Universität Bielefeld

Technische Fakultät
Abteilung Informationstechnik
Forschungsberichte

On the Similarity of Sets of Permutations and its Applications to Genome Comparison

Anne Bergeron Jens Stoye

Report 2003-01

Impressum: Herausgeber:
Robert Giegerich, Ralf Hofestädt, Peter Ladkin, Helge Ritter,
Gerhard Sagerer, Jens Stoye, Ipke Wachsmuth

Technische Fakultät der Universität Bielefeld,
Abteilung Informationstechnik, Postfach 10 01 31,
33501 Bielefeld, Germany

ISSN 0946-7831

On the Similarity of Sets of Permutations and its Applications to Genome Comparison

Anne Bergeron
LaCIM
Université du Québec à Montréal
Canada

Jens Stoye
Technische Fakultät
Universität Bielefeld
Germany

Abstract

The comparison of genomes with the same gene content relies on our ability to compare permutations, either by measuring how much they differ, or by measuring how much they are alike. With the notable exception of the breakpoint distance, which is based on the concept of conserved adjacencies, measures of distance do not generalize easily to sets of more than two permutations. In this paper, we present a basic unifying notion, *conserved intervals*, as a powerful generalization of adjacencies, and as a key feature of genome rearrangement theories. We also show that sets of conserved intervals have elegant nesting and chaining properties that allow the development of compact graphic representations, and linear time algorithms to manipulate them.

1 Introduction

Gene order analysis in a set of organisms is a powerful technique for phylogenetic inference. Current methods are based on notions of *distances* between genomes, which are usually defined as the minimum number of such and such operations needed to transform one genome into the other one. Distance matrices can either be used directly as data for phylogenetic reconstruction, or in more qualitative attempts to reconstruct ancestral genomes [8]. All these methods, with the notable exception of the *breakpoint distance* [5], are closely tied to initial choices of allowable rearrangement operations. They are also *pure* distances, in the sense that similarities between genomes are purposefully ignored.

The breakpoint distance is based on the notion of conserved adjacencies. Compared to other distances, it is easy to compute, but it often fails to capture more global relations between genomes [16]. Nevertheless, conserved adjacencies have two highly desirable properties:

1. They can be defined on a set of more than two genomes, allowing for the identification of similar features in a family of organisms.

2. They are invariant under optimal rearrangement scenarios, in the sense that it is not necessary to break adjacencies to explain how a genome evolved from another one [9, 14, 20].

A first generalization of adjacencies is the notion of *common intervals* that identify subsets of genes that appear consecutively in two or more genomes [12, 21]. Common intervals identify more global relations between genomes, but often lose the invariant property of adjacencies with respect to optimal rearrangement scenarios. For example, all optimal sortings by reversals of the permutation $(1\ \ 3\ \ 2\ \ 5\ \ -4\ \ 6)$ break, in some of the intermediate permutations, the common interval $(2\ \ 3)$.

Are adjacencies the only structures that are invariant under biologically meaningful rearrangement operations? No. There exists a class of common intervals, called *conserved intervals*, that may be the best of two worlds. We will show that these structures both capture local and global properties of genomes; are invariant under most rearrangement scenarios; and their number and nature can be computed in linear time.

In the next section, we discuss the main issues of gene order comparison, with biological data. In Section 3, we give the formal definitions and properties of conserved intervals, and we present the associated algorithms in Section 4. Section 5 shows how conserved intervals can be used as either a similarity or a distance measure, and Section 6 discusses the links between conserved intervals and rearrangement theories. Section 7 concludes with a few provocative statements.

2 Permutations, Gene Order, and Rearrangements

In the following we will take for granted the simplifying hypothesis that the genes of an organism are ordered and oriented along linear or circular DNA molecules. For example

the 37 mitochondrial genes of the Fruit Fly are listed in [6], with minus signs to reflect orientation, as:

```
cox1, L2, cox2, K, D, atp8, atp6, cox3, G, nad3, A, R, N, S1, E, -F, -nad5, -H, -nad4, -nad4L,
T, -P, nad6, cob, S2, -nad1, -L1, -rrnL, -V, -rrnS, UNK, I, -Q, M, nad2, W, -C, -Y
```

The first gene is arbitrary, since mitochondrial genomes are circular molecules. When organisms with the same gene content are compared, one of them is chosen as a base organism, and all identical strips of genes are converted to integers. By extension, these are also called "genes". Table 1 presents the result of this transformation applied to the mitochondrial genomes of six *Arthropoda*, with Fruit Fly as base organism. The original 37 genes have been divided in 17 blocks: some represent isolated genes, and others represent longer strips. For example, 10 stands for `S1`, and 11 for `E, -F, -nad5, -H, -nad4, -nad4L, T, -P, nad6, cob, S2, -nad1`.

Fruit Fly	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Mosquito	1	2	3	4	5	6	8	7	9	-10	11	12	13	14	15	16	17
Silkworm	1	2	3	4	5	6	7	8	9	10	11	12	14	13	15	16	17
Locust	1	2	3	5	4	6	7	8	9	10	11	12	13	14	15	16	17
Tick	1	3	4	5	6	7	8	9	10	11	-2	12	13	14	15	16	17
Centipede	1	3	4	5	6	7	8	9	10	11	-2	12	16	13	14	15	17

Table 1: Condensed mitochondrial genomes of six Arthropoda

Various techniques are then used to compare the resulting permutations. The *distance* approaches focus on the differences between two particular genomes. For example, Fruit Fly differs from Mosquito by the *reversal* of gene 10, and the *transposition* of genes 7 and 8. One can count the minimal number of reversals and/or transpositions necessary to transform each genome into any other, yielding a distance matrix for the set of species. Explicit rearrangement scenarios, that is, sequences of operations that transform optimally one genome into another, are also used to reconstruct ancestral genomes.

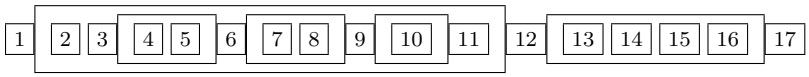
Another approach, the *breakpoint distance*, counts the lost adjacencies between genomes. It does not rely on particular rearrangement operations or an evolutionary model, and has an associated measure of similarity: the number of *conserved* adjacencies. For example, given the circularity of the genomes, Fruit Fly and Mosquito have 12 conserved adjacencies, and a breakpoint distance of 5.

Such a similarity measure extends easily to sets of species. For example, the first four species of Table 1 share 6 adjacencies: $[1, 2]$, $[2, 3]$, $[11, 12]$, $[15, 16]$, $[16, 17]$, and $[17, 1]$. When comparing all six species, the only left adjacency is $[17, 1]$: this lack of conserved adjacencies is a direct consequence of how the data was transformed. Does this mean that losing common adjacencies amounts to losing all common structures?

A quick glance at Table 1 reveals that the six permutations are very "similar". For example, the genes between in the interval $[1, 12]$ are all the same, with small variations in their ordering. This is also true for the genes in the intervals: $[3, 6]$, $[6, 9]$, $[9, 11]$, and $[12, 17]$. It turns out that such intervals, together with conserved adjacencies, play a

fundamental role in rearrangement and distance theories, ancestral genome reconstructions, and phylogeny.

The following *family portrait* gives a representation of the conserved intervals of the permutations of Table 1:



This representation boxes the elements in rectangles, which can be glued together to form larger objects. It takes its roots in PQ-trees [7] that are used to represent sets of permutations. All permutations of Table 1 fit the representation with the following conventions: (1) free objects within a rectangle can be reordered, or can change sign, (2) connections between rectangles are fixed. This representation also captures the features that should be invariant in biologically plausible rearrangement scenarios within the family.

In order to illustrate this last point, consider the two following rearrangement scenarios that transform Silkworm into Locust using a minimal number of *reversals* (operations that reverse the elements of a consecutive block while changing their signs).

1	2	3	4	5	6	7	8	9	10	11	12	14	13	15	16	17
1	2	3	-4	5	6	7	8	9	10	11	12	14	13	15	16	17
1	2	3	-4	-5	6	7	8	9	10	11	12	14	13	15	16	17
1	2	3	5	4	6	7	8	9	10	11	12	14	13	15	16	17
1	2	3	5	4	6	7	8	9	10	11	12	-14	13	15	16	17
1	2	3	5	4	6	7	8	9	10	11	12	-14	-13	15	16	17
1	2	3	5	4	6	7	8	9	10	11	12	13	14	15	16	17

1	2	3	4	5	6	7	8	9	10	11	12	14	13	15	16	17
1	2	3	4	-14	-12	-11	-10	-9	-8	-7	-6	-5	13	15	16	17
1	2	3	4	-14	5	6	7	8	9	10	11	12	13	15	16	17
1	2	3	4	-13	-12	-11	-10	-9	-8	-7	-6	-5	14	15	16	17
1	2	3	5	6	7	8	9	10	11	12	13	-4	14	15	16	17
1	2	3	5	4	-13	-12	-11	-10	-9	-8	-7	-6	14	15	16	17
1	2	3	5	4	6	7	8	9	10	11	12	13	14	15	16	17

Those two scenarios are fundamentally different, even if they both use six reversals. The right one uses much longer reversals than the left one, and the right one *breaks* conserved intervals between Silkworm and Locust in intermediate permutations, namely $[3, 6]$, $[1, 12]$, and $[12, 17]$. If a rearrangement scenario is expected to reflect the various intermediate species between Silkworm and Locust, the right one looks highly suspicious. Recent papers address these problems in various ways, for example by assigning weights to operations [1], or with probabilistic studies of the possible scenarios [15].

The two main flaws of the second scenario – long reversals and breaking conserved intervals – are closely tied: breaking conserved intervals, as we will show in Section 6, often involves long range operations that radically disturb a genome. In this sense, conserved intervals can be used as an intrinsic measure that allows to screen out rearrangement scenarios, or phylogenetic hypothesis, without the need of arbitrary weights or probability measures.

3 Conserved Intervals

This section presents a formalization of the notion of conserved intervals, together with properties that allow the development of linear time algorithms to manipulate them. We also discuss the necessary adaptations to model conserved intervals in circular and multi-chromosomal genomes.

Definition 1 *Let G be a set of signed permutations on n elements. An interval $[a, b]$ is a* conserved interval *of the set G if:*

1) either a precedes b, or $-b$ precedes $-a$, in each permutation, and

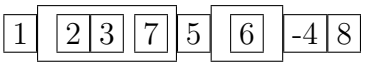
2) the set of unsigned elements that appear between a and b is the same for all permutations in G.

An elementary consequence of this definition is the fact that if $[a, b]$ is a conserved interval, so is $[-b, -a]$. We will consider these intervals as equivalent.

Table 1 contains several examples of conserved intervals. Their description is eased by the fact that the identity permutation belongs to the set G. When this is the case, all conserved intervals can be identified with their positive endpoints $a < b$, and the set of elements that appear between a and b is $\{a + 1, \ldots, b - 1\}$. The following example illustrates a more general case. Consider the two permutations:

$$\begin{array}{lcl} P & = & 1\ \ 2\ \ 3\ \ 7\ \ 5\ \ 6\ -4\ \ 8 \\ Q & = & 1\ \ 7\ -3\ -2\ \ 5\ -6\ -4\ \ 8 \end{array}$$

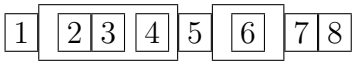
In this example, [1, 5] and [2, 3] are conserved intervals, but not [1, 6]. The other conserved intervals of P and Q are [1, -4], [1, 8], [5, -4], [5, 8] and [-4, 8]. The diagram representation of these intervals, with respect to the permutation P, is:



When the identity permutation is not in G, it is always possible to *rename* the elements of G such that conserved intervals will be intervals of consecutive elements. For example, if one composes[1] the permutations P and Q of the above example with the inverse permutation P^{-1}, one gets the set:

$$\begin{array}{lcl} P' & = & P^{-1} \circ P \\ Q' & = & P^{-1} \circ Q \end{array} \quad = \quad \begin{array}{l} 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8 \\ 1\ \ 4\ -3\ -2\ \ 5\ -6\ \ 7\ \ 8 \end{array}$$

whose diagram representation of conserved intervals, with respect to P' is equivalent to the one of P and Q, up to renaming of the elements:



In general, it is elementary to transform a set of conserved intervals to its equivalent up to renaming. It is a consequence of the following proposition:

Proposition 1 *Let R be a permutation and G a set of permutations, denote by $R \circ G$ the set of permutations obtained by composing each permutation in G with R. The interval $[a, b]$ is conserved in G if and only if the interval $[R(a), R(b)]$ is conserved in $R \circ G$.*

[1]Here, composition is understood as the standard composition of functions. Dealing with signed permutations requires the additional axiom that $P(-a) = -P(a)$.

Proof: Note that if a permutation P is written as:

$$P \;=\; p_1 \;\; p_2 \;\; \ldots \;\; p_n$$

then $R \circ P$ is:

$$R \circ P \;=\; R(p_1) \;\; R(p_2) \;\; \ldots \;\; R(p_n).$$

If $[a, b]$ is conserved in G, then each permutation in G has a consecutive block of elements beginning with a end ending with b, or beginning with $-b$ and ending with $-a$. These properties hold also for the set $R \circ G$, if one replaces a by $R(a)$ and b by $R(b)$. □

Some intervals, such as [1, 7] for the set $\{P', Q'\}$ in the above example, are the union of smaller intervals: $[1, 7] = [1, 5] \cup [5, 7]$. Intervals that are not unions are specially useful:

Definition 2 *Conserved intervals that are not the union of shorter conserved intervals are called* irreducible.

Sets of conserved intervals can be simply characterized by the corresponding set of irreducible intervals. Indeed, disjoint irreducible intervals, as highlighted in the diagram representation, are either *chained* or *nested*. The following proposition captures the basic properties of these structures.

Proposition 2 *Two different irreducible conserved intervals $[a, b]$ and $[c, d]$ of a set G of permutations, are either:*

1) disjoint,

2) nested with different endpoints, or

3) overlapping on one element.

Proof: Without loss of generality, by Proposition 1, we can assume that G contains the identity permutation, and that conserved intervals are intervals of consecutive elements. Suppose that $[a, b]$ and $[c, d]$ are nested with $a = c$ and $d < b$. Since $[c, d]$ is a conserved interval, it contains all integers between c and d, therefore, the interval $[d, b]$ contains all integers between d and b, and $[a, b]$ is not irreducible.

If $[a, b]$ and $[c, d]$ overlap with more than one element, we can suppose $a < c < b < d$. Since all elements between c and d are greater than c, then the interval between a and c must contain all elements between a and c, thus $[a, b]$ is not irreducible. □

The three cases of Proposition 2 are illustrated in Fig. 1.

Overlapping irreducible intervals form chains linked by their successive common elements. A chain of $k - 1$ intervals $[a_1, a_2][a_2, a_3] \ldots [a_{k-1}, a_k]$ will be denoted simply by its k links $[a_1, a_2, a_3 \ldots a_k]$. For example, $[1, 5, 7, 8]$ is a chain of the set of conserved intervals of P' and Q'. A *maximal* chain is a chain that cannot be extended. We have:

Proposition 3 *Every irreducible conserved interval belongs to a unique maximal chain.*

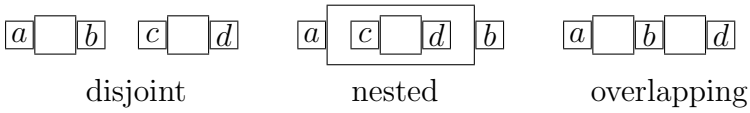


Figure 1: Chaining and nesting irreducible conserved intervals

Proof: By Proposition 2, if $[a, b]$ is an irreducible conserved interval, then no other can begin by a or end by b. □

One consequence of Proposition 3 is that maximal chains, as sets of links, together with isolated genes, form a partition of the set of genes. This will reveal useful to construct data structures to keep track of conserved intervals.

A set of permutations on n elements can have as many as $n(n-1)/2$ conserved intervals, but at most $n-1$ irreducible intervals. These bounds are achieved with sets containing only one permutation. A key observation, that will eventually lead to linear time algorithms to compute the number of conserved intervals, is the following:

Proposition 4 *Each maximal chain of k links contributes $k(k-1)/2$ to the total number of conserved intervals.*

Proof: Conserved intervals $[a, b]$, are in bijection with chains of the form

$$[a, x_1, \ldots x_k, b]$$

of irreducible intervals. Each maximal chain of k links has $k(k-1)/2$ such sub-chains. □

Finally, we will want to construct sets of conserved intervals for the union of two sets of permutations. Definition 1 implies that the set of conserved intervals of a union of two sets of permutations is the intersection of their sets of conserved intervals. The following properties relate these sets to their respective irreducible intervals when both sets of permutations have at least one permutation in common.

Proposition 5 *Let P be a permutation that is contained in both sets G_1 and G_2. The interval $[a, b]$ is a conserved interval of $G = G_1 \cup G_2$ if and only if there exist two chains of irreducible conserved intervals, with respect to P, with $k \geq 0$, $l \geq 0$:*

$$\begin{array}{ll} [a, x_1, \ldots, x_k, b] & \textit{in } G_1, \\ [a, y_1, \ldots, y_l, b] & \textit{in } G_2. \end{array}$$

The interval $[a, b]$ is irreducible if and only if $\{x_1, \ldots, x_k\}$ and $\{y_1, \ldots, y_l\}$ are disjoint.

Proof: The interval $[a, b]$ is a conserved interval of G if and only if it is a conserved interval in both G_1 and G_2, therefore there must exist chains beginning by a and ending by b for both sets G_1 and G_2. If $[a, b]$ is irreducible in G, and if $[a, x]$ and $[x, b]$ are conserved intervals of G_1, say, then x cannot belong to the set $\{y_1, \ldots, y_l\}$. If there is a common element x to both sets $\{x_1, \ldots, x_k\}$ and $\{y_1, \ldots, y_l\}$, then $[a, b] = [a, x] \cup [x, b]$, and both $[a, x]$ and $[x, b]$ are conserved intervals of G. □

Variable Geometry Genomes. Although the definition of conserved intervals was given for permutations that model genomes composed of single linear chromosomes, they can be adapted to other types of genomes.

For circular genomes, the easiest approach is to align all permutations of the set beginning with gene +1. This can always be done. Indeed, negative signs are used to code the strand of the DNA molecule on which the gene is read and, since the choice of strand is arbitrary, a chromosome can always be flipped by changing all the signs of its genes and reversing their order. It is also convenient to add an extra gene $n + 1$ at the end of each permutation to simulate the fact that the last gene of each permutation is adjacent to the first one. This creates artificial conserved intervals that can be discarded later.

Multi-chromosomal genomes can also be represented by permutations, with special marks that identify different chromosomes. For example, consider the following two genomes, for which each chromosome is on a separate line:

1 2 3 4 5 6 7 8 9 10 11	1 −3 −2 4 7 8 5 6 9 −10 11
genome 1	genome 2

Clearly, even if the adjacency $[5, 6]$ is conserved between the two permutations, the first genome does not even have those genes on the same chromosome. In the case of multi-chromosomal genomes, conserved intervals $[a, b]$ should have the added requirement that a and b belong to the same chromosome, in each genome.

4 Algorithms

This section discusses three algorithms. The first one is an adaptation of an existing algorithm that computes the conserved intervals of two permutations. The second one computes the conserved intervals of a set of permutations. The third one, finally, computes the conserved intervals of two sets of permutations, directly from their two individual sets of conserved intervals.

Conserved Intervals of Two Permutations. Conserved intervals between two permutations are strongly related to the notion of connected components of the *overlap graph* of a signed permutation. This graph plays a fundamental role in the sorting by reversals problem [10], and the sorting by reversals and translocations problem [11]. In the last few years, linear algorithms to identify these components have been devised [2]. The following algorithm is adapted from [4], and identifies all irreducible intervals[2] $[a, b]$ of a permutation π with the identity permutation such that $a > 0$ and $b > 0$ in π. The case of negative endpoints is treated by reversing π.

[2]In the original paper, these were called *framed common intervals*.

For example, for the permutation

$$P \quad = \quad 0 \ -4 \ -3 \ -2 \ \ 5 \ \ 8 \ \ 6 \ \ 7 \ \ 9 \ -1 \ \ 10,$$

Algorithm 1 identifies the positive irreducible intervals $[6, 7]$, $[5, 9]$, and $[0, 10]$. It will identify $[2, 3]$ and $[3, 4]$ on the reversed permutation.

The algorithm assumes that the input permutation is in the form:

$$\pi = (0, \pi_1, \ldots, \pi_{n-1}, n).$$

Define M_i to be the nearest unsigned element of the permutation that precedes π_i and is greater than $|\pi_i|$. (Set M_i to n, if such an element does not exist). The following lemma relates the values of M_i to conserved intervals.

Lemma 1 *If $[\pi_s, \pi_e]$ is a positive conserved interval of π and the identity permutation, then $M_s = M_e$.*

The algorithm uses two stacks: $\mathcal{S}$ contains the possible start positions of conserved intervals; $\mathcal{M}$ contains possible candidates for M_i. The top of $\mathcal{S}$ is always denoted by s. The top of $\mathcal{M}$ is always denoted by m.

Algorithm 1 (Positive irreducible intervals with the identity permutation)

stack 0 on $\mathcal{S}$
stack n on $\mathcal{M}$
$M_0 \leftarrow n$
for $i = 1, \ldots, n$ **do**
  // Computation of M_i
  unstack from $\mathcal{M}$ all elements m smaller than $|\pi_i|$
  $M_i \leftarrow m$
  stack the element $|\pi_i|$ on $\mathcal{M}$
  // Identification of irreducible intervals
  unstack from $\mathcal{S}$ all indices s such that ($|\pi_i| < \pi_s$ or $|\pi_i| > M_s$)
  if $i - s = \pi_i - \pi_s$ **and** $M_i = M_s$ **then**
    output $[\pi_s, \pi_i]$
  end if
  if π_i is positive **then**
    stack the index i on $\mathcal{S}$
  end if
end for

Proposition 6 *Algorithm 1 outputs the positive irreducible conserved intervals of a permutation π with the identity permutation in $O(n)$ time.*

Proof: Let $\pi_k \dots \pi_i$ be an interval in π such that $|\pi_k| = M_i$. Thus, $|\pi_k| > |\pi_i|$, and all unsigned elements between π_k and π_i are smaller than π_i. No element between π_k and π_i can unstack π_k in line 6, and π_i will unstack all of them, thus the algorithm correctly computes M_i, when it is different from n. If $M_i = n$, then π_i is larger than any element in $0, \pi_1, \dots, \pi_{i-1}$, and it will unstack all of them. In this case, the initialization of $\mathcal{M}$ yields the correct value for M_i.

Line 12 always outputs positive conserved intervals, since (1) $i - s = \pi_i - \pi_s$ provides the correct number of elements, (2) all unsigned elements in the interval are greater than π_s by the first condition of line 10, and (3) all unsigned elements in the interval are smaller than π_i since $M_i = M_s$.

If $[\pi_s, \pi_{s'}]$ and $[\pi_{s'}, \pi_e]$ are both conserved, then any element π_i that unstacks s' will also unstack s, since $M_{s'} = M_s$, and $|\pi_i| < \pi_{s'}$ implies $|\pi_i| < \pi_s$. Thus only irreducible intervals can be reported by the algorithm.

If $[\pi_s, \pi_e]$ is an irreducible interval and π_s is positive, then π_s will be stacked. Since $M_e = M_s$, and all unsigned elements between π_s and π_i are greater than π_s, π_s will not be unstacked before π_e is reached. If $s' \neq s$ is on the top of the stack once π_e is reached, then all unsigned elements between $\pi_{s'}$ and π_e must be greater than than $\pi_{s'}$, otherwise $\pi_{s'}$ would have been unstacked. But there is at least one element greater than $\pi_{s'}$ that lies between π_s and $\pi_{s'}$, otherwise $[\pi_{s'}, \pi_e]$ would be a conserved interval, thus $M_{s'} < \pi_e$, and s' will be unstacked once π_e is reached. Thus all irreducible intervals will be reported by the algorithm.

The analysis of the time complexity is elementary, since each index is stacked or unstacked at most once. □

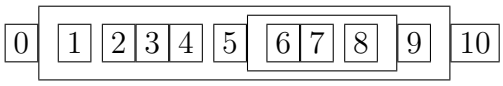
Corollary 1 *By applying Algorithm 1 both to $\pi = P^{-1} \circ Q$ and to the reverse of π, the irreducible conserved intervals of two permutations P and Q can be found in $O(n)$ time.*

Proof: This follows immediately from Propositions 1 and 6 and the fact that each irreducible interval must have either two positive or two negative endpoints in π. □

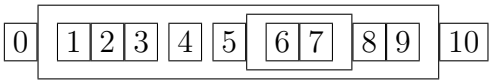
Conserved Intervals of a Set of Permutations. In order to find the irreducible conserved intervals of a set of permutations, the first step is to compute the irreducible intervals of each permutation with one particular permutation from the set, say π_1, using Algorithm 1, and then merge together the resulting sets of irreducible intervals. For example, computing the irreducible intervals of the set:

$$\begin{array}{rcccccccccccc} Id & = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ P & = & 0 & -4 & -3 & -2 & 5 & 8 & 6 & 7 & 9 & -1 & 10 \\ Q & = & 0 & 5 & -7 & -6 & 8 & 9 & 1 & 2 & 3 & -4 & 10 \end{array}$$

would first yield the following sets of conserved intervals, in graphic representation, of P and the identity:



and of Q and the identity:



In terms of maximal chains, the two sets are described by:

$$\{[0,10],[2,3,4][5,9],[6,7]\} \text{ and } \{[0,10],[1,2,3],[5,8,9],[6,7]\},$$

and their intersection by:

$$\{[0,10],[2,3],[5,9],[6,7]\}.$$

Assume that each set of irreducible conserved intervals is given by its maximal chains. Since these form partitions of the genes that are endpoints of conserved intervals, there exists a data structure with the following properties:

1. For each index from 1 to n, it is possible to determine in constant time the interval, if any, that starts and/or ends at this index.

2. It is possible to determine in constant time if two intervals belong to the same chain.

Let I_1 and I_2 be two sets of irreducible conserved intervals of sets of permutations G_1 and G_2 that have one permutation π_1 in common. For the moment we will assume that π_1 is the identity permutation. Then Algorithm 2 finds all irreducible conserved intervals of $G_1 \cup G_2$. It uses a stack $\mathcal{S}$ that contains possible start positions – or, equivalently, elements of the identity permutation. The top of the stack $\mathcal{S}$ is always denoted by s.

The correctness and time complexity of Algorithm 2 are established by the following theorem.

Theorem 1 *Algorithm 2 outputs the irreducible intervals of $G = G_1 \cup G_2$ in $O(n)$ time, given I_1 and I_2, the irreducible intervals of two sets of permutations G_1 and G_2 that both contain the identity permutation.*

Proof: If a and b belong to the same chain both in I_1 and I_2, then $[a,b]$ is certainly a conserved interval of G by Proposition 5. Thus the algorithm outputs only conserved intervals.

If an interval $[a,b]$ of G is irreducible, element a will be stacked in line 13, since there exists an interval that starts at a in I_1, and one in I_2. Since a is the beginning of a unique irreducible interval, it can be unstacked in line 10 only when b is reached. If a is unstacked in either line 4 or line 7, then $[x,i]$ is an irreducible interval either in I_1 or I_2, and $x < a < i$. By Proposition 2, $b < i$, thus a cannot be unstacked before b is reached.

We must also make sure that, when b is reached, a is on the top of the stack. Suppose that $[a,b]$ corresponds to the following chains

$$\begin{array}{ll} [a, x_1, \ldots, x_k, b] & \text{in } I_1 \\ [a, y_1, \ldots, y_l, b] & \text{in } I_2. \end{array}$$

Algorithm 2 (Irreducible intervals of $G_1 \cup G_2$, both containing the identity permutation)

```
 1: stack 0 on S
 2: for i = 1, ..., n do
 3:    if there is an interval [x, i] in I_1 then
 4:       unstack from S all elements larger than x
 5:    end if
 6:    if there is an interval [x, i] in I_2 then
 7:       unstack from S all elements larger than x
 8:    end if
 9:    if s and i belong to the same chain both in I_1 and I_2 then
10:       unstack s from S and output [s, i]
11:    end if
12:    if there is an interval that starts at i in I_1, and one in I_2 then
13:       stack i on S
14:    end if
15: end for
```

Any value x stacked after a and before b cannot belong to the same chain both in I_1 and I_2, since $[a, b]$ would not be irreducible, by Proposition 5. Thus x falls, say in I_2, between two consecutive values of the chain $[a, y_1, \ldots, y_l, b]$. When the second value is reached, x will be unstacked in line 7.

Finally, if a conserved interval $[a, b]$ of G is not irreducible, then it can be written as $[a, b] = [a, x] \cup [x, b]$, and we can assume that $[a, x]$ is irreducible. Element a will be unstacked in line 10, after identifying the interval $[a, x]$, thus the algorithm will never output $[a, b]$.

The analysis of the time complexity is elementary, since each index is stacked or unstacked at most once. □

Corollary 2 *Let I_1 and I_2 be the irreducible intervals of two sets of permutations G_1 and G_2 that both contain a permutation P. The irreducible intervals of $G = G_1 \cup G_2$ can be found in $O(n)$ time by applying Algorithm 2 to $I'_1 = \{[P^{-1}(a), P^{-1}(b)] \mid [a, b] \in I_1\}$ and $I'_2 = \{[P^{-1}(a), P^{-1}(b)] \mid [a, b] \in I_2\}$.*

Proof: This follows immediately from Proposition 1 and Theorem 1. □

Corollary 3 *The set of irreducible conserved intervals of a set of permutations G can be computed in $O(|G|n)$ time and $O(n)$ additional space.*

Proof: Let $G = \{\pi_1, \ldots, \pi_k\}$. For $i = 2, \ldots, k$ denote the set of irreducible conserved intervals of $\{\pi_1, \ldots, \pi_i\}$ by I_i. Intially generate I_2 by applying Algorithm 1 to π_1 and π_2. Then, for $i = 3, \ldots, k$, apply Algorithm 1 to the pair π_1 and π_i generating I_i^*, and apply Algorithm 2 to I_i^* and I_{i-1} generating I_i. □

Conserved Intervals of Disjoint Sets. Finally we are interested in computing the conserved intervals of two sets of permutations $G_1 = \{P_1, \ldots, P_k\}$ and $G_2 = \{Q_1, \ldots, Q_l\}$ that not necessarily have a permutation in common, given their sets of irreducible conserved intervals I_1 and I_2, respectively.

This can be done by properly combining Algorithms 1 and 2. The idea is to select one permutation from each set, say P_1 from G_1 and Q_1 from G_2, and compute the conserved intervals of these two by Algorithm 1. Then observe that the two sets $\{P_1, Q_1\}$ and $G_1 = \{P_1, \ldots, P_k\}$ have a joint permutation P_1, and hence their common irreducible intervals can be computed by Algorithm 2. Similarly, $\{Q_1, P_1, \ldots, P_k\}$ and $G_2 = \{Q_1, \ldots, Q_l\}$ contain a joint permutation Q_1, so their common irreducible intervals can also be computed by Algorithm 2.

This procedure is formalized in Algorithm 3.

Algorithm 3 (Irreducible intervals of $G_1 \cup G_2$)

1: Select a permutation $P_1 \in G_1$ and a permutation $Q_1 \in G_2$.
2: Compute I', the irreducible conserved intervals of P_1 and Q_1, by Algorithm 1.
3: Compute I'_1, the irreducible intervals of $G_1 \cup \{Q_1\}$, by applying Algorithm 2 to I_1 and I'.
4: Compute I, the irreducible intervals of $G_1 \cup G_2$, by applying Algorithm 2 to I'_1 and I_2.

Theorem 2 *Algorithm 3 finds the irreducible intervals of $G = G_1 \cup G_2$, given I_1 and I_2, the irreducible intervals of two sets of permutations G_1 and G_2, in $O(n)$ time.*

Proof: This follows immediately from Corollaries 1 and 2. □

5 Similarity and Distance

The number of conserved intervals of a set of permutations is a measure of similarity, but it can easily be transformed into a distance between two permutations, or two sets of permutations. The basic idea is that two sets of conserved intervals can be compared with the cardinality of their symmetric difference.

Definition 3 *Let G_1 and G_2 be two sets of permutations on n elements, with respectively N_1 and N_2 conserved intervals. Let N be the number of conserved intervals in $G_1 \cup G_2$, the* interval distance *between G_1 and G_2 is defined by:*

$$d(G_1, G_2) = N_1 + N_2 - 2N.$$

Note: The interval distance satisfies the fundamental properties of a mathematical distance since one can prove that the relation is symmetric, reflexive, and satisfies the *triangle inequality*: $d(P,Q) + d(Q,R) \geq d(P,R)$.

When comparing two permutations, the interval distance counts the total number of intervals that are unique to one of them. For example, the distance between:

$$\begin{array}{rcccccccccccc} P & = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ Q & = & 0 & 5 & -7 & -6 & 8 & 9 & 1 & 2 & 3 & 4 & 10 \end{array}$$

is given by $d(P,Q) = (11 * 10)/2 + (11 * 10)/2 - 2 * 11 = 88$.

Table 2 gives a comparison of the matrix of interval distances of the six species of Section 2, – left columns –, together with the matrix of reversal distances of the same species – right columns.

	Fruit Fly		Mosquito		Silkworm		Locust		Tick		Centipede	
Fruit Fly	-	-	90	4	62	3	62	3	158	2	188	5
Mosquito	90	4	-	-	140	7	140	7	200	6	230	9
Silkworm	62	3	140	7	-	-	116	6	180	5	194	7
Locust	62	3	140	7	116	6	-	-	188	5	218	8
Tick	158	2	200	6	180	5	188	5	-	-	110	3
Centipede	188	5	230	9	194	7	218	8	110	3	-	-

Table 2: Interval distances (left) and reversal distances (right)

It can readily be seen, in Table 2, that the two measures sometimes disagree. For example, Fruit Fly and Tick have the smallest reversal distance, but certainly not the smallest interval distance. The behavior of the interval distance is a consequence of the fact that it is affected be the length – or number of genes – involved in a rearrangement operation: short reversals, for example, are less disturbing than long ones. In particular, the amount of disruption due to a single rearrangement operation can readily be computed. For example, we have the following:

Proposition 7 *Suppose that P and Q have n elements, then:*

1) if P is obtained from Q by reversing k elements, then the interval distance between P and Q is $k(n-k)$;

2) if P is obtained from Q by transposing two consecutive blocks of a and b elements, then the interval distance between P and Q is $(a+b)(n-(a+b)) + ab$.

Since the interval distance is affected by length, the practice of collapsing identical strips of genes should be questioned. Indeed, as we saw in the example of Section 2, the integers resulting from such a transformation stand for strips of genes that vary greatly in length. We believe that whole genome comparison should use all available information, and

that length of segments is relevant to the study of rearrangement scenarios, as advocated in [18].

Although the above discussion focused on the reversal distance, similar results can be observed with the breakpoint distance (see Table 3). The only measure that agrees in trend with the interval distance is the transposition+reversal distance, but, as can bee seen in Table 4, interval distance has a much broader range.

	Fruit Fly		Mosquito		Silkworm		Locust		Tick		Centipede	
Fruit Fly	-	-	90	5	62	3	62	3	158	3	188	6
Mosquito	90	5	-	-	140	8	140	8	200	8	230	11
Silkworm	62	3	140	8	-	-	116	6	180	6	194	8
Locust	62	3	140	8	116	6	-	-	188	6	218	9
Tick	158	3	200	8	180	6	188	6	-	-	110	3
Centipede	188	6	230	11	194	8	218	9	110	3	-	-

Table 3: Interval distances (left) and breakpoint distances (right)

	Fruit Fly		Mosquito		Silkworm		Locust		Tick		Centipede	
Fruit Fly	-	-	90	2	62	1	62	1	158	2	188	3
Mosquito	90	2	-	-	140	3	140	3	200	4	230	5
Silkworm	62	1	140	3	-	-	116	2	180	3	194	4
Locust	62	1	140	3	116	2	-	-	188	3	218	4
Tick	158	2	200	4	180	3	188	3	-	-	110	1
Centipede	188	3	230	5	194	4	218	4	110	1	-	-

Table 4: Interval distances (left) and transposition+reversal distances (right)

6 Links With Rearrangement Theories

In Section 2, we gave an example of how conserved intervals could be used to evaluate optimal reversal scenarios between two genomes. Reversals are one of the many operations that are currently used to model genome evolution: the main other ones – among those that do not need to model duplication of genes – are transpositions, reverse transpositions, translocations, fusions, and fissions.

In this section, we want to characterize the rearrangement operations, or scenarios, that *preserve* conserved intervals:

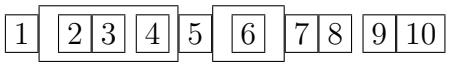
Definition 4 *Let P and Q be two permutations, and ρ a rearrangement operation applied to P yielding P'. We say that ρ* preserves *the conserved intervals of P and Q if the conserved intervals of $\{P, Q\}$ are contained in those of $\{P', Q\}$.*

Keeping in mind the graphical representation of the conserved intervals, it is easy to identify the operations that preserve conserved intervals: only rearrangements within blocks are preserving. To be more formal, note that all operations, except fusions, destroy some adjacencies that existed in the original permutation: the number and nature of these adjacencies is a key concept.

Definition 5 *Let ρ be a rearrangement operation that transforms P into P'. A* breakpoint *of ρ is a pair of elements that are adjacent in P but not in P'.*

In other words, breakpoints are where one has to cut P in order to apply ρ. Reversals and translocations have 2 breakpoints, transpositions have 3, and fissions have 1.

Consider the irreducible intervals of P and P' with respect to P. Adjacencies in P either belong to a (smallest) irreducible interval, or are *free*. For example, in the following diagram:



The adjacency $(3, 4)$ belongs to the interval $[1, 5]$, $(2, 3)$ belongs to $[2, 3]$, and $(8, 9)$ is free. Note that when two or more adjacencies belong to the same irreducible interval, then none of these adjacencies is conserved between P and P'.

Theorem 3 *Reversals, transpositions, and reverse transpositions are preserving if and only if all their breakpoints belong to the same irreducible interval, or are free.*

Translocations and fissions are preserving if and only if all their breakpoints are free.

Proof: If the breakpoints of any operation are free, then clearly no conserved interval is cut. If the breakpoints of a reversal, transposition, or reverse transposition belong to the same irreducible interval, then the operation reorders, or reverses, some blocks within that interval, thus preserving conserved intervals.

If a reversal has its two breakpoints in different intervals, it will break those two intervals. If it has only one free breakpoint, it will break the interval containing the other breakpoint. The same kind of arguments hold for transpositions and reverse transpositions.

If a breakpoint of a translocation or fission is not free, then it belongs to an irreducible interval whose extremities will end up in two different chromosomes. □

It turns out that most rearrangement operations used in optimal scenarios are indeed preserving. It is outside the scope of this paper to discuss these results in detail: they involve the *cycle* structure of a permutation, which are special subsets of the breakpoints of a permutation P with respect to a permutation P'. The following result has been proved in various disguises in recent years [4, 10, 13]:

Theorem 4 *All the breakpoints of a cycle belong to the same irreducible interval.*

In the sorting by reversals theory, a *sorting* reversal is defined as a reversal that decreases the reversal distance by 1. It is shown [10, 19] that the breakpoints of sorting reversals, except one type called *hurdle merging*, belong to a single cycle, thus we have:

Corollary 4 *All sorting reversals, except hurdle merging, are preserving.*

Hurdle mergings are a rare type of reversals in optimal scenarios: they break at least two disjoint irreducible intervals, thus involving long reversals. In Section 2, for example, the first reversal of the second scenario that transformed Silkworm into Locust was a hurdle merging.

The theory of translocations, fusions and fissions [11, 17] relies on the properties of sorting by reversals, thus most sorting reversals are preserving. Finally, transpositions are a more delicate matter since sorting transpositions are not (yet) characterized. Nevertheless, it is known that transpositions that increase the number of cycles – a desirable property when sorting permutations – have all their breakpoints in the same cycle [3]. Thus we have:

Corollary 5 *All transpositions that create two adjacencies are preserving.*

7 Conclusion

We have introduced a new similarity measure for permutations, based on the concept of conserved intervals. Conserved intervals have very interesting properties with respect to preserving the usual genome rearrangement operations. We believe that conserved intervals are a fundamental concept of rearrangement theory. They provide the unifying grounds to understand the variety of operations that are used to model genome evolution. Supported by recent results on the expected size of rearranged genome segments, one could go as far and claim that any rearrangement scenario that breaks conserved intervals is mathematical rambling without connection to evolutionary reality.

References

[1] Y. Ajana, J.-F. Lefebvre, E. R. M. Tillier, and N. El-Mabrouk. Exploring the set of all minimal sequences of reversals – An application to test the replication-directed reversal hypothesis. In *Proceedings of WABI 2002*, volume 2452 of *LNCS*, pages 300–315. Springer Verlag, 2002.

[2] D. A. Bader, B. M. E. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *J. Comput. Biol.*, 8(5):483–492, 2001.

[3] V. Bafna and P. A. Pevzner. Sorting by transpositions. *SIAM J. Disc. Math.*, 11(2):224–240, 1998.

[4] A. Bergeron, S. Heber, and J. Stoye. Common intervals and sorting by reversals: A marriage of necessity. *Bioinformatics*, 18(Suppl. 2):S54–S63, 2002. (Proceedings of ECCB 2002).

[5] M. Blanchette, T. Kunisawa, and D. Sankoff. Gene order breakpoint evidence in animal mitochondrial phylogeny. *J. Mol. Evol.*, 49(2):193–203, 1999.

[6] J. L. Boore. Mitochondrial gene arrangement source guide. `www.jgi.doe.gov/programs/comparative/Mito_top_level.html`.

[7] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs and graph planarity using *PQ*-tree algorithms. *J. Comput. Syst. Sci.*, 13(3):335–379, 1976.

[8] G. Bourque and P. A. Pevzner. Genome-scale evolution: Reconstructing gene orders in the ancestral species. *Genome Res.*, 12(1):26–36, 2002.

[9] D. A. Christie. *Genome Rearrangement Problems*. PhD thesis, The University of Glasgow, 1998.

[10] S. Hannenhalli and P. A. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proceedings of FOCS 1995*, pages 581–592. IEEE Press, 1995.

[11] S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. *J. ACM*, 46(1):1–27, 1999.

[12] S. Heber and J. Stoye. Finding all common intervals of k permutations. In *Proceedings of CPM 2001*, volume 2089 of *LNCS*, pages 207–218. Springer Verlag, 2001.

[13] H. Kaplan, R. Shamir, and R. E. Tarjan. A faster and simpler algorithm for sorting signed permutations by reversals. *SIAM J. Computing*, 29(3):880–892, 1999.

[14] J. D. Kececioglu and D. Sankoff. Efficient bounds for oriented chromosome inversion distance. In *Proceedings of CPM 1994*, volume 807 of *LNCS*, pages 307–325. Springer Verlag, 1994.

[15] B. Larget, J. Kadane, and D. Simon. A Markov chain Monte Carlo approach to reconstructing ancestral genome rearrangements. Technical report, Carnegie Mellon University, Pittsburgh, 2002.

[16] B. M. E. Moret, A. C. Siepel, J. Tang, and T. Liu. Inversion medians outperform breakpoint medians in phylogeny reconstruction from gene-order data. In *Proceedings of WABI 2002*, volume 2452 of *LNCS*, pages 521–536. Springer Verlag, 2002.

[17] M. Ozery-Flato and R. Shamir. Two notes on genome rearrangements. Manuscript, 2002.

[18] D. Sankoff. Short inversions and conserved gene clusters. *Bioinformatics*, 18(10):1305–1308, 2002.

[19] A. Siepel. An algorithm to find all sorting reversals. In *Proceedings of RECOMB 2002*, pages 281–290. ACM Press, 2002.

[20] G. Tesler. Efficient algorithms for multichromosomal genome rearrangement. *J. Comput. Syst. Sci.*, 65(3):587–609, 2002.

[21] T. Uno and M. Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309, 2000.

Bisher erschienene Reports an der Technischen Fakultät
Stand: 2003-02-05

94-01 Modular Properties of Composable Term Rewriting Systems
(Enno Ohlebusch)

94-02 Analysis and Applications of the Direct Cascade Architecture
(Enno Littmann, Helge Ritter)

94-03 From Ukkonen to McCreight and Weiner: A Unifying View of Linear-Time Suffix Tree Construction
(Robert Giegerich, Stefan Kurtz)

94-04 Die Verwendung unscharfer Maße zur Korrespondenzanalyse in Stereo Farbbildern
(André Wolfram, Alois Knoll)

94-05 Searching Correspondences in Colour Stereo Images – Recent Results Using the Fuzzy Integral
(André Wolfram, Alois Knoll)

94-06 A Basic Semantics for Computer Arithmetic
(Markus Freericks, A. Fauth, Alois Knoll)

94-07 Reverse Restructuring: Another Method of Solving Algebraic Equations
(Bernd Bütow, Stephan Thesing)

95-01 PaNaMa User Manual V1.3
(Bernd Bütow, Stephan Thesing)

95-02 Computer Based Training-Software: ein interaktiver Sequenzierkurs
(Frank Meier, Garrit Skrock, Robert Giegerich)

95-03 Fundamental Algorithms for a Declarative Pattern Matching System
(Stefan Kurtz)

95-04 On the Equivalence of E-Pattern Languages
(Enno Ohlebusch, Esko Ukkonen)

96-01 Static and Dynamic Filtering Methods for Approximate String Matching
(Robert Giegerich, Frank Hischke, Stefan Kurtz, Enno Ohlebusch)

96-02 Instructing Cooperating Assembly Robots through Situated Dialogues in Natural Language
(Alois Knoll, Bernd Hildebrand, Jianwei Zhang)

96-03 Correctness in System Engineering
(Peter Ladkin)

96-04 An Algebraic Approach to General Boolean Constraint Problems
(Hans-Werner Güsgen, Peter Ladkin)

96-05 Future University Computing Resources
(Peter Ladkin)

96-06 Lazy Cache Implements Complete Cache
(Peter Ladkin)

96-07 Formal but Lively Buffers in TLA+
(Peter Ladkin)

96-08 The X-31 and A320 Warsaw Crashes: Whodunnit?
(Peter Ladkin)

96-09 Reasons and Causes
(Peter Ladkin)

96-10 Comments on Confusing Conversation at Cali
(Dafydd Gibbon, Peter Ladkin)

96-11 On Needing Models
(Peter Ladkin)

96-12 Formalism Helps in Describing Accidents
(Peter Ladkin)

96-13 Explaining Failure with Tense Logic
(Peter Ladkin)

96-14 Some Dubious Theses in the Tense Logic of Accidents
(Peter Ladkin)

96-15 A Note on a Note on a Lemma of Ladkin
(Peter Ladkin)

96-16 News and Comment on the AeroPeru B757 Accident
(Peter Ladkin)

97-01 Analysing the Cali Accident With a WB-Graph
(Peter Ladkin)

97-02 Divide-and-Conquer Multiple Sequence Alignment
(Jens Stoye)

97-03 A System for the Content-Based Retrieval of Textual and Non-Textual Documents Based on Natural Language Queries
(Alois Knoll, Ingo Glöckner, Hermann Helbig, Sven Hartrumpf)

97-04 Rose: Generating Sequence Families
(Jens Stoye, Dirk Evers, Folker Meyer)

97-05 Fuzzy Quantifiers for Processing Natural Language Queries in Content-Based Multimedia Retrieval Systems
(Ingo Glöckner, Alois Knoll)

97-06 DFS – An Axiomatic Approach to Fuzzy Quantification
(Ingo Glöckner)

98-01 Kognitive Aspekte bei der Realisierung eines robusten Robotersystems für Konstruktionsaufgaben
(Alois Knoll, Bernd Hildebrandt)

98-02 A Declarative Approach to the Development of Dynamic Programming Algorithms, applied to RNA Folding
(Robert Giegerich)

98-03 Reducing the Space Requirement of Suffix Trees
(Stefan Kurtz)

99-01 Entscheidungskalküle
(Axel Saalbach, Christian Lange, Sascha Wendt, Mathias Katzer, Guillaume Dubois, Michael Höhl, Oliver Kuhn, Sven Wachsmuth, Gerhard Sagerer)

99-02 Transforming Conditional Rewrite Systems with Extra Variables into Unconditional Systems
(Enno Ohlebusch)

99-03 A Framework for Evaluating Approaches to Fuzzy Quantification
(Ingo Glöckner)

99-04 Towards Evaluation of Docking Hypotheses using elastic Matching
(Steffen Neumann, Stefan Posch, Gerhard Sagerer)

99-05 A Systematic Approach to Dynamic Programming in Bioinformatics. Part 1 and 2: Sequence Comparison and RNA Folding
(Robert Giegerich)

99-06 Autonomie für situierte Robotersysteme – Stand und Entwicklungslinien
(Alois Knoll)

2000-01 Advances in DFS Theory
(Ingo Glöckner)

2000-02 A Broad Class of DFS Models
(Ingo Glöckner)

2000-03 An Axiomatic Theory of Fuzzy Quantifiers in Natural Languages
(Ingo Glöckner)

2000-04 Affix Trees
(Jens Stoye)

2000-05 Computergestützte Auswertung von Spektren organischer Verbindungen
(Annika Büscher, Michaela Hohenner, Sascha Wendt, Markus Wiesecke, Frank Zöllner, Arne Wegener, Frank Bettenworth, Thorsten Twellmann, Jan Kleinlützum, Mathias Katzer, Sven Wachsmuth, Gerhard Sagerer)

2000-06 The Syntax and Semantics of a Language for Describing Complex Patterns in Biological Sequences
(Dirk Strothmann, Stefan Kurtz, Stefan Gräf, Gerhard Steger)

2000-07 Systematic Dynamic Programming in Bioinformatics (ISMB 2000 Tutorial Notes)
(Dirk J. Evers, Robert Giegerich)

2000-08 Difficulties when Aligning Structure Based RNAs with the Standard Edit Distance Method
(Christian Büschking)

2001-01 Standard Models of Fuzzy Quantification
(Ingo Glöckner)

2001-02 Causal System Analysis
(Peter B. Ladkin)

2001-03 A Rotamer Library for Protein-Protein Docking Using Energy Calculations and Statistics
(Kerstin Koch, Frank Zöllner, Gerhard Sagerer)

2001-04 Eine asynchrone Implementierung eines Microprozessors auf einem FPGA
(Marco Balke, Thomas Dettbarn, Robert Homann, Sebastian Jaenicke, Tim Köhler, Henning Mersch, Holger Weiss)

2001-05 Hierarchical Termination Revisited
(Enno Ohlebusch)

2002-01 Persistent Objects with O2DBI
(Jörn Clausen)

2002-02 Simulation von Phasenübergängen in Proteinmonoschichten
(Johanna Alichniewicz, Gabriele Holzschneider, Morris Michael, Ulf Schiller, Jan Stallkamp)

2002-03 Lecture Notes on Algebraic Dynamic Programming 2002
(Robert Giegerich)

2002-04 Side chain flexibility for 1:n protein-protein docking
(Kerstin Koch, Steffen Neumann, Frank Zöllner, Gerhard Sagerer)

2002-05 ElMaR: A Protein Docking System using Flexibility Information
(Frank Zöllner, Steffen Neumann, Kerstin Koch, Franz Kummert, Gerhard Sagerer)

2002-06 Calculating Residue Flexibility Information from Statistics and Energy based Prediction
(Frank Zöllner, Steffen Neumann, Kerstin Koch, Franz Kummert, Gerhard Sagerer)

2002-07 Fundamentals of Fuzzy Quantification: Plausible Models, Constructive Principles, and Efficient Implementation
(Ingo Glöckner)

2002-08 Branching of Fuzzy Quantifiers and Multiple Variable Binding: An Extension of DFS Theory
(Ingo Glöckner)