot-matrix. This speeds up comparison of long sequences enormously. In contrast, if the optimal chain is sought, the whole dot-matrix must be scanned for even the weakest similarities that can potentially be included into this chain.

Biologically, it makes sense to keep a stronger similarity regardless of its conflicts with any number of weaker similarities since a stronger similarity is likely to reflect orthology (Fig. 2). Of course, the backbone chain and the optimal chain may coincide, in particular, if similar sequences are compared.

Abstract

Introduction

Informal overview...

Formal background

Algorithms

Examples

Discussion

Acknowledgements

References



GO BACK

CLOSE FILE

In addition to relying exclusively on P-values, pairwise resolution of conflicts can also, as an option, be done manually. This makes it possible for an operator to use some hard-to-formalize clues for deciding which similarity to erase (and, perhaps, to store as a ‘footnote’ to the backbone chain, [Ogurtsov \*et al.\*, 2002](#)). As long as the decisions by the operator are transitive (i.e. it never happens that conflict between  $A_1$  and  $A_2$  is resolved in favor of  $A_1$ , conflict between  $A_2$  and  $A_3$  in favor of  $A_2$ , and conflict between  $A_1$  and  $A_3$  in favor of  $A_3$ ), human interventions does not lead to any modification of our algorithms. Such intervention, which are hardly possible if an optimal chain is sought, may be very useful in practical work.

[Abstract](#)

[Introduction](#)

[Informal overview...](#)

[Formal background](#)

[Algorithms](#)

[Examples](#)

[Discussion](#)

[Acknowledgements](#)

[References](#)



GO BACK

CLOSE FILE

# Acknowledgements

The authors thank four anonymous reviewers for many useful comments. Participation of M.A.R. in this and the accompanying paper was supported by INTAS grant 99-01476 by RFFI grant 00-04-48246 and by Netherland Organization for Scientific Research.

*Abstract*

*Introduction*

*Informal overview...*

*Formal background*

*Algorithms*

*Examples*

*Discussion*

*Acknowledgements*

*References*



GO BACK

CLOSE FILE

## References

- Aho,A.V., Hopcroft,J.E. and Ulman,J.D. (1974) *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA.
- Altschul,S.F., Madden,T.L., Schaffer,A.A., Zhang,J.H., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402. [MEDLINE Abstract](#)
- Arslan,A.N., Egecioglu,O. and Pevzner,P.A. (2001) A new approach to sequence comparison: normalized sequence alignment. *Bioinformatics*, **17**, 327–337. [MEDLINE Abstract](#)
- Bagheri-Fam,S., Ferraz,C., Demaille,J., Scherer,G. and Pfeifer,D. (2001) Comparative genomics of the SOX9 region in human and *Fugu rubripes*: conservation of short regulatory sequence elements within large intergenic regions. *Genomics*, **78**, 73–82. [MEDLINE Abstract](#)
- Chao,K.M., Zhang,J.H., Ostell,J. and Miller,W. (1995) A local alignment tool for very long DNA sequences. *Comput. Appl. Biosci.*, **11**, 147–153. [MEDLINE Abstract](#)
- Doolittle,W.F. (2000) The nature of the universal ancestor and the evolution of the proteome. *Curr. Opin. Struct. Biol.*, **10**, 355–358. [MEDLINE Abstract](#)
- Durbin,R., Eddy,D., Krogh,A. and Mitchison,G. (1998) Pairwise alignment. *Biological Sequence Analyses. Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, pp. 12–45.

[Abstract](#)

[Introduction](#)

[Informal overview...](#)

[Formal background](#)

[Algorithms](#)

[Examples](#)

[Discussion](#)

[Acknowledgements](#)

[References](#)



[GO BACK](#)

[CLOSE FILE](#)

- Eckardt,N.A. (2001) Everything in its place: conservation of gene order among distantly related plant species. *Plant Cell*, **13**, 723–725. [MEDLINE Abstract](#)
- van Emde Boas,P. (1977) Preserving order in a forest in Less than logarithmic time. *Inf. Proc. Lett.*, **6**, 80–82.
- Eppstein,D., Galil,Z., Giancarlo,R. and Italiano,G.F. (1992) Sparse dynamic programming I: linear cost functions. *J. ACM*, **39**, 519–545.
- Hannenhalli,S. and Pevzner,P.A. (1999) Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *J. ACM*, **46**, 1–27.
- Jareborg,N., Birney,E. and Durbin,R. (1999) Comparative analysis of noncoding regions of 77 orthologous mouse and human gene pairs. *Genome Res.*, **9**, 815–824. [MEDLINE Abstract](#)
- Johnson,D.B. (1982) A priority queue in which initialization and queue operations take  $O(\log \log D)$  time. *Math. Syst. Theory*, **15**, 295–309.
- Kent,W.J. and Zahler,A.M. (2000) Conservation, regulation, synteny, and introns in large-scale *C. briggsae*–*C. elegans* genomic alignment. *Genome Res.*, **10**, 1115–1125. [MEDLINE Abstract](#)
- Lipman,D.J. and Pearson,W.R. (1985) Rapid and sensitive protein similarity searches. *Science*, **227**, 1435–1441. [MEDLINE Abstract](#)
- Melhorn,K. and Nahler,S. (1990) Bounded ordered dictionaries in  $O(\log \log N)$  time and  $O(n)$  space. *Inf. Proc. Lett.*, **35**, 183–189.
- Miller,W. (2001) Comparison of genomic DNA sequences: solved and unsolved problems. *Bioinformatics*, **17**, 391–397. [MEDLINE Abstract](#)

[Abstract](#)

[Introduction](#)

[Informal overview...](#)

[Formal background](#)

[Algorithms](#)

[Examples](#)

[Discussion](#)

[Acknowledgements](#)

[References](#)



[GO BACK](#)

[CLOSE FILE](#)

- Mott,R. (2000) Accurate formula for  $p$ -values of gapped local sequence and profile alignments. *J. Mol. Biol.*, **300**, 649–659. [MEDLINE Abstract](#)
- Ogurtsov,A.Y., Roytberg,M.A., Shabalina,S.A. and Kondrashov,A.S. (2002) OWEN: aligning long collinear regions of genomes. *Bioinformatics*, **18**, submitted.
- Rossberg,M., Theres,K., Acarkan,A., Herrero,R., Schmitt,T., Schumacher,K., Schmitz,G. and Schmidt,R. (2001) Comparative sequence analysis reveals extensive microcolinearity in the Lateral suppressor regions of the tomato, *Arabidopsis*, and *Capsella* genomes. *Plant Cell*, **13**, 979–988. [MEDLINE Abstract](#)
- Schwartz,S., Zhang,Z., Frazer,K.A., Smit,A., Riemer,C., Bouck,J., Gibbs,R., Hardison,R. and Miller,W. (2000) PipMaker—a Web server for aligning two genomic DNA sequences. *Genome Res.*, **10**, 577–586. [MEDLINE Abstract](#)
- Shabalina,S.A. and Kondrashov,A.S. (1999) Pattern of selective constraint in *C. elegans* and *C. briggsae* genomes. *Genet. Res.*, **74**, 23–30. [MEDLINE Abstract](#)
- Shabalina,S.A., Ogurtsov,A.Y., Kondrashov,V.A. and Kondrashov,A.S. (2001) Selective constraint in intergenic regions of human and mouse genomes. *Trends Genet.*, **17**, 373–376. [MEDLINE Abstract](#)
- Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197. [MEDLINE Abstract](#)
- Venkatesh,B., Gilligan,P. and Brenner,S. (2000) Fugu: a compact vertebrate reference genome. *FEBS Lett.*, **476**, 3–7. [MEDLINE Abstract](#)

[Abstract](#)

[Introduction](#)

[Informal overview...](#)

[Formal background](#)

[Algorithms](#)

[Examples](#)

[Discussion](#)

[Acknowledgements](#)

[References](#)



GO BACK

CLOSE FILE

- Wolf,Y.I., Rogozin,I.B., Kondrashov,A.S. and Koonin,E.V. (2001) Genome alignment, evolution of prokaryotic genome organization, and prediction of gene function using genomic context. *Genome Res.*, **11**, 356–372. [MEDLINE Abstract](#)
- Zafar,N., Mazumder,R. and Seto,D. (2001) Comparisons of gene colinearity in genomes using GeneOrder2.0. *Trends. Biochem. Sci.*, **26**, 514–516. [MEDLINE Abstract](#)
- Zhang,Z., Berman,P. and Miller,W. (1998) Alignments without low-scoring regions. *J. Comput. Biol.*, **5**, 197–210. [MEDLINE Abstract](#)
- Zhang,Z., Raghavachari,B., Hardison,R.C. and Miller,W. (1994) Chaining multiple-alignment blocks. *J. Comput. Biol.*, **1**, 217–226. [MEDLINE Abstract](#)

[Abstract](#)

[Introduction](#)

[Informal overview...](#)

[Formal background](#)

[Algorithms](#)

[Examples](#)

[Discussion](#)

[Acknowledgements](#)

[References](#)



GO BACK

CLOSE FILE