

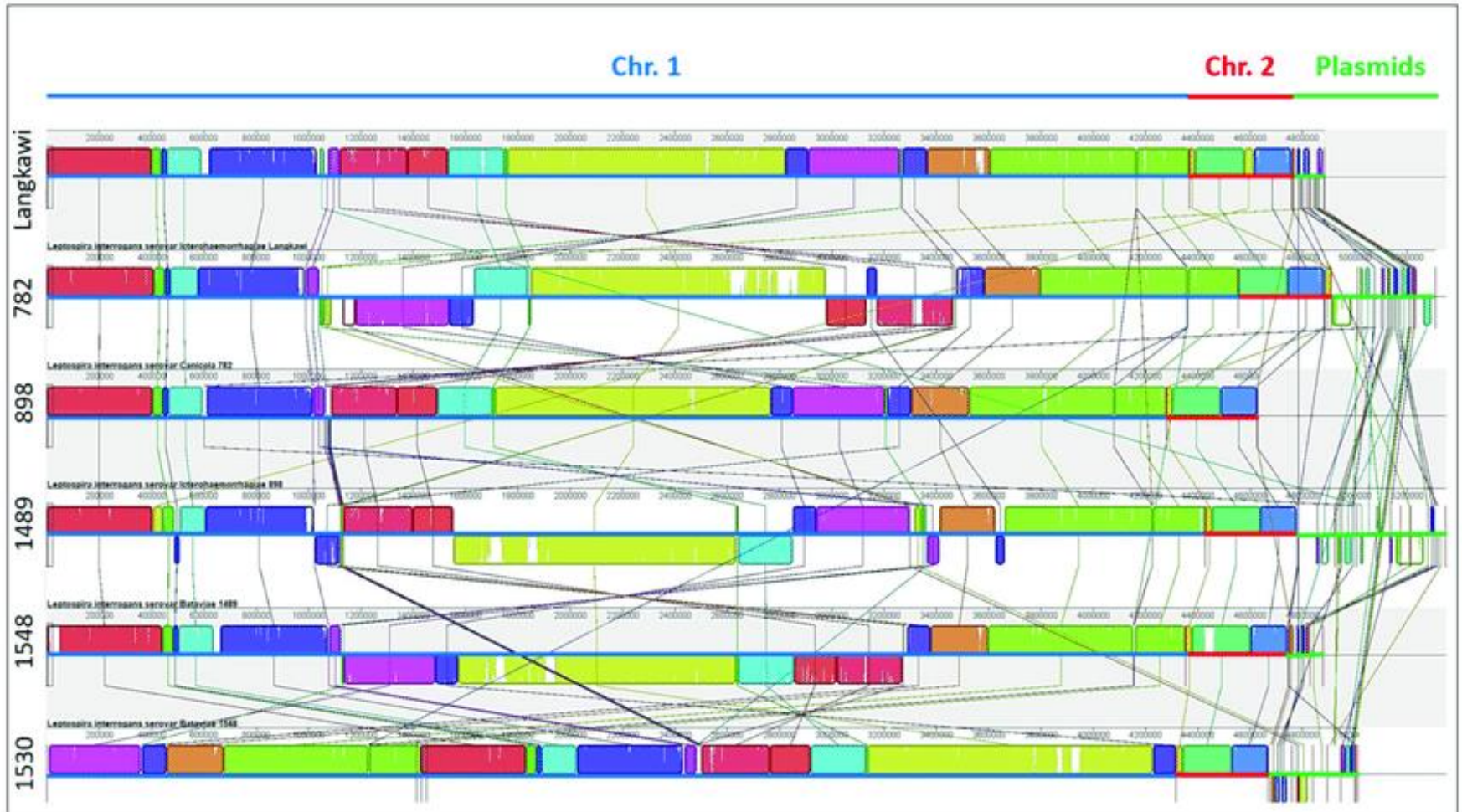
THE COMPLEXITY OF FINDING COMMON PARTITIONS OF GENOMES WITH PREDEFINED BLOCK SIZES

Manuel Lafond¹, Adiesha Liyanage², Binhai Zhu², Peng Zou²

¹Université de Sherbrooke, Canada

²Montana State University, USA

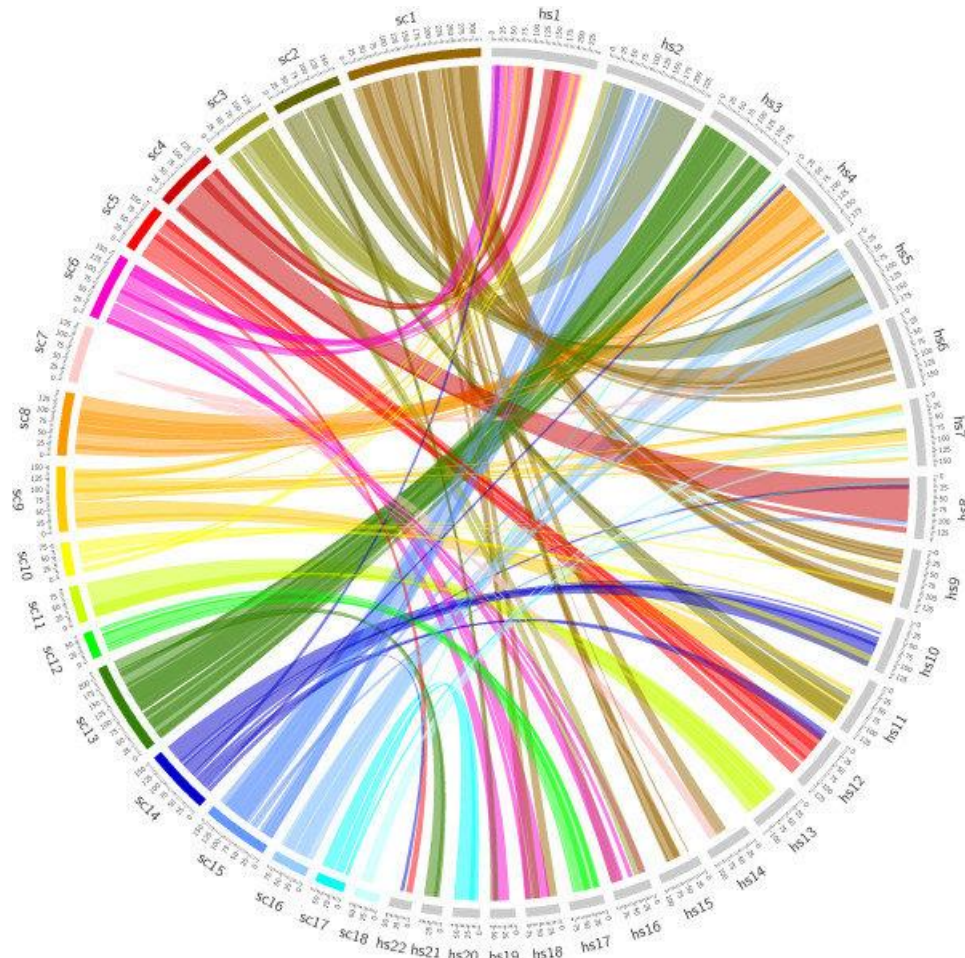
Synten : block of at least two genes that was conserved across multiple species.



Leptospira strains synteny

Image taken from Ramli et al.: <https://doi.org/10.3390/pathogens10091198>

Synten : block of at least two genes that was conserved across multiple species.



Human vs pig syntenic map

Image taken from Kim et al.: <https://doi.org/10.1186/1471-2164-13-711>

Synteny : block of at least two genes that was conserved across multiple species.

To find syntenic blocks between two genomes S and T :

- Partition genes into homologous families
- Represent S and T as strings (each symbol = 1 gene family)
- Partition S and T into identical substrings (blocks)

a a b a a b a a a b a b

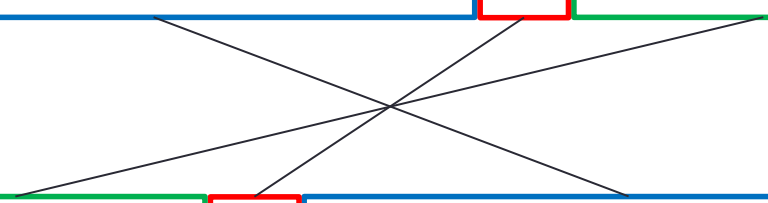
a b a b a a a b a a b a

a a b a a b a a a b a b

a b a b a a a b a a b a

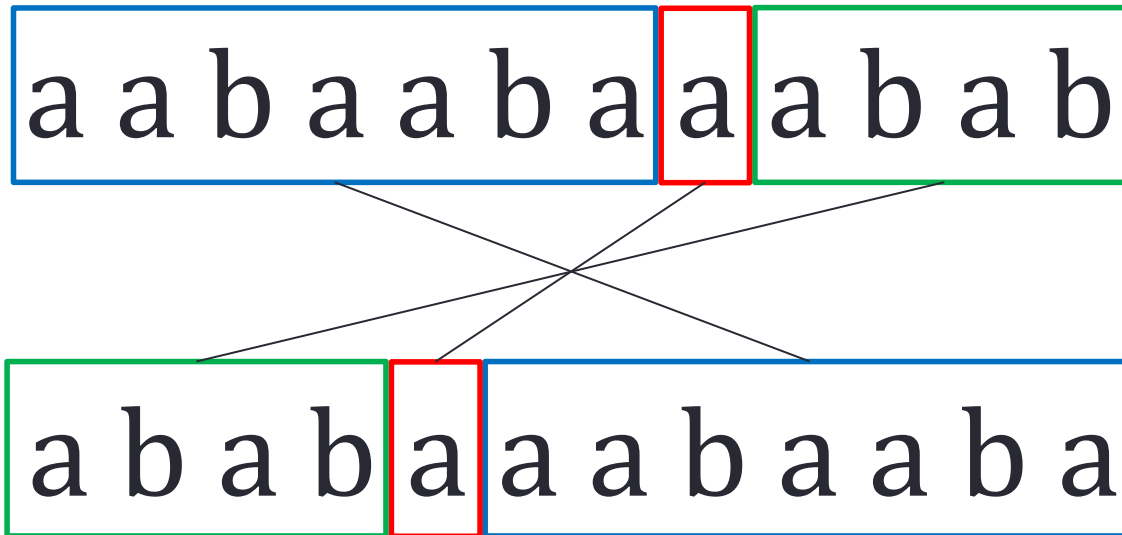
a a b a a b a a a b a b

a b a b a a a b a a b a



Usual formulation: Minimum Common String Partition (MCSP)

Given strings S, T , split them into two identical (multi)sets of blocks, while minimizing the number of blocks.



Usual formulation: Minimum Common String Partition (MCSP)

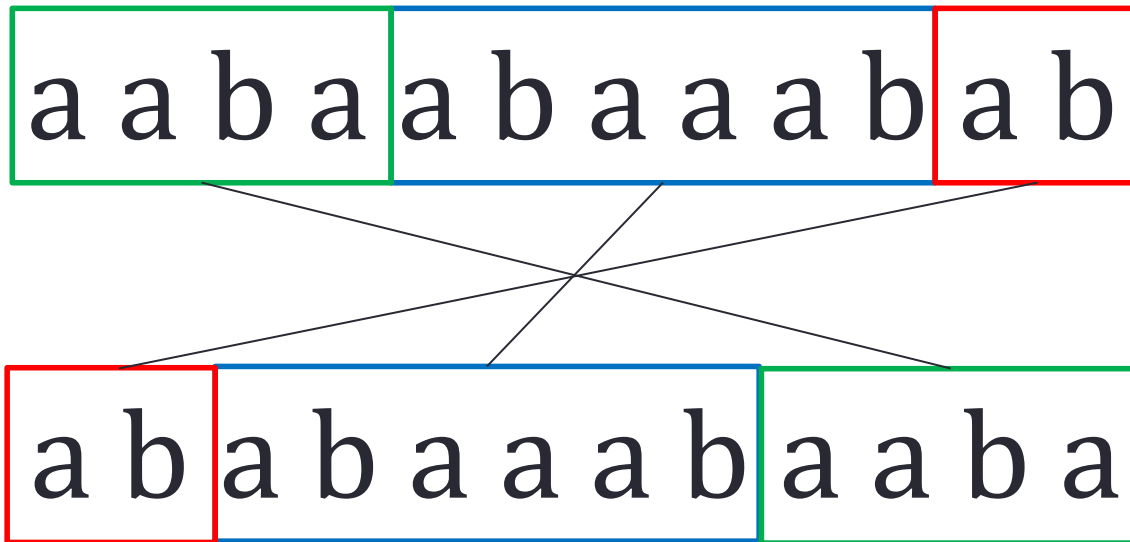
Given strings S, T , split them into two identical (multi)sets of blocks, while minimizing the number of blocks.

a a b a a b a a a b a b

a b a b a a a b a a b a

Usual formulation: Minimum Common String Partition (MCSP)

Given strings S, T , split them into two identical (multi)sets of blocks, while minimizing the number of blocks.



a a b a a b a a a b a b

a b a b a a a b a a b a

a a b a a b a a a b a b

a b a b a a a b a a b a

a a b a a b a a a b a b

a b a b a a a b a a b a

a a b a a b a a a b a b

a b a b a a a b a a b a

NO SINGLETONS
ONLY SYNTENIES

The Exact Strip Recovery problem

Given two strings S , T and min block size b , **does there exist** a common partition of S and T in which the size of each block is at least b ?

The Exact Strip Recovery problem

Given two strings S , T and min block size b , **does there exist** a common partition of S and T in which the size of each block is at least b ?

The Min-Strip Recovery Problem

Given two strings S , T and min block size b , **delete a minimum number** of characters from the strings, so that they admit a common partition with blocks of size at least b .

The Min-Strip Recovery Problem

Given two strings S, T and min block size b , **delete a minimum number** of characters from the strings, so that they admit a common partition with blocks of size at least b .

$$b = 4$$

a a b a a b a a a b a b

a b a b b a a b a a b a

The Min-Strip Recovery Problem

Given two strings S, T and min block size b , **delete a minimum number** of characters from the strings, so that they admit a common partition with blocks of size at least b .

$$b = 4$$

a a b a a b a ~~a~~ a b a b

a b a b ~~b~~ a a b a a b a

Past results

- Min Strip Recovery
- Formulation in [Zheng, Zhu & Sankoff 2007]
- Heuristics based on Maximum Clique [Choi et al 2007]

RICE																					
1	+001	+003	+004	+006	+007	+008	+010	+013	+014	+016	+017	+019	+020	+021	+022	+023	+024	+028	+029	+032	+035
+037	+038	+039	+040	+041	+042	+043	+044	+049	+051	+052	+053	+054	+055	+056	+057	+058	+059	+060	+061	+062	+063
+064	+065	+067	+069	+070	+071	+072	+073	+076													
2	+080	+081	+083	+084	+085	+086	+087	+088	+089	+090	+092	+093	+094	+095	+096	+100	+102	+104	+105	+106	+109
+110	+111	+112	+114	+115	+117	+118	+119	+120	+121	+122	+123	+124	+125	+126	+127	+128	+129				
3	+131	+132	+133	+135	+136	+137	+138	+139	+140	+144	+145	+147	+148	+149	+150	+151	+152	+153	+154	+155	+156
+157	+158	+159	+160	+161	+162	+163	+164	+165	+169	+170	+172	+175	+177	+179	+181	+182	+183	+184	+185	+187	+188
+191	+192	+193	+194	+195	+198	+200	+201	+202	+203												
4	+206	+207	+210	+211	+212	+214	+217	+218	+220	+222	+223	+224	+226	+229	+230	+234	+235	+237	+238		
5	+239	+242	+243	+244	+245	+246	+249	+250	+251	+252	+260	+261	+262	+263	+265	+268	+269	+272	+273	+274	+276
+277	+279																				
6	+280	+281	+283	+284	+285	+286	+288	+289	+290	+291	+292	+293	+295	+299	+304	+305	+307	+308	+310	+311	+313
+314	+315	+317	+318	+320	+321	+322	+323	+324	+325	+326											
7	+327	+328	+329	+330	+332	+334	+336	+337	+338	+340	+342	+343	+344	+347	+350	+351	+352	+355	+356	+357	
8	+358	+362	+363	+365	+367	+371	+372	+373	+374	+377	+378	+379	+380	+381	+382	+383	+386	+387			
9	+388	+392	+393	+394	+395	+396	+397	+398	+399	+402	+403	+405	+409	+410	+413						
10	+416	+418	+420	+423	+425	+426	+427	+429													
11	+431	+432	+434	+435	+436	+438	+439	+440	+442	+443	+447	+448	+449	+450	+452	+454	+455	+456			
12	+460	+464	+466	+470	+474	+477															
SORGHUM																					
A	-013	-010	-008	-007	-006	-004	-003	-001	+014	+016	+017	+019	+020	+021	+022	+023	+024	+028	+029	+032	+035
+037	+038	+039	+040	+041	+042	+043	+044	+049	+051	+054	+055	+056	+057	+058	+059	+060	+061	+062	+063	+064	+065
+067	+069	+070	+071	+072	+073	+076															
B	-357	+161	+162	+163	-356	-355	-352	-351	-350	-347	-344	+340	+342	+343	-338	-337	-413	-410	-409	-405	-403
-402	-399	-398	-397b+395	+396	+392	+393	+394	-388	-336	-334	-332	-330	-329	-328	-327						
C	+202	+203	+200	+201	-198	-195	-194	-193	+191	+192	+184	+185	+187	+188	-183	-182	-181	-179	-177	-175	-426
-425	-423	+052b	+053	+109	+110	+386	+387	+420	-418	-416	-429	-427	-172	-170	-169b	-165	-164	+156	+157	+158	+159
+160	+154	+155	-153	+150	+151	+152	+148	+149	-147	-145	-144	-137	+135	+136	-133	-132	-131				
D	-238	-237	-235	-234	-230	-229	-226	-224	+222	+223	-220	-218	-217	+210	+211	+212	+214	-207	-206		
E	-477b	-474	-470	-466b	-464	-460															
F	+128	+129	+126	+127	+124	+125	-123	-122	-121	+111	+112	+114	+115	+117	+118	+119	+120	+102	+104	+105	+106
-096	+094	+095b	-093	-092	-090	+088	+089	+086	+087	+084	+085	-083	-081	-080							
G	-279	-277	-276	-274	-273	-272	-269	-268b	-265	-263	+252	+260	+261b	+262	-251	-250	-249	+245b+246	-244	-243	
-242	-239																				
H	+431	+432	+434	+435c	+436c	+438	+439b	+440	+442	+443	+447	+448	-450	-449	+452	+454	+455	+456			
I	+325	+326	+323	+324b	-322	-321	-320	-318	-317b	-315	+313	+314	-311	+138	+139b	+140	+305	+307	+308	+310	-304
+293	+295	+299	+291	+292	-290	-289	-288	-286	-285	-284	-283	-281	-280								
J	+377	+378	+379	+380	+381b	+382	+383	-100	+367b	+371	+372	+373	+374	-365	-363b	-362	-358				

Past results

- Min Strip Recovery
- Formulation in [Zheng, Zhu & Sankoff 2007]
- Heuristics based on Maximum Clique [Choi et al 2007]
- NP-hard even on permutations [Wang & Zhu 2010]
- Some approximation results
 - No arbitrarily good approx. [Jiang 2011]
 - Some constant factor approx. [Chen et al. 2009]
- $O(3^k * n)$ time algorithm, $k = \#$ of genes to delete [Jiang et al. 2012]

Past results

- Min Strip Recovery
- Formulation in [Zheng, Zhu & Sankoff 2007]
- Heuristics based on Maximum Clique [Choi et al 2007]
- NP-hard even on permutations [Wang & Zhu 2010]
- Some approximation results
 - No arbitrarily good approx. [Jiang 2011]
 - Some constant factor approx. [Chen et al. 2009]
- $O(3^k * n)$ time algorithm, $k = \#$ of genes to delete [Jiang et al. 2012]
- This paper: polynomial time on fixed alphabets

Past results

- **Exact Strip Recovery**
- Exact Recovery = special case with 0 deletions allowed
- Easy if input strings are permutation (no duplicates)
- Otherwise, complexity = open problem [Bulteau & Weller 2019]

Past results

- Exact Strip Recovery
- Exact Recovery = special case with 0 deletions allowed
- Easy if input strings are permutation (no duplicates)
- Otherwise, complexity = open problem [Bulteau & Weller 2019]

- This paper: NP-hard if genes have duplicates

NP-hardness of a generalized Exact-Strip Recovery problem

The Exact Strip Recovery problem

Given two strings S, T and min block size b , does there exist a common partition of S and T in which the size of each block is at least b ?

The Exact Strip Recovery problem

Given two strings S, T and min block size b , does there exist a common partition of S and T in which the size of each block is at least b ?

$b = 2$: the answer is yes if and only if there is a common partition with block sizes in $\{2, 3\}$

Intuition: size 4 block = 2+2, size 5 block = 2+3, ...

The Exact Strip Recovery problem

Given two strings S, T and min block size b , does there exist a common partition of S and T in which the size of each block is at least b ?

$b = 2$: the answer is yes if and only if there is a common partition with block sizes in $\{2, 3\}$

$b = 3$: the answer is yes if and only if there is a common partition with block sizes in $\{3, 4, 5\}$

The Exact Strip Recovery problem

Given two strings S, T and min block size b , does there exist a common partition of S and T in which the size of each block is at least b ?

$b = 2$: the answer is yes if and only if there is a common partition with block sizes in $\{2, 3\}$

$b = 3$: the answer is yes if and only if there is a common partition with block sizes in $\{3, 4, 5\}$

$b = 4$: the answer is yes if and only if there is a common partition with block sizes in $\{4, 5, 6, 7\}$

The Exact Strip Recovery problem

Given two strings S, T and min block size b , does there exist a common partition of S and T in which the size of each block is at least b ?

Lemma

For any infinite set F of allowed block sizes, there exists a finite set F' such that the strip recovery problem with allowed block sizes F or F' are equivalent (i.e. admit same set of solutions).

The Exact Strip Recovery problem

Given two strings S, T and min block size b , does there exist a common partition of S and T in which the size of each block is at least b ?

Lemma

For any infinite set F of allowed block sizes, there exists a finite set F' such that the strip recovery problem with allowed block sizes F or F' are equivalent (i.e. admit same set of solutions).

- We might as well **generalize to arbitrary allowed block sizes F** .

The Exact F-Strip Recovery problem (XSR-F)

Input: two strings S, T

Question: does there exist a common partition of S and T in which the size of each block is in F ?

Here, F is a **fixed** set of integers.

The Exact F-Strip Recovery problem (XSR-F)

Input: two strings S, T

Question: does there exist a common partition of S and T in which the size of each block is in F ?

Here, F is a **fixed** set of integers.

Theorem

If all sizes in F are a multiple of $\min(F)$, then the problem can be solved in polynomial time.

For **any other** F , the problem is NP-hard, even if each symbol **occurs at most 6 times** in the input strings.

Theorem

If all sizes in F are a multiple of $\min(F)$, then the problem can be solved in polynomial time.

$$F = \{2, 4, 6\}$$

a c a b a a c b a c

c b a a a c a c a b

Theorem

If all sizes in F are a multiple of $\min(F)$, then the problem can be solved in polynomial time.

$F = \{2, 4, 6\}$ **equivalent to** $F = \{2\}$

a c a b a a c b a c

c b a a a c a c a b

Theorem

If all sizes in F are a multiple of $\min(F)$, then the problem can be solved in polynomial time.

$F = \{2, 4, 6\}$ **equivalent to** $F = \{2\}$

a	c	a	b	a	a	c	b	a	c
---	---	---	---	---	---	---	---	---	---

c	b	a	a	a	c	a	c	a	b
---	---	---	---	---	---	---	---	---	---

Theorem

If all sizes in F are a multiple of $\min(F)$, then the problem can be solved in polynomial time.

$F = \{2, 4, 6\}$ **equivalent to** $F = \{2\}$

a	c	a	b	a	a	c	b	a	c
---	---	---	---	---	---	---	---	---	---

c	b	a	a	a	c	a	c	a	b
---	---	---	---	---	---	---	---	---	---

Theorem

If all sizes in F are a multiple of $\min(F)$, then the problem can be solved in polynomial time.

Theorem (continued)

For **any other F** , the problem is NP-hard, even if each symbol **occurs at most 6 times** in the input strings.

(in particular, hard for $F = \{2,3\}$ representing blocks sizes of size at least 2)

Theorem (continued)

For **any other F** , the problem is NP-hard, even if each symbol **occurs at most 6 times** in the input strings.

Reduction from *Positive Cubic 1-in-3 SAT* to XSR-F with **string sets**.

- Given a boolean formula in which each clause has 3 **positive** variables and each variable x_i occurs exactly three times.
- Goal: assign x_i 's to *true* or *false* so that for each clause, **exactly** one of its variables is true.

$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_4 \vee x_5) \wedge (x_1 \vee x_2 \vee x_5) \wedge \dots \Rightarrow S, T$
Satisfiable iff Exact partition exists

for each variable x_i

Define X_i and X'_i as

$$\begin{aligned} X_i &= P_i^1 P_i^2 \dots P_i^r P_i^* L_a M_a R_a Q_i L_b M_b R_b S_i L_c M_c R_c T_i^* T_i^1 T_i^2 \dots T_i^r \\ X'_i &= P_i^1 P_i^2 \dots P_i^r P_i^* L_a M'_a R_a Q_i L_b M'_b R_b S_i L_c M'_c R_c T_i^* T_i^1 T_i^2 \dots T_i^r \end{aligned}$$

We put $\mathcal{W}_1 = \{X_1, X'_1, X_2, X'_2, \dots, X_n, X'_n\}$.

$$\begin{aligned} X_i &= \underline{P_i^1} \underline{P_i^2} \dots \underline{P_i^r} \underline{P_i^*} \underline{L_a} \underline{M_a} \underline{R_a} \underline{Q_i} \underline{L_b} \underline{M_b} \underline{R_b} \underline{S_i} \underline{L_c} \underline{M_c} \underline{R_c} \underline{T_i^*} \underline{T_i^1} \underline{T_i^2} \dots \underline{T_i^r} \\ X'_i &= \underline{P_i^1} \underline{P_i^2} \dots \underline{P_i^r} \underline{P_i^*} \underline{L_a} \underline{M'_a} \underline{R_a} \underline{Q_i} \underline{L_b} \underline{M'_b} \underline{R_b} \underline{S_i} \underline{L_c} \underline{M'_c} \underline{R_c} \underline{T_i^*} \underline{T_i^1} \underline{T_i^2} \dots \underline{T_i^r} \end{aligned}$$

Fig. 2. Partition of X_i and X'_i corresponding to $x_i = true$.

$$\begin{aligned} X_i &= \underline{P_i^1} \underline{P_i^2} \dots \underline{P_i^r} \underline{P_i^*} \underline{L_a} \underline{M_a} \underline{R_a} \underline{Q_i} \underline{L_b} \underline{M_b} \underline{R_b} \underline{S_i} \underline{L_c} \underline{M_c} \underline{R_c} \underline{T_i^*} \underline{T_i^1} \underline{T_i^2} \dots \underline{T_i^r} \\ X'_i &= \underline{P_i^1} \underline{P_i^2} \dots \underline{P_i^r} \underline{P_i^*} \underline{L_a} \underline{M'_a} \underline{R_a} \underline{Q_i} \underline{L_b} \underline{M'_b} \underline{R_b} \underline{S_i} \underline{L_c} \underline{M'_c} \underline{R_c} \underline{T_i^*} \underline{T_i^1} \underline{T_i^2} \dots \underline{T_i^r} \end{aligned}$$

Fig. 3. Partition of X_i and X'_i corresponding to $x_i = false$.

for each variable x_i

Define X_i and X'_i as

$$X_i = P_i^1 P_i^2 \dots P_i^r P_i^* L_a M_a R_a Q_i L_b M_b R_b S_i L_c M_c R_c T_i^* T_i^1 T_i^2 \dots T_i^r$$

$$X'_i = P_i^1 P_i^2 \dots P_i^r P_i^* L_a M'_a R_a Q_i L_b M'_b R_b S_i L_c M'_c R_c T_i^* T_i^1 T_i^2 \dots T_i^r$$

We put $\mathcal{W}_1 = \{X_1, X'_1, X_2, X'_2, \dots, X_n, X'_n\}$.

As for \mathcal{W}_2 , each of its strings has length ℓ or h . First, for each variable x_i , with C_a, C_b, C_c the clauses containing x_i , add the following strings to \mathcal{W}_2 :

- $P_i^1, P_i^2, \dots, P_i^r$ (r strings of length ℓ)
- $P_i^1 P_i^2 \dots P_i^r P_i^*$ (one string of length $r\ell + \ell - s = h$)
- $P_i^* L_a$ (one string of length $\ell - s + s = \ell$)
- $R_a Q_i, Q_i L_b, R_b S_i, S_i L_c$ (four strings of length $s + \ell - s = \ell$)
- $R_c T_i^*$ (one string of length $s + \ell - s = \ell$)
- $T_i^* T_i^1 T_i^2 \dots T_i^r$ (one string of length $r\ell + \ell - s = h$)
- $T_i^1, T_i^2, \dots, T_i^r$ (r strings of length ℓ)

Then, for each clause C_a , add the following strings to \mathcal{W}_2 :

$$- L_a M_a, M'_a R_a \quad \text{(two strings of length } h)$$

$$- M_a R_a, R_a M'_a, M'_a L_a, L_a M_a \quad \text{(four strings of length } h)$$

Second step:

Reduction from XSR-F with **string sets** to XSR-F with **single sequences**.

Denote $\mathcal{W}_1 = \{A_1, \dots, A_n\}$ and $\mathcal{W}_2 = \{B_1, \dots, B_m\}$

Now build single strings S and T as follows:

$$S = A_1 X_1 Y_1 Z_1 A_2 X_2 Y_2 Z_2 \dots A_n X_n Y_n Z_n$$

$$T = B_1 Z_1 Y_1 X_1 B_2 Z_2 Y_2 X_2 \dots B_m Z_m Y_m X_m Z_{m+1} Y_{m+1} X_{m+1} \dots Z_n Y_n X_n$$

We now show that $\mathcal{W}_1, \mathcal{W}_2$ admit a common F -partition if and only if S, T admit a common F -partition.

A polynomial time algorithm for fixed
alphabet and fixed F

The General F-Strip Recovery problem (GSR-F)

Input: two strings S, T

Goal: delete a **minimum number of characters** from S and T so that the resulting strings admit a common partition into blocks of sizes in F .

The General F-Strip Recovery problem (GSR-F)

Input: two strings S, T

Goal: delete a **minimum number of characters** from S and T so that the resulting strings admit a common partition into blocks of sizes in F .

Theorem

If the alphabet Σ is fixed and F is also fixed, then GSR-F can be **solved in polynomial time**.

The General F-Strip Recovery problem (GSR-F)

Input: two strings S, T

Goal: delete a **minimum number of characters** from S and T so that the resulting strings admit a common partition into blocks of sizes in F .

Theorem

If the alphabet Σ is fixed and F is also fixed, then GSR-F can be **solved in polynomial time**.

Time: $O(n^{|F| |\Sigma|^{\max(F)} + 3})$

The General F-Strip Recovery problem (GSR-F)

Input: two strings S, T

Goal: delete a **minimum number of characters** from S and T so that the resulting strings admit a common partition into blocks of sizes in F .

Theorem

If the alphabet Σ is fixed and F is also fixed, then GSR-F can be **solved in polynomial time**.

Time: $O(n^{|F| |\Sigma|^{\max(F)} + 3})$

For the standard $\Sigma = \{A, C, G, T\}$ and $F = \{2, 3\}$, this is $O(n^{131})$

Theorem

If the alphabet Σ is fixed and F is also fixed, then GSR-F can be **solved in polynomial time**.

Time: $O(n^{|F| |\Sigma|^{\max(F)} + 3})$

Dynamic programming algorithm

Block = any string over Σ whose length is in F .

Block count table = table C that assigns a number to each possible block.

aaa	aab	aba	abb	baa	bab	bba	bbb
0	0	1	0	1	0	2	0

Theorem

If the alphabet Σ is fixed and F is also fixed, then GSR-F can be **solved in polynomial time**.

Time: $O(n^{|F| |\Sigma|^{\max(F)} + 3})$

Dynamic programming algorithm

Block = any string over Σ whose length is in F .

Block count table = table C that assigns a number to each possible block.

For a string S and block count table C ,

$D(S, C)$ = min number of deletions to make in S so that the resulting string can be split into blocks, such that the number of each block is the same as in C (or infinity if not possible).

For a string S and block count table C ,
 $D(S, C) = \text{min number of deletions to make in } S \text{ so that the}$
 $\text{resulting string can be split into blocks, such that the number of}$
 $\text{each block is the same as in } C \text{ (or infinity if not possible).}$

C:

aaa	aab	aba	abb	baa	bab	bba	bbb
0	0	1	0	1	0	2	0

a b b a b a a a a b a b b a

For a string S and block count table C ,
 $D(S, C) = \text{min number of deletions to make in } S \text{ so that the}$
 $\text{resulting string can be split into blocks, such that the number of}$
 $\text{each block is the same as in } C \text{ (or infinity if not possible).}$

C:

aaa	aab	aba	abb	baa	bab	bba	bbb
0	0	1	0	1	0	2	0

~~a~~ b b a b a a ~~a~~ a b a b b a

$$D(S, C) = 2$$

The algorithm

- On input strings S, T

bestSolution = ∞

For each possible block count table C

 Compute $D(S, C)$ using DP

 Compute $D(T, C)$ using DP

If $D(S, C) + D(T, C) < \text{bestSolution}$ **then**

 bestSolution = $D(S, C) + D(T, C)$

For a string S and block count table C ,
 $D(S, C) = \text{min number of deletions to make in } S \text{ so that the}$
 $\text{resulting string can be split into blocks, such that the number of}$
 $\text{each block is the same as in } C \text{ (or infinity if not possible).}$

Idea: for any i , let $S[1..i]$ be the prefix of S of length i .
 $D(S[1..i], C)$ can be computed from all the $D(S[1..j], C')$ values,
where $j < i$.

For a string S and block count table C ,

$D(S, C) = \min$ number of deletions to make in S so that the resulting string can be split into blocks, such that the number of each block is the same as in C (or infinity if not possible).

Idea: for any i , let $S[1..i]$ be the prefix of S of length i .

$D(S[1..i], C)$ can be computed from all the $D(S[1..j], C')$ values, where $j < i$.

$$D(S[1..i], C) = \min_{X, j < i} D(S[1..j], C - X) + \text{cost}(S[j + 1..n], X)$$

$C - X$ means reduces the number of X s by 1

$\text{cost}(S[j + 1..n], X)$ is the number of deletions to make the substring become X

The algorithm

- On input strings S, T

bestSolution = ∞

For each possible block count table C

 Compute $D(S, C)$ using DP

 Compute $D(T, C)$ using DP

If $D(S, C) + D(T, C) < \text{bestSolution}$ **then**

 bestSolution = $D(S, C) + D(T, C)$

- Complexity: dominated by the number of possible block count tables, which is $O(n^{|F|} |\Sigma|^{\max(F)})$

Down the theory rabbit hole: NP-hardness
for given F

Complexity status

F	Σ	Complexity
Fixed	Not fixed	NP-hard (unless F is trivial)
Fixed	Fixed	P (but horrible)
Not fixed	Not fixed	?
Not fixed	Fixed	?

Complexity status

F	Σ	Complexity
Fixed	Not fixed	NP-hard (unless F is trivial)
Fixed	Fixed	P (but horrible)
Not fixed	Not fixed	NP-hard
Not fixed	Fixed	NP-hard

Exact F-Strip Recovery, given F

Theorem

If **F is part of the input**, then the Exact F-Strip recovery problem is **NP-hard**, even if given two strings on alphabet of size 4.

Exact F-Strip Recovery, given F

Theorem

If **F is part of the input**, then the Exact F-Strip recovery problem is **NP-hard**, even if given two strings on alphabet of size 4.

Reduction from 3-Partition.

- Given a set of numbers S and an integer D , partition S into triples that all have the same sum D .

Exact F-Strip Recovery, given F

Theorem

If **F** is part of the input, then the Exact F-Strip recovery problem is **NP-hard**, even if given two strings on alphabet of size 4.

Reduction from 3-Partition.

- Given a set of numbers S and an integer D , partition S into triples that all have the same sum D .
- $S = \{a_1, \dots, a_{3m}\}, D \Rightarrow$ string A, B and allowed sizes F

Exact F-Strip Recovery, given F

Reduction from 3-Partition.

- Given a set of numbers S and an integer D , partition S into triples that all have the same sum D .
- $S = \{a_1, \dots, a_{3m}\}, D \Rightarrow$ string A, B and allowed sizes F

$$F = \{a_1, a_2, \dots, a_{3m}\}$$

$$A = \overbrace{\$ \dots \$}^{D+1} \overbrace{1 \dots 1}^{a_1} \overbrace{\$ \dots \$}^{D+1} \overbrace{1 \dots 1}^{a_2} \dots \overbrace{\$ \dots \$}^{D+1} \overbrace{1 \dots 1}^{a_{3m}} \overbrace{\# \dots \#}^{D+1} \overbrace{0 \dots 0}^{m(D+1)},$$

$$B = \overbrace{\# \dots \#}^{D+1} \overbrace{\$ \dots \$}^{3m(D+1)} \overbrace{(0 \dots 0 \ 1 \dots 1)}^{D+1 \ D}{}^m.$$

Conclusion

- Experiments? Nope.
 - It remains to explore the potential of strip recovery to find syntenies in practice.
- Exact algorithms? Probably not.
 - Need approximation, good heuristics, ILP, ...

THX