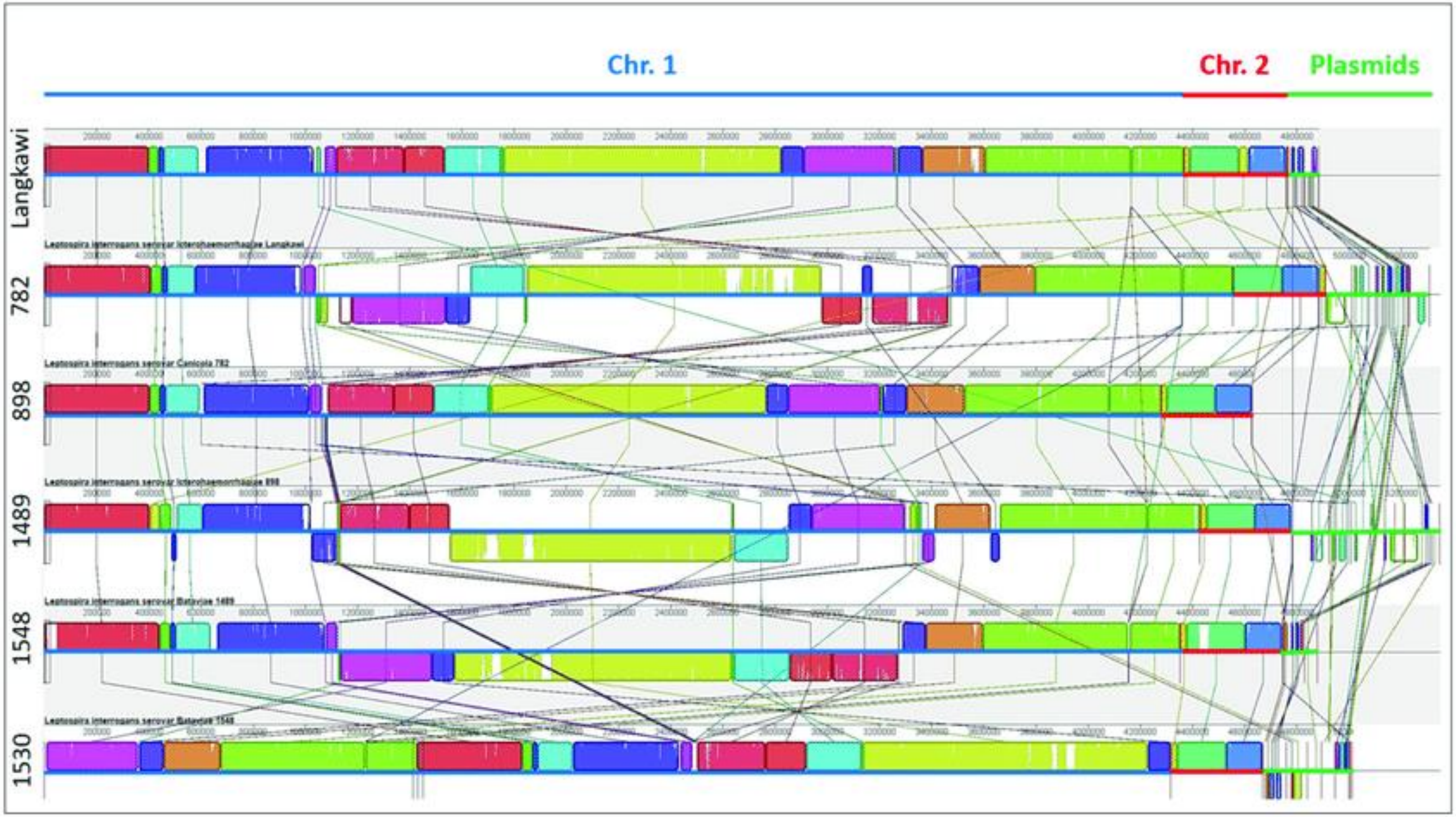
THE COMPLEXITY OF FINDING COMMON PARTITIONS OF GENOMES WITH PREDEFINED BLOCK SIZES

Manuel Lafond[1], Adiesha Liyanage[2], Binhai Zhu[2], Peng Zou[2]

[1] Université de Sherbrooke, Canada

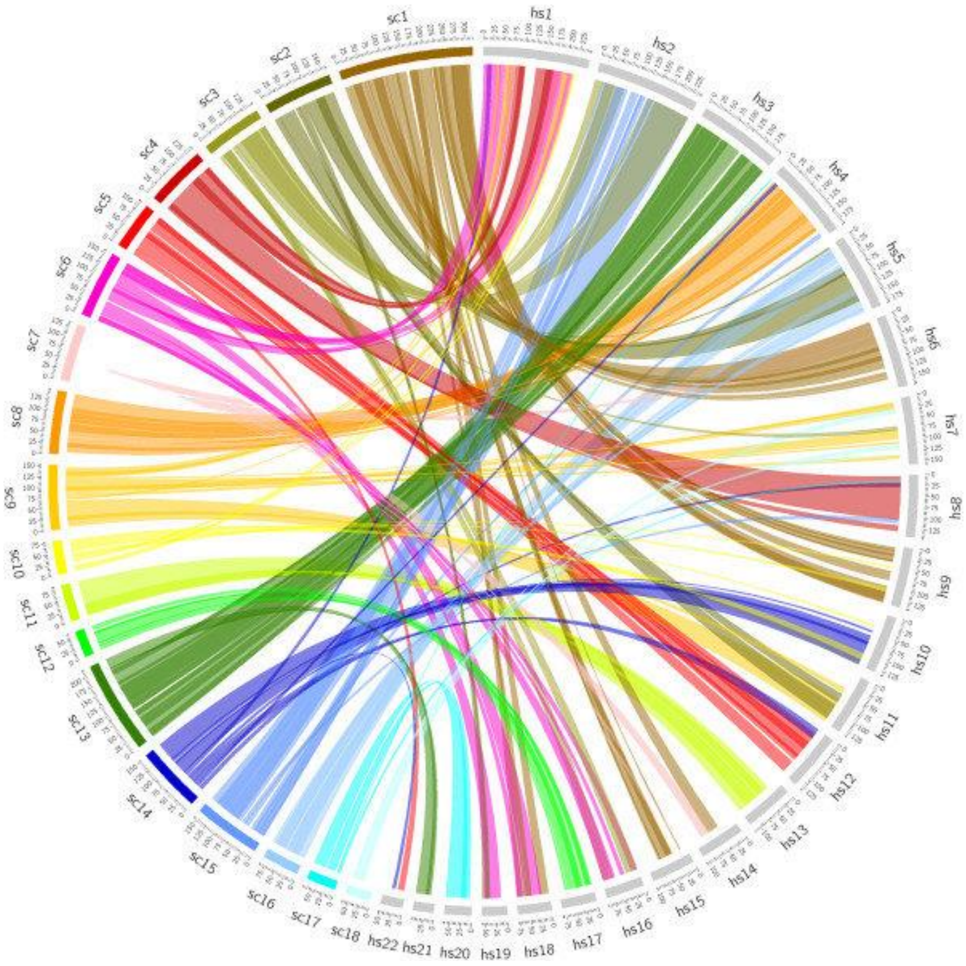
[2] Montana State University, USA

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



Leptospira strains syntenies
Image taken from Ramli et al.: [**https://doi.org/10.3390/pathogens10091198**](https://doi.org/10.3390/pathogens10091198)

<u>Synteny</u> : block of at least two genes that was conserved across multiple species.



Human vs pig synteny 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

$$\boxed{a\ a\ b\ a\ a\ b\ a}\boxed{a}\boxed{a\ b\ a\ b}$$

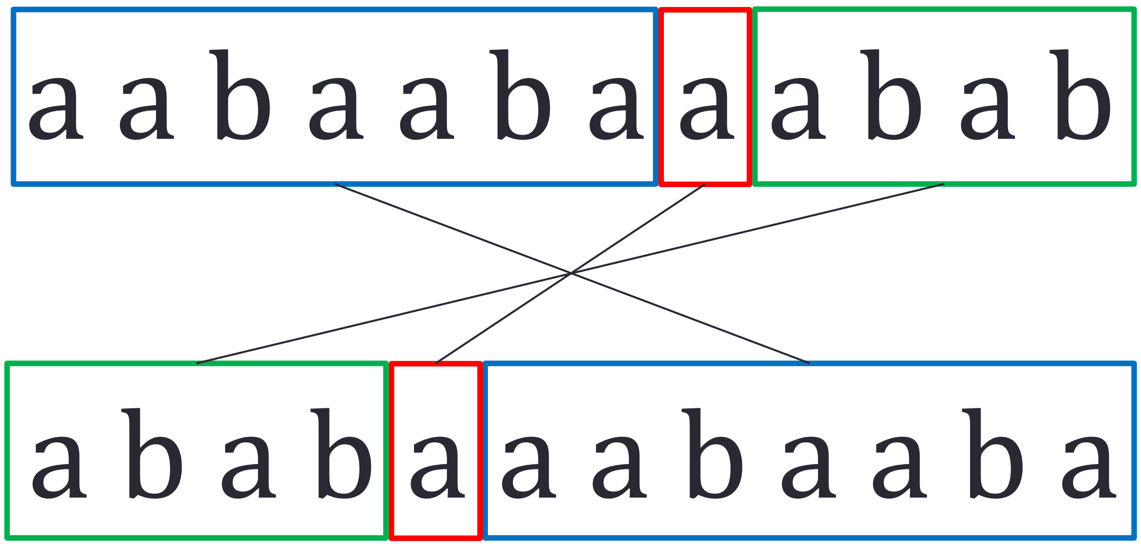
$$\boxed{a\ b\ a\ b}\boxed{a}\boxed{a\ a\ b\ a\ a\ b\ a}$$

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)

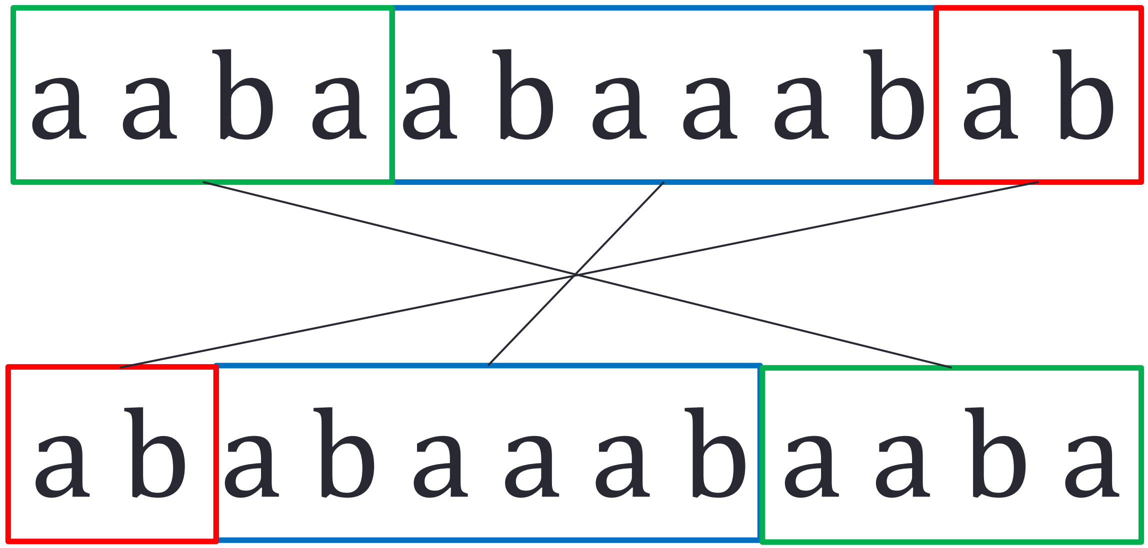
Given strings S, T, split them into two identical (multi)sets of blocks, while minimizing the number of blocks.

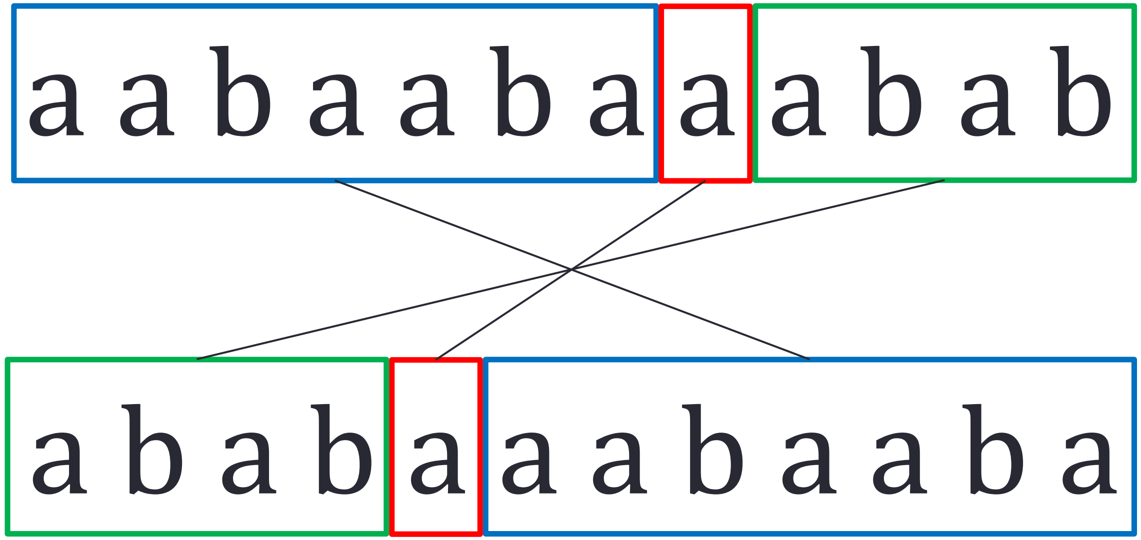
a a b a a b a a a b a b

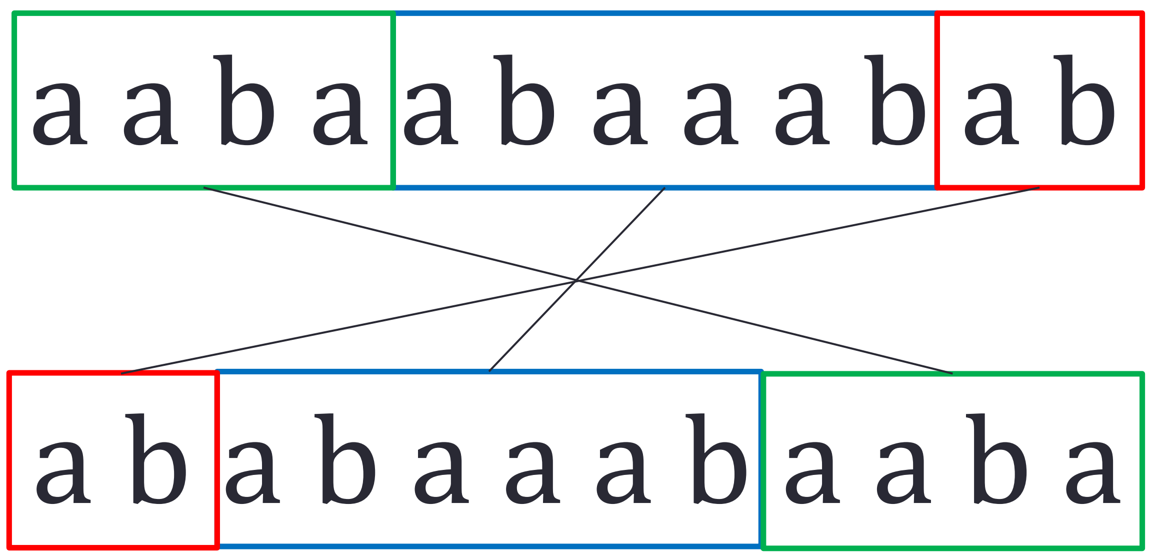
a b a b a a a b a a b a

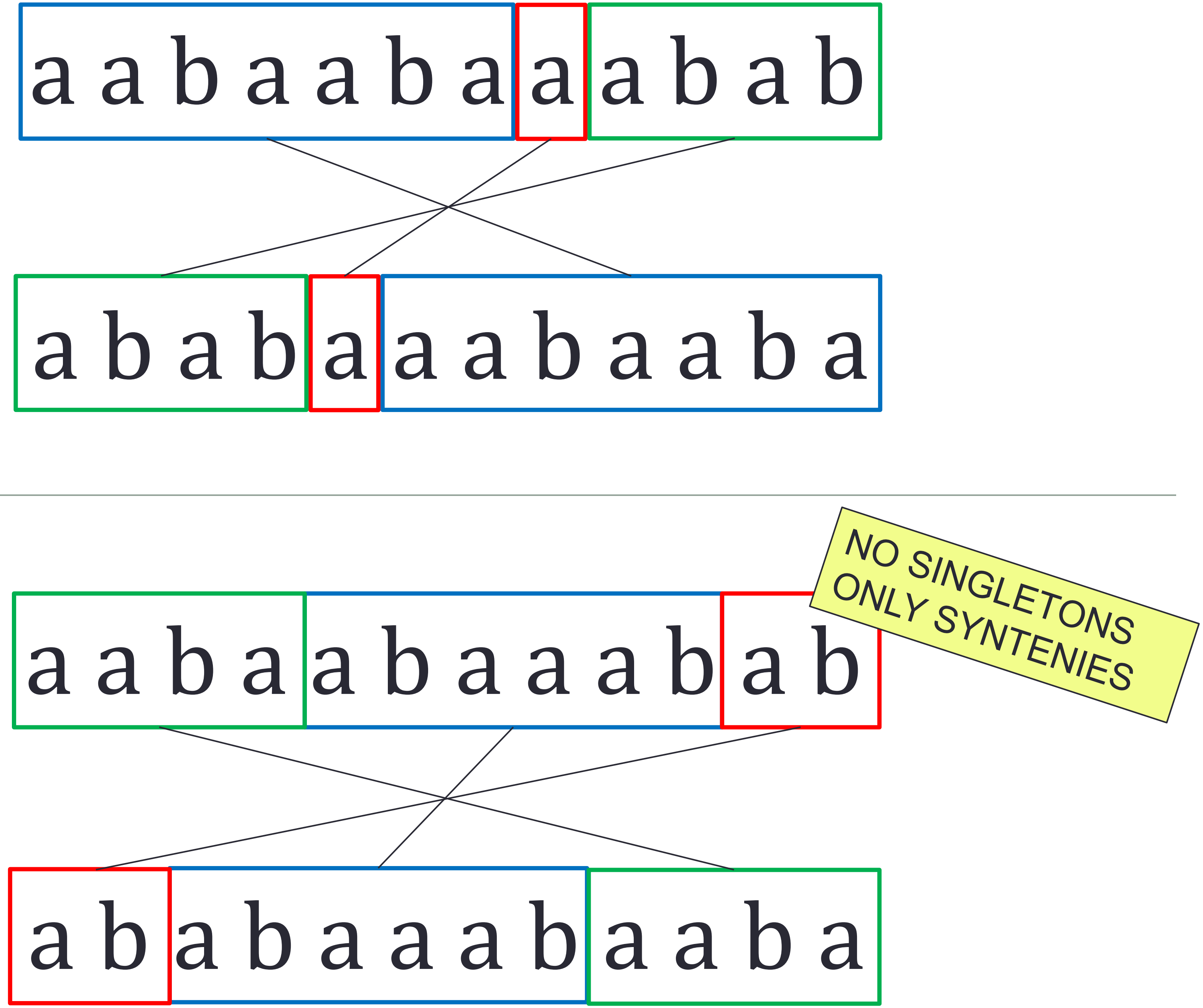
Usual formulation: Minimum Common String Partition (MCSP)

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
NO SINGLETONS
ONLY SYNTENIES
a b a b a a a b a a b a

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.

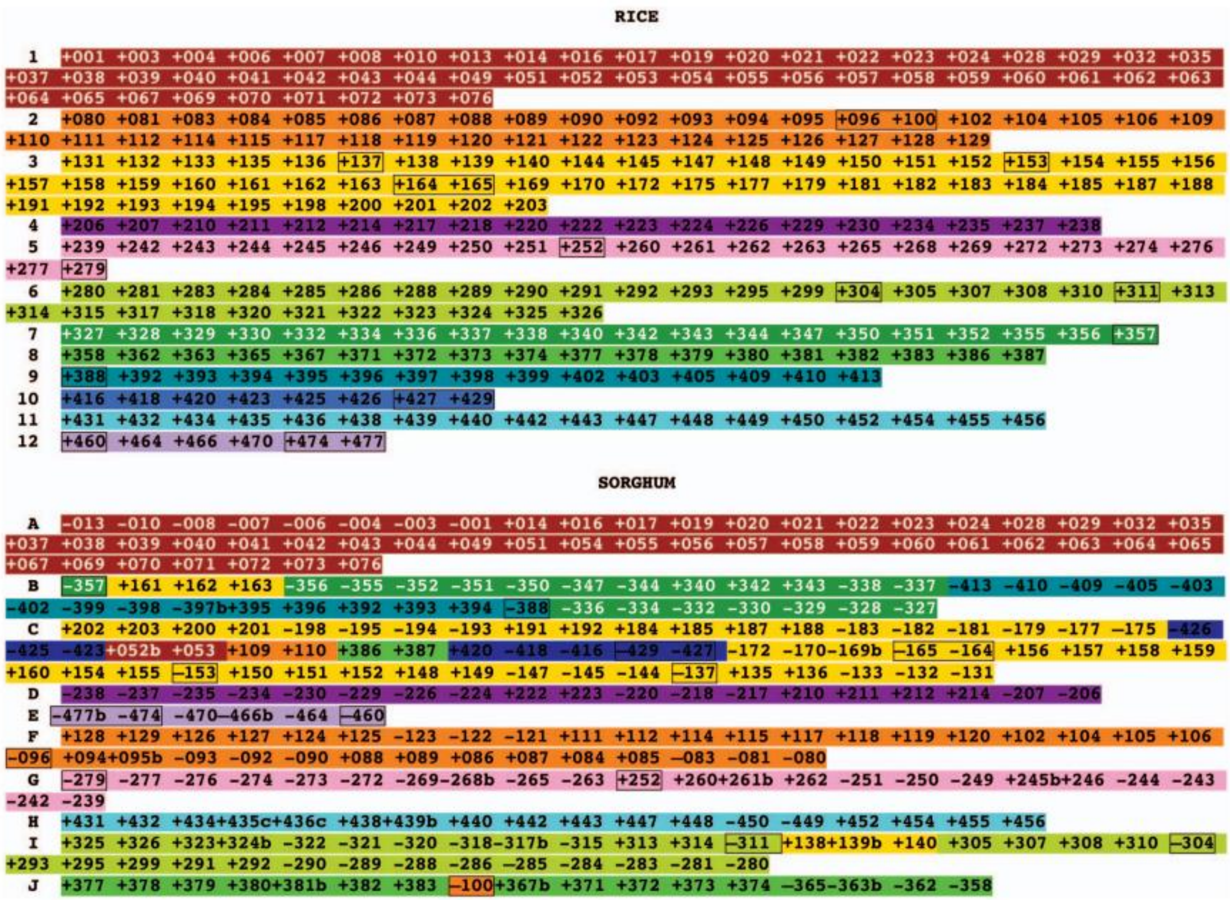
$b = 4$

$$\boxed{a\ a\ b\ a\ a\ b\ a}\ \cancel{a}\ \boxed{a\ b\ a\ b}$$

$$\boxed{a\ b\ a\ b}\ \cancel{b}\ \boxed{a\ a\ b\ a\ a\ b\ a}$$

Past results

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



Past results

- <u>**Min Strip Recovery**</u>
- 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

- <u>**Min Strip Recovery**</u>
- 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]

- <u>This paper: polynomial time on fixed alphabets</u>

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?

$\boldsymbol{b = 2}$ **:** 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?

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

$\boldsymbol{b = 3}$ **:** 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?

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

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

$\boldsymbol{b = 4}$ **:** 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}**

$$\boxed{a\ c}\boxed{a\ b}\boxed{a\ a}\boxed{c\ b}\boxed{a\ c}$$

$$\boxed{c\ b}\boxed{a\ a}\boxed{a\ c}\boxed{a\ c}\boxed{a\ b}$$

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.

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

for each variable x_i

Define X_i and X_i' as

$$X_i = P_i^1 \ P_i^2 \ \ldots \ P_i^r \ P_i^* \ \mathbb{L}_a \ \mathbb{M}_a \ \mathbb{R}_a \ Q_i \ \mathbb{L}_b \ \mathbb{M}_b \ \mathbb{R}_b \ S_i \ \mathbb{L}_c \ \mathbb{M}_c \ \mathbb{R}_c \ T_i^* \ T_i^1 \ T_i^2 \ \ldots \ T_i^r$$

$$X_i' = P_i^1 \ P_i^2 \ \ldots \ P_i^r \ P_i^* \ \mathbb{L}_a \ \mathbb{M}_a' \ \mathbb{R}_a \ Q_i \ \mathbb{L}_b \ \mathbb{M}_b' \ \mathbb{R}_b \ S_i \ \mathbb{L}_c \ \mathbb{M}_c' \ \mathbb{R}_c \ T_i^* \ T_i^1 \ T_i^2 \ \ldots \ T_i^r$$

We put $\mathcal{W}_1 = \{X_1, X_1', X_2, X_2', \ldots, X_n, X_n'\}$.

$$X_i = \underline{\underline{P_i^1 \ P_i^2 \ \ldots \ P_i^r \ P_i^*}} \ \underline{\underline{\mathbb{L}_a \ \mathbb{M}_a}} \ \underline{\mathbb{R}_a \ Q_i} \ \underline{\underline{\mathbb{L}_b \ \mathbb{M}_b}} \ \underline{\mathbb{R}_b \ S_i} \ \underline{\underline{\mathbb{L}_c \ \mathbb{M}_c}} \ \underline{\mathbb{R}_c \ T_i^*} \ \underline{T_i^1} \ \underline{T_i^2} \ \ldots \ \underline{T_i^r}$$

$$X_i' = \underline{P_i^1} \ \underline{P_i^2} \ \ldots \ \underline{P_i^r} \ \underline{P_i^* \ \mathbb{L}_a} \ \underline{\underline{\mathbb{M}_a' \ \mathbb{R}_a}} \ \underline{Q_i \ \mathbb{L}_b} \ \underline{\underline{\mathbb{M}_b' \ \mathbb{R}_b}} \ \underline{S_i \ \mathbb{L}_c} \ \underline{\underline{\mathbb{M}_c' \ \mathbb{R}_c}} \ \underline{\underline{T_i^* \ T_i^1 \ T_i^2 \ \ldots \ T_i^r}}$$

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

$$X_i = \underline{P_i^1} \ \underline{P_i^2} \ \ldots \ \underline{P_i^r} \ \underline{P_i^* \ \mathbb{L}_a} \ \underline{\underline{\mathbb{M}_a \ \mathbb{R}_a}} \ \underline{Q_i \ \mathbb{L}_b} \ \underline{\underline{\mathbb{M}_b \ \mathbb{R}_b}} \ \underline{S_i \ \mathbb{L}_c} \ \underline{\underline{\mathbb{M}_c \ \mathbb{R}_c}} \ \underline{\underline{T_i^* \ T_i^1 \ T_i^2 \ \ldots \ T_i^r}}$$

$$X_i' = \underline{\underline{P_i^1 \ P_i^2 \ \ldots \ P_i^r \ P_i^*}} \ \underline{\underline{\mathbb{L}_a \ \mathbb{M}_a'}} \ \underline{\mathbb{R}_a \ Q_i} \ \underline{\underline{\mathbb{L}_b \ \mathbb{M}_b'}} \ \underline{\mathbb{R}_b \ S_i} \ \underline{\underline{\mathbb{L}_c \ \mathbb{M}_c'}} \ \underline{\mathbb{R}_c \ T_i^*} \ \underline{T_i^1} \ \underline{T_i^2} \ \ldots \ \underline{T_i^r}$$

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

$$X_i = P_i^1 \; P_i^2 \; \ldots \; P_i^r \; P_i^* \; \mathbb{L}_a \; \mathbb{M}_a \; \mathbb{R}_a \; Q_i \; \mathbb{L}_b \; \mathbb{M}_b \; \mathbb{R}_b \; S_i \; \mathbb{L}_c \; \mathbb{M}_c \; \mathbb{R}_c \; T_i^* \; T_i^1 \; T_i^2 \; \ldots \; T_i^r$$
$$X_i' = P_i^1 \; P_i^2 \; \ldots \; P_i^r \; P_i^* \; \mathbb{L}_a \; \mathbb{M}_a' \; \mathbb{R}_a \; Q_i \; \mathbb{L}_b \; \mathbb{M}_b' \; \mathbb{R}_b \; S_i \; \mathbb{L}_c \; \mathbb{M}_c' \; \mathbb{R}_c \; T_i^* \; T_i^1 \; T_i^2 \; \ldots \; T_i^r$$

We put $\mathcal{W}_1 = \{X_1, X_1', X_2, X_2', \ldots, X_n, X_n'\}$.

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 $\mathcal{W}_2$:

- $P_i^1,\; P_i^2,\; \ldots,\; P_i^r$ (r strings of length ℓ)
- $P_i^1 P_i^2 \ldots P_i^r P_i^*$ (one string of length $r\ell + \ell - s = h\}$
- $P_i^* \mathbb{L}_a$ (one string of length $\ell - s + s = \ell$)
- $\mathbb{R}_a Q_i,\; Q_i \mathbb{L}_b,\; \mathbb{R}_b S_i,\; S_i \mathbb{L}_c$ (four strings of length $s + \ell - s = \ell$)
- $\mathbb{R}_c T_i^*$ (one string of length $s + \ell - s = \ell$)
- $T_i^* T_i^1 T_i^2 \ldots T_i^r$ (one string of length $r\ell + \ell - s = h\}$
- $T_i^1,\; T_i^2,\; \ldots,\; T_i^r$ (r strings of length ℓ)

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

- $\mathbb{L}_a \mathbb{M}_a,\; \mathbb{M}_a' \mathbb{R}_a$ (two strings of length h)
- $\mathbb{M}\mathbb{R},\; \mathbb{L}\mathbb{M}',\; \mathbb{M}\mathbb{R},\; \mathbb{L}\mathbb{M}'$ (four strings of length h)

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

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

Now build single strings S and T as follows:

$$S = A_1\ X_1\ Y_1\ Z_1\ A_2\ X_2\ Y_2\ Z_2\ \ldots\ A_n\ X_n\ Y_n\ Z_n$$
$$T = B_1\ Z_1\ Y_1\ X_1\ B_2\ Z_2\ Y_2\ X_2\ \ldots\ B_m\ Z_m\ Y_m\ X_m\ Z_{m+1}\ Y_{m+1}\ X_{m+1}\ \ldots\ Z_n\ Y_n\ X_n$$

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 $\Sigma = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$ and $\mathrm{F} = \{2, 3\}$, 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

<u>Theorem</u>

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

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

<u>Dynamic programming algorithm</u>

<u>Block</u> $=$ any string over Σ whose length is in F.

<u>Block count table</u> $=$ 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,
$D(S, C) =$ 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).

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

C:

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

$$\cancel{a}\,\boxed{b\ b\ a}\boxed{b\ a\ a}\,\cancel{a}\,\boxed{a\ b\ a}\boxed{b\ b\ a}$$

D(S, C) = 2

The algorithm

- On input strings S, T

bestSolution $= \infty$

For each possible block count table C

Compute $D(S, C)$ using DP

Compute $D(T, C)$ using DP

If $D(S, C) + D(T, C) <$ bestSolution **then**

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

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 $j < i$.

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 $j < i$.

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

$C - X$ means reduces the number of Xs by 1
$cost(S[j + 1..n], X)$ is the number of deletions to make the substring become X

The algorithm

- On input strings S, T

  bestSolution $= \infty$

  For each possible block count table C

      Compute $D(S, C)$ using DP

      Compute $D(T, C)$ using DP

      If $D(S, C) + D(T, C) <$ 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.
- $S = \{a_1, \ldots, a_{3m}\}, D \Rightarrow$ 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

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

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

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

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