The Footprint Sorting Problem

Claudia Fried[†,‡], Wim Hordijk[⊗], Sonja J. Prohaska[†,‡], Claus R. Stadler[§], and Peter F. Stadler[†,‡,*]

[†] Bioinformatics, Department of Computer Science, University of Leipzig, Germany; www.bioinf.uni-leipzig.de
[‡] Institute for Theoretical Chemistry and Structural Biology, University of Vienna, Austria
[⊗] Department of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand
[§] Department of Computer Science, University of Leipzig, Germany
[*] For correspondence: Tel: ++49 341 14951 20, Fax: ++49 341 14951 19,
Email: peter.stadler@bioinf.uni-leipzig.de

Jul 22 2003

Phylogenetic footprints are short pieces of non-coding DNA sequence in the vicinity of a gene that are conserved between evolutionary distant species. A seemingly simple problem is to sort footprints in their order along the genomes. It is complicated by the fact that not all footprints are co-linear: they may cross each other. The problem thus becomes the identification of the crossing footprints, the sorting of the remaining co-linear cliques, and finally the insertion of the non-colinear ones at "reasonable" positions. We show that solving the footprint sorting problem requires the solution of the "Minimum Weight Vertex Feedback Set Problem", which is known to be NP-complete and APX-hard. Nevertheless good approximations can be obtained for datasets of interest. The remaining steps of the sorting process are straight forward: computation of the transitive closure of an acyclic graph, linear extension of the resulting partial order, and finally sorting w.r.t. the linear extension. Alternatively, the footprint sorting problem can be rephrased as a combinatorial optimization problem for which approximate solutions can be obtained by means of general purpose heuristics. Footprint sortings obtained with different methods can be compared using a version of multiple sequence alignment that allows the identification of unambiguously ordered sub-lists.

1 INTRODUCTION

Phylogenetic Footprints are short pieces of non-coding DNA sequence in the vicinity of a gene that are conserved between evolutionary distant species [1]. Automatic procedures for phylogenetic footprinting such as `footprinter` [2] or `tracker` [3] can produce large amounts of data that require automatized analysis tools. A seemingly simple problem is to sort footprints in their order along the genomes. It is complicated by the fact that not all footprints are co-linear: they may cross each other. The problem thus becomes to identify the crossing footprints, to sort the remaining co-linear cliques, and finally to insert the non-colinear ones at "reasonable" positions. In this contribution we show that the footprint sorting problem is in fact a hard combinatorial optimization problem for realistic data and we describe an implementation that produces exact results in acceptable time for data sets of practical interest.

Mathematically speaking, we are given N intervals $\mathbb{X}^i$, $i = 1, \dots, N$ representing the DNA sequences. Let us denote by $[i; a, \ell]$ the sub-interval $[a, a+\ell-1] \subseteq \mathbb{X}^i$ where i identifies the DNA sequence, a is the initial position of the sub-interval, and ℓ is the length of the interval. A *footprint clique* J is a collection of sub-intervals with the property that $\alpha = [i; a, \ell] \in J$ and $\alpha' = [i'; a', \ell'] \in J$ implies either $\alpha = \alpha'$ or $i \neq i'$, i.e., a footprint clique contains at most one sub-interval from each sequence $\mathbb{X}^i$. The output of a footprinting program is a collection $\mathfrak{J}$ of M footprint cliques J_k, $k = 1, \dots, M$.

Since not all footprint cliques are of equal importance (or have been determined with equal certainty) it is useful to assign a weight $\omega : \mathfrak{J} \to [0, 1]$ to each footprint clique. For instance, one might use

$$\omega([i; a, \ell]) = \frac{\ell}{\max_j \ell_j} \qquad (1)$$

where the maximum is taken over all intervals in all footprint cliques. More sophisticated weight functions, that e.g. take the sequence conservation into account, might also be useful. For each subset $\mathfrak{I} \subseteq \mathfrak{J}$ we define the weight of the subset

$$\omega(\mathfrak{I}) = \sum_{\alpha \in \mathfrak{I}} \omega(\alpha) \qquad (2)$$

For two sub-intervals $\alpha = [i; a, l]$ and $\alpha' = [i; a', \ell']$ on the same sequence i we have the (trivial) order relation

$$\alpha < \alpha' \quad \Longleftrightarrow \quad \begin{cases} a < a' \\ \ell < \ell' \quad \text{and} \quad a = a' \end{cases} \qquad (3)$$

Clearly, the ordering (3) implies an order-like relation $\prec^*$ on $\mathfrak{J}$:

Definition 1 *For $J, J' \in \mathfrak{J}$ we set $J \preceq^* J'$ if and only if for all $i \in \{1, \dots, N\}$ for which there is an $\alpha = [i; a, \ell] \in J$ and an $\alpha' = [i; a', \ell'] \in J'$ we have $\alpha \leq \alpha'$.*

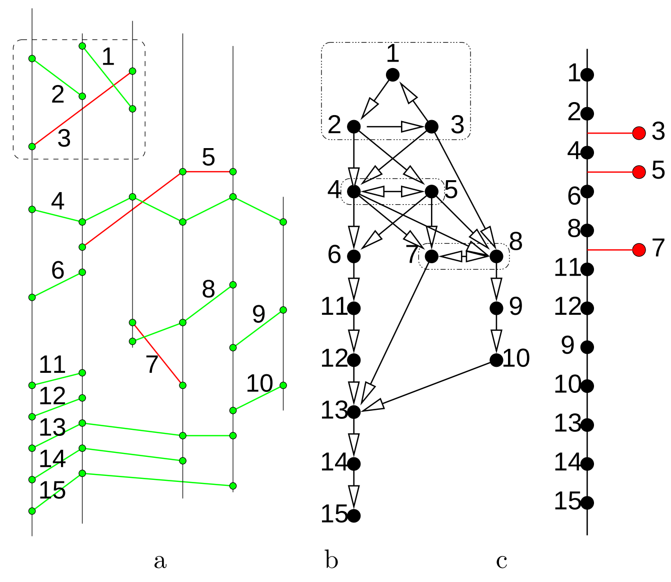

Figure 1: (a) Phylogenetic footprints (small balls) of different sequences (vertical lines) belonging to the common footprint clique are connected by lines. (b) Directed graph $\vec{G}$ describing their relative locations (not all arcs are shown for clarity). 2-connected components, i.e., obstructions to partial ordering are shown by boxes. (c) A well-ordering on a maximal set of co-linear cliques. The diagram also indicates the obstructing cliques at positions with a minimal number of conflicts.

Our task is hence to find a well-order on $\mathfrak{J}$ that is consistent with the order of the sub-intervals on each sequence, i.e., that is an extension of $\prec^*$ on $\mathfrak{J}$.

Recall that a relation $\leq$ on a set X is a partial order if the following three axioms are satisfied

(O1) $x \leq x$ for all $x \in X$ (reflexivity).

(O2) $x \leq y$ and $y \leq x$ implies $x = y$ (antisymmetry).

(O3) $x \leq y$ and $y \leq z$ implies $x \leq z$ (transitivity).

A partial order is totally ordered if in addition we have

(O4) $x \leq y$ or $y \leq x$ for all $x, y \in X$.

A total order consistent with any given partial order (a so-called *linear extension*) can be computed efficiently, see e.g. [4]. As a necessary condition, the transitive closure $\overline{\prec^*}$ of $\prec^*$ must therefore be a partial order. In general, however, this is not the case for realistic data. Consider the following two simple examples:
(1) $J = \{[1; a_1, \ell_1], [2; a_2, \ell_2], [3; a_3, \ell_3]\}$ and $J' = \{[1; a_1, \ell_1], [2; a_2, \ell_2], [4; a_4, \ell_4]\}$. In this case we have $J \preceq^* J'$ and $J' \preceq^* J$ but $J \neq J'$, i.e., antisymmetry (O2) is violated.
(2) $J_1 = \{[1; a_1, \ell_1], [2; a_2, \ell_2]\}$, $J_2 = \{[1; a_1', \ell_1'], [3; a_3, \ell_3]\}$, and $J_3 = \{[2; a_2', \ell_2'], [3; a_3', \ell_3']\}$, such that $a_1 < a_1'$, $a_2' < a_2$, $a_3 < a_3'$. This implies by definition $J_1 \preceq^* J_2$, $J_3 \preceq^* J_1$, and $J_2 \preceq^* J_3$. For the transitive closure, hence, we have $J_1 \overline{\preceq^*} J_3$ and $J_3 \preceq^* J_1$ but $J_1 \neq J_3$, again violating (O2).

The relation $\overline{\preceq^*}$ therefore is not antisymmetric in general. Our task therefore becomes to identify a maximal subset $\mathfrak{I} \subseteq \mathfrak{J}$ of footprint cliques that can be well-ordered. Maximality is defined conveniently w.r.t. some weight function such as equ.(2). The remaining footprint cliques that have to be removed from $\mathfrak{J}$ are those that are called "non-colinear" in [3]. We remark that in the case of just two sequences the maximum increasing subsequence algorithm [5, 12.5.1] can be used.

2 MINIMUM FEEDBACK-VERTEX SETS

The set $\mathfrak{J}$ can be regarded as the vertex set of a directed graph $\vec{G}$ with arcs $\alpha \to \beta$ if and only if $\alpha \preceq^* \beta$ and $\alpha \neq \beta$. The following result is obvious from the definition of a partial order:

Lemma 1 *The transitive closure of the relation* $\preceq^*$ *is a partial order if and only if the associated graph* $\vec{G}$ *is acyclic.*

Proof. The relation $\overline{\preceq^*}$ is anti-symmetric if and only if for any two vertices x and y with a direct path from x to y there is no directed path from y to x, i.e., no two vertices in $\vec{G}$ are contained in a circuit. $\square$

The problem of detecting non-colinear footprint cliques can therefore be rephrased as follows:
Given the vertex-weighted directed graph $\vec{G} = (V, A)$ with vertex set V and arc-set A, find a maximal (w.r.t. ω) subset of vertices $U \subseteq V$ such that the induced subgraph $\vec{G}[U]$ is acyclic.
In other words, we want to remove a set $W = V \setminus U$ of *non-colinear* footprints with minimal total weight. This problem is known as the *Minimum Feedback-Vertex Set Problem* [6], see [7] for a recent review. It is known to be NP-hard [8] and has applications in many diverse areas, including program verification [9] and Bayesian inference [10].

Unfortunately, the exact algorithm described in [11] is not fast enough for large examples with a larger number of non-colinear footprints. Well-tested heuristics based on a greedy randomized adaptive search procedure are available [12]. For the purpose of this study we have re-implemented both the exact and the heuristic algorithms in `C`.

3 TOPOLOGICAL SORTING

For the next step we have two options: (1) We may sort the acyclic subset U and then re-insert the feedback set W at appropriate positions. (2) Alternatively, we may modify $\vec{G} = (V, A)$ by removing some of the arcs incident with the feedback vertex set W in order to obtain an acyclic graph $G(V, A')$ with $A' \subset A$. Superficially, this suggest to consider the *feedback arc set problem*, i.e., to find a maximal subset $A' \subset A$ such that $\vec{G}' = (V, A')$ is acyclic. In the present context, however, we are interested in the set of non-colinear cliques W, while the feedback arcs $A \setminus A'$ do not seem to have an interpretation in terms of the phylogenetic footprints.

In both cases we compute the transitive closure of the acyclic graph by connecting two vertices with an arc from i to j if and only if there is a directed path from i to j. This can be achieved e.g. by Warshall's algorithm in $\mathcal{O}(|U|^3)$ time [13]. The resulting graph $\vec{G}^\bullet$ is again acyclic and represents a partial order.

The computation of a well-ordering of the vertex set of a directed acyclic graph $\vec{G}$ (such that arcs go only from vertices with smaller labels to vertices with larger labels) is known as *topological sorting* and can be solved in $\mathcal{O}(|A|)$ time [14, 15].

4 SORTING AS OPTIMIZATION PROBLEM

A completely different approach starts with the observation that the standard sorting problem can be reformulated as a combinatorial optimization problem. Let $(X,<)$ be a finite ordered set, which, without loosing generality, we can identify with the set $\{1,2,\dots,|X|\}$ endowed with the standard order on $\mathbb{N}$. Furthermore let $\mathbf{U}=(u_{ij})$ be a symmetric weight matrix with $u_{ij}=u_{ji}>0$. We use $u_{ij}=(\omega(i)+\omega(j))/2$ in terms of the footprint weights (1). Write $\pi(i)$ for the X-element at the i-th position in the sorted list and set

$$f_{ij}(\pi)=\begin{cases} u_{\pi(i)\pi(j)} & \text{if } i<j \text{ and } \pi(i)>\pi(j)\\ u_{\pi(i)\pi(j)} & \text{if } i>j \text{ and } \pi(i)<\pi(j)\\ 0 & \text{otherwise}\end{cases} \tag{4}$$

The quantity $f_{ij}(\pi)=0$ if $\pi(i)$ and $\pi(j)$ are correctly sorted with respect to each other in the particular ordering (permutation) π, while $f_{ij}(\pi)>0$ if $\pi(i)$ and $\pi(j)$ are missorted. Thus the total cost of a particular ordering π is conveniently defined as

$$f(\pi)=\sum_{ij} f_{ij}(\pi) \tag{5}$$

which can be interpreted as the total weight of all conflicting pairs. It can be interpreted as a weighted variant of Kendall's τ parameter [16].

The problem of sorting an n-element set (given that the comparison of two elements can be evaluated in constant time) is solved in $\mathcal{O}(n\log n)$ by standard algorithms such as `quick sort`, implemented e.g. in the `C` standard library function `qsort`. It is clear therefore, that a heuristic approach based on minimizing f will not be computationally efficient. In contrast to classical sorting algorithms, however, we can generalize the combinatorial optimization approach as we shall see below. Let us first consider the properties of equ.(4) for well-orders and partial orders, however.

Theorem 1 *If $(X,<)$ is well-ordered then every adaptive walk that makes use of the canonical transpositions reaches the correct sorting.*

Proof. If π is not the correct sorting then there is $m\in X$ such that $\pi(m)<\pi(m-1)$. It is convenient to define to $f_k(\pi)=\sum_i f_{ik}(\pi)$, the total weight of the conflicts of the k-th object in the list. Let π' be the ordering obtained by exchanging m and $m-1$, i.e, $\pi'(i)=\pi(i)$ for $i\neq m,m-1$, $\pi'(m)=\pi(m-1)$ and $\pi'(m-1)=\pi(m)$. One easily verifies that $f_k(\pi)=f_k(\pi')$ for $k\neq m-1,m$. Tedious but straightforward computation shows that $f_{m-1}(\pi')+f_m(\pi')-f_{m-1}(\pi)+f_m(\pi)=-2u_{\pi(m-1)\pi(m)}<0$, i.e, every canonical transposition that exchanges a mis-sorted pair of adjacent objects strictly decreases the cost function f. Therefore there is no local minimum of f with the exception of the correct sorting (where a mis-sorted object m does not exist by definition). □

It follows immediately that adaptive walks using arbitrary transpositions and/or reversals are also guaranteed to find the correct sorting because these movesets contain the canonical transpositions.

In this formulation the sorting problem can be extended in an obvious way to an arbitrary relation $\prec$ on X which need not be complete or even antisymmetric. Clearly, there is a perfect solution π satisfying $f(\pi)=0$ if and only if $(X,\prec)$ is a partially ordered set. A permutation thus satisfies $f(\pi)=0$ if and only if π codes for a linear extension of $\prec$.

Theorem 2 *If $(X,\prec)$ is a partially ordered set then every adaptive walk that makes use of the (general) transpositions reaches a correct topological sorting.*

Proof. The argument in the proof above can be modified in the following form in the case of partial orders. We first observe that if π is not a linear extension of $\prec$ then there are $m,m'\in X$ such that $\pi(m)\prec\pi(m')$ while $\pi(k)$ is incomparable with $\pi(m)$ and $\pi(m')$ for all $m<k<m'$. In the extreme case $m\neq m'-1$. Now consider the permutation π' defined by $\pi'(m)=\pi(m')$, $\pi'(m)=\pi(m')$, and $\pi'(j)=\pi(j)$ for $j\neq m,m'$. Since the positions k between m and m' are uncomparable with both m and m' we have $f_j(\pi)=f_j(\pi')$ for all $j<m$, $j>m'$, and $m<j<m'$. Furthermore $f_m(\pi')-f_m(\pi)=-u_{\pi(m)\pi(m')}<0$ since these terms differ only by the contribution from comparing $\pi(m)$ and $\pi(m')$. By the same argument $f_{m'}(\pi')-f_{m'}(\pi)=-u_{\pi(m')\pi(m)}<0$. Thus $f(\pi')<f(\pi)$. □

Let us now turn to the general case where we are given a directed graph $\vec{G}=(V,A)$ that is not acyclic in general. Let $\mathbf{U}=(u_{ij})$ be a symmetric weight matrix satisfying $u_{ij}=u_{ji}>0$ whenever $(i,j)\in A$ or $(j,i)\in A$ is an arc in $\vec{G}$. Furthermore we define the cost function f as

$$f_{ij}(\pi)=\begin{cases} u_{\pi(i)\pi(j)} & \text{if } i<j \text{ and } (\pi(j),\pi(i))\in A\\ u_{\pi(i)\pi(j)} & \text{if } i>j \text{ and } (\pi(i),\pi(j))\in A\\ 0 & \text{otherwise}\end{cases} \tag{6}$$

Equ.(6) reduces to (4) provided $(x,y)\in A$ if and only if $x<y$ and $<$ is a partial order. We can hope to obtain a useful ordering π of the nodes of $\vec{G}$ by minimizing f. We do not know of an exact solution to this problem. It would be easy to use adaptive walks or other simple local search algorithms such as simulated annealing [17], however. Given the two theorems above, it appears that transpositions, and possibly also reversals of permutations, will be a good move set at least as long as the graph is nearly acyclic.

Given the permutation π that represents our best approximation to the true ordering of the vertices, we would also like to identify a minimum feedback vertex set. It is not clear whether this can be done exactly. A heuristic approximation proceeds by iteratively removing the vertex k with largest total weight

$$g_k(\pi) = \sum_i \begin{cases} \omega(i) & \text{if } k < i \text{ and } (\pi(i), \pi(k)) \in A \\ \omega(i) & \text{if } k > i \text{ and } (\pi(k), \pi(i)) \in A \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

of all those objects with which it conflicts in the given ordering π.

More formally we have the following algorithm:

```
Input: $\vec{G} = (V, A)$, $\mathbf{U}$, $\pi$
Output: $W$ /* feedback vertex set */
1: $W \leftarrow \emptyset$
2: while $f(\pi) > 0$ do
3:     $\ell := \arg\max g_k(\pi)$.
4:     $W \leftarrow W \cup \{\pi(\ell)\}$.
5:     $u_{\pi(\ell)i}, u_{i\pi(\ell)} \leftarrow 0$ for all $i$.
```

Clearly, this procedure produces a feedback vertex set; its minimality is not guaranteed even if π minimizes f.

5 COMPARISON OF SORTED LISTS

In order to assess the quality of the orderings π obtained from the different approaches and variants described above we need a systematic way of comparing ordered lists. Since these lists are represented as permutations, π' and π'', of the same set n of objects, it seems natural to use distance measures $d(\sigma, \tau)$ on the symmetric group S_n. Natural metrics on S_n are associated with a length function that measures how much a group element is "different" from the identity element: $d(\sigma, \tau) = L(\sigma \circ \tau^{-1})$ [20]. For instance, the minimum number of transpositions that are necessary to generate π from the identity permutation satisfies $L_t(\pi) = n - \text{cyc}(\pi)$, where $\text{cyc}(\pi)$ is the number of cycles in the cycle representation of π. Length functions associated with sorting by canonical transpositions (in terms of so-called inversions) and reversals can also be computed [21].

Another possibility, which allows a more convenient comparison of the two lists, is to *align* the two permutations π' and π'' of n elements such that the number D of insertions and deletions is minimized, Fig. 2. This can be achieved by a simple dynamic programming scheme orginally invented by Needleman & Wunsch [22] for the alignment of two protein sequences. Starting from the initialization $D_{0i} = D_{i0} = i$ we have to compute

$$D_{ij} = \min \begin{cases} D_{i,j-1} + 1 \\ D_{i-1,j} + 1 \\ D_{i-1,j-1} & \text{whenever } \pi'(i) = \pi''(j) \end{cases} \tag{8}$$

The optimal number of insertions and deletions is then given by $D = D_{nn}$. The alignment of the orderings is then reconstructed from matrix (D_{ij}) by a standard backtracking procedure. Alignments of more than two lists can be computed by means of the iterative procedure familiar from popular multiple sequence alignment programs such as `clustalw` [23]. We note in passing that such list alignments can be used as an alternative approach to the Top-k List Comparison problem discussed in [24].

Figure 3 shows an example of a manually sorted footprint list in comparison with the exact solution of the MFVS-based procedure and the optimization heuristic described above. These data are fairly noisy containing a significant number of non-colinear footprints. The resulting sortings and assignments of non-colinearities are rather different: the manual list differs by 16 and 21 indels from the exact and the optimization results, the two automatic sortings differ by 26 indels, i.e., the manually sorted list is much closer to the exact MFVS-based listing than to the result of the simple optimization procedure. This is not surprising since we cannot expect adaptive walks to produce good results in problems with a large number of order conflicts. More sophisticated heuristics such as simulated annealing [25] or genetic algorithms [26] will have to be used for such data sets.

6 AN APPLICATION

For some applications a properly sorted list of phylogenetic footprints is not only a convenience but a necessary prerequisite for further data analysis. Fortunately, many datasets, in particular those with fewer input sequences, are much less noisy then the one shown in Fig. 3 so that non-colinear cliques can be identified (almost) unambiguously. In this section we briefly consider an example of an application where unambiguity is important.

The total size of a genome varies over evolutionary time scales. These variation can be caused by large scale changes such as gene and chromosome duplications and by the accumulation of small-scale, local insertion, deletions, and inversions. Length variations of the non-conserved sequences between *adjacent* homologous footprints in different sequences can be used to estimate the local changes. Clearly, this requires a reliable method for excluding non-colinear (and hence non-homologous) footprints and a proper ordering of the footprints so that adjacency can be determined.

Recent analysis showed that chromosome 10 of the rat differs by several chromosome rearrangements from both the mouse and the human homologs [27], see also [28]. Here we report on the evaluation of the insertion/deletion pattern in a sample consisting of immune genes (with all intron sequences and with 5000nt of flanking sequence on each side) from human, mouse, and rat.

Let ℓ_j^h and ℓ_j^k be the lengths of the intervening sequences between two adjacent footprints that are common to the input sequences h and k. Given a reference sequence, say human h, we are interested in the differences between the distributions of $\alpha_j^k = \log \ell_j^k / \ell_j^h$ for different species k, say mouse and rat. We assume that the total length of insertions and deletions will be approximately proportional to the length ℓ_j^h of the piece of the reference sequence, hence we consider the length ratio rather than the length difference. (Depending on the mechanism that one envisages for the insertion and deletion processes a different scaling might be more appropriate, however.) Here

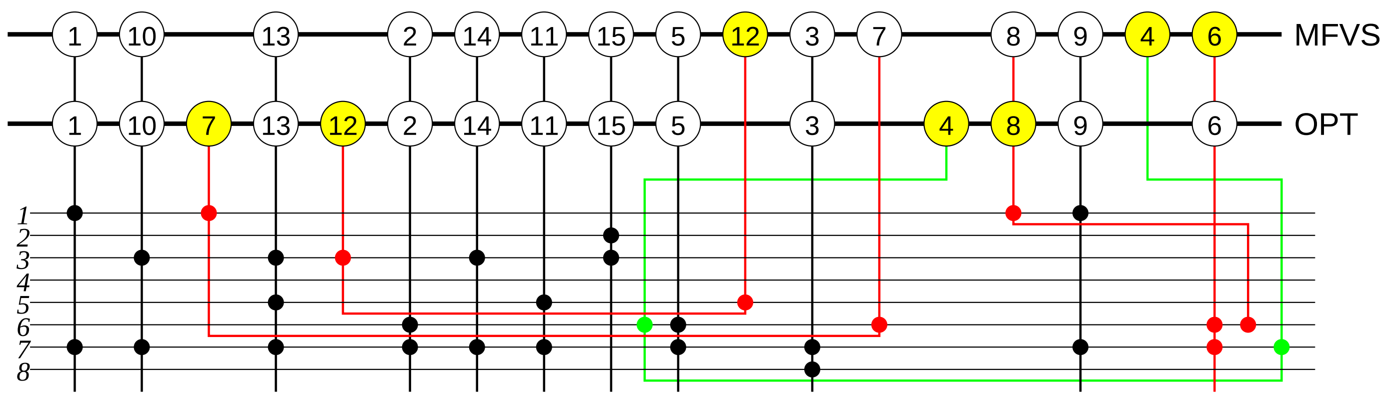

Figure 2: Alignment of two permutations. We use here the first 15 cliques from the comparison of eight *HoxA* clusters of various vertebrates discussed in [18]. The two rows on the top display the alignment of the exact ordering derived from an exact solution of the minimum feedback vertex set problem and the ordering obtained by optimizing equ.(6). The feedback vertices are indicated in grey. Below the positions of the underlying footprints on the 8 nucleotide sequences are shown.

Sorting				Footprints												
manual	MFVS	opt.	Exon	PmW	HsA	HsC	HsD	HfM	HfN	TrAa	TrAb	TrD	TrCa	Ci	TrBa	HsB
1	1	1 +		•	•	•	•	•	•	•	•	•				
2	2	2	*	•	•	•	•	•	•	•	•	•				
3	3	3										•	•			
—	—	5 +				•										•
—	—	12 +			•										•	
4	4	4		•									•			
5	5	—				•										•
6 +	6	6 +		•		•										•
7	7	7 +		•		•										•
—	12	—			•										•	
8	—	—		•		•										•
9	9	9 +		•												•
—	—	13 +				•	•									
10	10	10		•					•							
11	11	11 +		•												•
—	8 +	8 +		•		•										•
12	—	—			•										•	
13	13	—				•	•									
14 +	14 +	—		•												•
15	15	15	*	•	•	•	•	•	•	•	•	•	•	•		
—	30 +	—			•	•		•							•	
16 +	16 +	16 +		•		•										•
—	26 +	—		•											•	
—	40 +	—				•										•
17	17	17							•						•	
18	—	18					•					•				
19 +	19 +	—							•				•			
20	20	—							•							•
—	18	—					•					•				
—	31 +	—						•								•
21 +	—	—									•				•	•
22 +	—	—							•	•						
23	—	—							•						•	
24 +	—	—							•						•	
25	25 +	—		•					•							
—	32	—			•			•								
26 +	—	26 +		•											•	
27	27	27 +		•										•		
28	28	—		•												•
—	21	—									•				•	•
—	36 +	—				•										•
29	29	—		•										•		

Sorting				Footprints												
manual	MFVS	opt.	Exon	PmW	HsA	HsC	HsD	HfM	HfN	TrAa	TrAb	TrD	TrCa	Ci	TrBa	HsB
—	23	—							•						•	
30 +	—	30 +			•	•		•							•	
31	—	—						•								•
32	—	32			•			•								
—	—	40 +				•										•
—	—	20 +							•							•
33	33	33			•			•								
—	—	28 +		•												•
—	—	29 +		•										•		
34	34	34		•	•			•		•						
35	35	35 +						•								•
—	—	14 +		•												•
—	—	21 +									•				•	•
—	—	31 +						•								•
36	—	36 +				•										•
—	—	44 +								•	•					
—	—	47 +			•			•		•	•					
—	—	45 +								•	•					
—	—	22 +							•	•						
—	—	23 +							•						•	
37	37	37 +		•											•	
—	—	24 +							•						•	
38	38	38		•												•
—	—	25 +		•					•							
—	41	—											•			•
39	39	39 +		•											•	
40 +	—	—				•										•
41	—	41 +											•			•
42	42	42 +				•							•			
—	—	48 +			•											•
—	—	19 +							•				•			
43 =	43 +	43 +		•	•	•		•		•			•		•	•
—	22	—							•	•						
—	24	—							•						•	
44	44	—								•	•					
45	—	—								•	•					
46 =	—	46 +												•	•	•
47	47	—			•			•		•	•					
48 +	48	—			•											•
—	46 +	—												•	•	•
—	45	—								•	•					
49	49	49			•		•									

Figure 3: Some footprint datasets are quite noisy and contain many non-colinear entries. The example shown here is the Hox-10 region from the lamprey *Petromyzon marinus* (Pm), human (Hs), the hornshark *Hetrodontus francisci* (Hf), the pufferfish *Takifugu rubripes* (Tr), and the tunicate *Ciona intestinalis* (Ci). The first column gives the manual sorting based on the raw `tracker` output [19], the second column is the sorting based on solving the MFVS problem exactly, and the third column was obtained using the optimization approach. The + signs indicate footprint cliques that are identified as non-colinear; one match was removed from two footprints in [19] after visual inspection, indicated by a = sign. The two exons of the Hox-10 gene itself are indicated by a ∗. Footprints are marked by •.

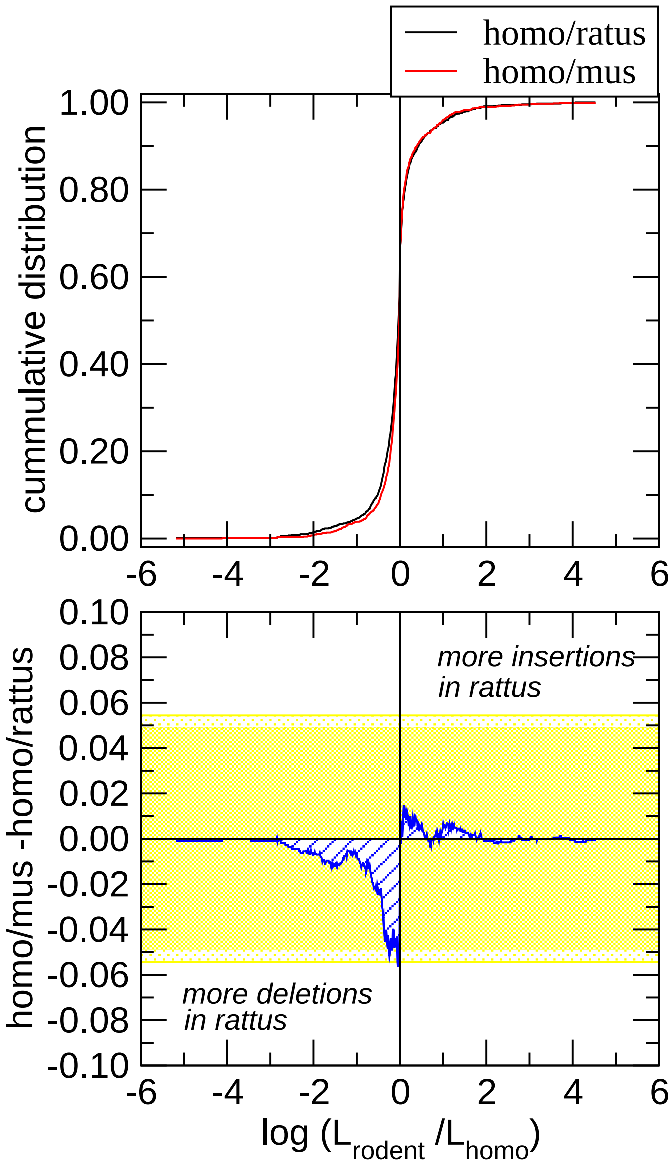

Figure 4: Comparison of length rations of $N_1 = 1122$ human/rat and $N_2 = 1404$ human/mouse non-conserved sequence fragment located between homologous footprints in the vicinity of various immune genes. The maximum difference between the two cummulative probability density functions is $D^* = 0.05659$, compared to the threshold value $D_{95\%} = 0.05438$ for the Kolmogorov-Smirnov test with effective sample size $N_{\text{eff}} = N_1N_2/(N_1 + N_2) \approx 623.6$. The two distributions are therefore (just barely) significantly different.

we use the logarithm of the length ratio because it produces a distribution that is symmetric if there is no bias between insertions and deletions.

Figure 4 summarizes the results. The effect of deletions and insertions appears to be slighly increased in the rat compared to the mouse. The difference is very small however: the Kolmogorov-Smirnov test just barely yields a 95% confidence for a significant difference of the distribution of rat/man and mouse/man sequence length ratios.

7 DISCUSSION

We have shown here that the seemingly trivial task of sorting a list of phylogenetic footprints properly by their location along the sequences in fact gives rise to a complex optimization problem. As a chemical application one might for instance consider the problem of sorting a list of samples of complex mixtures according to their compositions in the presence of missing data, i.e., when not all components are measured in all samples. In this case, simpler approaches such as lexicographic sorting cannot be applied.

Here we have described two approaches to solving this task. An exact algorithm that requires the solution of the NP-complete minimum feedback vertex set problem and a formulation as a combinatorial optimization problem that essentially generalizes a "landscape version" of bubble sort.

The "Footprint Sorting Problem" can also be viewed abstractly as a multi-objective optimization problem. Given a directed graph $\vec{G}(V, A)$, a vertex weight function $\omega : V \to \mathbb{R}^+$, and an edge weight function $u : A\mathbb{R}^+$ find a permutation $\pi : V \to V$ and a "feedback set" $W \subset V$ such that both the weight $\sum_{i\in W} \omega(i)$ of W and weight total of conflicts

$$u(V \setminus W) = \sum_{i,j\in V\setminus W} f_{ij}(\pi)$$

among the remaining vertices is minimized.

This rather general version of a topological sorting problem arises in many different contexts. For example we might want to sort the results from different database queries for the same topic. In this case u_{ij} is e.g. confidence or score-difference with which a particular database ranks the results i better than j and $\omega(i)$ measures how much information the result contains. The goal would be to rank the results as good as possible in accordance with rankings from the individual queries and to focus on the most detailed results.

ACKNOWLEDGEMENT

Financial support by the German DFG Bioinformatics Initiative (CF, SJP, PFS) and the Allan Wilson Centre for Molecular Ecology and Evolution in New Zealand (WH) is gratefully acknowledged.

REFERENCES

(1) Tagle, D. A.; Koop, B. F.; Goodman, M.; Slightom, J. L.; Hess, D. L.; Jones, R. T., Embryonic epsilon and gamma globin genes of a prosimian primate (Galago crassicaudatus). Nucleotide and amino acid sequences, developmental regulation and phylogenetic footprints. *J. Mol. Biol.* **1988**, *203*, 439–455.

(2) Blanchette, M.; Schwikowski, B.; Tompa, M., Algorithms for Phylogenetic Footprinting. *J. Comp. Biol.* **2002**, *9*, 211–223.

(3) Prohaska, S.; Fried, C.; Flamm, C.; Wagner, G.; Stadler, P. F., Surveying Phylogenetic Footprints in Large Gene Clusters: Applications to Hox Cluster Duplications. *Mol. Phylog. Evol.* **2003**, submitted; SFI preprint #03-02-011.

(4) Preusse, G.; Ruskey, F., Generating Linear Extensions Fast. *SIAM J. Comput.* **1994**, *23*, 373–386.

(5) Gusfield, D., *Algorithms on Strings, Trees, and Sequences*. Cambridge, UK: Cambridge University Press, **1997**.

(6) Lempel, A.; Cederbaum, I., Minimum feedback arc and vertex sets of a directed graph. *IEEE Trans. Circuit Theory* **1966**, *13*, 399–403.

(7) Festa, P.; Pardalos, P.; Resende, M. G. C., Feedback set problems. In: *Encyclopedia of Optimization*, vol. 2, Kluwer, **2001** 94–106.

(8) Garey, M. R.; Johnson, D. S., *Computers and Intractability. A Guide to the Theory of NP-Completeness*. San Francisco: Freeman, **1979**.

(9) Shamir, A., A linear time algorithm for finding minimum cutsets in reduced graphs. *SIAM J. Comput.* **1979**, *8*, 654–655.

(10) Yehuda, B.; Geiger, D.; Naor, J.; Roth, R. M., Approximation algorithms for the vertex feedback set problem with applications to constraint satisfaction and Bayesian inference. In: *Proc. 5th Annual ACM-SIAM Symp. on Discrete Algorithms*, **1994** 344–354.

(11) Ashar, P.; Malik, S., Implicit Computation of Minimum-Cost Feedback Vertex Sets for Partial Scan and Other Applications. In: *Proc. 31st Annual Design Automation Conf. (DAC)*, ACM Press, **1994** 77–80.

(12) Festa, P.; Pardalos, P.; Resende, M. G. C., Algorithm 815: Fortran subroutines for computing approximate solutions of feedback set problems using GRASP. *ACM Transactions Math. Software* **2001**, *27*, 456–464.

(13) Warshall, S., A theorem on boolean matrices. *J. ACM* **1962**, *9*, 11–12.

(14) Toda, S., On the complexity of topological sorting. *Inf. Process. Lett.* **1990**, *35*, 229–233.

(15) Hagerup, T.; Maas, M., Generalized topological sorting in linear time. *Nord. J. Comput.* **1994**, *1*, 38–49.

(16) Kendall, M. G., A new measure of rank correlation. *Biometrika* **1938**, *30*, 81–93.

(17) Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P., Optimization by simulated annealing. *Science* **1983**, *220*, 671–680.

(18) Chiu, C.-H.; Dewar, K.; Wagner, G. P.; Takahashi, K.; Ruddle, F. H.; Ledje, C.; Bartsch, P.; Scemama, J.-L.; Stellwag, E.; Fried, C.; Prohaska, S. J.; Stadler, P. F.; Amemiya, C. T., Bichir *HoxA* cluster sequence reveals surprising trends in ray-finned fish genomic evolution. *Genome Res.* **2003**, submitted.

(19) Fried, C.; Prohaska, S. J.; Stadler, P. F., Independent Hox-Cluster Duplications in Lampreys. *J. Exp. Zool. MDE* **2003**, in press.

(20) Lyndon, R. C., Length Functions in Groups. *Math. Scand.* **1963**, *12*, 209–234.

(21) Kececioglu, J. D.; Sankoff, D., Exact and Approximation Algorithms for Sorting by Reversals, with Application to Genome Rearrangement. *Algorithmica* **1995**, *13*, 180–210.

(22) Needleman, S. B.; Wunsch, C. D., A general method applicable to the search for similarities in the aminoacid sequences of two proteins. *J. Mol. Biol.* **1970**, *48*, 443–452.

(23) Thompson, J. D.; Higgs, D. G.; Gibson, T. J., CLUSTALW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties, and weight matrix choice. *Nucl. Acids Res.* **1994**, *22*, 4673–4680.

(24) Fagin, R.; Kumar, R.; Sivakumar, D., Comparing Top-k Lists. *SIAM J. Discr. Math.* **2003**, to appear.

(25) Aarts, E.; Korst, J., *Simulated Annealing and Boltzman Machines*. New York: J. Wiley & Sons, **1990**.

(26) Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading (Mass.): Addison Wesley, **1989**.

(27) Behboudi, A.; Sjostrand, E.; Gomez-Fabre, P.; Sjoling, A.; Taib, Z.; Klinga-Levan, K.; Stahl, F.; Levan, G., Evolutionary aspects of the genomic organization of rat chromosome 10. *Cytogenet. Genome Res.* **2002**, *96*, 52–59.

(28) Millwood, I. Y.; Bihoreau, M. T.; Gauguier, D.; Hyne, G.; Levy, E. R.; Kreutz, R.; Lathrop, G. M.; Monaco, A. P., A gene-based genetic linkage and comparative map of the rat X chromosome. *Genomics* **1997**, *40*, 253–261.