Some Recent Combinatorial Approaches To Genome Comparison

Riccardo Dondi – Università di Bergamo

Comatege – SeqBio2012

Talk Outline

Introduction

Variants of LCS

- Repetition Free Longest Common Subsequence (RFLCS)
- Exemplar Longest Common Subsequence (ELCS)
- RFLCS and ELCS: complexity and algorithms

Genome Alignment

- Duplication-Loss Model of evolution
- Duplication-Loss Alignment problem
- Minimum Labeling Alignment problem

Conclusion

Comparative genomics

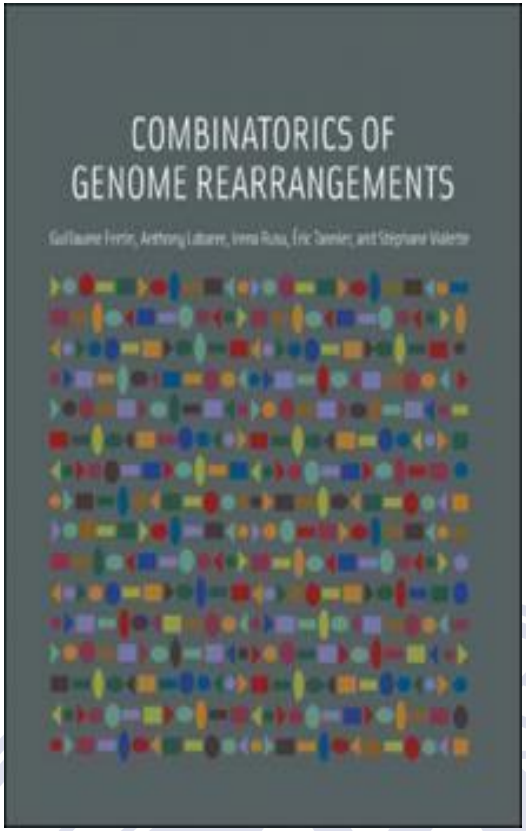
- **Comparative genomics**: study of genome structure and function in different species
- Goals: understand
  - Structure and function relationship
  - Evolutionary histories of gene families

- From a **combinatorial** point of view: genomes can be considered as **strings** or **permutations**

Comparative genomics

Genome comparison inspiration for many interesting combinatorial problems *[Fertin, Labarre, Rusu, Tannier and Vialette, Combinatorics of Genome Rearrangements, 2009]*

- Genome rearrangements
- Phylogenetic problems
- Variants of LCS
- …



Comparative genomics

Recent approach *[Holloway et al, RECOMB 2012]:*

- Consider an **evolutionary model** for genomes
- **Goal:** inference of ancestral genomes and evolutionary events
- Approach based on **alignment of genomes**

Variants of LCS

Exemplar model

- Genomes contain multiple copies of a gene
- **Exemplar model** *[Sankoff, Bioinformatics, 1999]*
- For each family of duplicated genes infer an **exemplar**
- **Exemplar**: **representative** from which all other genes have originated

Replacement approach

- Differences in gene order in two genomes: limited number of rearrangement operations
- The problem is easy when there are no duplicates, **hard** when there are **several copies of the same gene**
- Specific subsequences of genomes → **highly conserved sets of genes**
- **Greedy approach**: replace each substring containing such subsequences by a symbol in both genomes
- **Replacement approach** → each gene family must have (at least) an occurrence in the common subsequence

Variants of LCS

LCS-like problems with constraints on the symbols:

- **Exemplar model** → ***no repetition*** of a symbol in a subsequence
- **Replacement approach** → ***mandatory*** and ***optional symbols***

Longest Common Subsequence

- LCS Well-known problem in Computational Biology
- Strings $s = s[1], s[2], \ldots, s[m]$ and $t = t[1], t[2], \ldots, t[l]$
- s is a **subsequence** of t if for some $j_1 < j_2 < \ldots < j_m$

$$s[h] = t[j_h]$$

- A **longest common subsequence** of s_1 and s_2: a sequence s subsequence of both s_1 and s_2 of maximum length
- *Longest common subsequence* of a set S of sequences: a longest possible sequence s subsequence of each sequence in S.

Longest Common Subsequence

LCS - previous results:

- Polynomial time algorithm for fixed number of strings via dynamic programming algorithms *[Hsu and Du, JCSS, 1984]*
- NP-hard even for sequences over an alphabet of size 2 *[Maier, Journal of the ACM, 1978]*
- Not approximable within factor $O(n^{1-\varepsilon})$, even if all symbols appear at most twice in each string *[Jiang and Li. , SIAM Journal on Computing, 1995]*

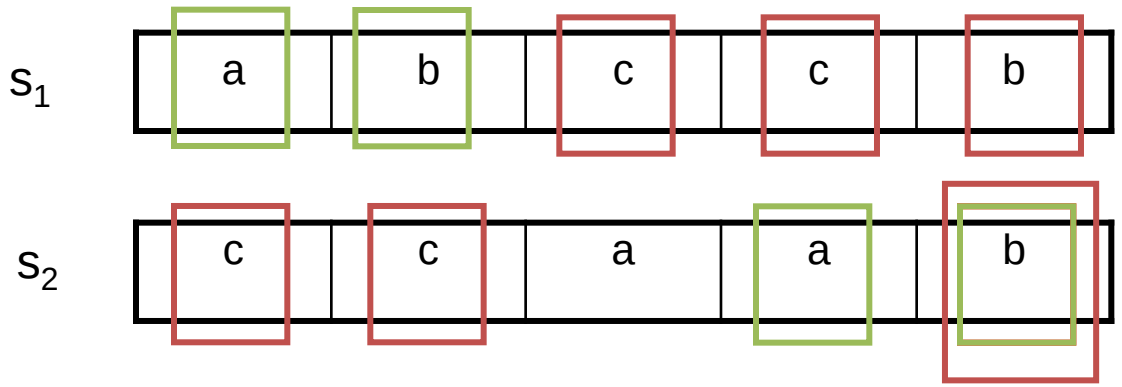
Repetition Free LCS

Repetition Free LCS (RFLCS)

Input: two strings s_1, s_2 over alphabet *A*

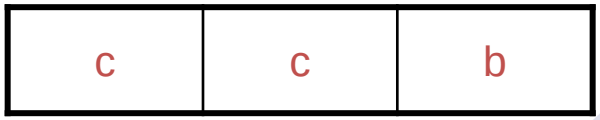
Output: a longest common subsequence s of s_1, s_2 such that each symbol in A occurs at most once in s

RFLCS

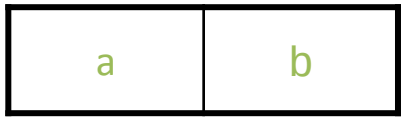


A={ a,b,c }

A LCS



A RFLCS



Exemplar LCS

Exemplar LCS (ELCS)

Input: two strings s_1, s_2 over alphabet A

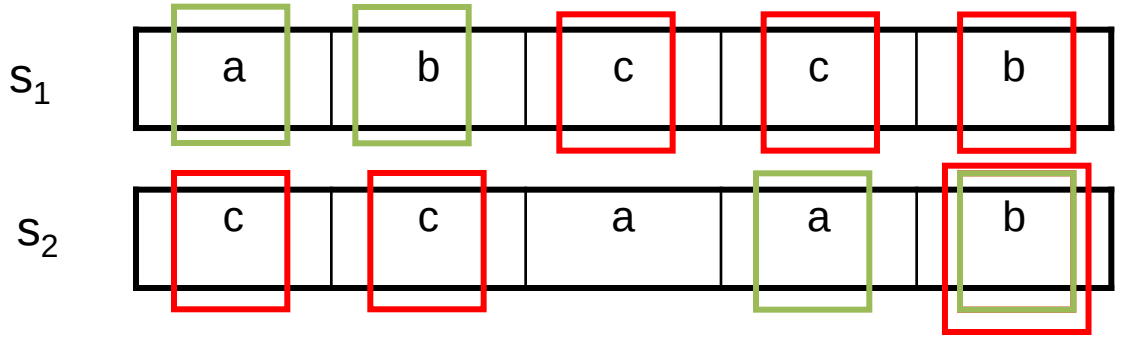
$$A=A_o \cup A_m,\ A_o \cap A_m = \emptyset \text{ where}$$

A_o: set of optional symbols

A_m: set of mandatory symbols

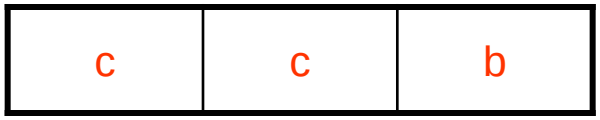
Output: a longest common subsequence s of s_1, s_2 that contains each symbol in A_m

Exemplar LCS

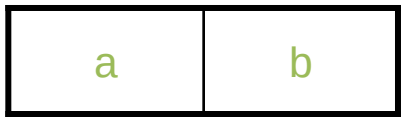


A_o={ b,c}, A_m={ a }

A LCS



An ELCS



Exemplar LCS

Problem	Occurrences of mandatory symbols	Occurrences of optional symbols
ELCS(1;≤ 1)	exactly 1	at most 1
ELCS(1)	exactly 1	unrestricted
ELCS(≥1;≤ 1)	at least 1	at most 1
ELCS(≥1)	at least 1	unrestricted

Different versions of the problem according to the number of occurrences of each symbol in the solution

RFLCS → ELCS(*,≤1) without mandatory symbols

RFLCS - complexity

RFLCS **poly-time cases** *[Adi et al, DAM, 2010]*:

- each symbol occurs **at most once** in one of the input strings → LCS
- the number of symbols with **multiple occurrences** is **bounded** by a parameter → guess the right subsequence of these symbols and add other symbols

RFLCS - complexity

Theorem *[Adi et al, DAM, 2010]: RF-LCS is APX-hard, even when restricted to instances in which each input string contains at most two occurrences of each symbol.*

Proof.

L-reduction from **MAX 2,3-SAT**

MAX 2,3-SAT: restriction of MAX SAT where

- Each clause has at most two literals
- Each variable occurs in at most three clauses

RFLCS - complexity

Proof.

$s_1 = s(x_1)s(\neg x_1)\ D_1D_2 \ldots\ D_6\ s(x_2)s(\neg x_2)\ D_7D_8 \ldots\ D_{12}\ \ldots\ s(x_n)s(\neg x_n)$

$s_2 = s(\neg x_1)s(x_1)\ D_1D_2 \ldots\ D_6\ s(\neg x_2)s(x_2)\ D_7D_8 \ldots\ D_{12}\ \ldots\ s(\neg x_n)s(\neg x_n)$

$D_1D_2 \ldots\ D_k$ separation symbols

RFLCS - complexity

Proof.

- Each symbol D_i in an RFLCS
- Solution of MAX 2-3 satisfies q clauses iff RLCS of length q + |D|
- Each clause satisfied retained in the corresponding block

Approximating RFLCS

h-approximation algorithm (where **h** is the **maximum number** of occurrences of a symbol in an input string) *[Adi et al, DAM, 2010]*

1. compute a LCS
2. remove repetitions

Properties:

- LCS is an upper bound on the length of a RFLCS
- At most h removal

Approximating RFLCS

Randomized h-approximation algorithms *[Adi et al, DAM, 2010]*

In the input string containing more occurrences of a symbol x in A

- Choose one of the occurrences of x
- Remove the other occurrences

RFLCS – FPT algorithm

Theorem *[Bonizzoni et al, IPL, 2010]: RFLCS is fixed parameter tractable when the parameter is the length of the solution.*

k → size of the solution

Algorithm: computes if there exists a solution of RFLCS of size at least k

RFLCS – FPT algorithm

Application of the **color-coding technique**

Two phases:

Phase 1) **color** the symbols in alphabet *A* with k colors such that each symbol in the solution is assigned a **distinct color**

Phase 2) by dynamic programming compute if a solution with k distinct colors exists

RFLCS – FPT algorithm

Phase 1

Use family F of *perfect hash functions* from *A* to the set of colors $\{c_1, \ldots, c_k\}$

By the properties of F, there exists a function f in F such that **each symbol in the solution** is assigned a **distinct color**

RFLCS – FPT algorithm

Phase 2)

Dynamic Programming

L[i,j,C] represents a RFLCS for $s_1[1,i]$, $s_2[1,j]$ that contains symbols colored by the set of colors C

L[i,j,C] = max

- L[i-1,j,C]
- L[i,j-1,C]
- L[i-1,j-1,C- $\{c_h\}$] if $s_1[i] = s_2[j] = a$ and $f(a) = c_h$

RFLCS – FPT algorithm

Example

s_1= a b c b d d

s_2= d b d c d a

A = { a,b,c,d}

s_1= a b c b d d

s_2= d b d c d a

A = { a,b,c,d}

Solution s = b c d

RFLCS – FPT algorithms

- Randomized FPT algorithm *[Blin et al, IPL, 2012]* that improves upon the time and space complexity, based on the **multilinear detection technique**
- Reduction to the problem of detecting a multilinear monomial (of degree *k*) in an arithmetic circuit

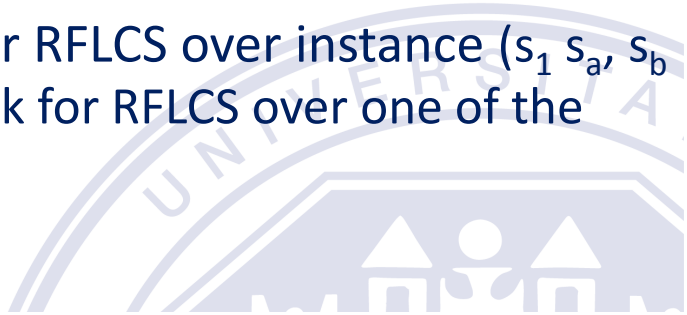
RFLCS – Parameterized complexity

Theorem *[Blin et al, IPL, 2012]: RFLCS does not admit a polynomial size kernel unless NP in coNP/Poly.*

Proof.

Recent technique: **composition algorithm**

- Two instances of RFLCS (s_1, s_2), (s_a, s_b)
- An instance $(s_1\ s_a, s_b\ s_2)$ of RFLCS such that
  - There exists a solution of size k for RFLCS over instance $(s_1\ s_a, s_b\ s_2)$ iff there exists a solution of size k for RFLCS over one of the instance (s_1, s_2), (s_a, s_b)



Exemplar LCS

Problem	Occurrences of mandatory symbols	Occurrences of optional symbols
ELCS(1;≤ 1)	exactly 1	at most 1
ELCS(1)	exactly 1	unrestricted
ELCS(≥1;≤ 1)	at least 1	at most 1
ELCS(≥1)	at least 1	unrestricted

1. Complexity of ELCS (existence of a feasible solution)
2. Complexity of ELCS(1;≤ 1), ELCS(>=1;≤ 1)

ELCS -complexity

ELCS: general version of the problem

Does a feasible solution exist?

Input: strings s_1, s_2 over alphabet $A=A_o \cup A_m$, $A_o \cap A_m = \emptyset$, where

A_o: set of optional symbols

A_m: set of mandatory symbols

Output: does a common subsequence of sequences s_1, s_2 that contains all mandatory symbols exist?

Only mandatory symbols are relevant

ECLS - complexity

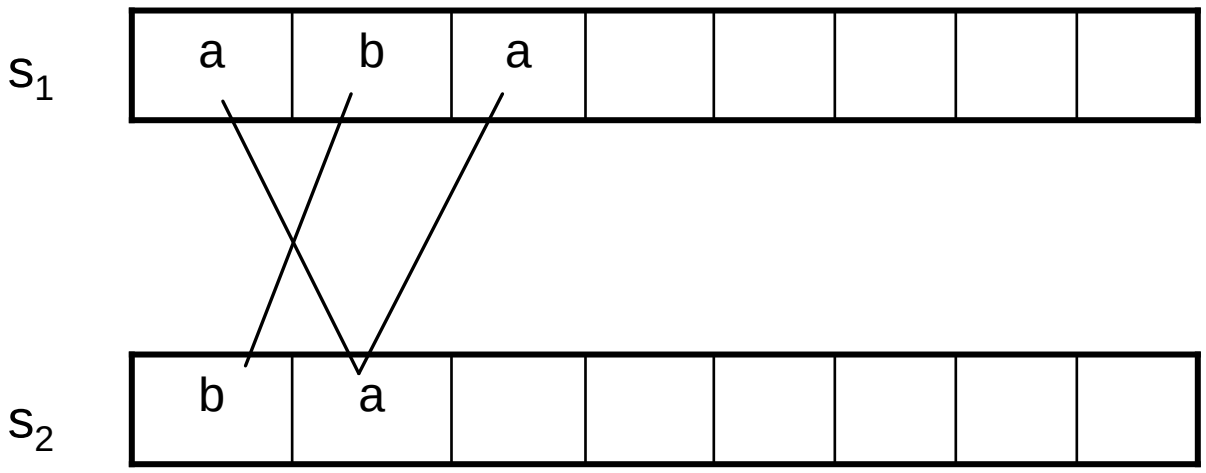
Theorem *[Bonizzoni et al, TCBB, 2007]: ELCS problem is polynomial time solvable when each mandatory symbol appears totally at most three times in the input strings.*

Proof.

Each mandatory symbol can have **at most two occurrences** in each input string

ELCS can be reduced to 2SAT

ECLS - complexity



Feasible solution: **no crossing lines**

1. Boolean variable for each occurrence of a symbol in an input string
2. Clause for each pair of crossing line

ECLS - complexity

Theorem *[Bonizzoni et al, TCBB, 2007]: ELCS problem is NP-hard when each mandatory symbol appears at most three times in each input string.*

Proof.

Reduction from 3SAT similar to the reduction for RFLCS

ELCS (1, ≤1)

Theorem *[Bonizzoni et al, TCBB, 2007]: ELCS(1;≤ 1) problem is* ***APX-hard*** *even when each symbol appears at most twice in each input string.*

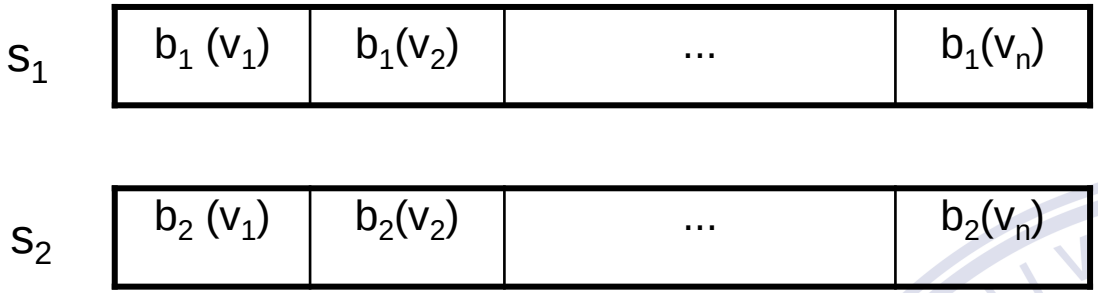
Proof.

Reduction from Max Independent Set on Cubic Graphs

ELCS (1, ≤1)

Proof.

- Input strings s_1, s_2 are divided in **blocks**
- For each vertex v_i of V → a block $b_j(v_i)$ in string s_j (j=1,2)



ELCS (1, ≤1)

v_i	$e_1(v_i)$	$e_2(v_i)$	$e_3(v_i)$	x_i

i-th block of s_1

v_k	$e_1(v_k)$	$e_2(v_k)$	$e_3(v_k)$	x_k

k-th block of s_1

- Edge $\{v_i, v_k\}$:
  - first edge incident on v_i,
  - second edge incident on v_k
- Encoded by a **mandatory** symbol

ELCS (1, ≤1)

i-th block of s_1

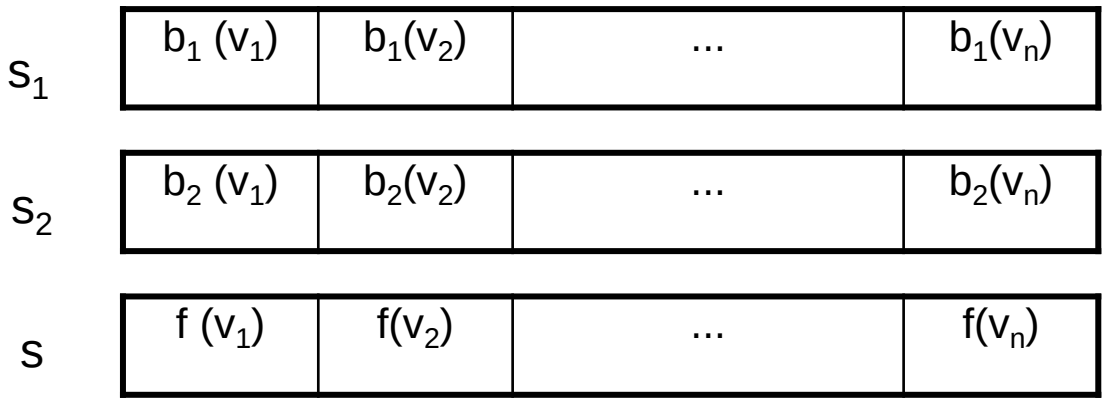
v_i	$e_1(v_i)$	$e_2(v_i)$	$e_3(v_i)$	x_i

i-th block of s_2

$e_1(v_i)$	$e_2(v_i)$	$e_3(v_i)$	v_i	x_i

- Symbol x_i is **mandatory**
- Symbol v_i is **optional**
- $e_j(v_i)$: j-th edge incident on v_i encoded by a **mandatory** symbol

ELCS (1, ≤1)



- Any feasible solution ***s*** must contain symbol x_i
- Any feasible solution ***s*** can be divided in blocks
- Each block $f(v_i)$ is either $v_i x_i$ (Max Ind Set) or a subsequence of $e_1(v_i)e_2(v_i)e_3(v_i)\ x_i$

ELCS (≥1, ≤1)

Theorem *[Bonizzoni et al, TCBB, 2007]: ELCS(≥1; ≤1) is APX-hard even when each symbol appears at most twice in each input string.*

Proof.

- Similar to the previous reduction
- Each mandatory symbol must have at least one occurrence
- Each optional symbol v_i is encoded with four optional symbols: $v_i^a\, v_i^b\, v_i^c\, v_i^d$

ELCS - Parameterized Complexity

Restriction of ELCS and ELCS(≥1) when the set A_m of mandatory symbols is a **parameter** *[Bonizzoni et al, TCBB, 2007]:*

- Dynamic programming algorithm to
  - Store the mandatory symbols used
  - Fill the gaps between a pair of mandatory symbols

Variant of LCS – Open problems

Approximation complexity of RFLCS

- Constant factor approximation algorithms?
- Hardness results?

ELCS

- Complexity when each symbol occurs less than three times in one input string, more than three times in the other

Other variants with combined constraints

Genome Alignment

Genome comparison

Genome comparison → infer mutations inside genomes

- macro-evolutionary events
  - **rearrangements** (inversions, transpositions...)
  - **content modifying operations** (duplications, losses, horizontal gene transfers,...)

Duplication-loss model

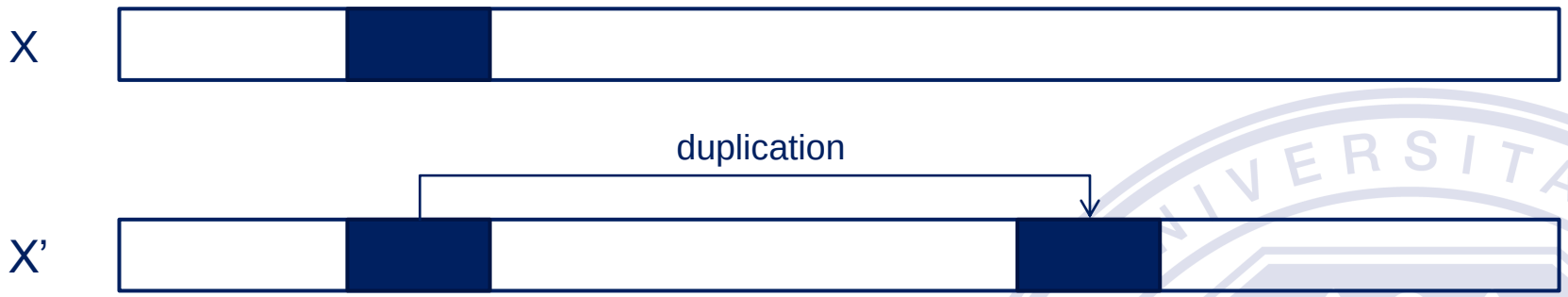
Duplication-loss model *[Holloway et al., RECOMB 2012]*: evolutionary model restricted to two evolutionary events

- **duplications**
- **losses**

- Goal: inference of **ancestral genomes** and **evolutionary events**
- Rearrangements operations ignored: organization preserved
- Application to tRNA in bacteria

Duplication-loss model

Duplication of size k: operation that copies a substring of size k of a genome somewhere else in the genome



Duplication-loss model

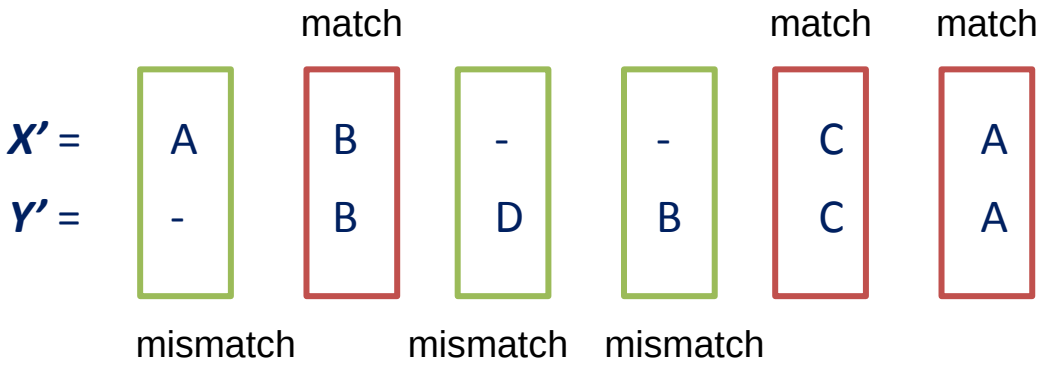
A **loss** of size k is an operation that removes a substring of size k from a genome

Genome Alignment

An **alignment** of genome X and Y → pair (X' , Y') of strings obtained by filling X and Y respectively with gaps (i.e. -), such that:

- |X'| = |Y'|
- For each position i
  - X'[i] = Y'[i] ≠ – (a **match**)
  - Either X'[i] = - or Y'[i] = - (a **mismatch**)

Genome Alignment



Genome Alignment

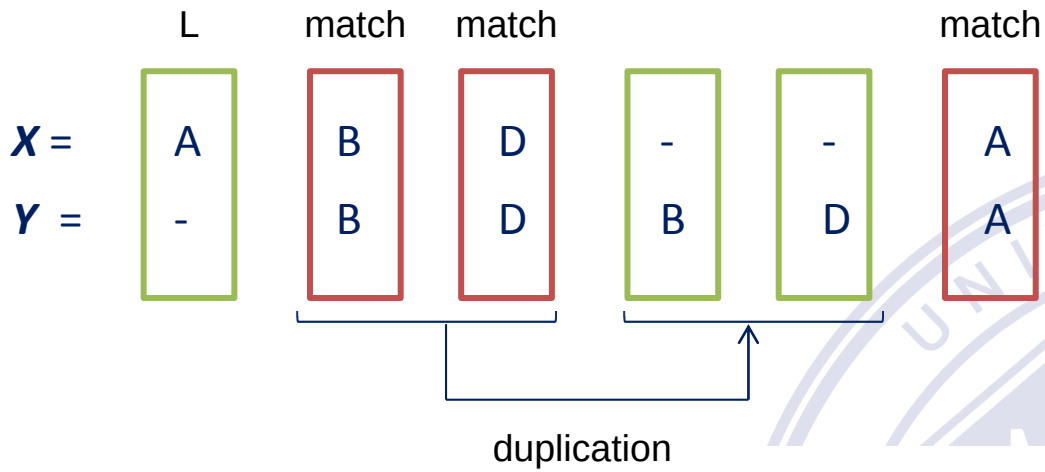
Given two aligned genomes:

- **matches**: genes in both genomes
- **mismatches**: genes (copies of genes) in one of the two genome

Labeling of the mismatched positions of the aligned genomes in terms of duplications and losses

Genome Alignment

Labeling L(X) of an aligned genome X: set of losses and duplications, such that each mismatched position of X is labeled either as a loss or as a duplication



Genome Alignment

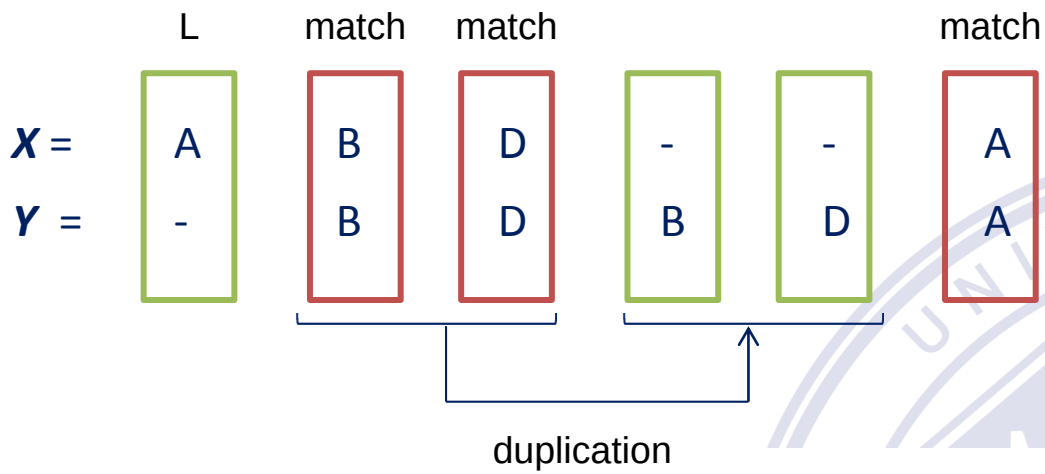
- The cost of a labeling L(X) is the cost of the **underlying operations** (losses and duplications)
- The cost of a labeled alignment (L(X), L(Y)) is the sum of cost of the two labeling L(X) and L(Y)

Usually cost *C(L(k))=k, c(D(k))=1*

Genome Alignment

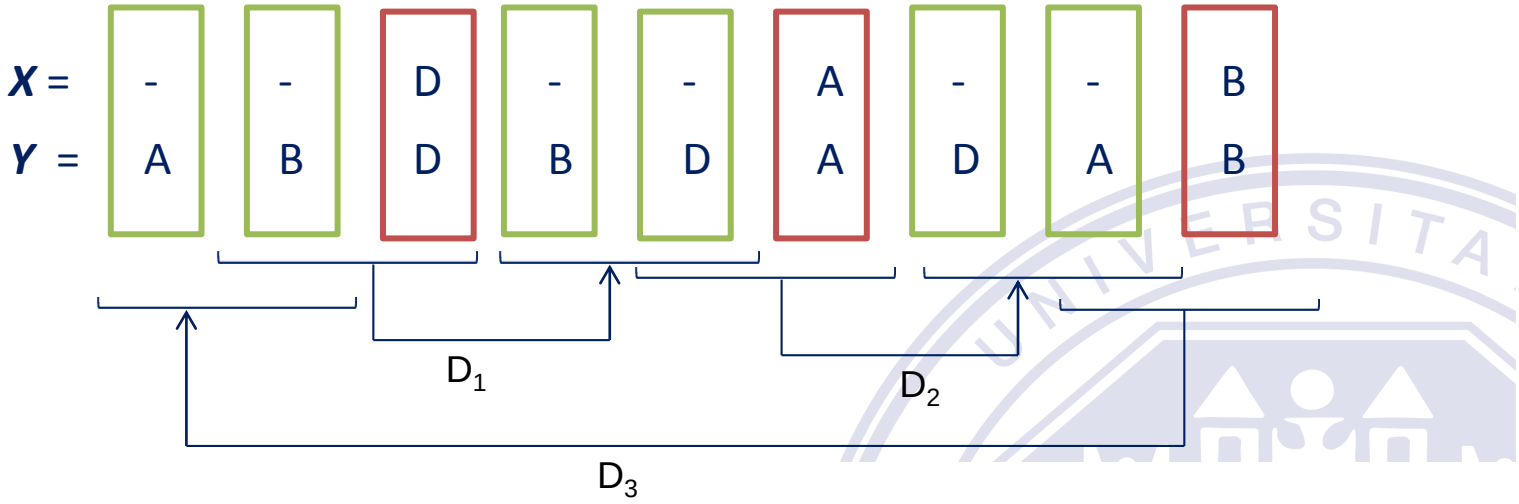
Alignment of cost **two**:

- **one loss**
- **one duplication**



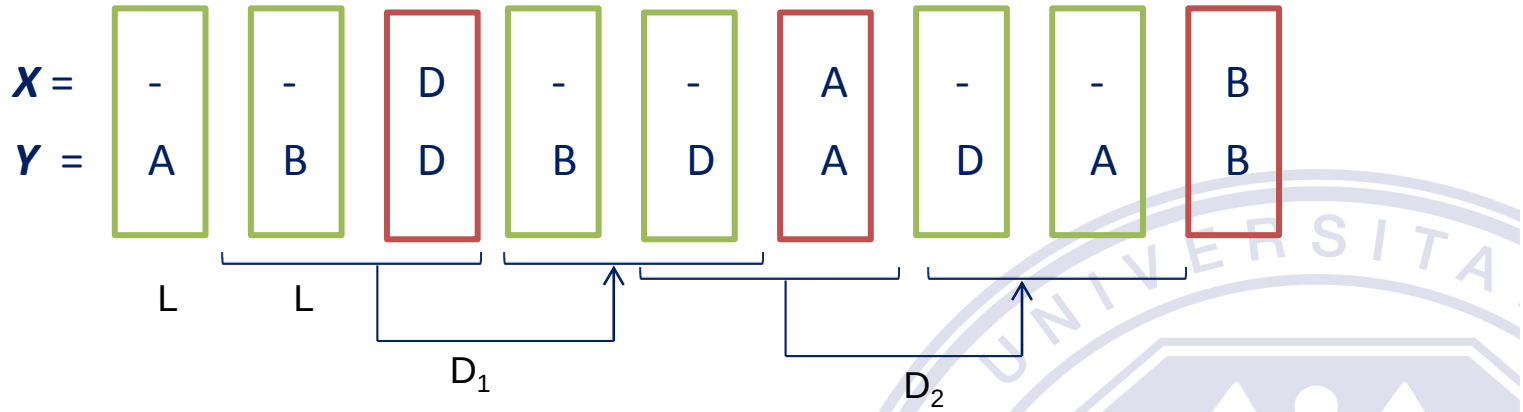
Genome Alignment

A labeling of an aligned genome can be **cyclic** → not biologically consistent



Genome Alignment

Given an aligned genomes, a labeling is **feasible** if there is no subset of duplications that induces a **duplication cycle**



Genome Alignment

Duplication-Loss Alignment problem [DLA]

Input: Two genomes X and Y.

Output: A Feasible Labeled Alignment (L(X), L(Y)) of minimum cost.

Genome Alignment

Previous results

- Dynamic programming does not work *[Holloway et al., RECOMB 2012]*
- Exact Pseudo Boolean programming *[Holloway et al., RECOMB 2012]*

More recently *[Canzar and Andreotti, Arxiv, 2012]*

- DLA is NP-hard
- Branch and Cut Algorithm

Genome Alignment – New approach

Possible heuristic for DLA:

1. Align optimally two genomes → **dynamic programming**
2. **Label** the given aligned genomes

Property

Each genome can be labeled **independently**

Minimum Labeling Alignment

Minimum Labeling Alignment Problem [MLA]

Input: An aligned genome X.

Output: A Feasible Labeling L(X) of minimum cost.

MLA - Complexity

Theorem *[Dondi and El-Mabrouk, Arxiv, 2012]: Minimum Label Alignment is APX-hard.*

Proof.

L-reduction from *Minimum Vertex Cover on Cubic Graphs*

$$X = B(v_1)...B(v_n)\ B(e_{1,a})...B(e_{z,w})$$
$$B(A,1,v_1)...B(A,2,v_1)...B(A,1,v_n)...B(A,2,v_n)$$

MLA - Complexity

High level idea:

- B(A,x,v_i) → matched
- Labeling of B(v_i):
  - **duplications** from B($e_{i,j}$), B($e_{i,h}$), B($e_{i,k}$) B(A,1,v_i) → cost 7 (**independent set**)
  - **duplications** from B(A,2,v_i) → cost 8 (**vertex cover**)

MLA - Complexity

High level idea:

- Labeling of $B(e_{i,j})$:
  - A duplication from one of $B(v_i)$, $B(v_j)$
- **To avoid cycles**
  - If there is a duplication from one of $B(v_i)$ to $B(e_{i,j})$ → no duplication from $B(e_{i,j})$ to B(vi)

MLA - Complexity

***Lemma**: there exists a vertex cover V' of G iff there exists a feasible labeling of X of cost 8|V'| + 7 |V-V'|+2|E|.*

***Theorem:** MLA is APX-hard even if each symbol has at most 5 occurrences in X.*

Label Alignment – Open Problems

- Approximation complexity of DLA and MLA
- New (heuristics) approaches to DLA
- Complexity of MLA with [3,4] occurrences for each symbol

Conclusion

Variants of LCS

- **Repetition Free Longest Common Subsequence**
  - Complexity
  - Approximation Algorithms
  - FPT algorithms
- **Exemplar Longest Common Subsequence (ELCS)**
  - Complexity of variants of ELCS

Genome Alignment

- **Duplication-Loss Model of evolution**
- **Duplication-Loss Alignment** problem
  - Complexity
- **Minimum Label Alignment** problem
  - Complexity of Minimum Label Alignment problem

References - RFLCS and ELCS

- Guillaume Blin, Paola Bonizzoni, Riccardo Dondi, Florian Sikora: On the parameterized complexity of the repetition free longest common subsequence problem. *Inf. Process. Lett.* 112(7): 272-276 (2012)
- Paola Bonizzoni, Gianluca Della Vedova, Riccardo Dondi, Yuri Pirola: Variants of constrained longest common subsequence. *Inf. Process. Lett.* 110(20): 877-881 (2010)
- Paola Bonizzoni, Gianluca Della Vedova, Riccardo Dondi, Guillaume Fertin, Raffaella Rizzi, Stéphane Vialette: Exemplar Longest Common Subsequence. *IEEE/ACM Trans. Comput. Biology Bioinform.* 4(4): 535-543 (2007)
- Carlos Eduardo Ferreira, Christian Tjandraatmadja: A branch-and-cut approach to the repetition-free longest common subsequence problem. *Electronic Notes in Discrete Mathematics* 36: 527-534 (2010)
- Said Sadique Adi, Marília D. V. Braga, Cristina G. Fernandes, Carlos Eduardo Ferreira, Fábio Viduani Martinez, Marie-France Sagot, Marco A. Stefanes, Christian Tjandraatmadja, Yoshiko Wakabayashi: Repetition-free longest common subsequence. *Discrete Applied Mathematics* 158(12): 1315-1324 (2010)
- Cristina G. Fernandes, Carlos Eduardo Ferreira, Christian Tjandraatmadja, Yoshiko Wakabayashi: A Polyhedral Investigation of the LCS Problem and a Repetition-Free Variant. *LATIN* 2008: 329-338

References - DLA and MLA

- Stefan Canzar, Sandro Andreotti: A Branch-and-Cut Algorithm for the 2-Species Duplication-Loss Phylogeny Problem. *CoRR* abs/1208.2698 (2012)
- Riccardo Dondi, Nadia El-Mabrouk: On the Complexity of Minimum Labeling Alignment of Two Genomes. *CoRR* abs/1206.1877 (2012)
- Patrick Holloway, Krister M. Swenson, David H. Ardell, Nadia El-Mabrouk: Evolution of Genome Organization by Duplication and Loss: An Alignment Approach. *RECOMB* 2012: 94-112

Some Recent Combinatorial Approaches To Genome Comparison

Thank you!

Questions?