Universität Bielefeld

Technische Fakultät
Abteilung Informationstechnik
Forschungsberichte

Alignment of Tandem Repeats with Excision, Duplication, Substitution and Indels (EDSI)

Michael Sammeth Jens Stoye

Report 2005-05

Impressum: Herausgeber:
Robert Giegerich, Ralf Hofestädt, Franz Kummert,
Peter Ladkin, Ralf Möller, 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

Alignment of Tandem Repeats with Excision, Duplication, Substitution and Indels (EDSI)

Michael Sammeth and Jens Stoye

Technische Fakultät, Universität Bielefeld, Germany.

Abstract. Traditional sequence comparison by alignment applies a mutation model comprising two events, substitutions and indels (insertions or deletions) of single positions (SI). However, modern genetic analysis knows a variety of more complex mutation events (e.g., duplications, excisions and rearrangements), especially regarding DNA. With the ever more DNA sequence data becoming available, the need to accurately compare sequences which have clearly undergone more complicated types of mutational processes is becoming critical.
Herein we introduce a new model, where in total four mutational events are considered: excision and duplication of tandem repeats, as well as substitutions and indels of single positions (**EDSI**). Assuming the EDSI model, we develop a new algorithm for pairwisely aligning and comparing DNA sequences containing tandem repeats. To evaluate our method, we apply it to the *spa* VNTR (variable number of tandem repeats) of *Staphylococcus aureus*, a bacterium of great medical importance.

1 Introduction

Sequence alignment is a rather well established tool to compare biological sequences. To align sequences, so-called *edit operations* have been defined which represent the atomic steps of the biological phenomenon called evolution. By successively applying such edit operations, the compared sequences can be converted into each other and - assuming parsimony as a major characteristic of evolution - good sequence alignments minimize the number of operations for these conversions or, more precisely, the assigned *costs*. In the classical model of mutation, two different edit operations are considered: the substitution and the insertion or deletion (together indel) of single characters in a sequence. In accordance with other literature [2], we will refer to this as the **SI model** (for substitution and indels) further on.

In general sequence alignments, the SI model has proven to work well. However, modern genetics knows more complex sources of mutation, especially when regarding the evolution of DNA. These mechanisms affect no longer only single positions but complete subareas of a sequence. Common other edit operations are *duplications* (insertions of copied subsequences, in case of *tandem* duplications, immediately adjacent to the original), *excisions* (deletions of subareas of a sequence), and *rearrangements* (relocations or reorientations of substrings within the sequence, e.g. *transpositions* or *inversions*).

In recent years quite some work has been invested in the algorithmic investigation of tandem repeats. Tandem duplications and excisions follow different rules than regular, character-based indels. On one hand the inserted or deleted substrings are usually much bigger in duplications and excisions, and on the other hand they contain the pattern of the tandem repeats in the corresponding sequence. Preliminary work in this field roughly is categorized into (1) tandem repeat *detection*, (2) *alignment* of sequences containing tandem repeats (with or without knowledge of their positions), and (3) reconstruction of a tandem *repeat history* where the phylogenetic history of the tandem repeats of one sequence is tracked down to a single ancestor repeat. (1) concerns the detection of tandem repeat copies with an unknown pattern [10]. In the context of (2) various works extended the SI mutation model to additionally respect tandem duplication events (**DSI model**), e.g. in [2, 4, 1]. The research of (3) investigates possible duplication histories of the tandem repeats in a sequence. These are represented by duplication phylogenies which under certain conditions can be turned into rooted duplication trees, see [3, 11, 7, 5].

Staphylococcus aureus (*S. aureus*), a bacterium responsible for a wide range of human diseases (e.g., endocarditis, toxic shock syndrome, skin, soft tissue and bone infections etc. [17]), contains polymorphic 24-bp variable-number tandem repeats (VNTRs) in the 3' coding region of the staphylococcal protein A (the *spa* protein) [6]. The tandem repeats in this region undergo a mutational process including duplication and excision events in addition to nucleotide-based substitutions and indels [12], probably caused by slipped strand mispairing [18]. Further on, the microvariation of the *spa* VNTR cluster [13] seems to support the phylogenetic signal reported by other methods (e.g., by [15]). Therefore, an automated method to compare strains of *S. aureus* and classify them according to the microvariation of the *spa* tandem repeats is critical in order to determine the types of newly acquired sequences rapidly and accurately.

In this paper, we introduce a novel model of evolution, the **EDSI model** (excisions, duplicatons, substitutions and indels), which in addition to the DSI model includes repeat excision operations. Moreover, the restrictions on the order of mutation events are relaxed: all four edit operations may occur arbitrarily cascaded with each other. In Section 2 we formalize the EDSI model and give an overview of the problem addressed. Next, in Section 3, we propose an exact algorithm to align and compare a pair of sequences under the EDSI model of mutation. Finally, in Section 4, we give some practical examples for comparing *spa* sequences of *S. aureus* with the novel method and Section 5 summarizes the benefit of the EDSI model and outlines its potential for accurate phylogenetic investigations.

2 Description of the EDSI Model

Let s be a sequence of characters over the DNA alphabet $\Sigma = \{\text{A,C,G,T}\}$, and let s contain tandem repeats. If the boundaries of the repeats are known, s can be written directly as a sequence s' over the macro alphabet of the different

repeat types $\Sigma' = \{-, A, B, C, D, \ldots\}$. The additional gap character ($- \in \Sigma'$) is used later on when aligning repeat sequences (Section 3). $(\Sigma')^+$ denotes the set of all nonempty strings over the repeat alphabet Σ'. On s' we define the EDSI evolution, allowing duplication and excision of repeats (characters in s'), as well as substitutions and indels of nucleotides within the repeats. Note that the commonly used substitution and indel operations work on the DNA bases of s, and therefore are comprised in the term *mutation* of a tandem repeat. In contrast, the duplication and excision events affect complete repeats of s' (Fig. 1).

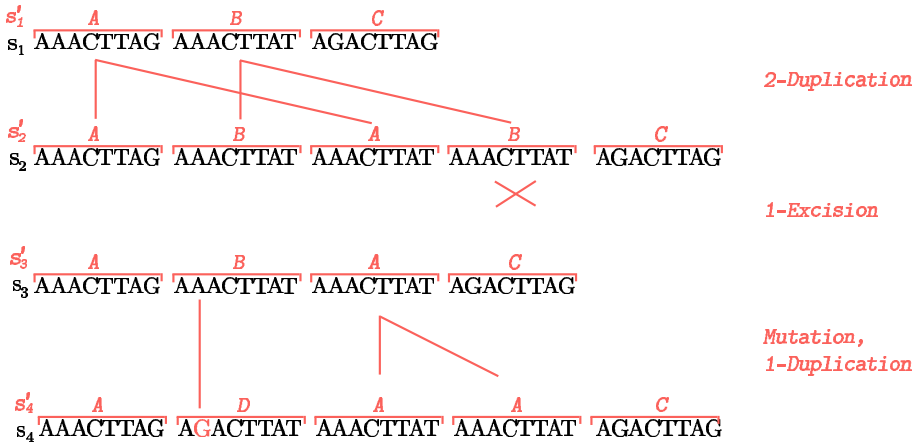


Fig. 1. An example for cascaded duplication, excision, and mutation events. Shown are DNA sequences s_i and the corresponding sequences s'_i on the macro alphabet Σ' of repeat types (superimposed on s_i in grey). Some edit operations (as given to the right) successively are performed on the sequence. It can be easily seen that after a couple of cascaded operations the sequence of characters is rather scrambled.

Precisely, the duplication events occurring in the evolution of the *spa* repeat cluster are *multi-copy* duplications (1-duplication, 2-duplication, etc.) copying one or more adjacent repeat copies at a time. The *boundaries* of the duplicated repeats are restricted to the boundaries of tandem repeats on the nucleotide sequence s. However, on s' the duplication boundaries are *free* in the characters of the macro alphabet Σ', i.e., duplicated substrings may start and end anywhere in s'. Finally, the duplication operation in the EDSI model is *single-step*, denoting that no more than one copy of a duplicated substring is produced in one evolutionary step. In the same manner, the excision operation of the model is characterized as multi-copy (1-excision, 2-excision, etc.) with free boundaries on s'. The order of events in EDSI is unrestricted. To be specific, all four edit operations described by the model may be applied arbitrarily cascaded with each other (Fig. 1).

In order to assess the evolutionary distance between two given sequences, we assign *costs* to all operations comprised in the EDSI model: $cost_e(w)$ for the excision of the tandem repeat copies in string w, $cost_d(w)$ for a duplication of the tandem repeats in string w, and $cost_m(w_1, w_2)$ for a mutation of a repeat type w_1 into the repeat type w_2. The **cost model** of EDSI evolution then can be freely adjusted[1] with respect to the following criteria:

- Excision costs should be positive, $cost_e(w) > 0$ for all $w \in (\Sigma')^+$, since excision events can replace all other operations. To be specific, any non-identical pair of sequences (s', t') can be derived from a concatenated ancestor string $s't'$ by two excisions (once excising s' to reconstruct t' and once exising t' to deduce s' from the ancestor). Hence, finding the minimum distance for sequences in a cost model with $cost_e(w) = 0$ is trivial.
- Duplication costs should be non-negative, $cost_d(w) \geq 0$ for all $w \in (\Sigma')^+$.
- Mutation costs should comply with the properties of a metric: symmetry ($cost_m(w_1, w_2) = cost_m(w_2, w_1)$ for all $(w_1, w_2) \in \Sigma'$), zero property ($cost_m(w, w) = 0$ for all $w \in \Sigma'$), and the triangle inequality ($cost_m(w_1, w_2) + cost_m(w_2, w_3) \geq cost_m(w_1, w_3)$ for all $w_1, w_2, w_3 \in (\Sigma')^+$).

The problem of sequence evolution comprising EDSI operations can now be formulated as an optimization problem with the goal of minimizing the EDSI distance defined in the following: Given two sequences (s', t') and cost measures $cost_e$, $cost_d$, and $cost_m$, find the distance $d(s', t')$ under the EDSI evolution that is the minimum sum of costs of all series of operations possible to reproduce one sequence from the other. This can be interpreted such that both sequences s' and t' are subjected to evolutionary operations in order to transform them into a common string – a possible common ancestor according to the biological model. The operations that produce a common ancestor of s' and t' with the least costs define the EDSI distance $d(s', t')$.

Theorem 1 (finiteness). *The edit operations under EDSI evolution and their unrestricted order basically force us to explore an infinite search space of possible ancestor sequences. However, the space of operation sets to be explored in order to find the minimal distance between two sequences $d(s', t')$ is finite.*

Proof. When reconstructing possible evolutionary histories from a given pair of sequences (s', t'), theoretically there could have been present an arbitrarily large number of repeats between two adjacent positions x and $x + 1$ of s' (or, symmetrically, t') which later were deleted with cost $cost_e > 0$. There are two possible sources for these deleted repeats in the tandem-repeat history: (i) they may have emerged from duplication events, or (ii) they may have been repeats from non-duplicated sequence areas. Cnsider case (i). If a deleted repeat has originated from a duplication event, the corresponding excision can be detected by investigating all possible duplication events to the left (in $s'[1, x]$) and to the

[1] For a definition of the costs used for the *Staphylococcus aureus* evolution see Section 4.

right (in $s'[x+1, |s'|]$). The number of single duplication events on a finite string is limited, and so is the number of possibly excised repeat units between x and $x+1$. Moreover, all character insertions between x and $x+1$ induced by possible duplication events with consecutive excisions are collected (Section 3.1 and 3.2) and taken into account in the final comparison between s' and t' (Section 3.3). Consider now case (ii). If deleted substrings have originated from non-duplicated sequence areas (or whenever the second repeat copy of a duplication has been excised as well), they are not relevant in the search for a minimal distance: in the comparison of s' with t' an appropriate excision event will be detected (Section 3.3), whenever t' has a substring that aligns between $s'[x]$ and $s'[x+1]$. However, if the alignment with t' does not indicate any presumptive excision between x and $x+1$ in s', all such theoretically possible excisions are not contained in the operations determining the minimum distance since additional excision costs $cost_e > 0$ produced an ancestor sequence that is not closer to t' than the original sequence s'. □

3 Pairwise Alignment under the EDSI Model

After the definition of the EDSI model, we can describe an exact algorithm to compare and align sequences with respect to the four edit operations. The main idea of our method is to find possible repeat histories, assign costs to them according to the edit operations, and consider them as alternatives during an alignment procedure. Thereby the alignment possibility between both sequences with the least cost is selected, regarding the original sequences with all contracted substrings generated by the repeat histories. Assuming the parsimony principle for nature we take these costs as a distance measure for the compared sequences. So basically our algorithm works in two steps: first it finds possible duplications on each sequence under the rules given by the EDSI model. Afterwards, we determine the distance between a sequence pair in a high-dimensional multiple sequence alignment (MSA) using the duplication events found before as alternative alignment possibilities between the compared sequences.

Although not observed in biology, we also use the term *contraction* for the mathematically inverse process of a duplication. Our technique is based on *contramers* $C = (s', b, m, e, A)$, representing contraction units. These are substrings of s' (the macro alphabet representation of the input string s) on which a contraction is performed. The substring to be contracted, $s'[b, e]$, is located within s' by its beginning b and its end position e. The *meridian* $m, (b < m \leq e)$ splits the contramer into two *segments*, also called the *prefix* (the first segment $s'[b, m-1]$) and the *suffix* (the second segment $s'[m, e]$). Finally, the alignment A of the prefix and the suffix describes how the characters of both segments are evolutionarily related according to the contramer. To be specific, aligned repeats correspond to each other (with respect to possible mutation events) and gaps indicate the excision of repeats. An example of a contramer representing a duplication event (including mutation and excision) is given in Fig. 2.

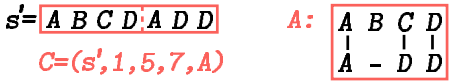


Fig. 2. A contramer $C = (s', 1, 5, 7, A)$ that implies the duplication of substring $s'[1, 4] = ABCD$ and its post-duplicational modification into ADD. The alignment (grey box) shows that repeat B was excised while repeat C mutated to repeat D. All vertically adjacent repeat pairs (i.e., non-gap characters) in an alignment layout correspond to each other w.r.t. possible mutations. These links (black lines) are not explicitly visualized in further representations of an alignment.

3.1 Primary library of contramers

The initial set of contramers is extracted directly from the repeat sequence s'. For each meridian position in s', $1 < m \leq |s'|$, all alignment possibilities of available non-empty prefixes $s'[b, m-1]$, $1 \leq b < m$, and non-empty suffixes $s'[m, e]$, $m \leq e \leq |s'|$, are generated. The contramers inferred thereby form the *primary library*. Note that at this stage the similarity of the aligned segments is not optimized by any objective function since links between amalgamated contramers later on can involve new repeat copies (i.e., characters of s'). Contramers in the primary library represent possible duplication events, i.e. links of positions in neighboring segments.

Algorithm 1 (Generate the contramers for the primary library L)

```
1: L ← ∅
2: for m ← 2 to |s'| do
3:    for b ← 1 to (m − 1) do
4:       for e ← m to |s'| do
5:          AP[] ← GeneratePossibleAlignments(b, m, e)
6:          for all A in AP[] do
7:             Store(L, C = (s', b, m, e, A))
8:          end for
9:       end for
10:   end for
11: end for
```

Algorithm 1 outlines the technique used to assemble the primary library of contramers. As input serves a sequence s' on the alphabet of tandem repeats Σ'. The resulting list L contains each possible contramer $C = (s', b, m, e, A)$ of s'. The *cost* of a contramer may be derived directly from the associated alignment A by adding, for each column of the alignment, the costs of mutations or excisions. In addition, to reflect the costs for the duplication event, $cost_d(s'[b, (m-1)])$ is added, yielding the final cost of contramer $C = (s', b, m, e, A)$:

$$cost(C) = cost_d(s'[b,(m-1)]) + \sum_{i=1}^{|A|} \begin{cases} cost_m(A_{1i}, A_{2i}) \text{ for each mutation } (A_{1i} \neq - \neq A_{2i}) \\ cost_e(A_{1i}) \text{ for each excision in the prefix } (A_{1i} \neq -,\ A_{2i} = -) \\ cost_e(A_{2i}) \text{ for each excision in the suffix } (A_{1i} = -,\ A_{2i} \neq -). \end{cases}$$

Theorem 2 (completeness of the primary library). *Contramers contained in the primary library exhaustively generate all ancestor strings that can be derived from a sequence by reversing exactly one duplication event.*

Proof. The exhaustive search over possible starts b, meridians m and end points e of contramers guarantees that for any pair of non-overlapping repeat substrings $(s'[x_1, y_1], s'[x_2, y_2])$, $y_1 < x_2$ there exists at least one contramer $C = (s', b, m, e, A)$ in the primary library that contains both substrings in different segments $b \leq x_1 \leq y_1 < m \leq x_2 \leq y_2 \leq e$. Since furthermore all possible alignments between the segments are generated (Algorithm 1), all possibilities to map characters from one segment $s'[x_1, y_1]$ on positions of the other segment $s'[x_2, y_2]$ are taken into account. It should be stressed that the segments $s'[x_1, y_1]$ and $s'[x_2, y_2]$ need not match each other's lengths, since by gap characters in the alignment (excision events) substrings of different lengths $(y_1 - x_1) \neq (y_2 - x_2)$ can be mapped on each other. The argumentation for the redundancy of contramers with an empty segment is similar to the one for limitation of the search space given in Theorem 1.

3.2 The secondary library

In order to infer cascaded duplication histories, overlapping primary contramers C_1 and C_2 are to be merged to form cascaded duplication events. Abusing notation, we define a contramer intersection (union) as the intersection (union) of the corresponding segments of s', i.e. $C_1 \cap C_2 = \{b_1, \ldots, e_1\} \cap \{b_2, \ldots, e_2\}$ $(C_1 \cup C_2 = \{b_1, \ldots, e_1\} \cup \{b_2, \ldots, e_2\})$.

If $C_1 \cap C_2$ comprises positions of both segments of C_1, we call C_1 a *contained* contramer (and C_2 a *containing* contramer). Otherwise, C_1 and C_2 are *connected* contramers. However, not all overlapping duplication events are necessarily *compatible* with each other. The precondition for a pair of compatible contramers (C_1, C_2) is that they can be realized in a common *evolutionary order*, i.e., there exists at least one repeat history tree comprising both described duplication events.

Evolutionary order of compatible contramers. The common evolutionary realizability can be deduced from analyzing the intersection of the two contramers $C_1 \cap C_2$. In evolution, the duplication events described by contained contramers must have happened before the duplication events of the contramer they are contained in (Fig. 3a), whereas for a pair of connected contramers the evolutionary order does not matter (Fig. 3b). Two contramers mutually contained in each other are not realizable in a common repeat history (Fig. 3c), even if they share the same meridian position m (Fig. 3d).

Fig. 3. Restrictions on compatible contramer pairs $C_1 = (s', b_1, m_1, e_1, A_1)$ and $C_2 = (s', b_2, m_2, e_2, A_2)$ (grey boxes, meridian position indicated by a dashed line). Possible repeat histories expressed by the amalgamation $C_1 \cup C_2$ are depicted on the right. (a) C_1 is *contained* in C_2, therefore the evolutionary order is fixed and the duplication captured in C_1 must have happened before the one described by C_2 (only one possible repeat history). (b) Merging two *connected* contramers imposes no order on the evolution (i.e., the duplication of C_1 or C_2 may have happened first). (c) and (d) If none of the contramers has a non-intersecting segment, $\{m_1 - 1, m_1, m_2 - 1, m_2\} \notin C_1 \cap C_2$, *no repeat history* can be found incorporating both duplication events captured by the contramers. This holds even if the meridians coincide, $m_1 = m_2$, see (d).

Lemma 1 (merging conditions). *Two contramers $C_1 = (s', b_1, m_1, e_1, A_1)$ and $C_2 = (s', b_2, m_2, e_2, A_2)$ are compatible and can be merged if:*

1. *they overlap, $C_1 \cap C_2 \neq \emptyset$ (*connectibility*) and*
2. *one of the reflected duplication events has happened after the other one. Therefore at least the contramer C_1 needs to have a segment outside of the intersection area, $(m_1 - 1) \notin C_1 \cap C_2$ or $m_1 \notin C_1 \cap C_2$ (*compatibility*).*

Lemma 1 describes the preconditions that are to be met to merge a pair of contramers. Afterwards, C_1 and C_2 are merged into $C_1 \cup C_2$ by combining their respective alignments: any repeat $s'[x]$ in the overlapping area $C_1 \cap C_2$ may be linked to two other repeat copies $s'[y] \in (C_1 \setminus C_2)$ and $s'[z] \in (C_2 \setminus C_1)$ by the alignments A_1 and A_2. Thereby a *transitive link* between both of the not directly associated repeats $s'[y]$ and $s'[z]$ is created. All three repeats $(s'[x], s'[y], s'[z])$ are then written in one column of the merged alignment (Fig. 4a). Problems arise when both contramers comprise excisions in between corresponding positions of the overlapping area (Fig. 4b). In this case the contramers do not provide a

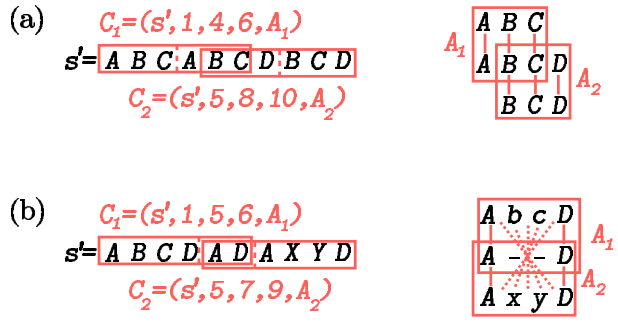


Fig. 4. Transitive links when merging contramers. (a) A pair of partially overlapping contramers where e.g. C_1 connects the positions $(2, 5)$ and C_2 links position 5 with 8. The transitive link created when merging A_1 with A_2 links all three B-characters together (2nd column of the merged alignment to the right). (b) The amalgamation of a pair of contramers (C_1, C_2) which are both inducing characters in the same area. Consequently, the phylogenetic relation of the characters (lowercase) cannot be exactly determined (possible relations are indicated by the dotted grey lines).

unique information about the transitive relation between the excised characters. One possibility would be to exhaustively generate all the alignment possibilities between the respective characters. However, since we are only interested in finding a "good" combination of characters minimizing the distance to another sequence, we let these ambiguous repeats unaligned for the moment and search for a combination similar to the compared sequence later on in the comparison step (Section 3.3).

The merging strategy is straightforwardly extendable to deal with more than two contramers. A set of combinable contramers $\{C_1, C_2, \ldots, C_k\}$ obviously requires that each of the contramers C_i, $i \in \{1, 2, \ldots, k\}$ has to fulfill the precondition of connectibility with at least one other contramer $j \in \{1, 2, \ldots, k\}, j \neq i$. Otherwise C_i is isolated and cannot be joined. Furthermore, it is required that each pair of overlapping contramers (C_i, C_j), $i, j \in \{1, 2, \ldots, k\}$, is compatible. The order in which the contramers are joined is not important since the merge operations are commutative:

Theorem 3 (commutativity). *The pairwise merging steps of multiply joined contramers are* commutative, $(C_1 \cup C_2) = (C_2 \cup C_1)$.

Proof. Two contramers $C_1 = (s', b_1, m_1, e_1, A_1)$ and $C_2 = (s', b_2, m_2, e_2, A_2)$ when joined merge the links realized in A_1 and in A_2 to form transitive links. A transitive link joining positions $(s'[x], s'[y], s'[z])$ is equally created either by taking the link $(s'[x], s'[y])$ and extending it by the link $(s'[y], s'[z])$ or by starting with $(s'[y], s'[z])$ and joining it to the link $(s'[x], s'[y])$. □

Figure 5 demonstrates that when performing a multiple amalgamation with all preconditions met by the contramers $\{C_1, C_2, \ldots, C_k\}$, we perform succes-

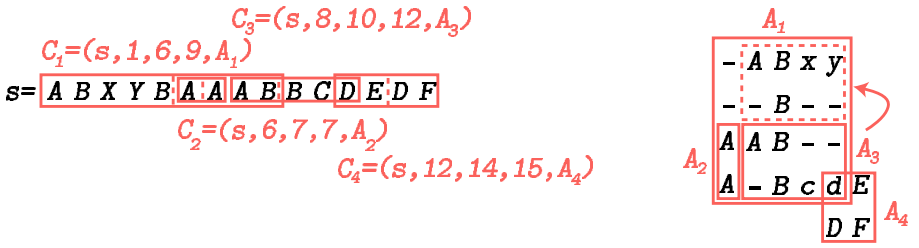


Fig. 5. A set of multiply merged contramers ($C_1 \cup C_2 \cup C_3 \cup C_4$) and the respective concatenated alignment. Note that lowercase characters are not uniquely aligned by the transitive links of the contramers, and their position is determined later during the comparison process (Section 3.3).

sively the merging step for each pair of overlapping contramers (C_i, C_j) such that $i \neq j$, $C_i \cap C_j \neq \emptyset$.

Algorithm 2 describes the construction of contramers in the secondary library. Initially, L comprises the contramers already included in the primary library. The set of contramers with beginning b, meridian m, and end e can be accessed via the function GETC(L, b, m, e). The functions FINDCONNECTEDC() and FINDCONTAINEDC() extract contramers in a given subarea (specified by the start and end point). For each pair of overlapping contramers the preconditions are checked before returned (set $DP[]$). In the end, compatible contramers F are merged with C, and the result is added to L.

Theorem 4 (completeness of the secondary library). *Contramers contained in the secondary library generate all ancestor strings that can be derived from a sequence under the EDSI model containing one or more duplication events.*

Proof. The iteration order (over ascending meridian, start and end positions) of Algorithm 2 guarantees to (i) find all cascaded duplication events of *contained* contramers by investigating only the prefix area of any iterated contramer C. Additionally, (ii) all overlapping chains of connected contramers are found by looking in the suffix of C.

(i) When processing a certain contramer $C = (s', b, m, e, A)$, all contramers C' that can be contained in the prefix of C have already been processed earlier. Since A links the characters of the prefix $s'[b, m-1]$ with characters in the suffix $s'[m, e]$, contramers contained in the suffix of C are implicitely induced by their counterparts contained in the prefix of C. The same argumentation holds for already concatenated groups of contramers.

(ii) Connected contramers are only to be searched in the suffix $s'[m, e]$ (i.e., the suffix with the biggest end position case of already merged chains contramers). A contramer $C' = (s', b', m', e', A')$ connected to the suffix of C will be iterated later in the cascaded loops of Algorithm 2 since $m < m'$. Formulated

Algorithm 2 (Amalgamate contramers to build the secondary library L)

```
 1: L ← PrimaryLibrary()
 2: for m ← 2 to |s'| do
 3:    for b ← 1 to (m − 1) do
 4:       for e ← m to |s'| do
 5:          CP[] ← getC(L, b, m, e)
 6:          for all C in CP[] do
 7:             DP[] ← findConnectedC(L, m, e)
 8:             for all D in DP[] do
 9:                F ← Merge(C,D)
10:                Store(L,F)
11:             end for
12:             DP[] ← findContainedC(L, b, m)
13:             for all D in DP[] do
14:                F ← Merge(C,D)
15:                Store(L,F)
16:             end for
17:          end for
18:       end for
19:    end for
20: end for
```

the other way round, all contramers connected to the prefix of C have already been processed when investigating C.

Note that in either case ((i) or (ii)) the product of amalgamation, $C \cup C'$, is inserted in L at a position that will be iterated again, so that chains of multiple connected contramers are merged step by step from left to right. □

3.3 Contramer Alignment

In the final alignment phase, the possible tandem repeat histories of two sequences (s', t') are used as alternative character combinations when comparing s' to t'. To this end, we extend the well established technique of dynamic programming (DP) for sequence alignment to additionally take into account the (cascaded) duplications. The contramers found along both sequences to be compared serve as additional alignment possibilities, i.e., cells extending the regular DP matrix. For each cell (i, j) to be computed in the DP recursion of the main matrix M of size $|s'| \times |t'|$, (merged) contramers ending at position i in s' (or at position j in t, respectively) are considered. The alignment profile of each comprimer $C = (s, b, m, e, A)$ can substitute the characters of the original sequence in the area $s'[b, e]$. Note that each cell (i, j) of the matrix M is connected by multiple contramers with any of the cells computed earlier during the DP process. Therefore, the resulting alignment is high-dimensional with multiple alternative submatrices for each contramer in both sequences (Fig. 6).

The matrix M can be computed by the following recursion formula:

$$M(i,j) = \min \begin{cases} M(i-1,j-1) + cost_m(s'[i], t'[j]) & //mutation \\ M(i-1,j) + cost_e(s'[i]) & //excision\ in\ s' \\ M(i,j-1) + cost_e(t'[i]) & //excision\ in\ t' \\ M(i-b_x,0) + cost(C_x) + align(t'[0,j], A_x) & \\ \quad \text{for all } C_x = (s', b_x, m_x, i, A_x) & //duplication\ in\ s' \\ M(0,j-b_y) + cost(C_y) + align(s'[0,i], A_y) & \\ \quad \text{for all } C_y = (s', b_y, m_y, j, A_y) & //duplication\ in\ t' \\ M(i-b_x,j-b_y) + cost(C_x) + cost(C_y) + align(A_x, A_y) & \\ \quad \text{for all } C_x = (s', b_x, m_x, i, A_x) & \\ \quad \text{and } C_y = (s', b_y, m_y, j, A_y) & //duplication\ in\ s'\ and\ t' \end{cases}$$

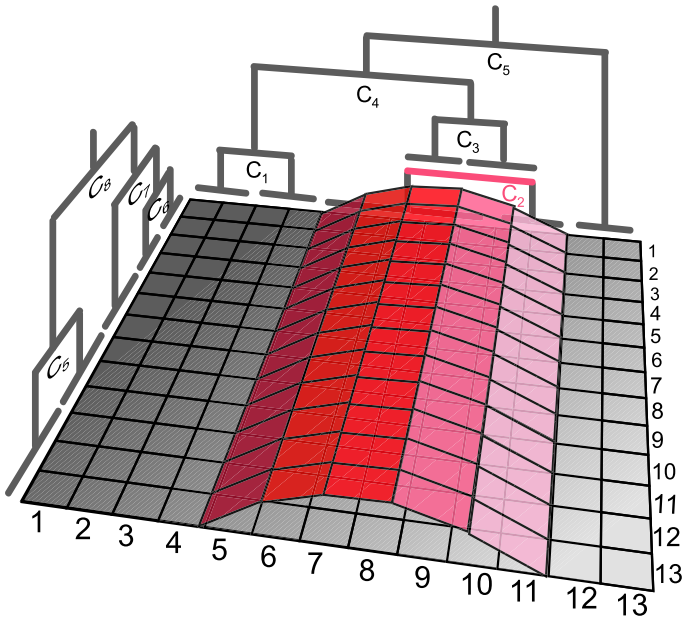


Fig. 6. An example for an alternative submatrix within a DP matrix $M = |s'| \times |t'|$. $C_2 = (t', 5, 10, 11, A_2)$ substitutes the substring $t'[5, 11]$ with the alignment A_2. Projected into M, C_2 spans the submatrix shown. During the DP process paths within the original and within the submatrix are taken into account when determining the optimum of the cells in column 12. Note that only contramers of one possible repeat history are depicted here, but all cascaded duplication events of the secondary library are investigated.

At each stage (i, j) of the alignment, the minimum cost is calculated for all edit operations of the EDSI model: mutation (line 1) of repeats (comprising substitutions and indels on the DNA alphabet Σ), excisions (lines 2 and 3) of

repeat copies (on the macro alphabet Σ') or duplication events (lines 4, 5 and 6). The preference of the algorithm is in the same order as given, and to optimize the performance a bounding step was added to only assess the alignment of contramers C which are not already exceeding the costs found earlier for a cell (i, j) by their cost $cost(C)$.

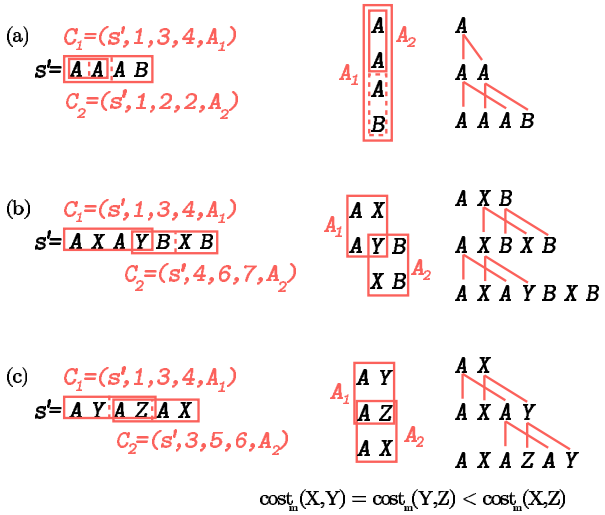


Fig. 7. Tandem duplication histories for contramer sets joined in different manners. Shown are the contramers on the linear sequence (left), the concatenated alignments (center), and a repeat history producing minimal cost (right). (a) A contained contramer always represents a duplication event that happened before the duplication of the contramer it is contained in. Note that the history displayed is the only one possible for the amalgamated contramers C_1 and C_2. (b) In partially overlapping contramers the cost-optimal list of evolutionary steps is to be found. Every segment has to be investigated as possible root from where the repeats evolved in order of their location within the sequence. The lists generated can produce different costs since it is not specified in the contramers, in which direction a substitution event happened and whether mutations happened before or after the duplication event. The given repeat history with $cost_m(X, Y)$ is cheaper than all other possible histories with $2cost_m(X, Y)$ (in addition to $2cost_d(s'[x, y])$ common to all histories inferred on the given contramers). (c) When overlapping completely, finding the history of connected contramers is a tandem repeat history problem first described by Fitch [9]. In the optimal history given here, w.r.t. the triangle inequality assumption for $cost_m$, the last evolutionary segment evolves directly from the first one. For only partially overlapping contramers such a non-linear evolutionary order is not possible.

As found earlier (Section 3.2, Fig. 4), in merged contramers not necessarily all of the transitive relations are clear. These positions are to be aligned within the amalgamated contramer taking also into account the sequence the contramer

is compared to. To this end we use a stable re-implementation of the hyperspace multiple sequence alignment procedure [16], which was modified to use the scoring function for repeat evolution, when aligning the amalgamated contramers with the corresponding compared sequence: all positions already aligned between the duplication events of the contramers are provided as constraints, whereas the ambiguous positions finally are aligned optimally according to the sequence information.

Costs of merged contramers. Joined contramers may contain concatenated alignment areas of three or more lines, corresponding to transitive links in the overlapping area of contramers. The sum-of-pairs-score normally used to measure multiple alignments is not suitable since it does not reflect the order of the segments, i.e., the lines in the alignment; in a regular alignment, the sequences (the lines) are independent of their order while the lines of contramer-alignments represent the order of evolutionary fragments within the input sequence. The scoring function has to respect this order of the evolutionary fragments, and all possible repeat histories are to be explored to find the minimum cost for transitively linked areas.

As described earlier, for *contained* contramers the evolutionary order is fixed, and they happened earlier than the contramers they are contained in (Fig. 7a). In case of *connecting* contramers, there are two possibilities. Either, the merged neighboring contramers C_1 and C_2 may *overlap partially*, i.e. $s'[m_1, e_1] \neq s'[b_2, m_2 - 1]$. Here, some of the characters of C_1 or C_2 are not transitively linked with the other contramer, implying that the evolutionary order must be a subsequent chain, starting at any of the involved evolutionary fragments (Fig. 7b). ALternatively, two connected contramers may *overlap completely*, to be specific, $s'[m_1, e_1] = s'[b_2, m_2 - 1]$. In this case the evolutionary order is arbitrary and all possible duplication tree topologies have to be investigated to find the history with least cost (Fig. 7c).

Figure 8 demonstrates that not only the topology of the repeat history, but also the time when a mutation event took place is crucial when calculating the costs. Specifically a mutation event performed before the corresponding duplication event creates a different pattern than if the same position is mutated only in one of the tandem copies after duplication. In a similar manner, the minimal costs of excisions in overlapping duplication events can be smaller than adding up the excision costs from the merged comprimers, respectively.

Therefore the scoring function for merged contramers has to explore all possible repeat histories w.r.t. the restrictions as set by the overlaps of the contramers (Fig. 7). Additionally, each modification (i.e., excisions and mutations) in the amalgamated contramers has to be tested before or after the respective duplication event. This exhaustive search yields the minimum cost for a cascade of contramers (Theorem 5). However, time and space complexity of the method are exponential w.r.t. the sequence length. Note that the input of the algorithm are sequences of already annotated repeats and the input size therefore is much smaller than the original sequences.

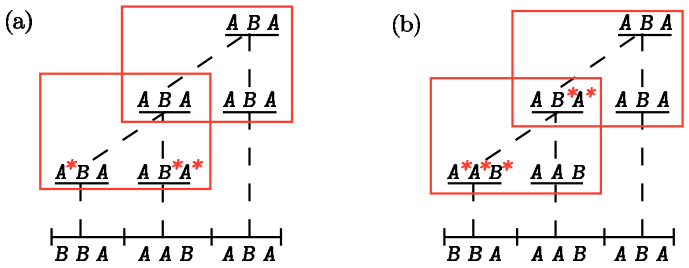


Fig. 8. Depicted is a repeat history for 3 repeats originating from two tandem duplication events (grey boxes). Although the topology of the repeat history is the same in (a) and (b), the calculated costs diverge when changing the time point a certain mutation (grey asterisk) happens. In (a), three mutation events are used ($3cost_d + 3cost_m(A, B)$), whereas in (b), in total five mutations are performed ($3cost_d + 5cost_m(A, B)$).

Theorem 5 (correctness of the method). *The distance $d(s', t')$ found by the DP recursion is the minimum distance possible in the comparison of s' and t', assuming the model of EDSI evolution.*

Proof. In the primary library all possible links between repeats of s' and t' that can originate from single duplication events, are generated (Theorem 2). Thus, by merging overlapping duplication events in the secondary library, we explore all possible cascades of duplications collected in the primary library (with restrictions to the biological model as given in Lemma 1, Theorem 4). On each of these cascades, excision events are tried before and after the respective duplication in order to yield the minimum costs according to the EDSI evolution (Fig. 7). Finally, in a high dimensional alignment all contramers extracted from s' and t' are used as alternative substrings imposing replacement costs as calculated. The minimum distance is then finally found by a DP recursion in a high dimensional alignment (Section 3.3). □

4 Results

To test our method, we applied it to the DNA sequences of *Staphylococcus aureus*. To be specific, the 5'-VNTR clusters in the gene encoding the *spa*-Protein were used as input for pairwise alignment under the EDSI model. Since the repeat patterns for all hitherto isolated strains (so-called *spa-types*) are known, the sequences are provided in characters of the macro alphabet Σ'. To this point, we use the *Kreiswirth* notation defined by identify the repeats $\Sigma' = \{A, A_2, B, B_2, C, C_2, \ldots, V, V_2, W, X, Y, Y_2, Z, Z_2\}$. In addition to the simplified alphabet used to introduce the model, in the Kreiswirth notation each letter may be used more than once in conjunction with a unique index [13].

Comparison of ST-254 Spa-types. Figure 10 shows an alignment of the *spa* repeat that we used to set up a simple cost scheme for the comparison of *spa*-types:

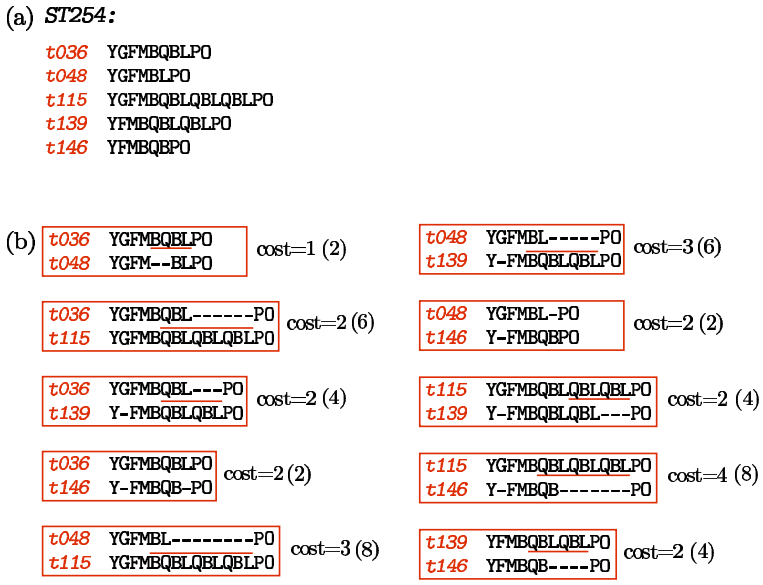
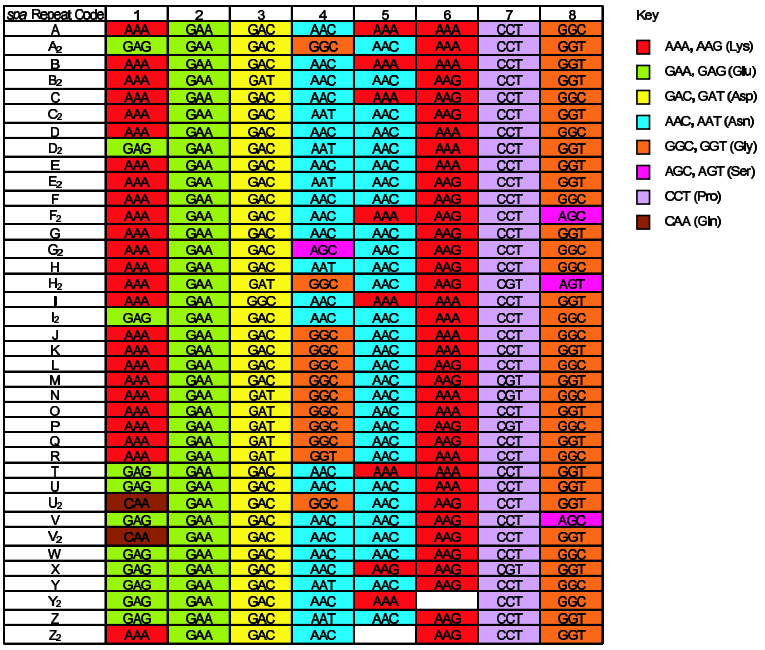


Fig. 9. Sequence comparison of the MLST sequence type ST-254. (a) a list of Spa-types found to have the ST-254 pattern. The data was acquired by sequencing from the same laboratory strains the VNTR cluster of the spa-protein and the MLST loci [14]. (b) One alignment that scores minimal costs for each pair of Spa-types from the ST-254 group. Substrings involved in duplication events leading to the minimum distance are underlined. To the right of the alignments the costs are given w.r.t. the EDSI model and in parentheses the costs under the SI model (without taking into account duplications). Under the EDSI model, the costs in the comparison of $t036$ and $t048$ are composed of a duplication event of the substring $t048[5,6] = BL$ and the mutation of repeat L into $t036[6] = Q$, yielding the distance $d(t036, t048) = cost_d(t048[5,6]) + cost_m(L,Q) = 1 + 0 = 1$.

since we are interested in a distance to measure evolutionary steps, we assign a unit cost u corresponding to the number of operations needed to perform the change. A duplication costs one operation ($cost_d(s'[x,y]) = u$), regardless of its length. The objective function to score mutation events (substitutions and indels of nucleotides) is based on a multiple alignment of all repeat types (Fig. 10). In order to contribute to the fact that the repeat cluster is coding, nucleotide substitutions changing effectively the corresponding codon are weighted with a cost of u, while silent mutations are omitted. In the same manner indels are penalized according to the number of codons x missing (xu). If not further specified, we set $u = 1$ in the tests. The mutation costs $cost_m(x,y)$ for $x, y \in \Sigma'$ are summed up along the pairwise DNA alignment of x and y which is projected from the global alignment of all repeats. Excisions are treated differently, we penalize them according to their length, such that $cost_c(s'[x,y]) = (y - x)$. The linear cost function prevents the algorithm from replacing all evolutionary events



spa Repeat Code	1	2	3	4	5	6	7	8
A	AAA	GAA	GAC	AAC	AAA	AAA	CCT	GGC
A_2	GAG	GAA	GAC	GGC	AAC	AAA	CCT	GGT
B	AAA	GAA	GAC	AAC	AAA	AAA	CCT	GGT
B_2	AAA	GAA	GAT	AAC	AAC	AAG	CCT	GGT
C	AAA	GAA	GAC	AAC	AAA	AAG	CCT	GGC
C_2	AAA	GAA	GAC	AAT	AAC	AAG	CCT	GGT
D	AAA	GAA	GAC	AAC	AAC	AAA	CCT	GGC
D_2	GAG	GAA	GAC	AAT	AAC	AAA	CCT	GGT
E	AAA	GAA	GAC	AAC	AAC	AAA	CCT	GGT
E_2	AAA	GAA	GAC	AAT	AAC	AAG	CCT	GGT
F	AAA	GAA	GAC	AAC	AAC	AAG	CCT	GGC
F_2	AAA	GAA	GAC	AAC	AAA	AAG	CCT	AGC
G	AAA	GAA	GAC	AAC	AAC	AAG	CCT	GGT
G_2	AAA	GAA	GAC	AGC	AAC	AAG	CCT	GGC
H	AAA	GAA	GAC	AAT	AAC	AAG	CCT	GGC
H_2	AAA	GAA	GAT	GGC	AAC	AAG	CGT	AGT
I	AAA	GAA	GGC	AAC	AAA	AAA	CCT	GGT
I_2	GAG	GAA	GAC	AAC	AAC	AAA	CCT	GGC
J	AAA	GAA	GAC	GGC	AAC	AAA	CCT	GGC
K	AAA	GAA	GAC	GGC	AAC	AAA	CCT	GGT
L	AAA	GAA	GAC	GGC	AAC	AAG	CCT	GGC
M	AAA	GAA	GAC	GGC	AAC	AAG	CGT	GGT
N	AAA	GAA	GAT	GGC	AAC	AAA	CGT	GGC
O	AAA	GAA	GAT	GGC	AAC	AAA	CCT	GGT
P	AAA	GAA	GAT	GGC	AAC	AAG	CGT	GGC
Q	AAA	GAA	GAT	GGC	AAC	AAG	CCT	GGT
R	AAA	GAA	GAT	GGT	AAC	AAA	CCT	GGC
T	GAG	GAA	GAC	AAC	AAA	AAA	CCT	GGT
U	GAG	GAA	GAC	AAC	AAC	AAA	CCT	GGT
U_2	CAA	GAA	GAC	GGC	AAC	AAG	CCT	GGT
V	GAG	GAA	GAC	AAC	AAC	AAG	CCT	AGC
V_2	CAA	GAA	GAC	AAC	AAC	AAG	CCT	GGT
W	GAG	GAA	GAC	AAC	AAC	AAG	CCT	GGC
X	GAG	GAA	GAC	AAC	AAG	AAG	CGT	GGT
Y	GAG	GAA	GAC	AAT	AAC	AAG	CCT	GGC
Y_2	GAG	GAA	GAC	AAC	AAA		CCT	GGC
Z	GAG	GAA	GAC	AAT	AAC	AAG	CCT	GGT
Z_2	AAA	GAA	GAC	AAC		AAG	CCT	GGT

Key

- AAA, AAG (Lys)
- GAA, GAG (Glu)
- GAC, GAT (Asp)
- AAC, AAT (Asn)
- GGC, GGT (Gly)
- AGC, AGT (Ser)
- CCT (Pro)
- CAA (Gln)

Fig. 10. DNA sequences of the *Spa* repeat types known hitherto. To the left, the letter denoting the repeat type in the *Kreiswirth* notation is given. The 24-bp repeats are grouped in triplets according to the reading frame. Shaded boxes indicate the different amino acids that are translated from the codons. Data adapted from [13].

by excisions when repeat copies are no longer exact. From another point of view, the scoring biases the algorithm to favor duplications and mutations and prefer them – up to a certain threshold – over possible excisions.

Since, to our knowledge, this is the first time the VNTR data of Spa-types is used to infer distance measures, we focus on one sequence type (ST-254), which by definition pools strains with the same types of the seven housekeeping genes used for MLST [8]. However, the resolution of STs found by MLST is lower than the microvariation within the *spa* repeat cluster. Thus, a ST group with an identical MLST pattern can pool several strains with diverging Spa-types (named by "t" and a 3-digit code), while the other way around a Spa-type may have evolved in different ST groups. Spa-types used in here to investigate the micro-variation of the repeats (i.e., t036, t048, t115, t139, and t146) were isolated in the laboratory from identical strain stocks [14]. Therefore, the microvariation of these Spa-types can be assumed to bear a phylogenetic marker (Fig. 9a).

Figure 9b summarizes the differences of applying the novel method based on the EDSI evolution in contrast to standard scoring functions for SI model comparisons. In order to compare the results, we adapted the scoring function of the SI alignment to the same values given for the EDSI evolution (xu for the insertion of x gaps and substitution costs according to non-synonymous muta-

tions, Figure 10). We want to stress on the fact, that the alignments shown are only one example from a set of alignments that can reproduce the minimal costs shown. As can be seen clearly, the distribution of the costs varies substantially between the EDSI model and the SI model of evolution. This results from the simple fact that the SI scoring scheme is not capable to distinguish duplication events from excisions and cannot score them differently. The resolution of the EDSI events is therefore higher, which results in a more subtle analysis when comparing the distances.

For instance, the minimal costs of the alignments in Fig. 9b can be calculated as follows. (Note that mutation costs with $cost_m = 0$ are omitted. Therefore, character changes are possible between repeats that do only differ by silent mutations.)

$$
\begin{aligned}
d(t036, t115) &= 2cost_d(t036[6,8]) = 2 \\
d(t036, t139) &= cost_d(t036[6,8]) + cost_e(t036[2,2]) = 2 \\
d(t036, t146) &= cost_e(t036[2,2]) + cost_e(t036[8,8]) = 2 \\
d(t048, t115) &= cost_d(t048[5,6]) + 2cost_d(QBL) = 3 \\
d(t048, t139) &= cost_e(t048[2,2]) + cost_d(t048[4,5]) + cost_d(QBL) = 3 \\
d(t048, t146) &= cost_e(t048[2,2]) + cost_e(t146[6,6]) = 2 \\
d(t115, t139) &= cost_e(t115[2,2]) + cost_e(t115[6,8]) = 2 \\
d(t115, t146) &= cost_d(t146[2,2]) + cost_e(t115[8,8]) + 2cost_d(QBL) = 4 \\
d(t139, t146) &= cost_d(t139[7,7]) + cost_d(QBL) = 2
\end{aligned}
$$

5 Conclusion

The EDSI model of evolution joins the events of tandem duplication, tandem copy excision, point mutation and deletion that may happen in arbitrary order throughout evolution. To our knowledge, this is the first time an evolutionary model of that complexity has been described for sequence comparison. Taking into account operations as captured in the EDSI model, we described an exact method to compare a pair of repeat sequences and to assign them a distance. In first tests we could show that the pairwise comparison under the EDSI model efficiently captures cascades of duplication events and expresses them in the distance measure. Regular (based on the SI model) scoring functions cannot resolve duplication events, and scoring schemes based on the DSI model do not support cascaded cycles of duplications, excisions and mutations. However, *in vivo* studies demonstrated how essential these mechanisms are, when investigating the evolution of *S. aureus* [12].

Acknowledgments

The work of Michael Sammeth was supported by a doctoral scholarship of the Ernst-Schering Research Foundation.

References

1. B. Behzadi and J.-M. Steyaert. The minisatellite transformational problem revisted. In *Proc. of WABI 2004*, volume 3240 of *LNBI*, pages 310–320, 2004.
2. G. Benson. Sequence alignment with tandem duplication. *J. Comput. Biol.*, 4:351–367, 1997.
3. G. Benson and L. Dong. Reconstructing the duplication history of a tandem repeat. In *Proc. of ISMB 1999*, pages 44–53, 1999.
4. S. B/'erard and E. Rivals. Comparison of minisatellites. In *Proc. of RECOMB 2002*, pages 67–76, 2002.
5. D. Bertrand and O. Gascuel. Topological rearrangements and local search method for tandem duplication trees. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 2:1–13, 2005.
6. M. de Macedo Brígido, C. R. M. Barardi, C. A. Bonjardin, C. L. S. Santos, M. de Lourdes Junqueira, and R. R. Brentani. Nucleotide sequence of a variant protein a of *Staphylococcus aureus* suggests molecular heterogeneity among strains. *J. Basic Microbiol.*, 31:337–345, 1991.
7. O. Elemento, O. Gascuel, and M.-P. Lefranc. Reconstructing the duplication history of tandemly repeated genes. *Mol. Biol. Evol.*, 19:278–288, 2002.
8. M. Enright, N. Day, C. Davies, S. Peacock, and B. Spratt. Multilocus sequence typing for characterization of methicillin-resistant and methicillin-susceptible clones of *Staphylococcus aureus*. *J. Clin. Microbiol.*, 38:1008–1015, 2000.
9. W. Fitch. Phylogenies constrained by cross-over process as illustrated by human hemoglobins in a thirteen-cycle, eleven amino-acid repeat in human apolipoprotein a-i. *Genetics*, 86:623–644, 1977.
10. R. Groult, M. Léonard, and L. Mouchard. A linear algorithm for the detection of evolutive tandem repeats. In *The Prague Stringology Conference 2003*, 2003.
11. D. Jaitly, P. Kearney, G.-H. Lin, and B. Ma. Reconstructing the duplication history of tandemly repeated genes. *J. Comput. Sys. Sci.*, 65:494–507, 2002.
12. B. Kahl, A. Mellmann, S. Deiwick, G. Peters, and D. Harmsen. Variation of the polymorphic region X of the protein A gene during persistent airway infection of cystic fibrosis patients reflects two independent mechanisms of genetic change in *Staphylococcus aureus*. *J. Clin. Microbiol.*, 43:502–505, 2005.
13. L. Koreen, S. V. Ramaswamy, E. A. Graviss, S. Naidich, J. M. Musser, and B. N. Kreiswirth. spa typing method for discriminating among staphylococcus aureus isolates: implications for use of a single marker to detect genetic micro- and macrovariation. *J. Clin. Microbiol.*, 47:792–799, 2004.
14. Ridom nomenclature server. `http://www.ridom.de/spaserver/nomenclature.shtml`.
15. D. A. Robinson and M. C. Enright. Evolutionary models of the emerge of methicillin-resistant staphylococcus aureus. *Antimicrob. Agents Chemother.*, 47:3926–3934, 2003.
16. M. Sammeth, J. Rothgänger, W. Esser, J. Albert, J. Stoye, and D. Harmsen. Qalign: quality based multiple alignments with dynamic phylogenetic analysis. *Bioinformatics*, 19:1592–1593, 2003.
17. N. N. I. S. System. National nocosomial infections surveillance (nnis) system report, data summary from january 1990-may 1999. *Am. J. Infect. Control*, 27:520–532, 1999.
18. A. van Belkum, S. Scherer, L. van Alphen, and H. Verbrugh. Short-sequence dna repeats in prokaryotic genomes. *Microbiol. Mol. Biol. Rev.*, 62:275–293, 1998.

Bisher erschienene Reports an der Technischen Fakultät
Stand: 2005-08-08

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)

2003-01 On the Similarity of Sets of Permutations and its Applications to Genome Comparison
(Anne Bergeron, Jens Stoye)

2003-02 SNP and mutation discovery using base-specific cleavage and MALDI-TOF mass spectrometry
(Sebastian Böcker)

2003-03 From RNA Folding to Thermodynamic Matching, including Pseudoknots
(Robert Giegerich, Jens Reeder)

2003-04 Sequencing from compomers: Using mass spectrometry for DNA de-novo sequencing of 200+ nt
(Sebastian Böcker)

2003-05 Systematic Investigation of Jumping Alignments
(Constantin Bannert)

2003-06 Suffix Tree Construction and Storage with Limited Main Memory
(Klaus-Bernd Schürmann, Jens Stoye)

2003-07 Sequencing from compomers in thepresence of false negative peaks
(Sebastian Böcker)

2003-08 Genalyzer: An Interactive Visualisation Tool for Large-Scale Sequence Matching – Biological Applications and User Manual
(Jomuna V. Choudhuri, Chris Schleiermacher)

2004-01 Sequencing From Compomers is NP-hard
(Sebastian Böcker)

2004-02 The Money Changing Problem revisited: Computing the Frobenius number in time $O(k\, a_1)$
(Sebastian Böcker, Zsuzsanna Lipták)

2004-03 Accelerating the Evaluation of Profile HMMs by Pruning Techniques
(Thomas Plötz, Gernot A. Fink)

2004-04 Optimal Group Testing Strategies with Interval Queries and Their Application to Splice Site Detection
(Ferdinando Cicalese, Peter Damaschke, Ugo Vaccaro)

2004-05 Compressed Representation of Sequences and Full-Text Indexes
(Paolo Ferragina, Giovanni Manzini, Veli Mäkinen, Gonzalo Navarro)

2005-01 Overlaps Help: Improved Bounds for Group Testing with Interval Queries
(Ferdinando Cicalese, Peter Damaschke, Libertad Tansini, Sören Werth)

2005-02 Two batch Fault-tolerant search with error cost constraints: An application to learning
(Ferdinando Cicalese)

2005-03 Searching for the Shortest Common Supersequence
(Sergio A. de Carvalho Jr., Sven Rahmann)

2005-04 Counting Suffix Arrays and Strings
(Klaus-Bernd Schürmann, Jens Stoye)