The Minisatellite Transformation Problem: The Run-Length-Encoding Approach and Further Enhancements

Behshad Behzadi & Jean-Marc Steyaert,
Ecole Polytechnique
Mohamed Abouelhoda, Cairo University
Robert Giegerich, Bielefeld University

Biology…

- Minisatellites consist of tandem arrays of short repeat units found in genome of most higher eukaryotes.

- High degree of polymorphism at minisatellites has applications from forensic studies to the investigation of the origins of modern human groups.

…Biology…

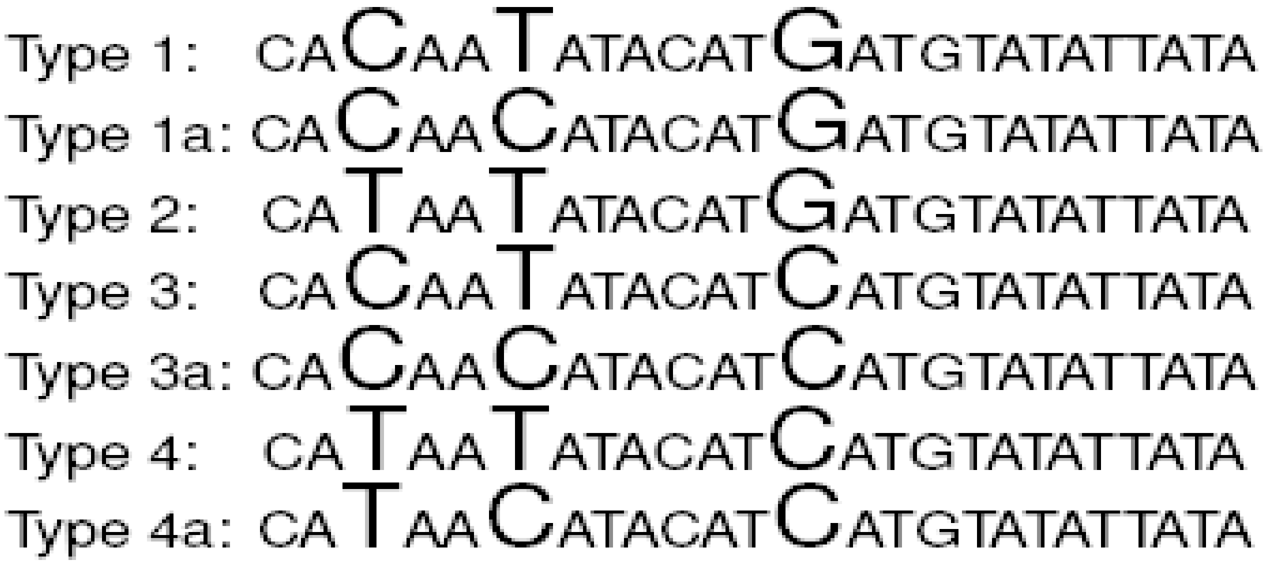
- These repeats are called variants.

- MVR-PCR is designed to find the variants.

- As an example, MSY1 is the minisatellite on the human Y-chromosomes. There are five different repeats (variants) in MSY1.

Different Repeat Types (Variants) of MSY1

Map Types:



Type 1: CACAATATACATGATGTATATTATA

Type 1a: CACAACATACATGATGTATATTATA

Type 2: CATAATATACATGATGTATATTATA

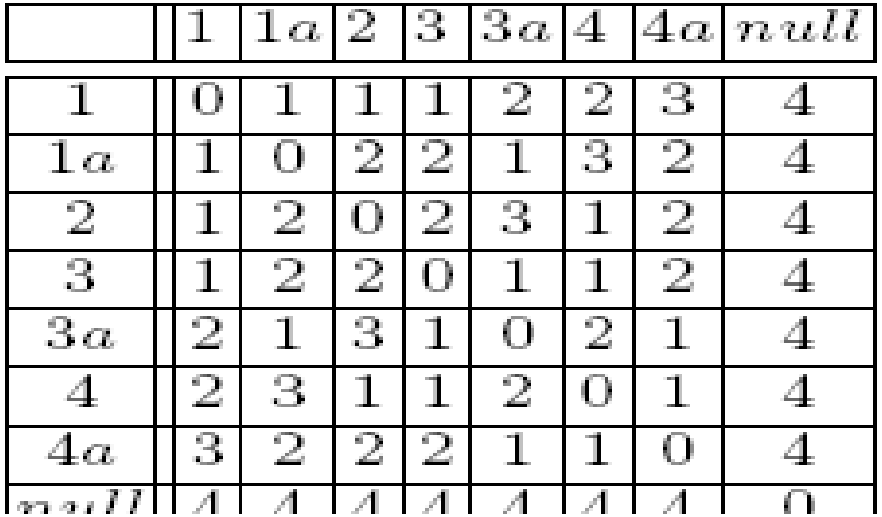
Type 3: CACAATATACATCATGTATATTATA

Type 3a: CACAACATACATCATGTATATTATA

Type 4: CATAATATACATCATGTATATTATA

Type 4a: CATAACATACATCATGTATATTATA

Distance between types:



	1	$1a$	2	3	$3a$	4	$4a$	$null$
1	0	1	1	1	2	2	3	4
$1a$	1	0	2	2	1	3	2	4
2	1	2	0	2	3	1	2	4
3	1	2	2	0	1	1	2	4
$3a$	2	1	3	1	0	2	1	4
4	2	3	1	1	2	0	1	4
$4a$	3	2	2	2	1	1	0	4
$null$	4	4	4	4	4	4	4	0

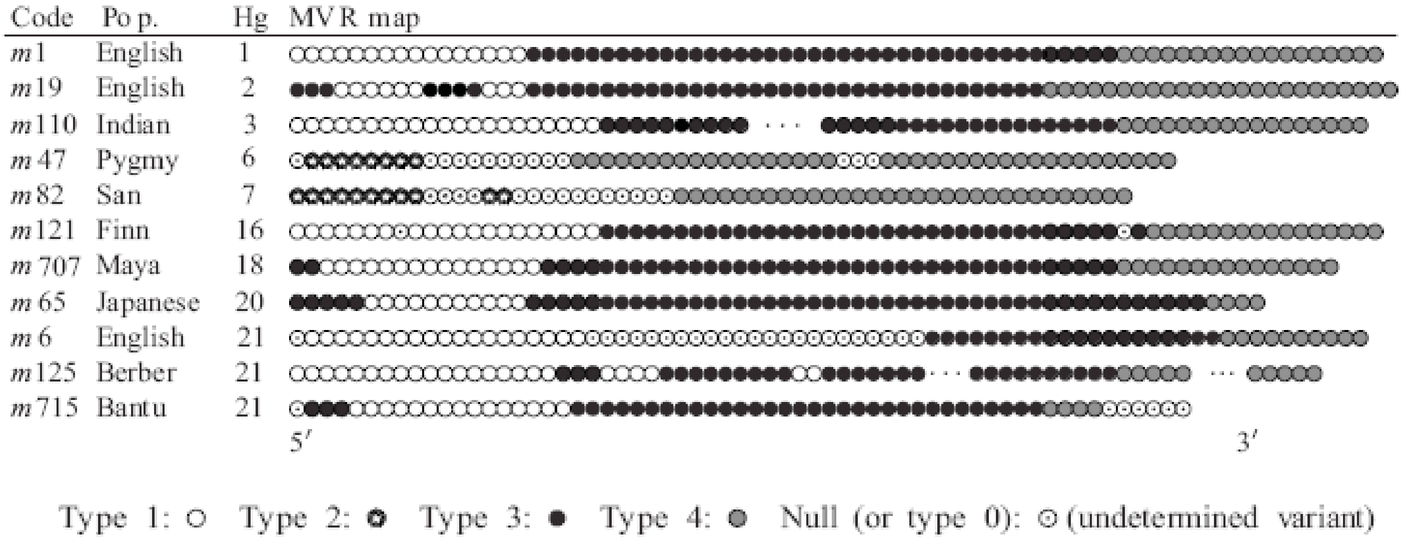
Minisatellite Maps: The MSY1 Dataset

DNA Sequence: … CGGCGAT CGGCGAC CGGCGAC CGGCGAC CGGAGAT…

Unit types (Alphabet): X= CGGCGAT Y= CGGCGAC Z= CGGAGAT

Minisatallite Map: XYYYZ

- **Example Maps from the MSY1 Dataset:**

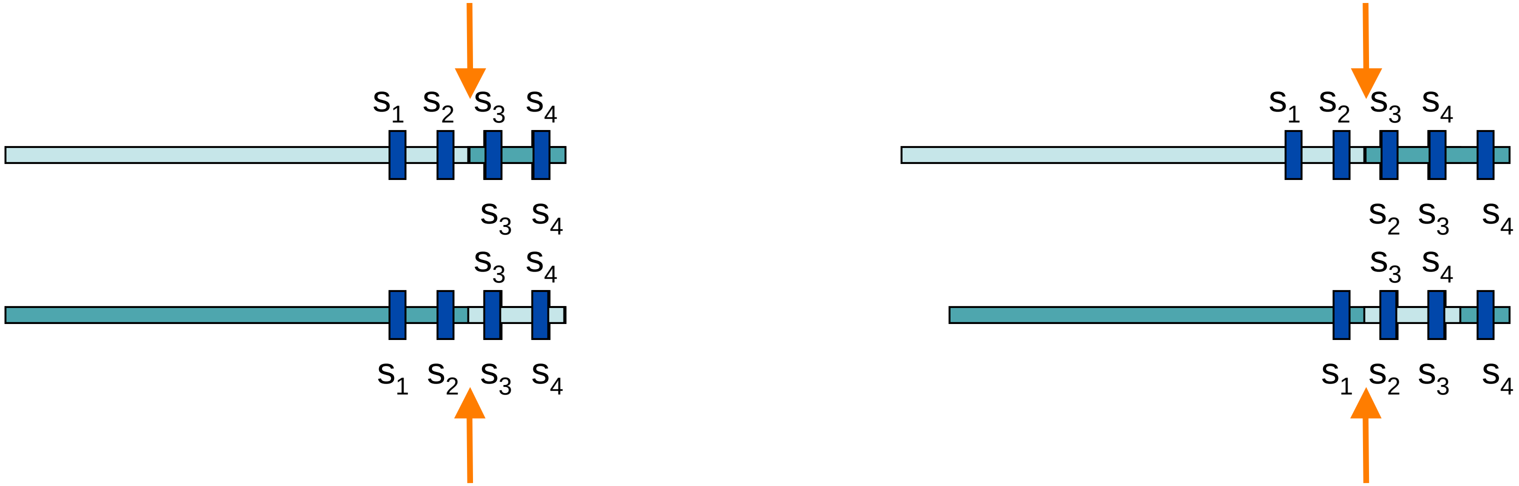


Code	Po p.	Hg
*m*1	English	1
*m*19	English	2
*m*110	Indian	3
*m*47	Pygmy	6
*m*82	San	7
*m*121	Finn	16
*m*707	Maya	18
*m*65	Japanese	20
*m*6	English	21
*m*125	Berber	21
*m*715	Bantu	21

Type 1: ○ Type 2: ✪ Type 3: ● Type 4: ◉ Null (or type 0): ⊙ (undetermined variant)

Evolution Mechanism of Minisatellites

The unequal crossover is a possible mechanism for tandem duplication:



Evolutionary Operations

- Insertion
- Deletion
- Mutation

- Amplification (*p*-plication)
- Contraction (*p*-contraction)

Examples of operations

- Insertion of *d*

  $abbc \rightarrow abb\underline{d}c$

- Deletion of *c*

  $abb\underline{c}b \rightarrow abbb$

- Mutation of *c* into *d*

  $\underline{c}aab \rightarrow \underline{d}aab$

- 4-plication of *c*

  $ab\underline{c}b \rightarrow ab\underline{cccc}b$

- 2-contraction of *b*

  $a\underline{bb}c \rightarrow a\underline{b}c$

Cost Functions

$I(x)$	insertion of symbol x
$D(x)$	deletion of symbol x
$M(x, y)$	mutation of symbol x to y
$A_p(x)$	p-plication of symbol x
$C_p(x)$	p-contraction of symbol x

Hypotheses

- All the costs are positive.
- The cost of duplications (and contractions) is less than all other operations.
- Triangle inequality holds:

  $M(x,y)+M(y,z) <= M(x,z)$; $M(x,x) = 0$

Transformation distance between *s* and *t*

- Applying a sequence of operations on *s* transforming it into *t*.
- The cost of a transformation is the sum of costs of its operations.
- TD = Minimum cost for a possible transformation of *s* into *t*.
- Any transformation which gives this minimum is called an *optimal transformation*.

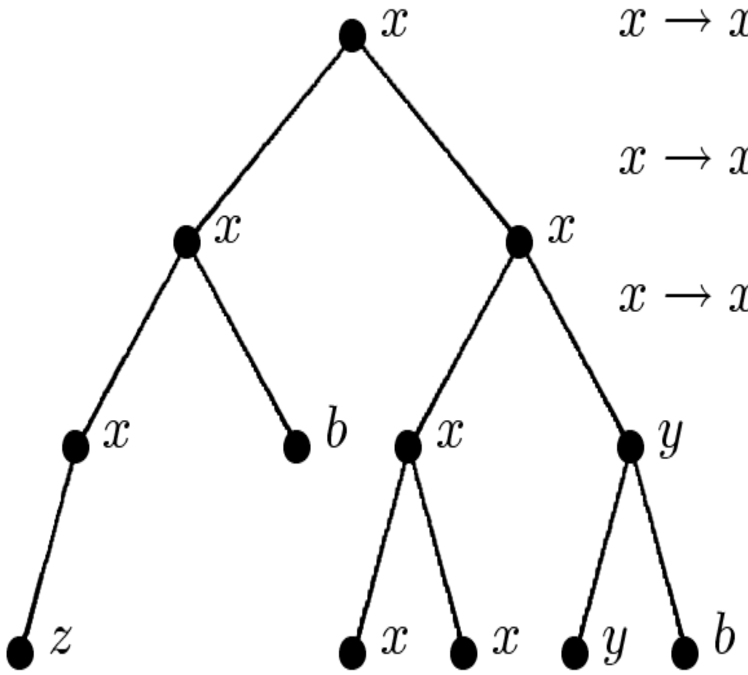
Previous Works

- Bérard & Rivals (RECOMB’02)
- Behzadi & Steyaert (CPM’03, JDA’04)
- Behzadi & Steyaert (WABI'04)

Generation vs. *Reduction*

- The symbols of *s* which generate a non-empty substring of *t* are called **generating symbols**.
- Other symbols of *s* are **vanishing symbols**. (These symbols are eliminated during the transformation by a deletion or contraction.)
- The transformation of symbol *x* into non-empty string *s* is called **generation**.
- The transformation of a non-empty string *s* into a unique symbol *x* is called **reduction**.

The Generation *x→ zbxxyb*



$$x \to xx \to xxy \to xxxy \to xbxxy \to xbxxyb \to zbxxyb$$

$$x \to xx \to xxy \to xxyb \to xbxyb \to zbxyb \to zbxxyb$$

$$x \to xx \to xbx \to zbx \to zbxy \to zbxxy \to zbxxyb$$

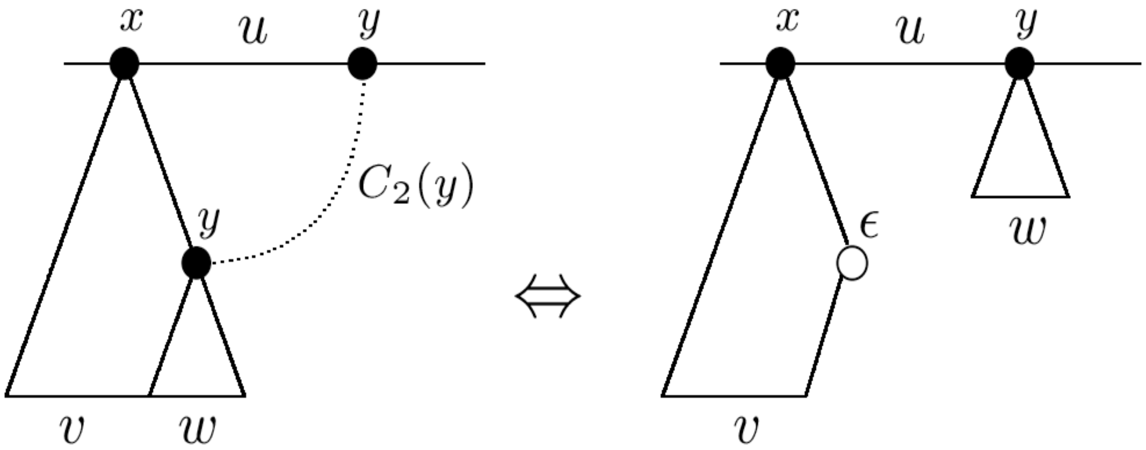
⋆ Different generation sequences for the same tree

$$\text{Generation Cost } = 2A_2(x) + 2I(b) + I(y) + M(x,z)$$

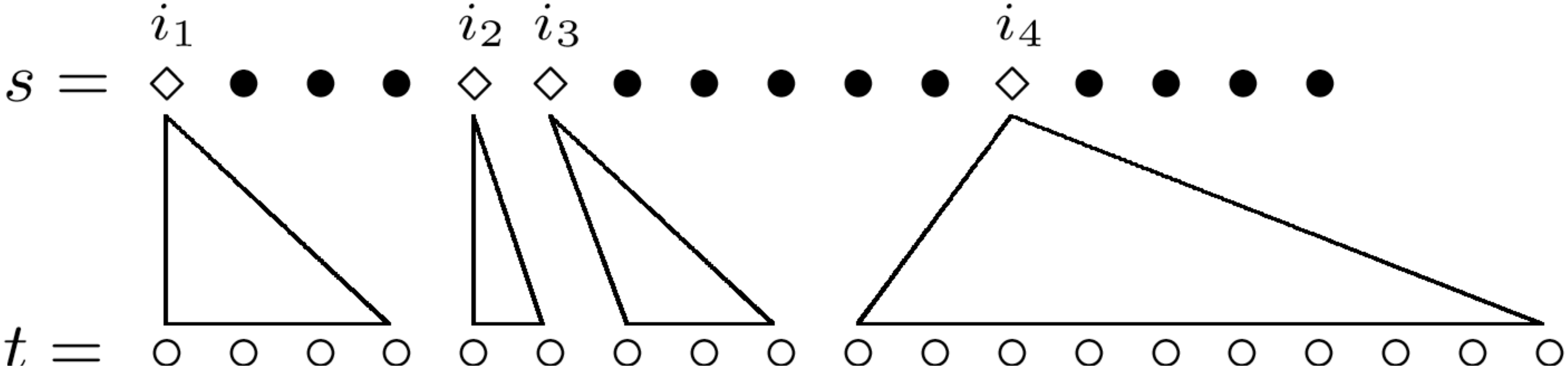
The optimal generation of a non-empty string *s* from a symbol *x* can be achieved by a *non-*

The schema for an optimal transformation

There exists an optimal transformation of *s* into t in which all the *contractions* are done *before* all amplifications.



$u \xrightarrow{*} \epsilon$


i_1
i_2
i_3
i_4
$s =$ ◇ ● ● ● ◇ ◇ ● ● ● ● ● ◇ ● ● ● ●
$t =$ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○

Run-Length Encoding and Run Generation

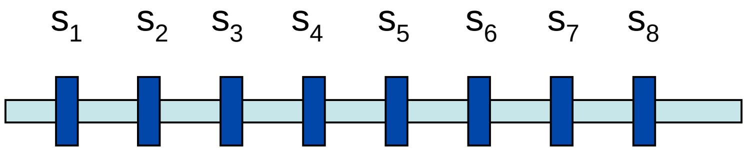
- The RLE encoding of $aaaabbbbcccabbbbcc$ is $a^4b^4c^3a^1b^4c^2$.
- The lengths of the encoded strings with length *n* and *m* is denoted by *m*' and *n'*.
- There exists an optimal generation of a non-empty string *t* from a single symbol *x* in which for every run of size *k > 1* in *t* the *k-1* right symbols of the run are generated by duplications of the leftmost symbol of the run

Preprocessing --> Core algorithm

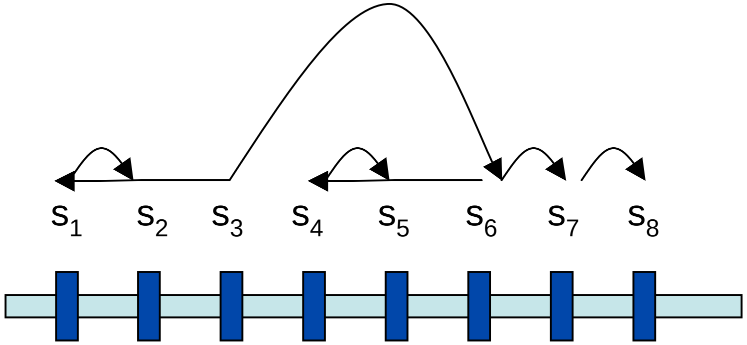
- Compute the generation cost of all substrings of the target string *t* from any symbol *x* of the alphabet: *G(t)[x,i,j]*
- Compute the optimal generation/reduction costs over the substrings by recurrence using dynamic programming.
- The running time is given by:

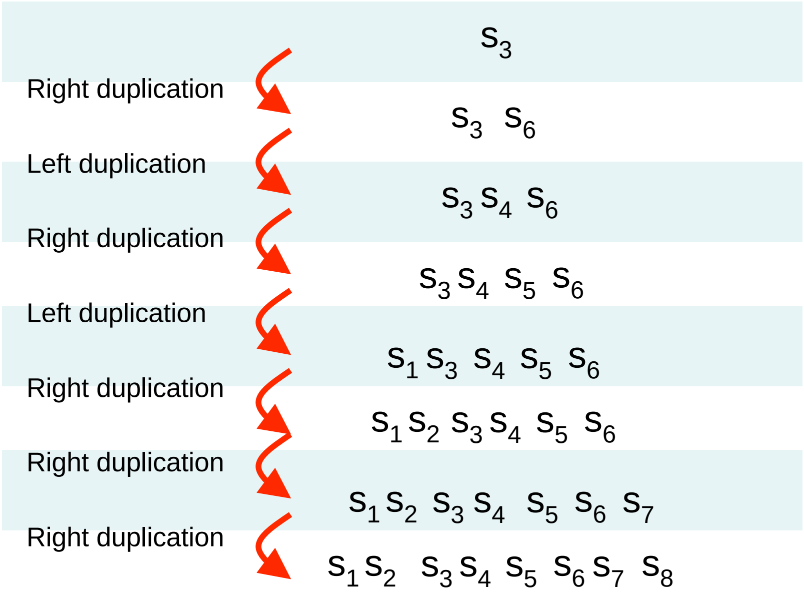
$$O((m'^3+n'^3)|Alpha|+mn'^2+nm'^3+mn)$$

A different look at Duplication History



observed





s_3

Right duplication

s_3 s_6

Left duplication

s_3 s_4 s_6

Right duplication

s_3 s_4 s_5 s_6

Left duplication

s_1 s_3 s_4 s_5 s_6

Right duplication

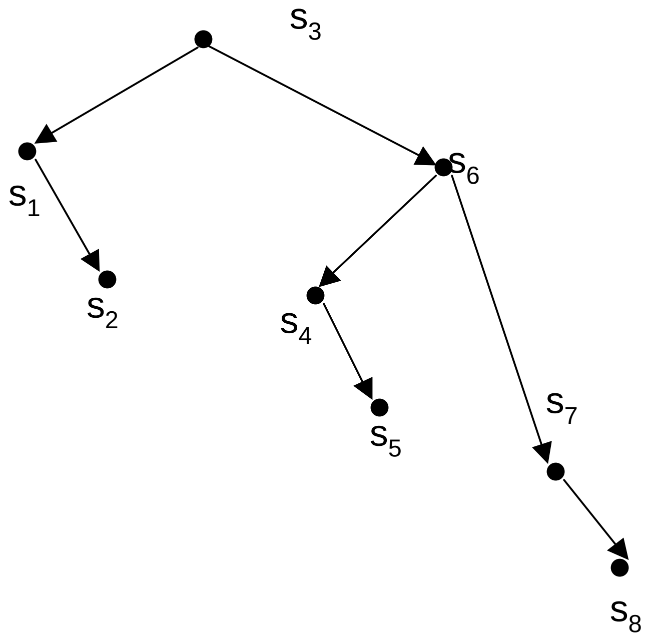
s_1 s_2 s_3 s_4 s_5 s_6

Right duplication

s_1 s_2 s_3 s_4 s_5 s_6 s_7

Right duplication

s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8



Alignment of Minisatellite Maps (1)

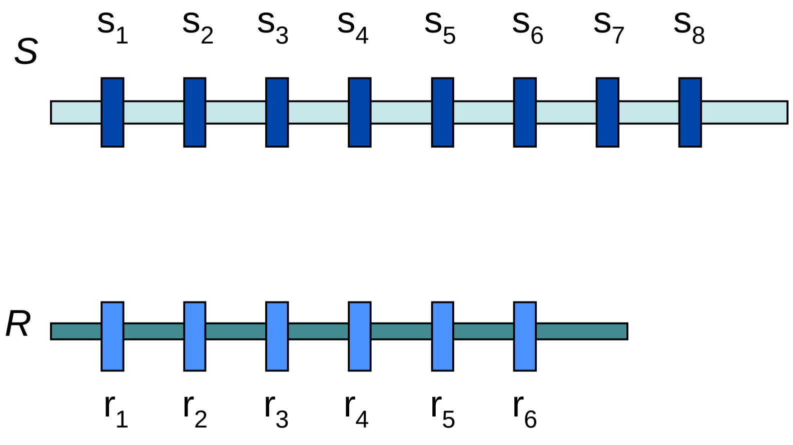
Complications: comparing maps is more than copy number

1)Types are not identical

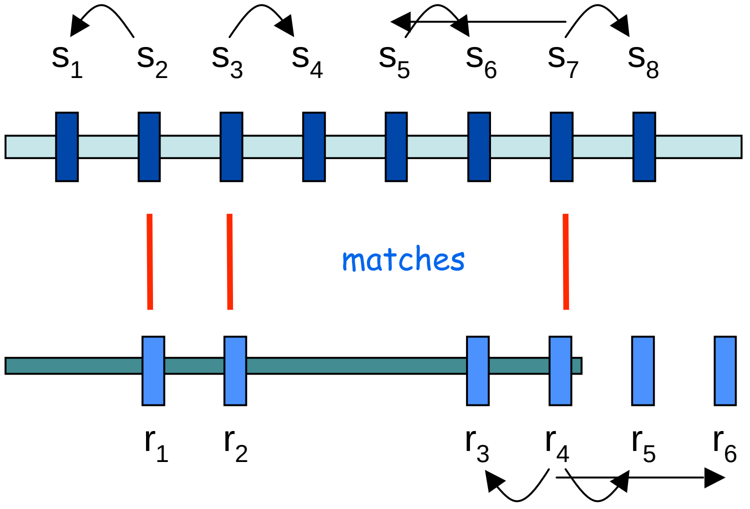
2)Types duplicate according to a duplication model

3)Parts of the map may be foreign, appeared by transposition

Example of an alignment:

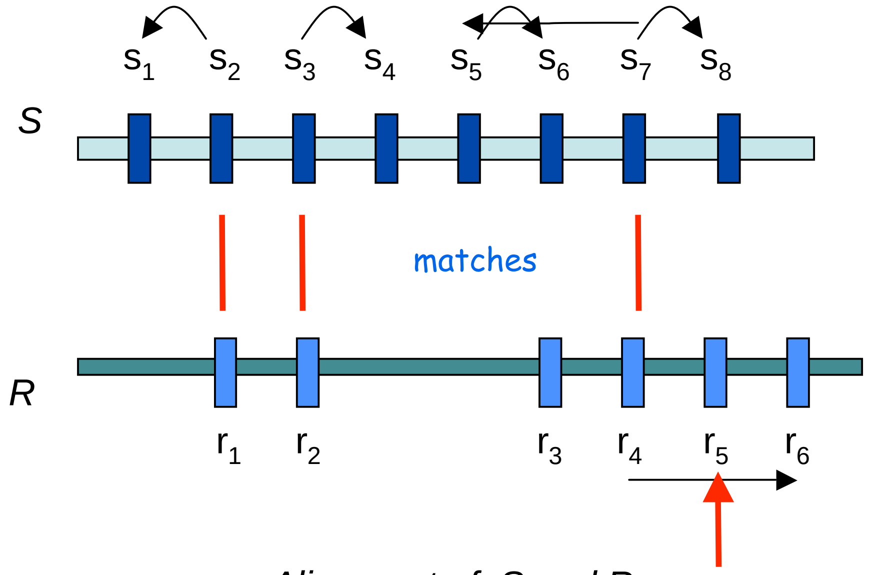


The two maps S and R



Alignment of S and R

Alignment of Minisatellite Maps (2)



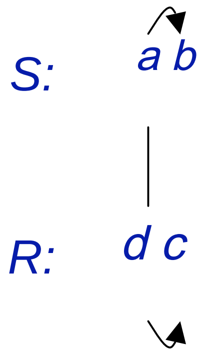
Alignment of S and R

- Matches refer to common history
- Duplication events refer to individual duplication history
- Insertions/Deletions refer to foreign units

Improved Model of Comparison
Left and Right Simultaneous Dups

Example:

Assume: $d(a,b)=d(d,c)=d(c,d) < d(a,c)=d(b,d) < d(a,d)$

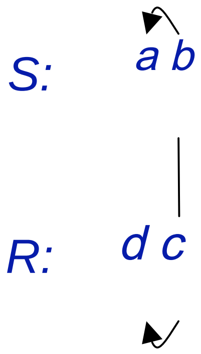


S: a b

R: d c

Bérard et al., Model

There is no rule to allow simultaneous left/right duplications in *S* and *R*



S: a b

R: d c

Our NEW Model

It has less score. Because there is a rule to allow simultaneous left/right duplications in *S* and *R*

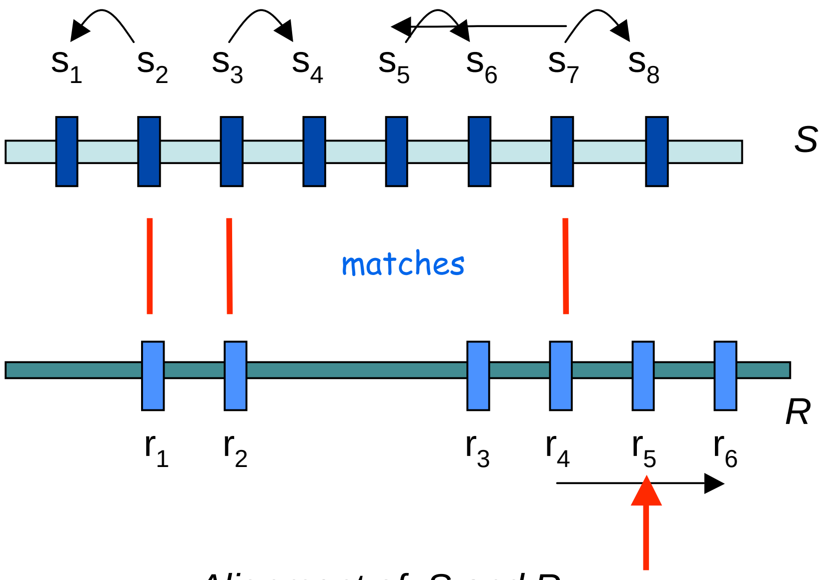
Algorithm Layout

Observations:

-- Duplications compose intervals in S/R

-- The duplications within an alignment originate either from the leftmost or from the rightmost unit of the interval containing the duplications

-- Optimal alignment must contain optimal duplication history of these intervals



Alignment of S and R

Therefore:

1. Pre-compute and store score of optimal history for all sub-intervals of *S* and *R originated from leftmost/rightmost unit*

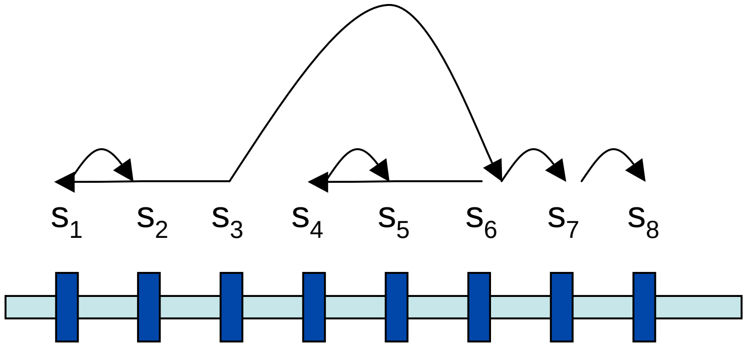
2. Use Dynamic programming alignment algorithm considering that intervals of *S/R* appeared as duplications (optimal scores are look-up

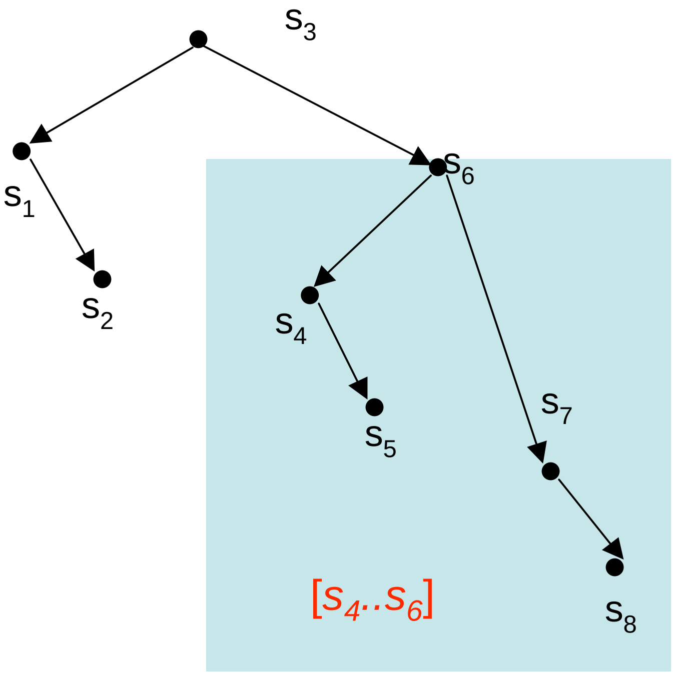
Finding an Optimal Duplication History

Duplication history can be represented by an *ordered directed tree ORDT: Nodes are the units*

Edges are directed and weighted by distance between the unit

Each sub-tree can be written as contiguous units $[s_i..s_j]$





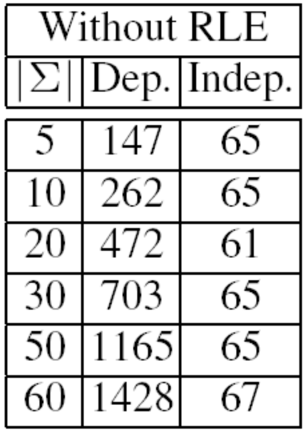
Optimal duplication history:≡ an optimal ORDT

An optimal ORDT can be found in *$O(n^3)$ time and $O(n^2)$ space* by partitioning contiguous non-overlapping intervals :

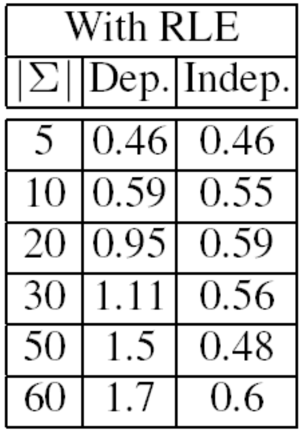
RLE: $O(n^2 + nn'^2 + n'^3)$ time

Experimental Running Times

Duplication history:

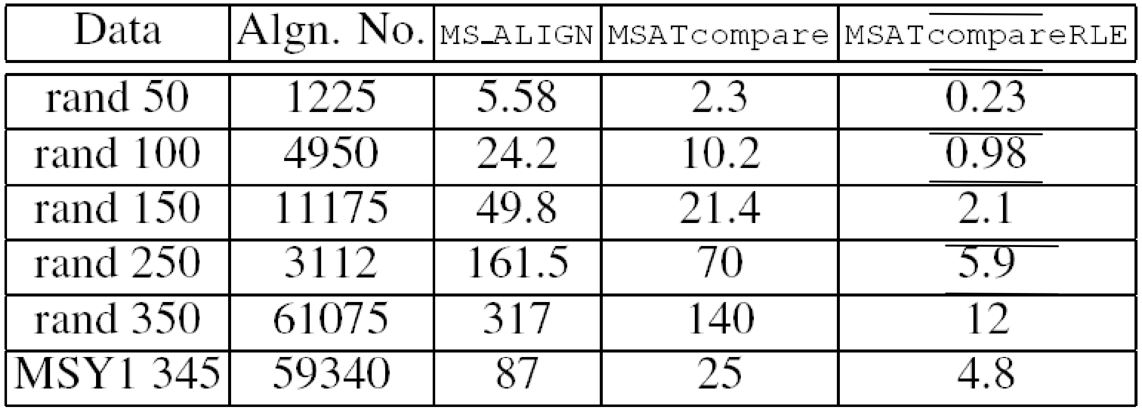


Without RLE		
$\|\Sigma\|$	Dep.	Indep.
5	147	65
10	262	65
20	472	61
30	703	65
50	1165	65
60	1428	67



With RLE		
$\|\Sigma\|$	Dep.	Indep.
5	0.46	0.46
10	0.59	0.55
20	0.95	0.59
30	1.11	0.56
50	1.5	0.48
60	1.7	0.6

Alignment algorithm:

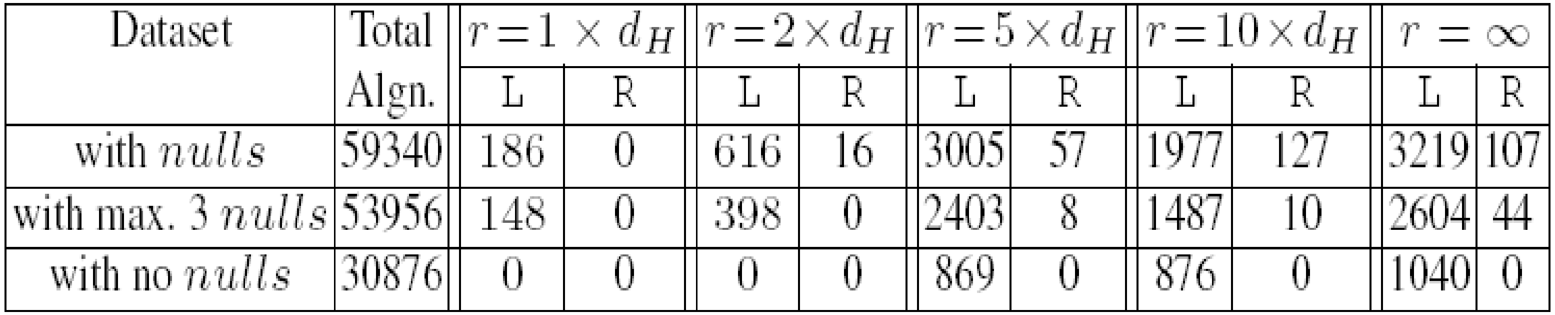


Data	Algn. No.	MS_ALIGN	MSATcompare	MSATcompareRLE
rand 50	1225	5.58	2.3	0.23
rand 100	4950	24.2	10.2	0.98
rand 150	11175	49.8	21.4	2.1
rand 250	3112	161.5	70	5.9
rand 350	61075	317	140	12
MSY1 345	59340	87	25	4.8

- MS_ALIGN is the algorithm of Bérard et al.
- MSATcompare is ours

Detection of Duplication Bias in MSY1 Dataset

E1: run algorithm allowing left- and right- duplications
EL: allow only left duplications
ER: allow only right duplications

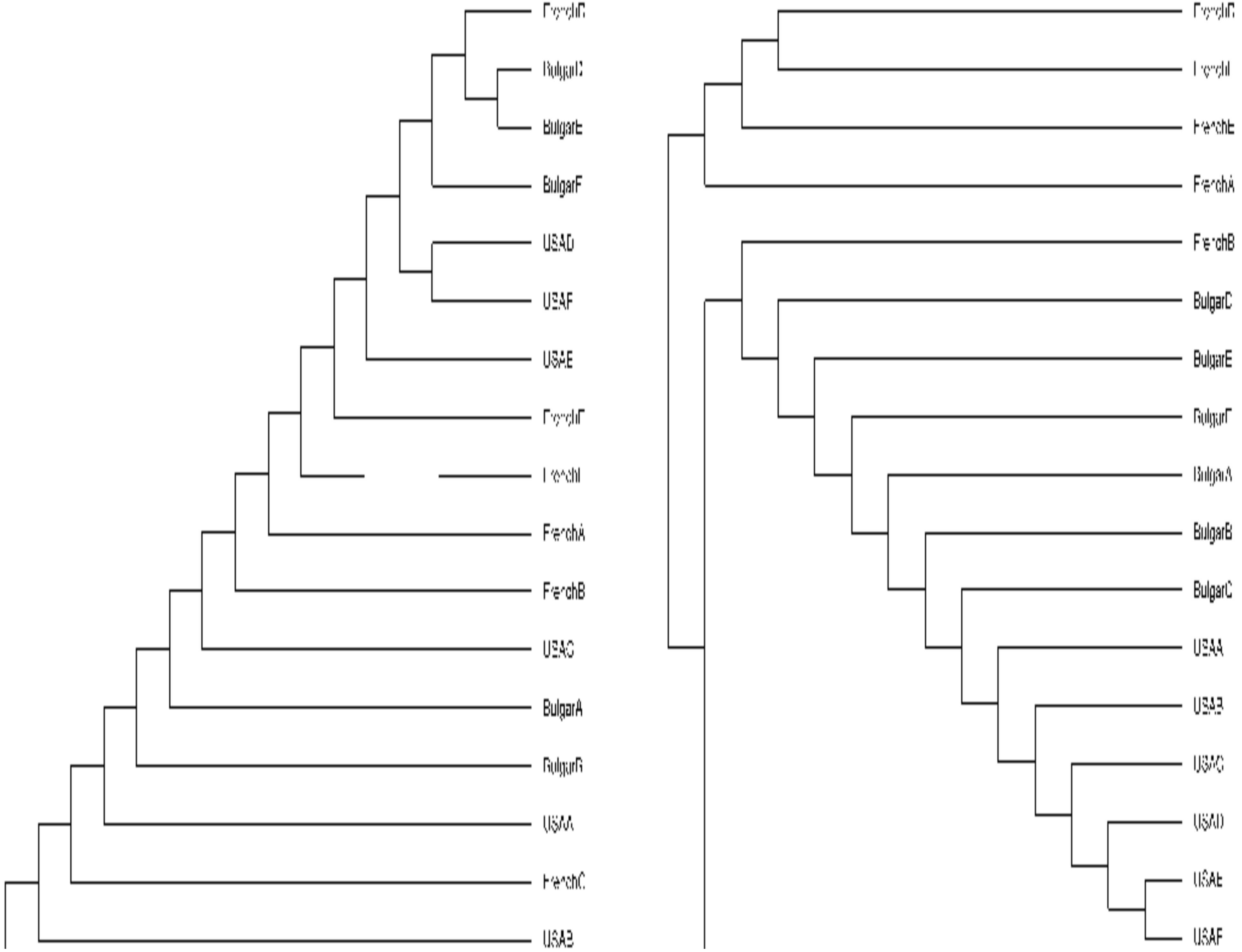


Dataset	Total Algn.	$r=1\times d_H$		$r=2\times d_H$		$r=5\times d_H$		$r=10\times d_H$		$r=\infty$	
		L	R	L	R	L	R	L	R	L	R
with $nulls$	59340	186	0	616	16	3005	57	1977	127	3219	107
with max. 3 $nulls$	53956	148	0	398	0	2403	8	1487	10	2604	44
with no $nulls$	30876	0	0	0	0	869	0	876	0	1040	0

L: number of alignments in EL with cost higher than that in E1
R: number of alignments in ER with cost higher than that in E1

There is an important bias: R keeps small while E increases quickly is nearly as r=M/DUP increases; this suggests that the units are most often generated from right to left

This raises questions about the underlying duplication mechanisms and


FrenchD
BulgarD
BulgarE
BulgarF
USAD
USAF
USAE
FrenchF
FrenchI
FrenchA
FrenchB
USAC
BulgarA
BulgarB
USAA
FrenchC
USAB
FrenchD
FrenchI
FrenchE
FrenchA
FrenchB
BulgarD
BulgarE
BulgarF
BulgarA
BulgarB
BulgarC
USAA
USAB
USAC
USAD
USAE
USAF