The Complexity of Finding Tangles

Oksana Firman, Boris Klemz,
Alexander Wolff, Johannes Zink

Julius-Maximilians-Universität Würzburg,
Germany

Alexander Ravsky

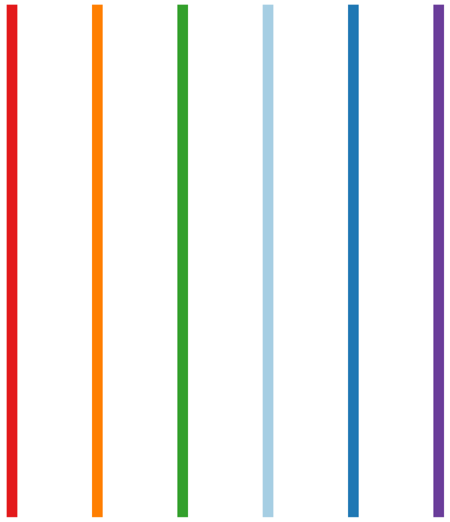
Pidstryhach Institute for Applied Problems
of Mechanics and Mathematics,
National Academy of Sciences of Ukraine,
Lviv, Ukraine

Philipp Kindermann

Universität Trier,
Germany

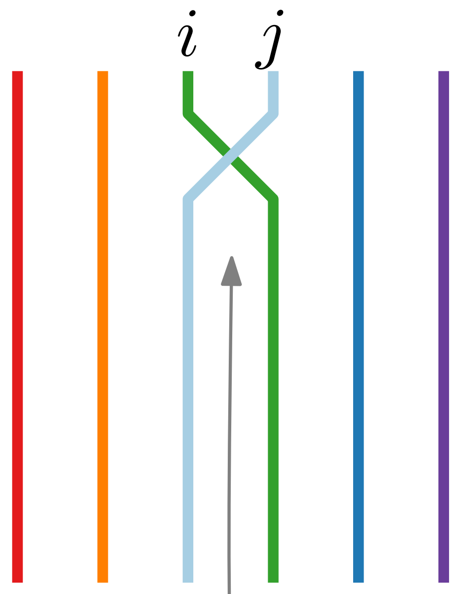
Introduction

Given an ordered set of n y-monotone wires



Introduction

Given an ordered set of n y-monotone wires

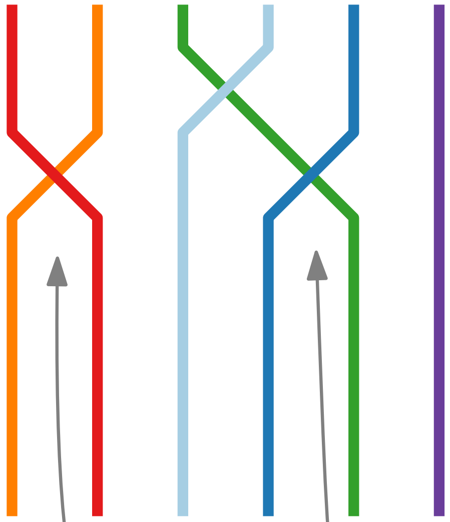


$1 \leq i < j \leq n$

swap ij

Introduction

Given an ordered set
of n y-monotone wires



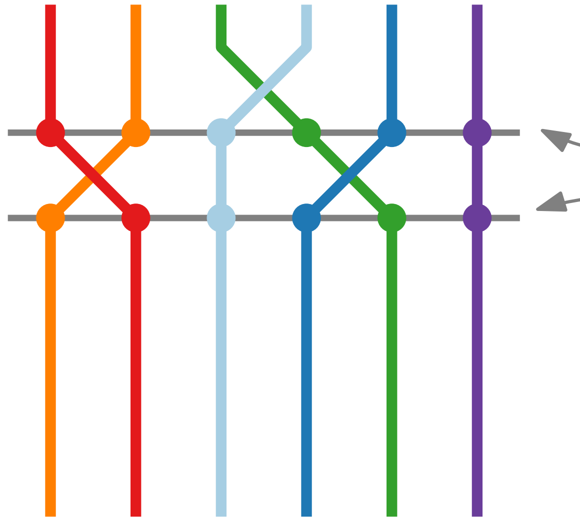
$1 \leq i < j \leq n$

swap ij

swaps ij and kl are *disjoint*
if $\{i, j\} \cap \{k, l\} = \emptyset$

Introduction

Given an ordered set
of n y-monotone wires



$1 \leq i < j \leq n$

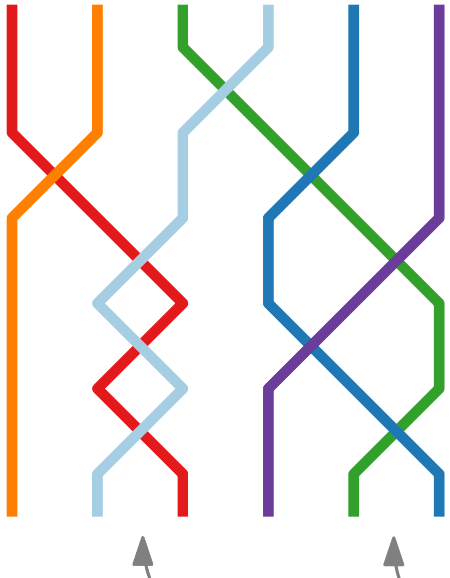
swap ij

swaps ij and kl are *disjoint*
if $\{i, j\} \cap \{k, l\} = \emptyset$

adjacent permutations

Introduction

Given an ordered set
of n y-monotone wires



$1 \leq i < j \leq n$

swap ij

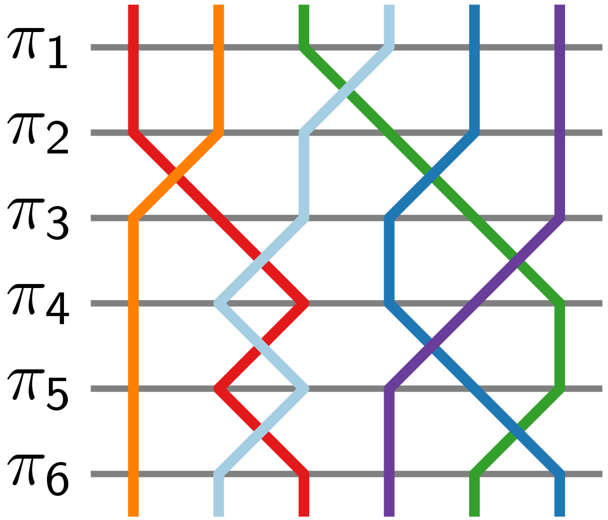
swaps ij and kl are *disjoint*
if $\{i, j\} \cap \{k, l\} = \emptyset$

adjacent permutations

multiple swaps

Introduction

Given an ordered set of n y-monotone wires



$1 \leq i < j \leq n$

swap ij

swaps ij and kl are *disjoint* if $\{i, j\} \cap \{k, l\} = \emptyset$

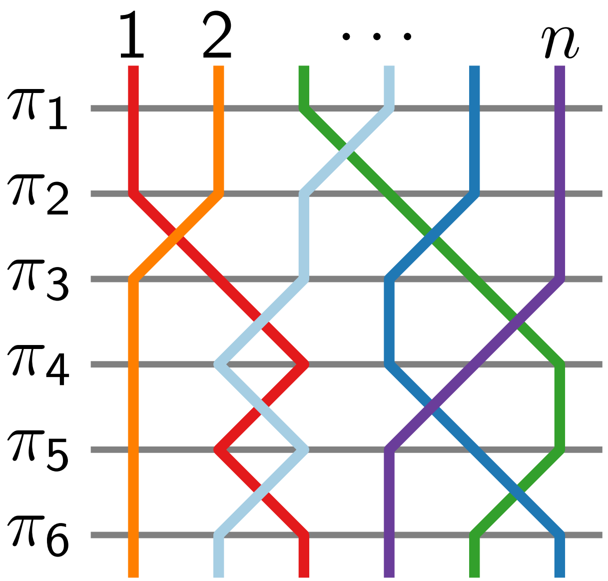
adjacent permutations

multiple swaps

tangle T of *height* $h(T)$

Introduction

Given an ordered set
of n y-monotone wires



$1 \leq i < j \leq n$

swap ij

swaps ij and kl are *disjoint*
if $\{i, j\} \cap \{k, l\} = \emptyset$

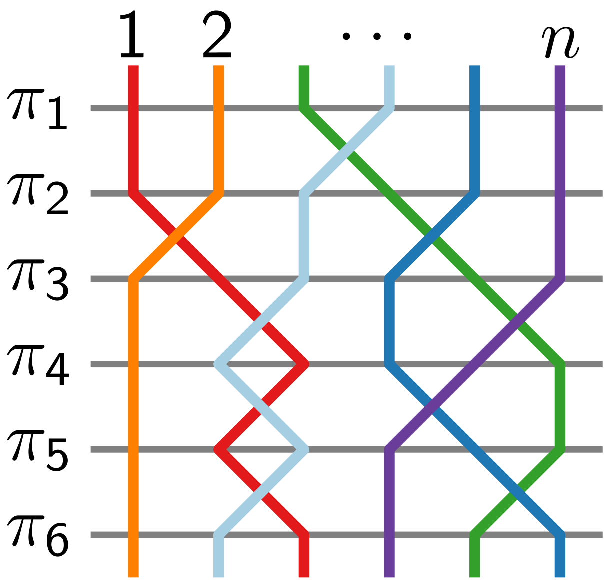
adjacent permutations

multiple swaps

tangle T of *height* $h(T)$

Introduction

Given an ordered set of n y-monotone wires



$1 \leq i < j \leq n$

swap ij

swaps ij and kl are *disjoint* if $\{i, j\} \cap \{k, l\} = \emptyset$

adjacent permutations

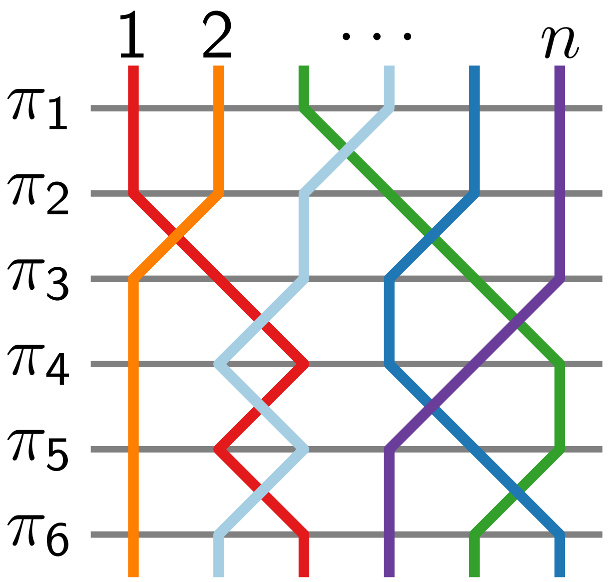
multiple swaps

tangle T of *height* $h(T)$

…and given a list of swaps L

Introduction

Given an ordered set of n y-monotone wires



$1 \leq i < j \leq n$

swap ij

swaps ij and kl are *disjoint* if $\{i, j\} \cap \{k, l\} = \emptyset$

adjacent permutations

multiple swaps

tangle T of *height* $h(T)$

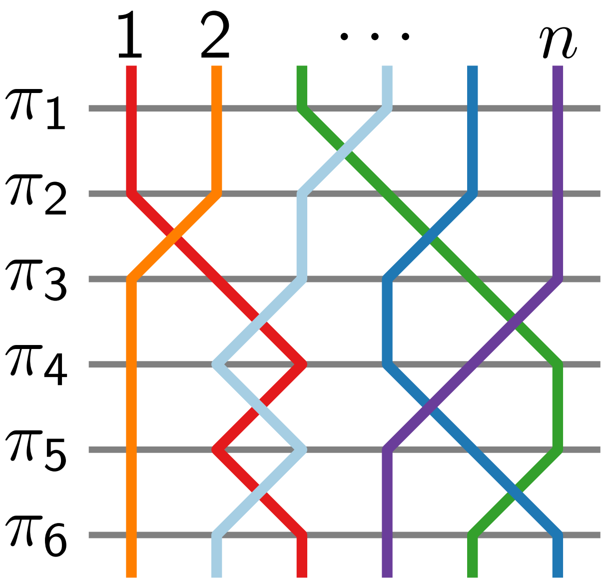
...and given a list of swaps L

as a multiset (ℓ_{ij})

1
3
1
2
1
1

Introduction

Given an ordered set of n y-monotone wires



$1 \leq i < j \leq n$

swap ij

swaps ij and kl are *disjoint* if $\{i, j\} \cap \{k, l\} = \emptyset$

adjacent permutations

multiple swaps

tangle T of *height* $h(T)$

…and given a list of swaps L

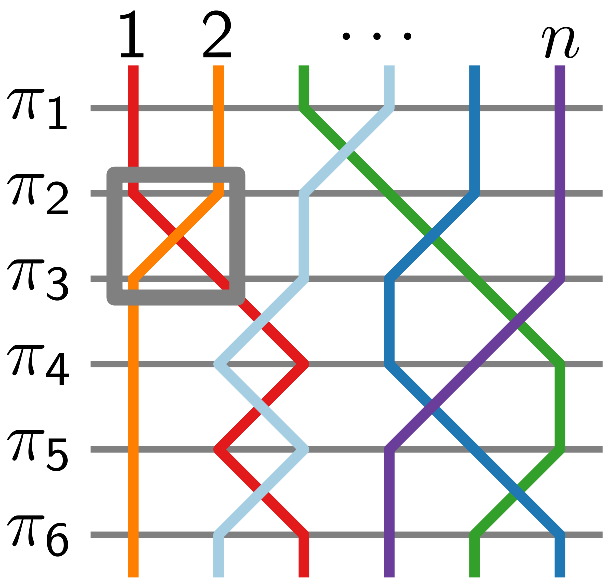
as a multiset (ℓ_{ij})

1
3
1
2
1
1

Tangle T *realizes* list L.

Introduction

Given an ordered set of n y-monotone wires



$1 \leq i < j \leq n$

swap ij

swaps ij and kl are *disjoint* if $\{i, j\} \cap \{k, l\} = \emptyset$

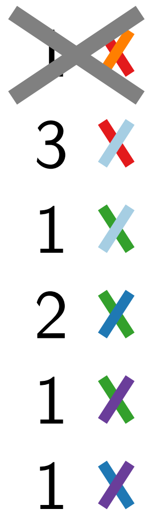
adjacent permutations

multiple swaps

tangle T of *height* $h(T)$

. . . and given a list of swaps L

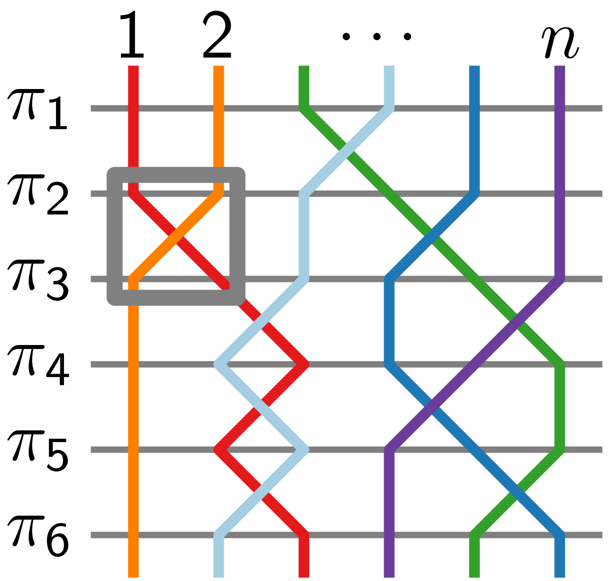
as a multiset (ℓ_{ij})



Tangle T *realizes* list L.

Introduction

Given an ordered set of n y-monotone wires



$1 \leq i < j \leq n$

swap ij

swaps ij and kl are *disjoint* if $\{i, j\} \cap \{k, l\} = \emptyset$

adjacent permutations

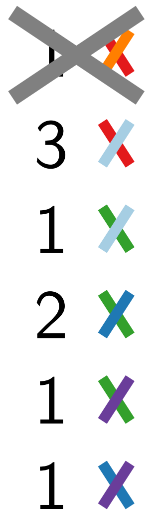
multiple swaps

tangle T *of height* $h(T)$

Tangle T *realizes* list L.

... and given a list of swaps L

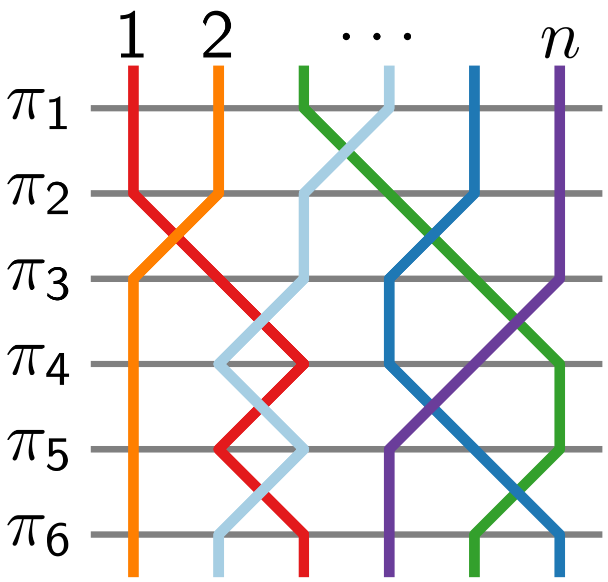
as a multiset (ℓ_{ij})



not *feasible*

Introduction

Given an ordered set of n y-monotone wires



$1 \leq i < j \leq n$

swap ij

swaps ij and kl are *disjoint* if $\{i,j\} \cap \{k,l\} = \emptyset$

adjacent permutations

multiple swaps

tangle T of *height* $h(T)$

…and given a list of swaps L

as a multiset (ℓ_{ij})

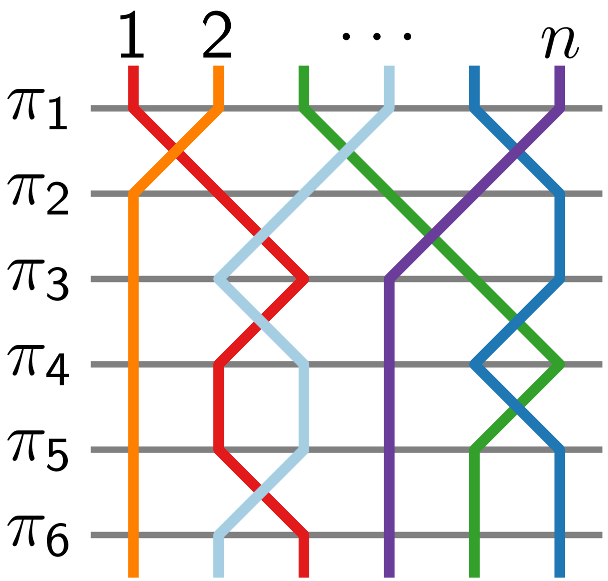
1
3
1
2
1
1

Tangle T *realizes* list L.

A list L of swaps is *feasible* if there exists a tangle that realizes L.
There may be multiple tangles realizing the same list of swaps.

Introduction

Given an ordered set of n y-monotone wires



$1 \leq i < j \leq n$

swap ij

swaps ij and kl are *disjoint* if $\{i,j\} \cap \{k,l\} = \emptyset$

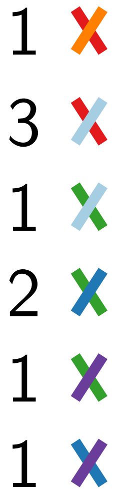
adjacent permutations

multiple swaps

tangle T of *height* $h(T)$

…and given a list of swaps L

as a multiset (ℓ_{ij})

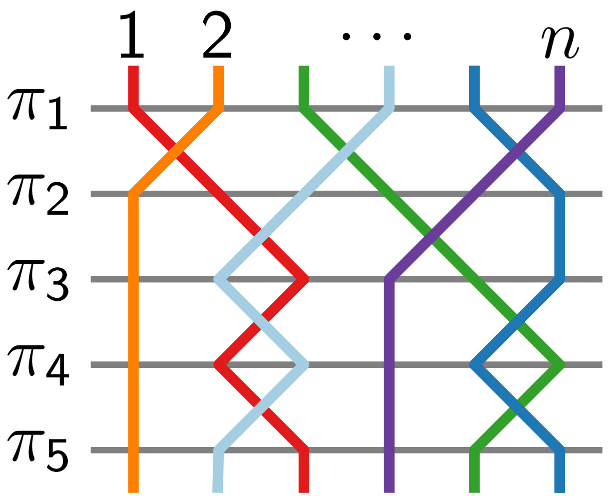


Tangle T *realizes* list L.

A list L of swaps is *feasible* if there exists a tangle that realizes L.
There may be multiple tangles realizing the same list of swaps.

Introduction

Given an ordered set of n y-monotone wires



$1 \leq i < j \leq n$

swap ij

swaps ij and kl are *disjoint* if $\{i,j\} \cap \{k,l\} = \emptyset$

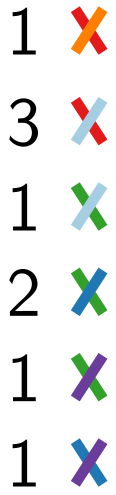
adjacent permutations

multiple swaps

tangle T of *height* $h(T)$

...and given a list of swaps L

as a multiset (ℓ_{ij})

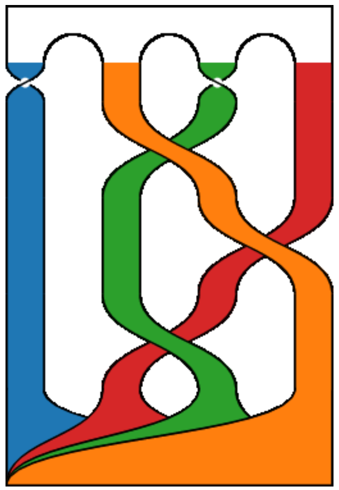


Tangle T *realizes* list L.

A list L of swaps is *feasible* if there exists a tangle that realizes L.
There may be multiple tangles realizing the same list of swaps.

Related Work

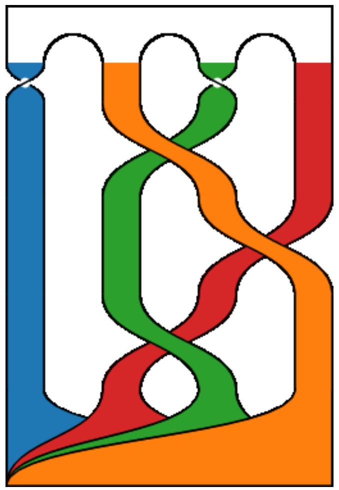
- Olszewski, Meder, Kieffer, Bleuse, Rosalie, Danoy, and Bouvry. [GD 2018]



Related Work

- Olszewski, Meder, Kieffer, Bleuse, Rosalie, Danoy, and Bouvry. [GD 2018]

Algorithm for optimal-height tangles, exponential in $|L|$



For a list of swaps $L = (l_{ij})$ with n wires:

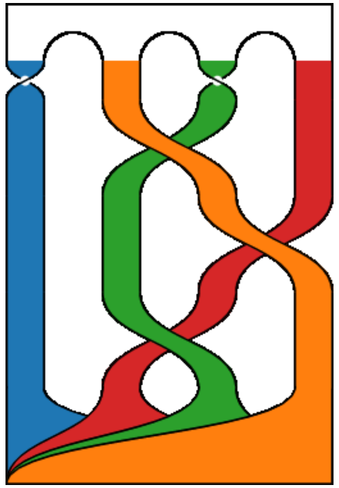
$|L| = \sum l_{ij}$

Related Work

- Olszewski, Meder, Kieffer, Bleuse, Rosalie, Danoy, and Bouvry. [GD 2018]

Algorithm for optimal-height tangles, exponential in $|L|$

Complexity ?



For a list of swaps $L = (l_{ij})$ with n wires:

$|L| = \sum l_{ij}$

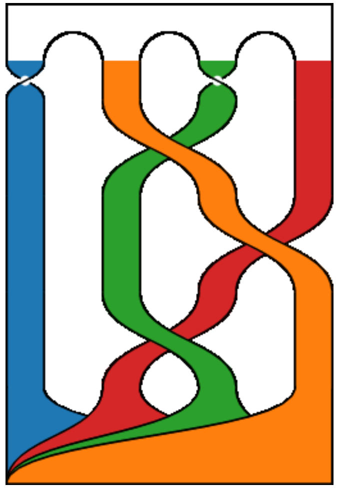
Related Work

- Olszewski, Meder, Kieffer, Bleuse, Rosalie, Danoy, and Bouvry. [GD 2018]

Algorithm for optimal-height tangles, exponential in $\lvert L \rvert$	*Complexity* ?

- F., Kindermann, Ravsky, Wolff, and Zink. [GD 2019]

Algorithm for optimal-height tangles, exponential in n^2	Finding optimal-height tangles is NP-hard



For a list of swaps $L = (l_{ij})$ with n wires:

$|L| = \sum l_{ij}$

Related Work

- Olszewski, Meder, Kieffer, Bleuse, Rosalie, Danoy, and Bouvry. [GD 2018]

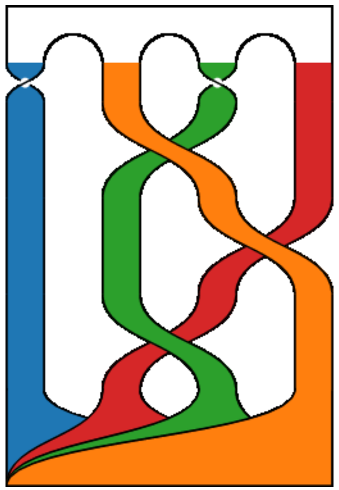
Algorithm for optimal-height tangles, exponential in $\lvert L \rvert$	*Complexity* **?**

- F., Kindermann, Ravsky, Wolff, and Zink. [GD 2019]

Algorithm for optimal-height tangles, exponential in n^2	Finding optimal-height tangles is NP-hard

- Sado and Igarashi. [TCS 1987]

Efficient algorithm for *simple* lists; height $\leq$ OPT $+1$



For a list of swaps $L = (l_{ij})$ with n wires:

$\lvert L \rvert = \sum l_{ij}$

L is *simple* if l_{ij} is 0 or 1

Related Work

- Olszewski, Meder, Kieffer, Bleuse, Rosalie, Danoy, and Bouvry. [GD 2018]

Algorithm for optimal-height tangles, exponential in $\lvert L\rvert$	*Complexity* **?**

- F., Kindermann, Ravsky, Wolff, and Zink. [GD 2019]

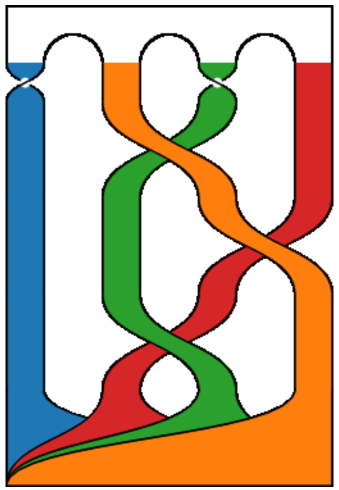
Algorithm for optimal-height tangles, exponential in n^2	Finding optimal-height tangles is NP-hard

- Sado and Igarashi. [TCS 1987]

  Efficient algorithm for *simple* lists; height $\leq$ OPT $+1$

- Bereg, Holroyd, Nachmanson, and Pupyrev. [GD 2013]

  Algorithms for minimizing the number of *bends*



For a list of swaps $L = (l_{ij})$ with n wires:

$|L| = \sum l_{ij}$

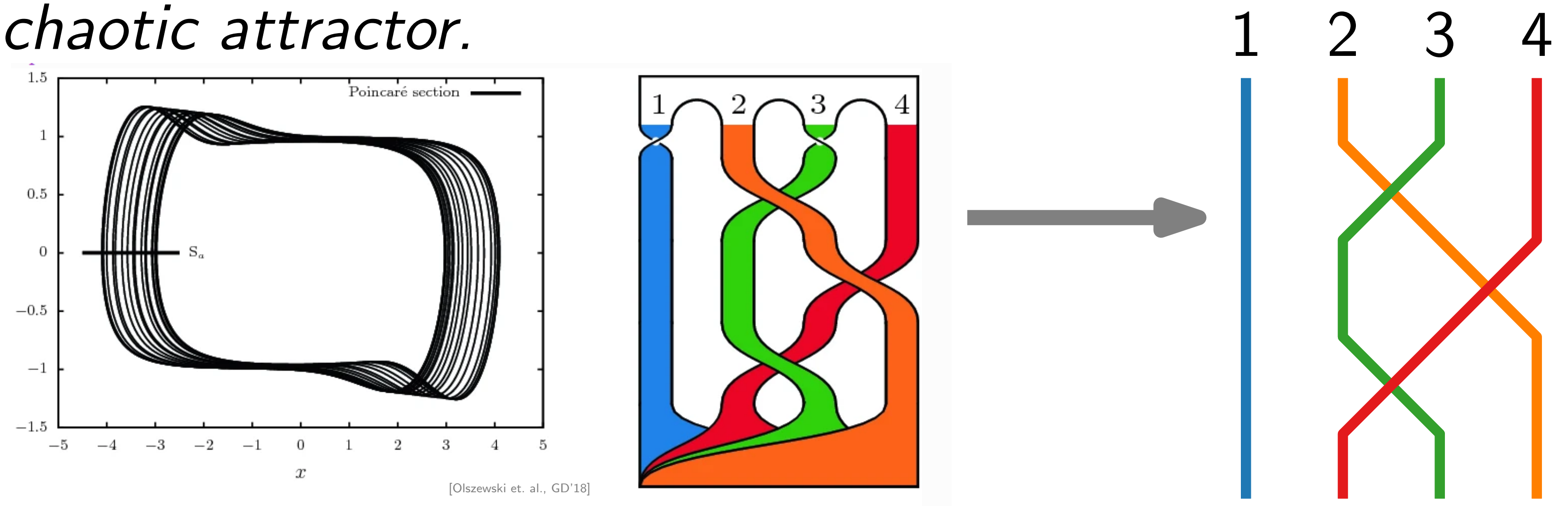
L is *simple* if l_{ij} is 0 or 1

Related Work and Motivation

- Olszewski, Meder, Kieffer, Bleuse, Rosalie, Danoy, and Bouvry.
  Visualizing the template of a chaotic attractor.
  [GD 2018]

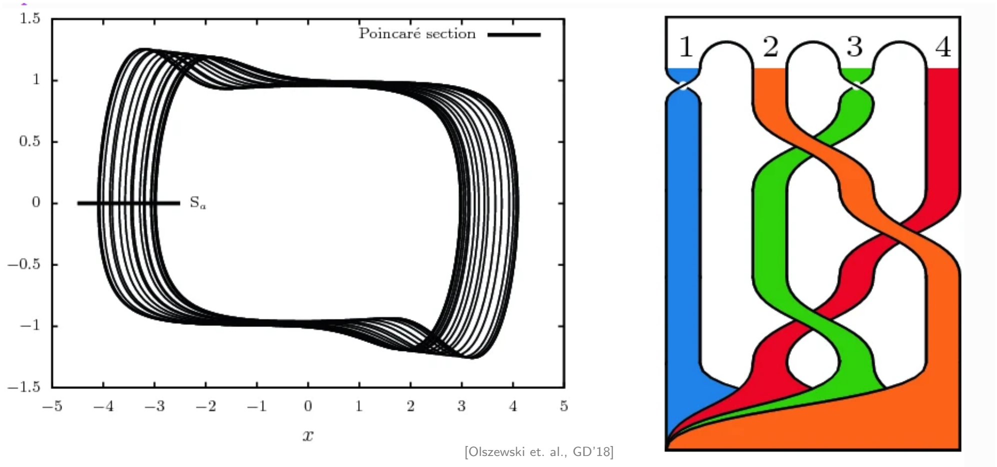
Related Work and Motivation

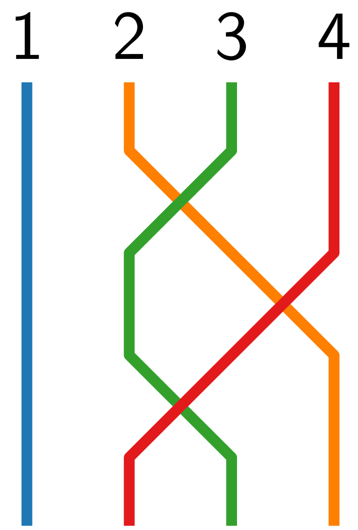
- Olszewski, Meder, Kieffer, Bleuse, Rosalie, Danoy, and Bouvry. *Visualizing the template of a chaotic attractor.* [GD 2018]



Related Work and Motivation

- Olszewski, Meder, Kieffer, Bleuse, Rosalie, Danoy, and Bouvry.
  Visualizing the template of a chaotic attractor.
  [GD 2018]

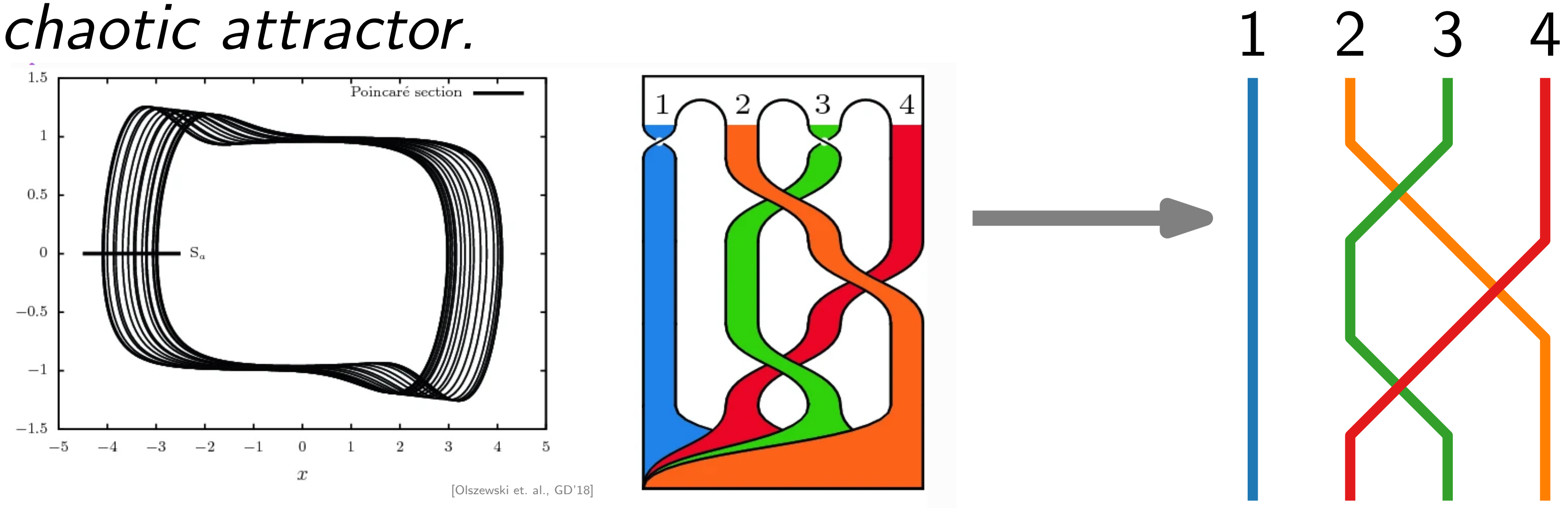




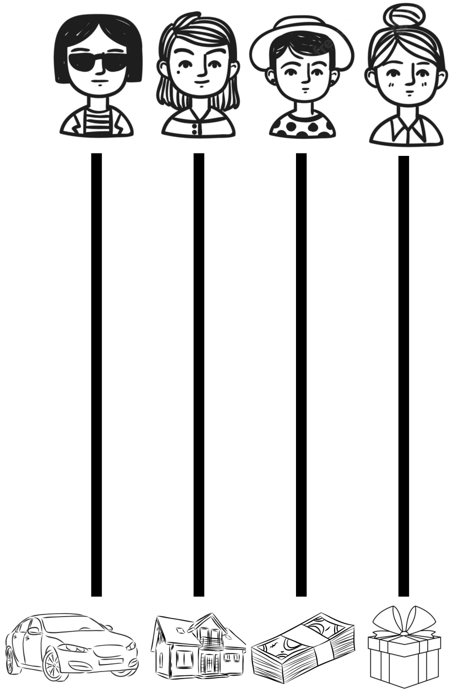
- Yamanaka, Horiyama, Uno, and Wasa.
  Ladder-lottery realization.
  [CCCG 2018]

Related Work and Motivation

- Olszewski, Meder, Kieffer, Bleuse, Rosalie, Danoy, and Bouvry.
  Visualizing the template of a chaotic attractor.
  [GD 2018]

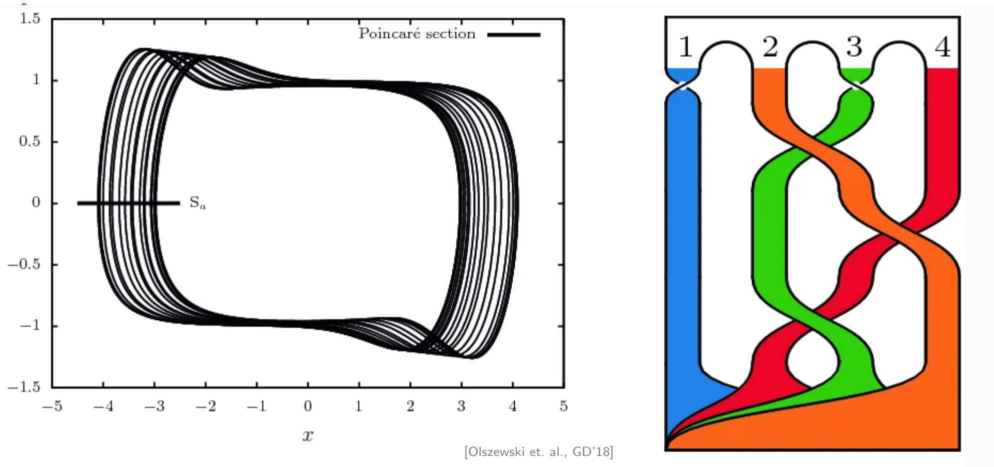


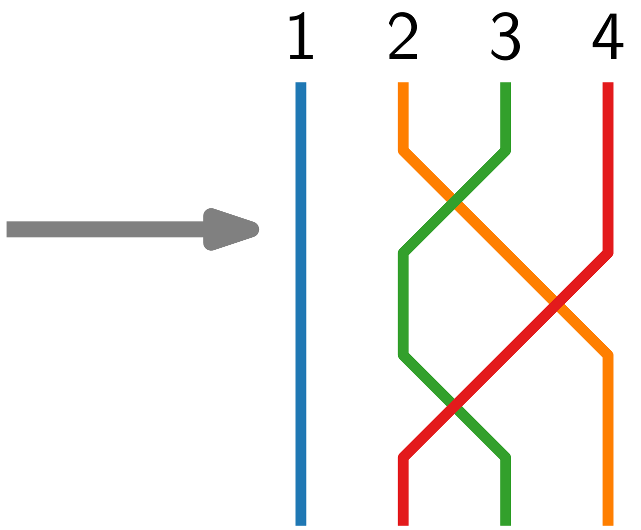
- Yamanaka, Horiyama, Uno, and Wasa.
  Ladder-lottery realization.
  [CCCG 2018]



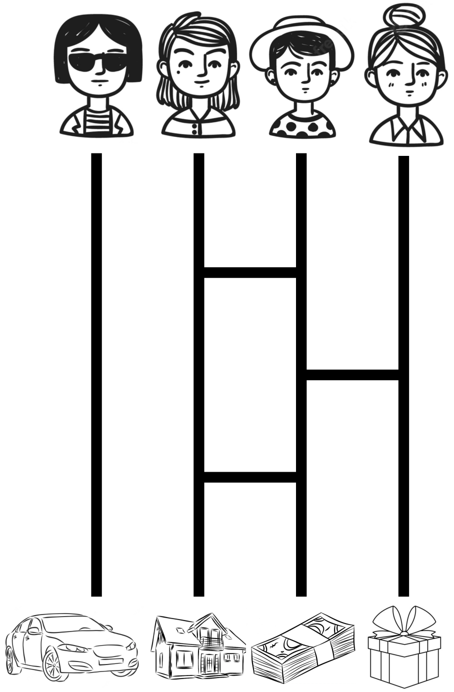
Related Work and Motivation

- Olszewski, Meder, Kieffer, Bleuse, Rosalie, Danoy, and Bouvry.
  Visualizing the template of a chaotic attractor.
  [GD 2018]



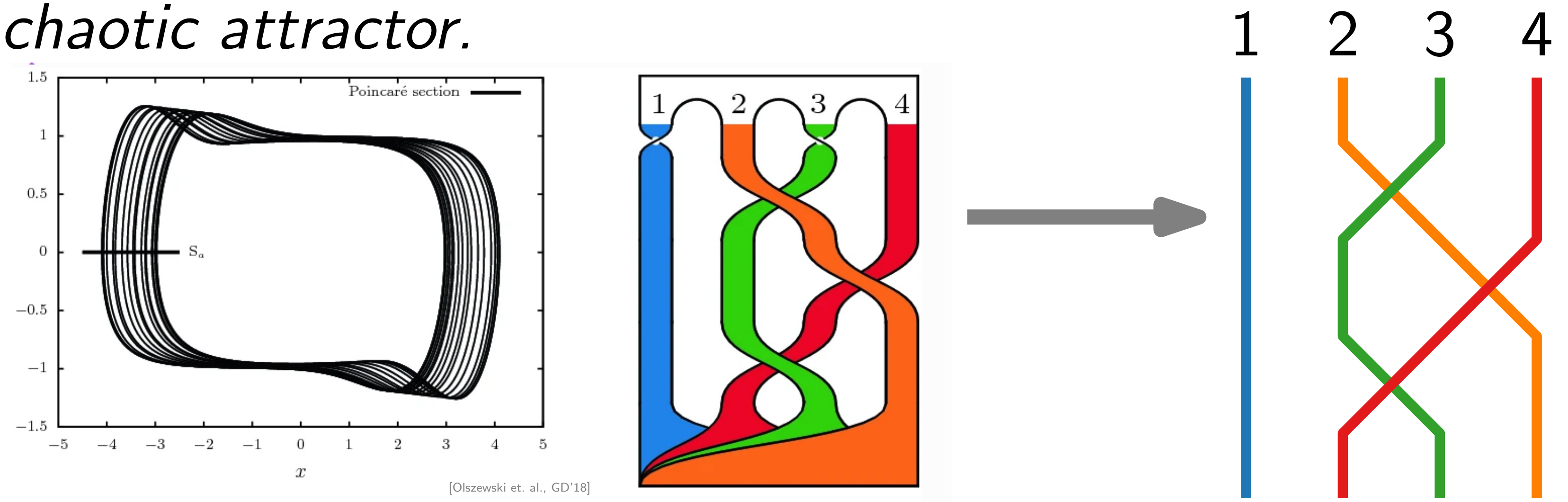


- Yamanaka, Horiyama, Uno, and Wasa.
  Ladder-lottery realization.
  [CCCG 2018]

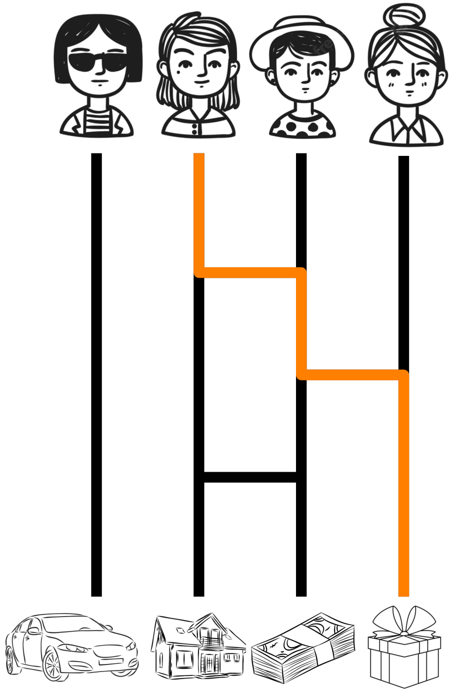


Related Work and Motivation

- Olszewski, Meder, Kieffer, Bleuse, Rosalie, Danoy, and Bouvry.
  Visualizing the template of a chaotic attractor.
  [GD 2018]

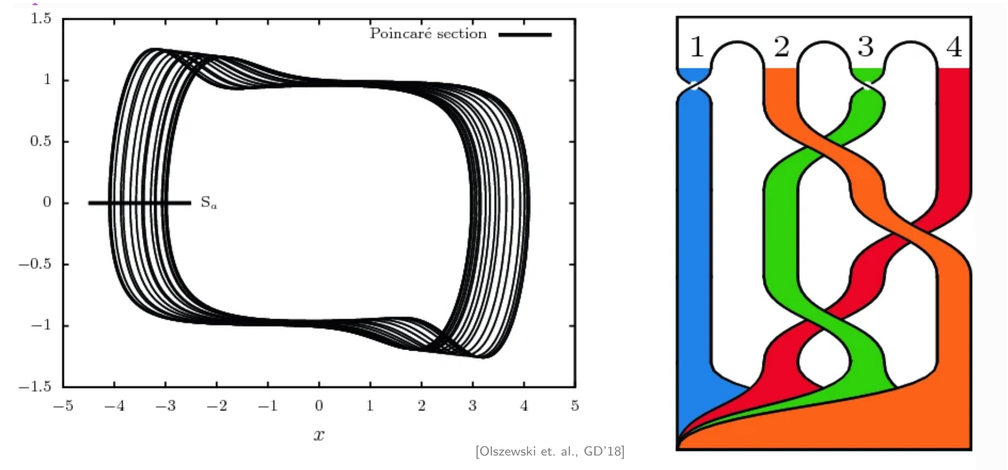


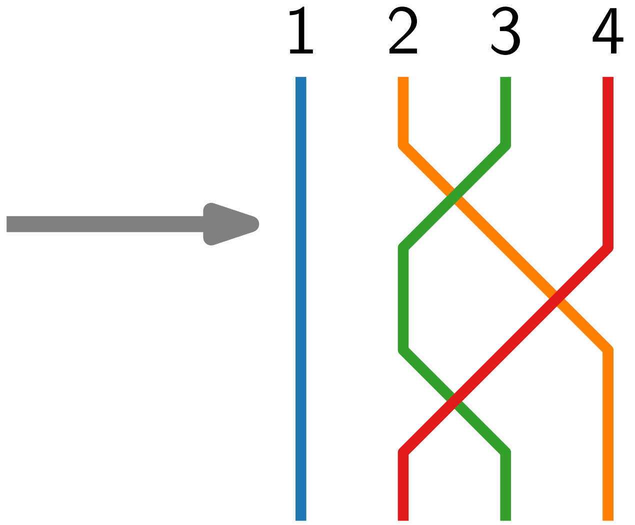
- Yamanaka, Horiyama, Uno, and Wasa.
  Ladder-lottery realization.
  [CCCG 2018]



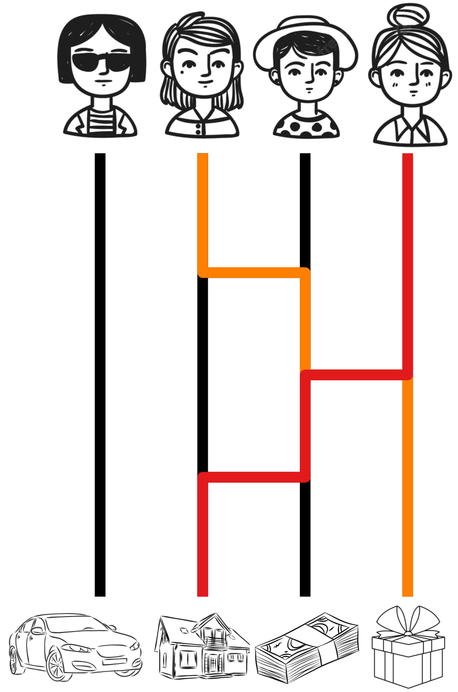
Related Work and Motivation

- Olszewski, Meder, Kieffer, Bleuse, Rosalie, Danoy, and Bouvry.
  Visualizing the template of a chaotic attractor.
  [GD 2018]



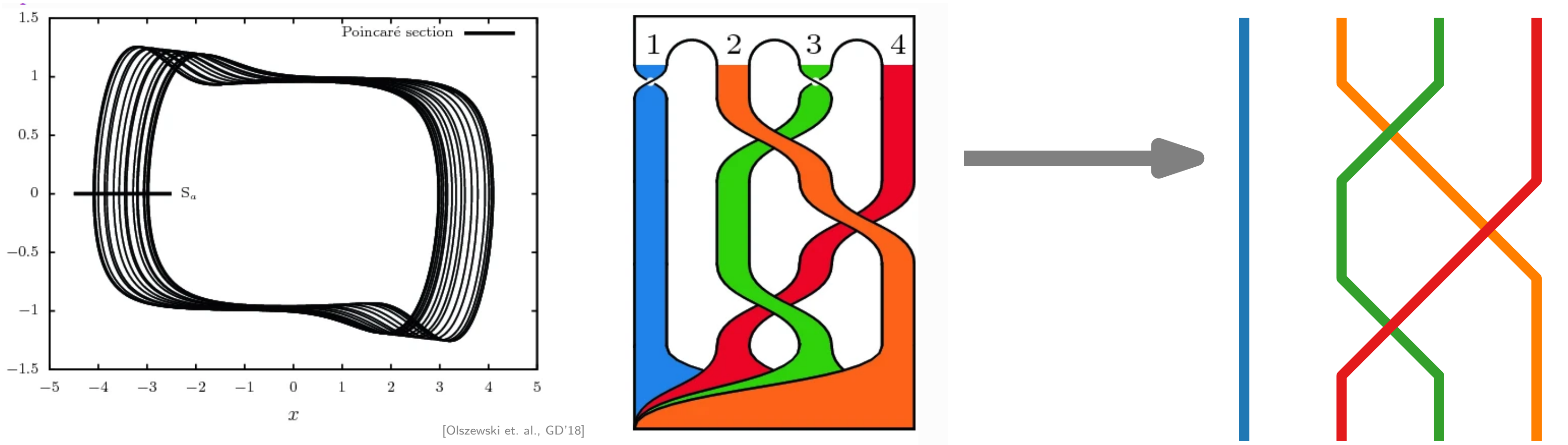


- Yamanaka, Horiyama, Uno, and Wasa.
  Ladder-lottery realization.
  [CCCG 2018]

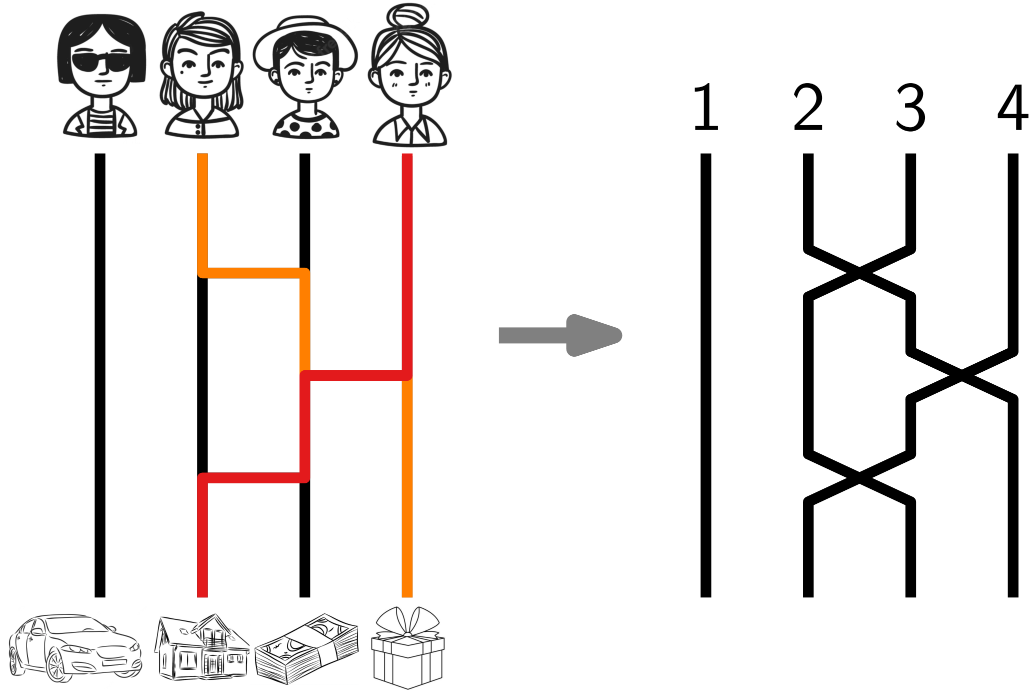


Related Work and Motivation

- Olszewski, Meder, Kieffer, Bleuse, Rosalie, Danoy, and Bouvry.
  Visualizing the template of a chaotic attractor.
  [GD 2018]

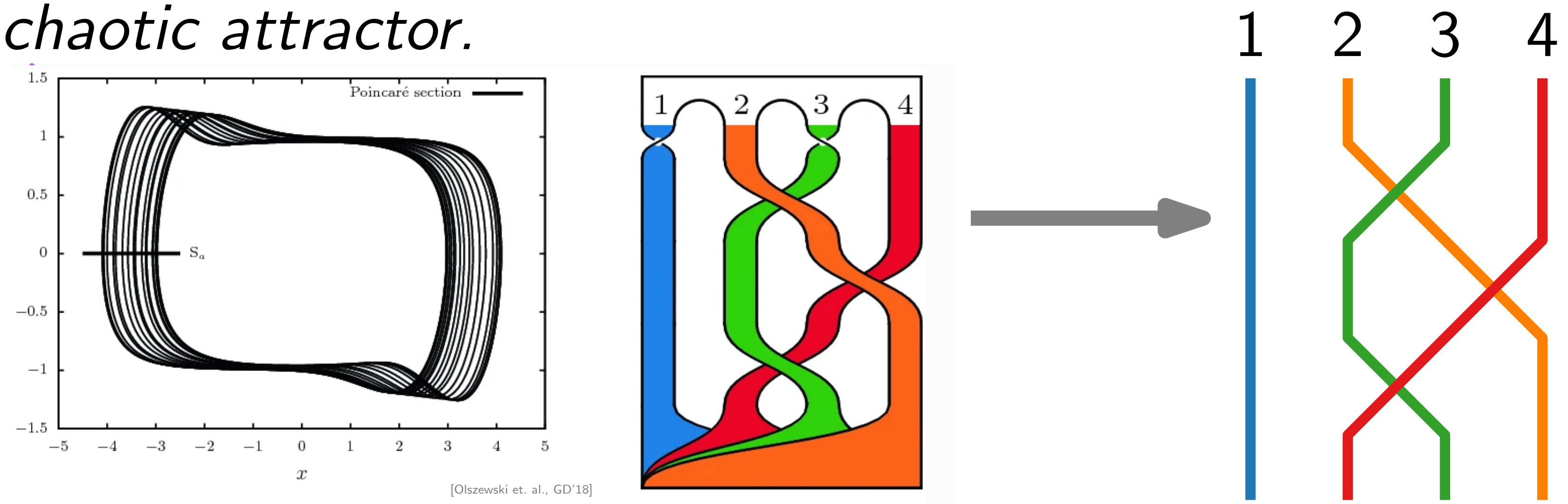


- Yamanaka, Horiyama, Uno, and Wasa.
  Ladder-lottery realization.
  [CCCG 2018]

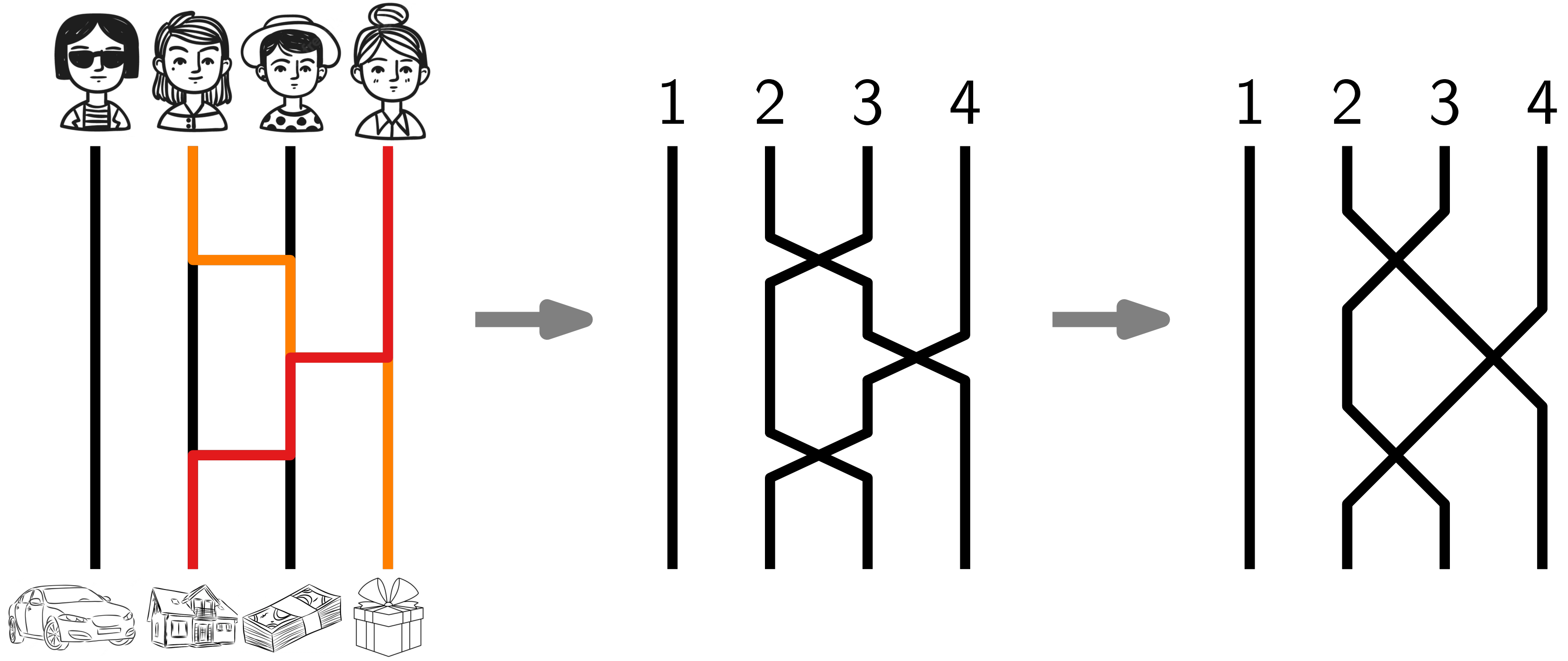


Related Work and Motivation

- Olszewski, Meder, Kieffer, Bleuse, Rosalie, Danoy, and Bouvry.
  Visualizing the template of a chaotic attractor.
  [GD 2018]

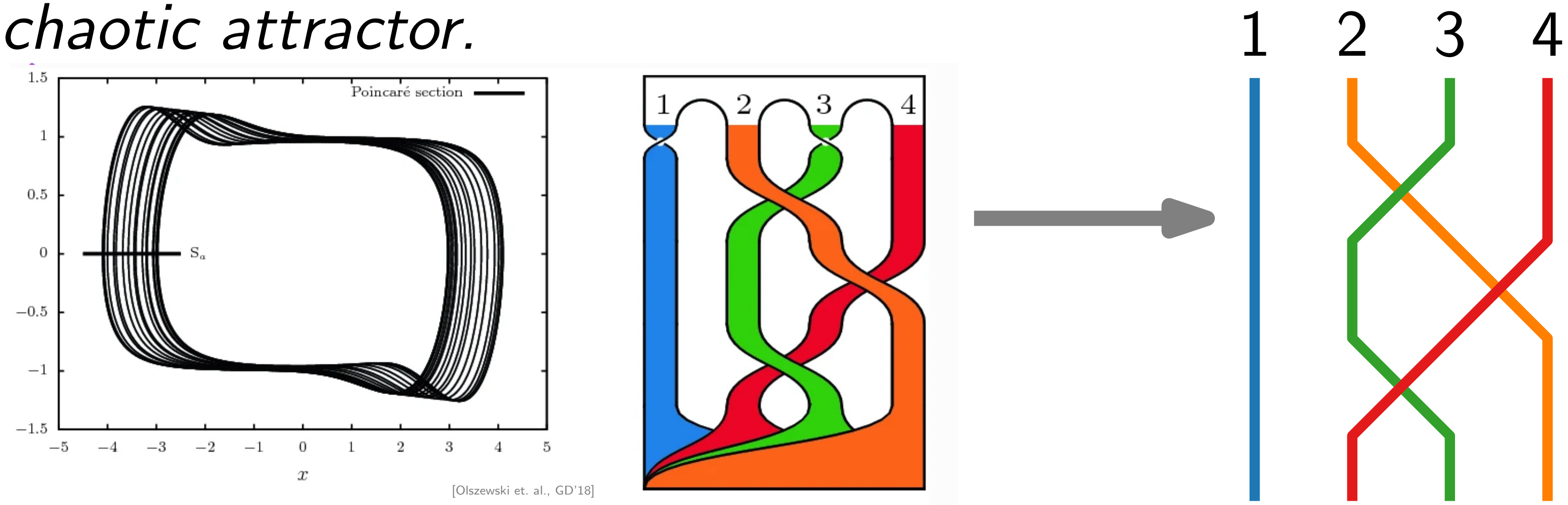


- Yamanaka, Horiyama, Uno, and Wasa.
  Ladder-lottery realization.
  [CCCG 2018]



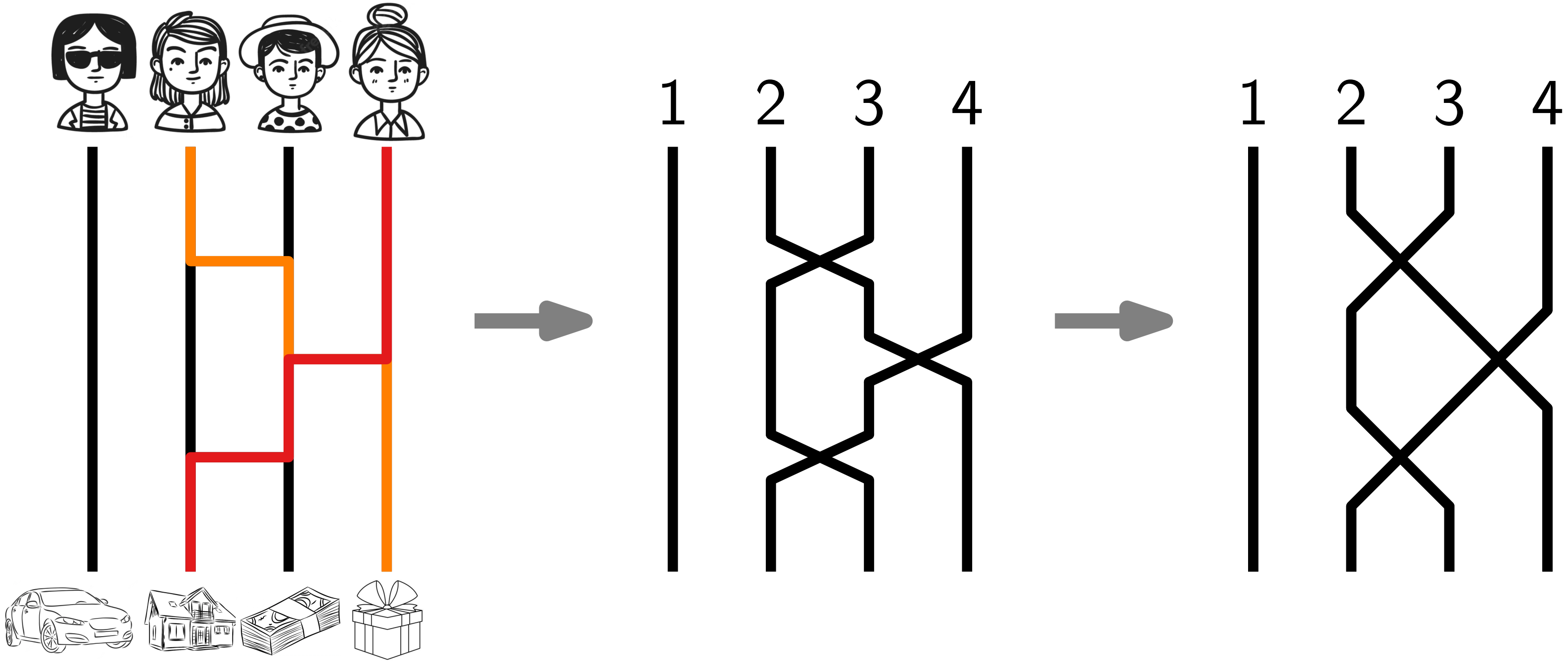
Related Work and Motivation

- Olszewski, Meder, Kieffer, Bleuse, Rosalie, Danoy, and Bouvry.
  Visualizing the template of a chaotic attractor.
  [GD 2018]



- Yamanaka, Horiyama, Uno, and Wasa.
  Ladder-lottery realization.
  [CCCG 2018]

It is NP-hard to decide
if a given list is feasible.



Our contribution

We consider the **feasibility** problem – whether a given list has a tangle.

Our contribution

We consider the **feasibility** problem – whether a given list has a tangle.

- **Complexity:** improve the NP-hardness result of [Yamanaka et al., CCCG'18]

Our contribution

We consider the **feasibility** problem – whether a given list has a tangle.

- **Complexity:** improve the NP-hardness result of [Yamanaka et al., CCCG'18]

  Given a list of swaps, it is NP-complete to decide if it is feasible *even if every pair of wires has* $O(1)$ *(eight) swaps*.

Our contribution

We consider the **feasibility** problem – whether a given list has a tangle.

- **Complexity:** improve the NP-hardness result of [Yamanaka et al., CCCG'18]

  Given a list of swaps, it is NP-complete to decide if it is feasible *even if every pair of wires has* $O(1)$ *(eight) swaps*.

- **Exp.-Time Algorithm:** can check feasibility faster than finding optimal-height tangles via [FKWRZ, GD'19]

Our contribution

We consider the **feasibility** problem – whether a given list has a tangle.

- **Complexity:** improve the NP-hardness result of [Yamanaka et al., CCCG'18]

  Given a list of swaps, it is NP-complete to decide if it is feasible
  even if every pair of wires has $O(1)$ *(eight) swaps*.

- **Exp.-Time Algorithm:** can check feasibility faster than finding optimal-height tangles via [FKWRZ, GD'19]

- **FPT Algorithm** parametrized by the number of wires

Our contribution

We consider the **feasibility** problem – whether a given list has a tangle.

- **Complexity:** improve the NP-hardness result of [Yamanaka et al., CCCG'18]

  Given a list of swaps, it is NP-complete to decide if it is feasible
  even if every pair of wires has $O(1)$ *(eight) swaps*.

- **Exp.-Time Algorithm:** can check feasibility faster than finding optimal-height tangles via [FKWRZ, GD'19]

- **FPT Algorithm** parametrized by the number of wires

Complexity

Theorem.
Given a list of swaps, it is NP-complete to decide if it is feasible even if every pair of wires has at most eight swaps.

Complexity

Theorem.
Given a list of swaps, it is NP-complete to decide if it is feasible even if every pair of wires has at most eight swaps.

Proof.
Reduction from POSITIVE NOT-ALL-EQUAL 3-SAT DIFF.

Complexity

Theorem.
Given a list of swaps, it is NP-complete to decide if it is feasible even if every pair of wires has at most eight swaps.

Proof.
Reduction from Positive Not-All-Equal $\boxed{\text{3-SAT}}$ Diff.

$$F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \neg x_1$$

Complexity

Theorem.
Given a list of swaps, it is NP-complete to decide if it is feasible even if every pair of wires has at most eight swaps.

Proof.
Reduction from POSITIVE NOT-ALL-EQUAL $\boxed{\text{3-SAT}}$ DIFF.

$$F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \neg x_1$$

$$\cancel{(F \vee F \vee F)}$$

Complexity

Theorem.
Given a list of swaps, it is NP-complete to decide if it is feasible even if every pair of wires has at most eight swaps.

Proof.
Reduction from POSITIVE $\boxed{\text{NOT-ALL-EQUAL 3-SAT}}$ DIFF.

$$F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \neg x_1$$

$$\cancel{(F \vee F \vee F)}$$

Complexity

Theorem.
Given a list of swaps, it is NP-complete to decide if it is feasible even if every pair of wires has at most eight swaps.

Proof.
Reduction from POSITIVE $\boxed{\text{NOT-ALL-EQUAL 3-SAT}}$ DIFF.

$F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \neg x_1$

$\cancel{(F \vee F \vee F)}$

$\cancel{(T \vee T \vee T)}$

Complexity

Theorem.
Given a list of swaps, it is NP-complete to decide if it is feasible even if every pair of wires has at most eight swaps.

Proof.
Reduction from $\boxed{\text{Positive Not-All-Equal 3-SAT}}$ Diff.

$F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \neg x_1$

$\cancel{(F \vee F \vee F)}$

$\cancel{(T \vee T \vee T)}$

Complexity

Theorem.
Given a list of swaps, it is NP-complete to decide if it is feasible even if every pair of wires has at most eight swaps.

Proof.
Reduction from $\boxed{\textsc{Positive Not-All-Equal 3-SAT}}$ $\textsc{Diff}$.

$F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \neg x_1$

$\cancel{(F \vee F \vee F)}$

$\cancel{(T \vee T \vee T)}$

~~negative literals~~

Complexity

Theorem.
Given a list of swaps, it is NP-complete to decide if it is feasible even if every pair of wires has at most eight swaps.

Proof.
Reduction from $\boxed{\text{Positive Not-All-Equal 3-SAT}}$ Diff.

$F = (x_1 \vee x_2) \wedge (x_1 \vee x_2 \vee x_3) \wedge x_1$

$\cancel{(F \vee F \vee F)}$

$\cancel{(T \vee T \vee T)}$

~~negative literals~~

Complexity

Theorem.
Given a list of swaps, it is NP-complete to decide if it is feasible even if every pair of wires has at most eight swaps.

Proof.
Reduction from $\boxed{\text{Positive Not-All-Equal 3-SAT Diff.}}$

$$F = (x_1 \vee x_2) \wedge (x_1 \vee x_2 \vee x_3) \wedge x_1$$

$\cancel{(F \vee F \vee F)}$

$\cancel{(T \vee T \vee T)}$

~~negative literals~~

Complexity

Theorem.
Given a list of swaps, it is NP-complete to decide if it is feasible even if every pair of wires has at most eight swaps.

Proof.
Reduction from $\boxed{\textsc{Positive Not-All-Equal 3-SAT Diff.}}$

$$F = (x_1 \vee x_2) \wedge (x_1 \vee x_2 \vee x_3) \wedge x_1$$

$\cancel{(F \vee F \vee F)}$

$\cancel{(T \vee T \vee T)}$

~~negative literals~~

all three variables in each clause differ from each other

Complexity

Theorem.
Given a list of swaps, it is NP-complete to decide if it is feasible even if every pair of wires has at most eight swaps.

Proof.
Reduction from $\boxed{\textsc{Positive Not-All-Equal 3-SAT Diff.}}$

$$F = (x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee x_4)$$

$\cancel{(F \vee F \vee F)}$

$\cancel{(T \vee T \vee T)}$

~~negative literals~~

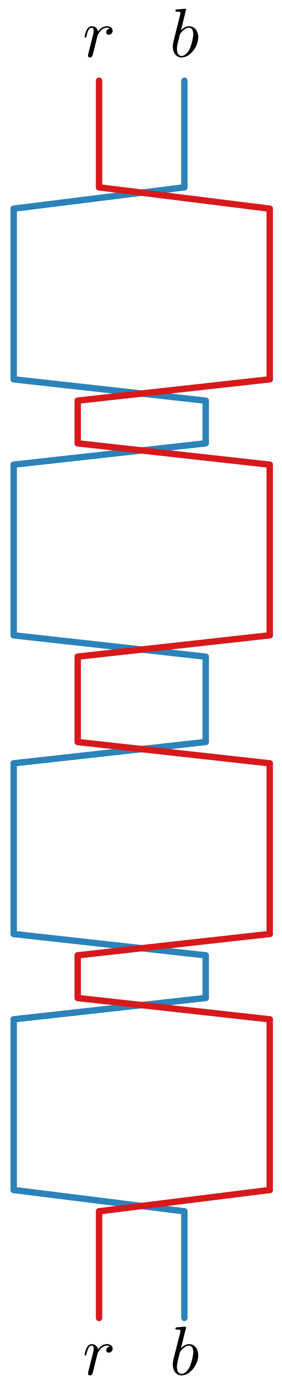
all three variables in each clause differ from each other

Idea

- Two wires build 4 *loops* that we consider.

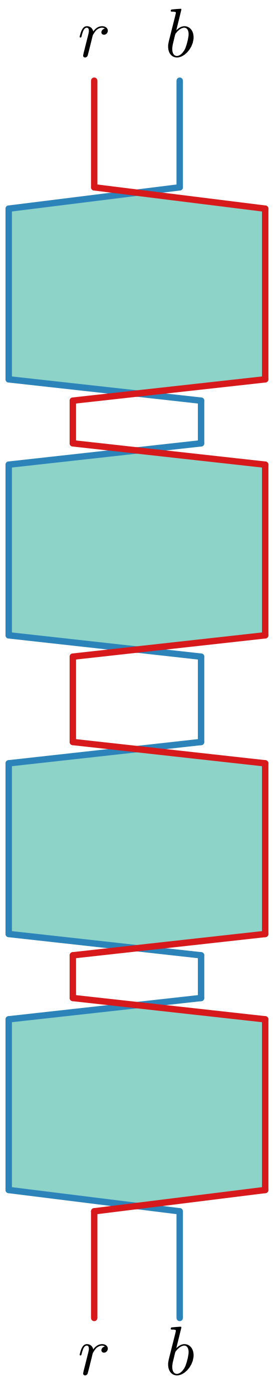
Idea

- Two wires build 4 *loops* that we consider.



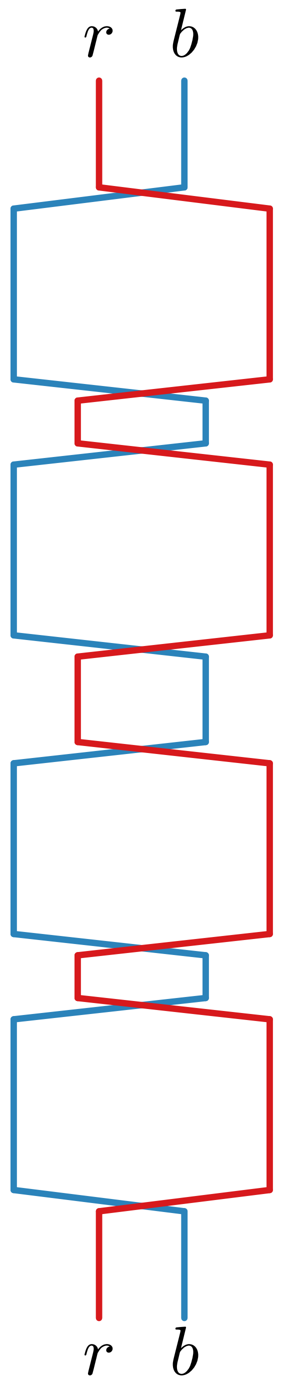
Idea

- Two wires build 4 *loops* that we consider.



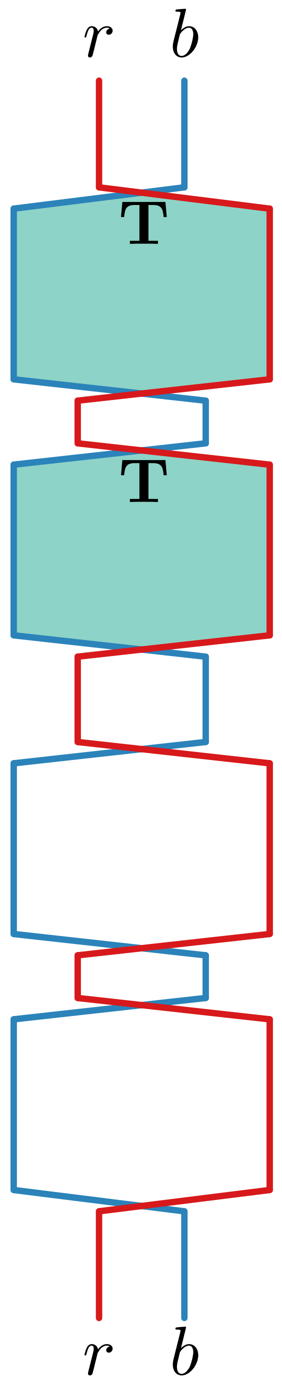
Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*,



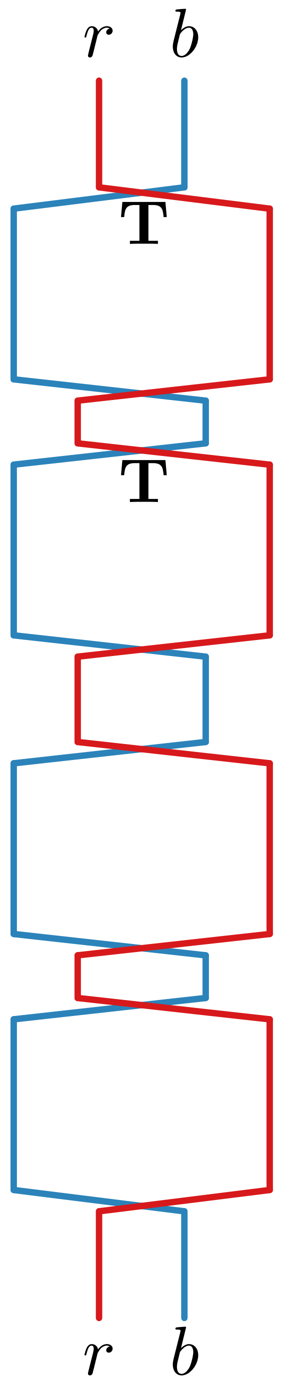
Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*,



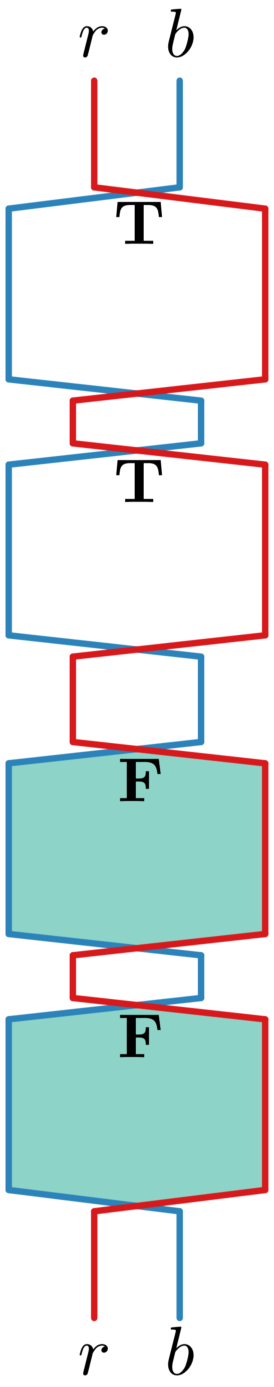
Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*, the other two *false*.



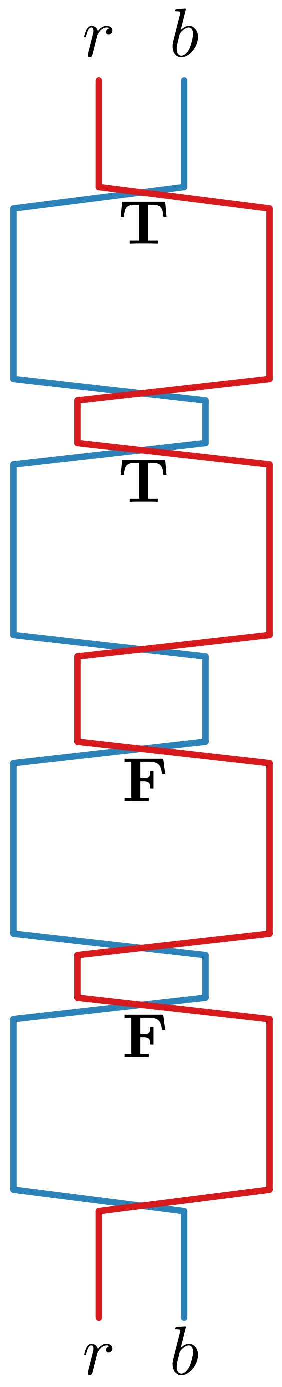
Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*, the other two *false*.



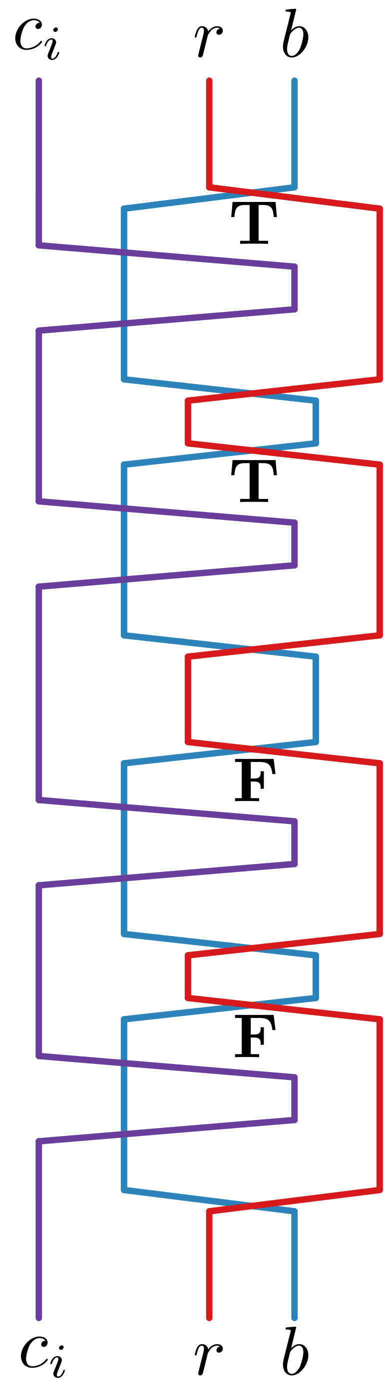
Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*, the other two *false*.
- For each clause, there is a wire with an *arm* in each of the 4 loops.



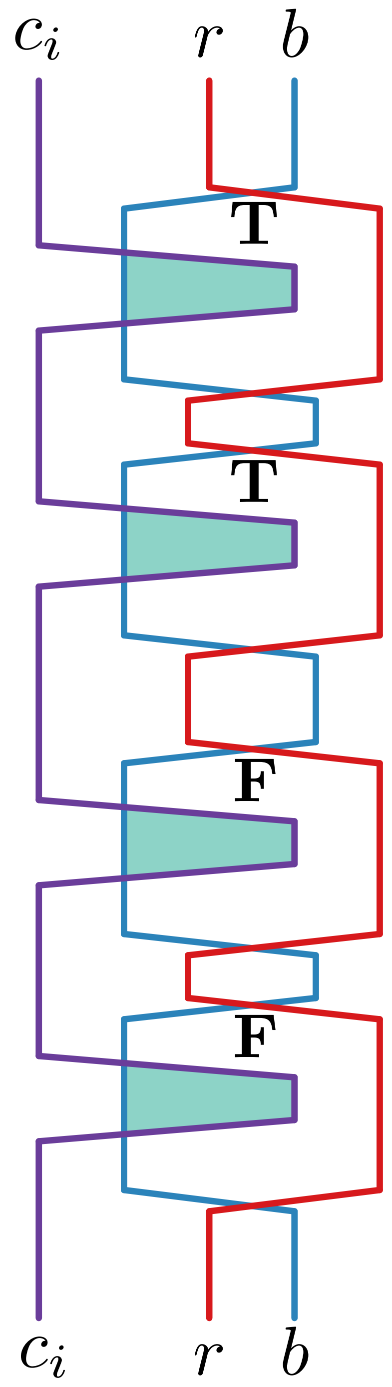
Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*, the other two *false*.
- For each clause, there is a wire with an *arm* in each of the 4 loops.



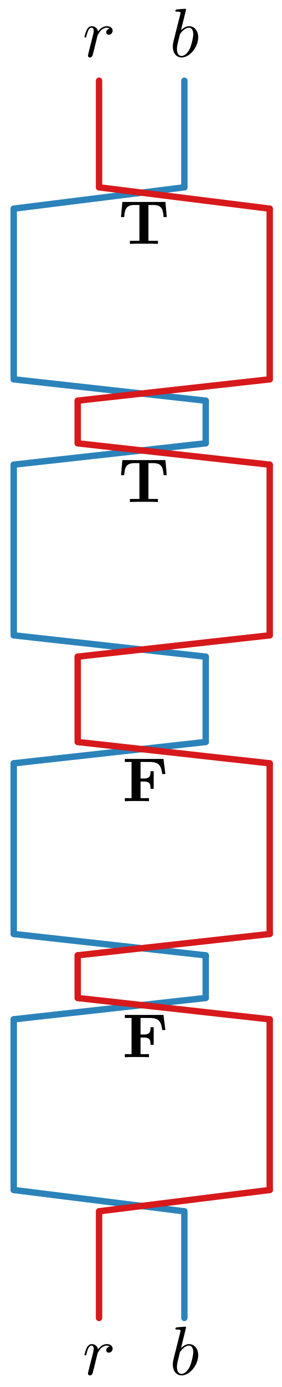
Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*, the other two *false*.
- For each clause, there is a wire with an *arm* in each of the 4 loops.



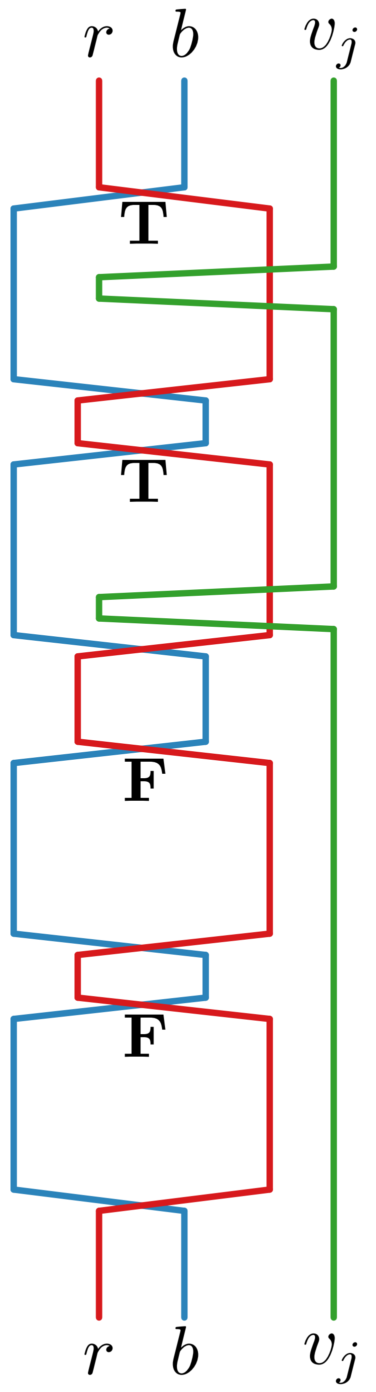
Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*, the other two *false*.
- For each clause, there is a wire with an *arm* in each of the 4 loops.
- For each variable, there is a wire entering either both *true* or both *false* loops.



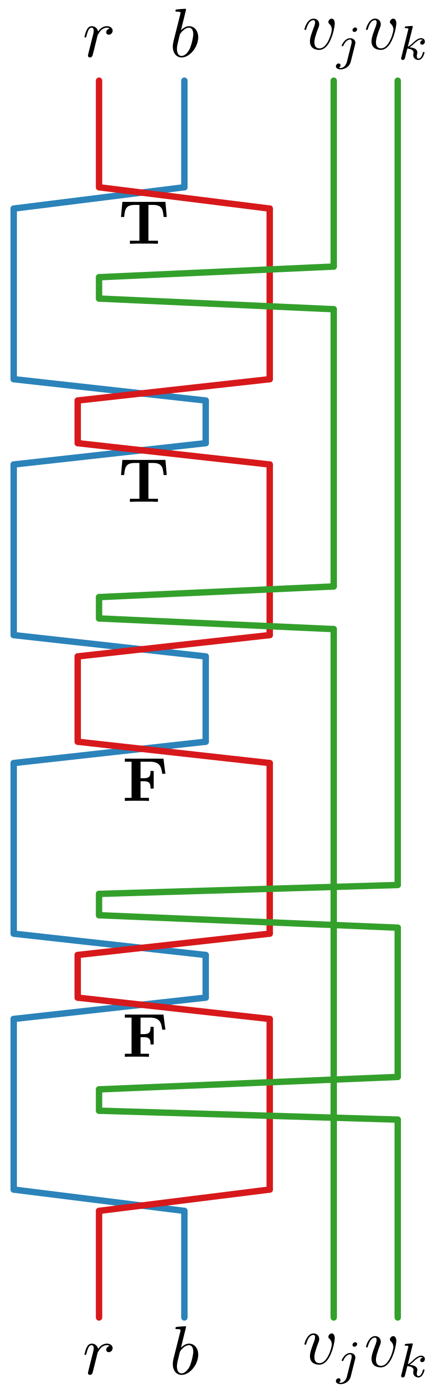
Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*, the other two *false*.
- For each clause, there is a wire with an *arm* in each of the 4 loops.
- For each variable, there is a wire entering either both *true* or both *false* loops.



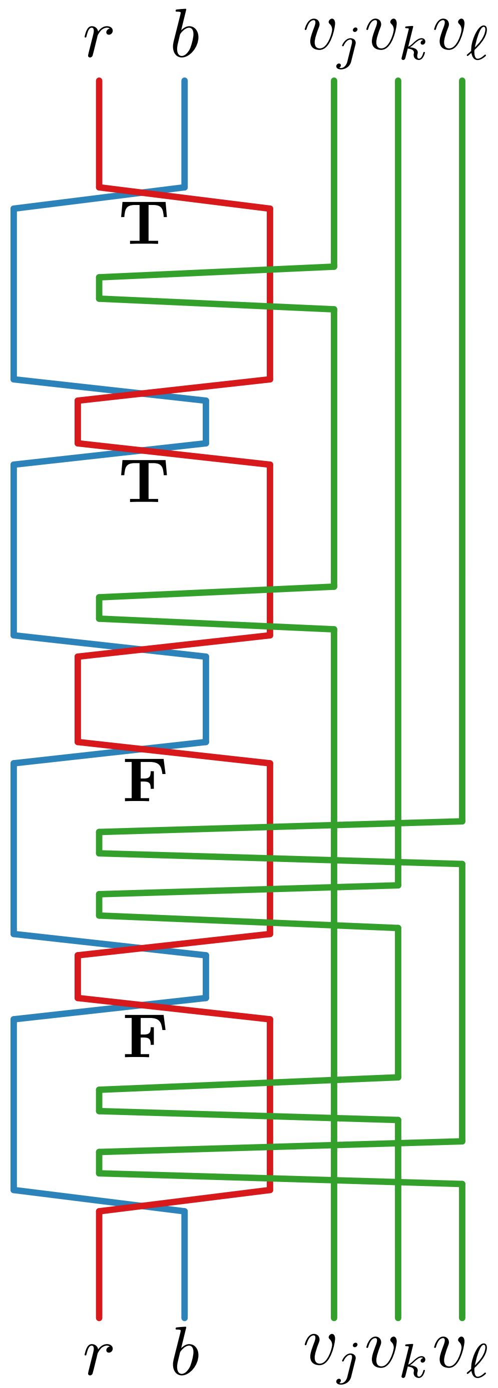
Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*, the other two *false*.
- For each clause, there is a wire with an *arm* in each of the 4 loops.
- For each variable, there is a wire entering either both *true* or both *false* loops.



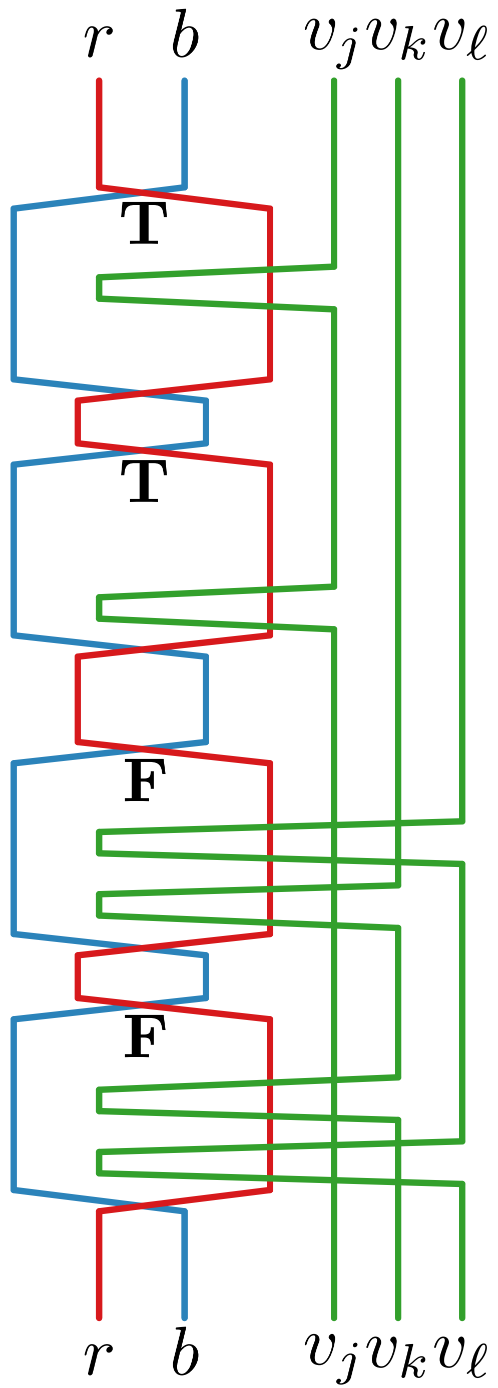
Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*, the other two *false*.
- For each clause, there is a wire with an *arm* in each of the 4 loops.
- For each variable, there is a wire entering either both *true* or both *false* loops.



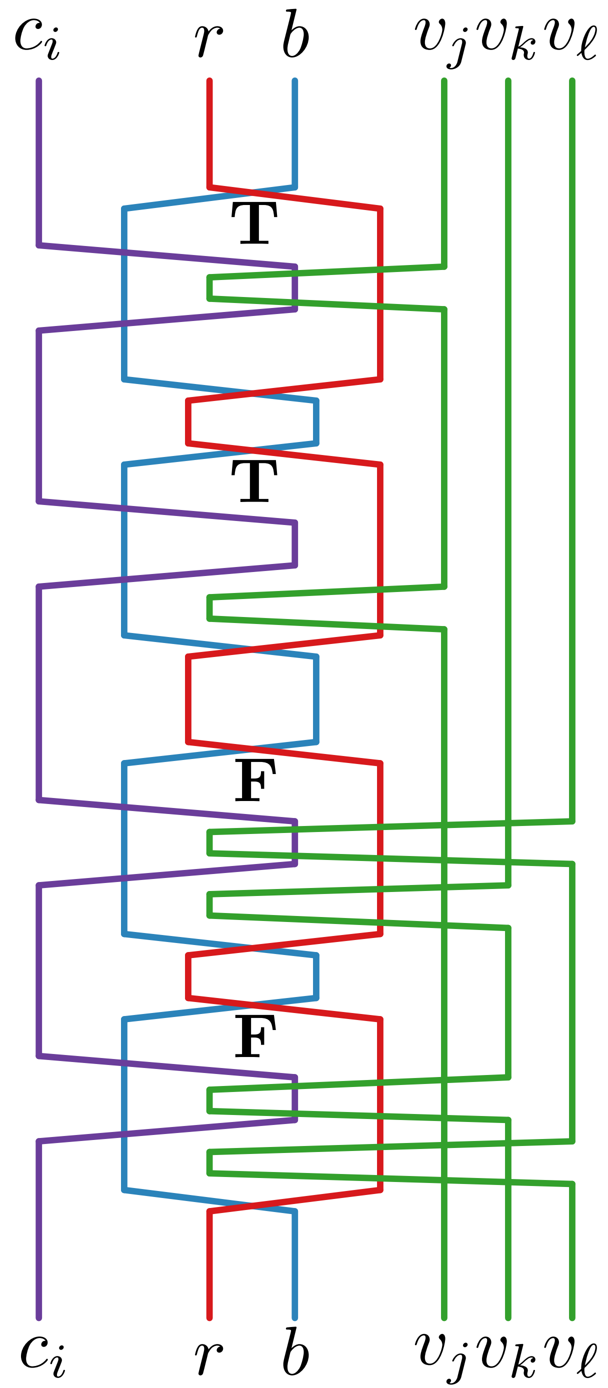
Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*, the other two *false*.
- For each clause, there is a wire with an *arm* in each of the 4 loops.
- For each variable, there is a wire entering either both *true* or both *false* loops.
- Each clause wire meets precisely its three corresponding variable wires – each one in a different loop.



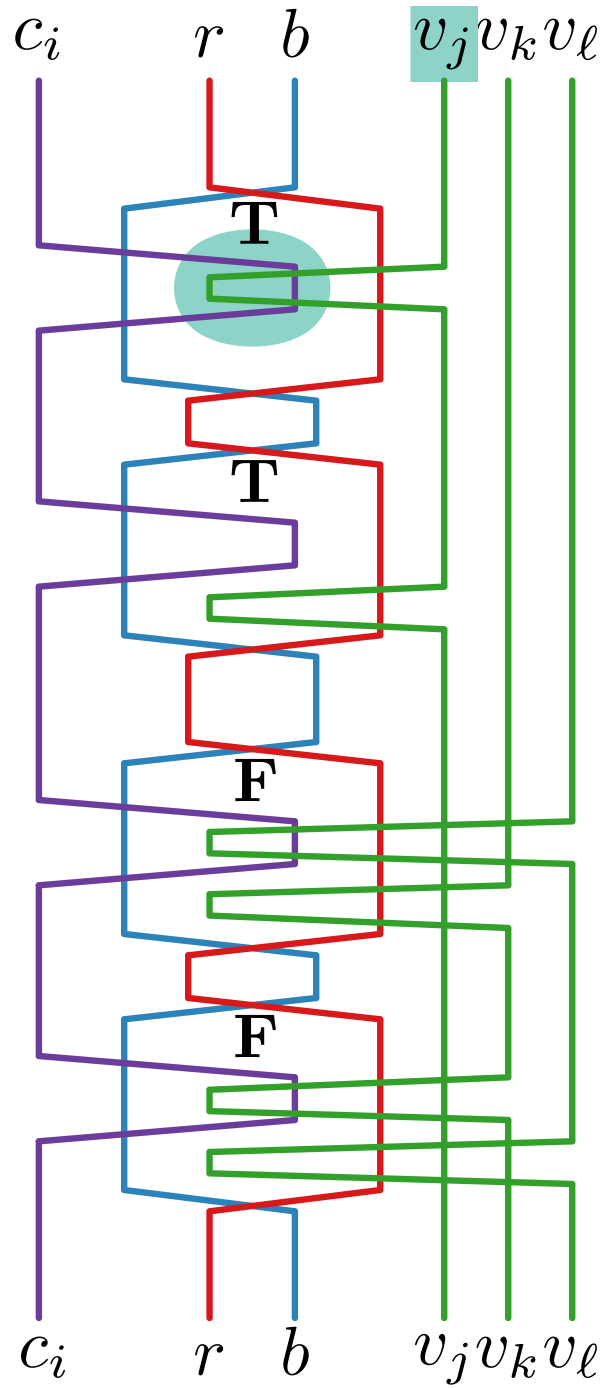
Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*, the other two *false*.
- For each clause, there is a wire with an *arm* in each of the 4 loops.
- For each variable, there is a wire entering either both *true* or both *false* loops.
- Each clause wire meets precisely its three corresponding variable wires – each one in a different loop.



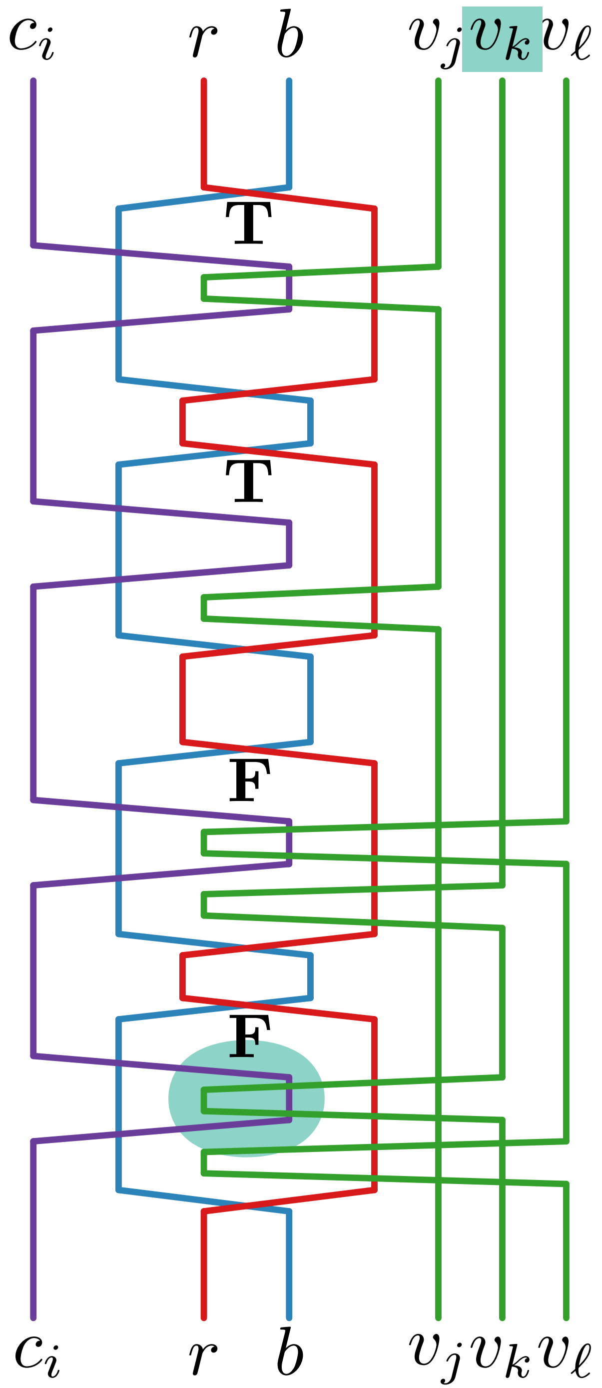
Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*, the other two *false*.
- For each clause, there is a wire with an *arm* in each of the 4 loops.
- For each variable, there is a wire entering either both *true* or both *false* loops.
- Each clause wire meets precisely its three corresponding variable wires – each one in a different loop.



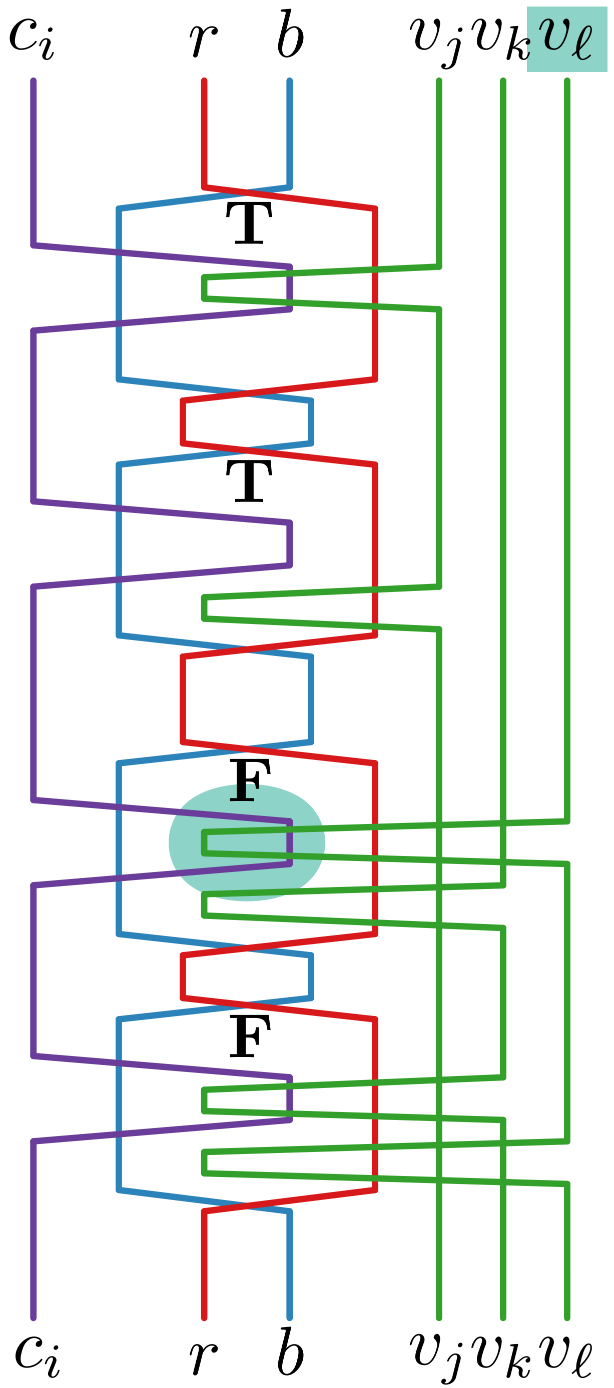
Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*, the other two *false*.
- For each clause, there is a wire with an *arm* in each of the 4 loops.
- For each variable, there is a wire entering either both *true* or both *false* loops.
- Each clause wire meets precisely its three corresponding variable wires – each one in a different loop.



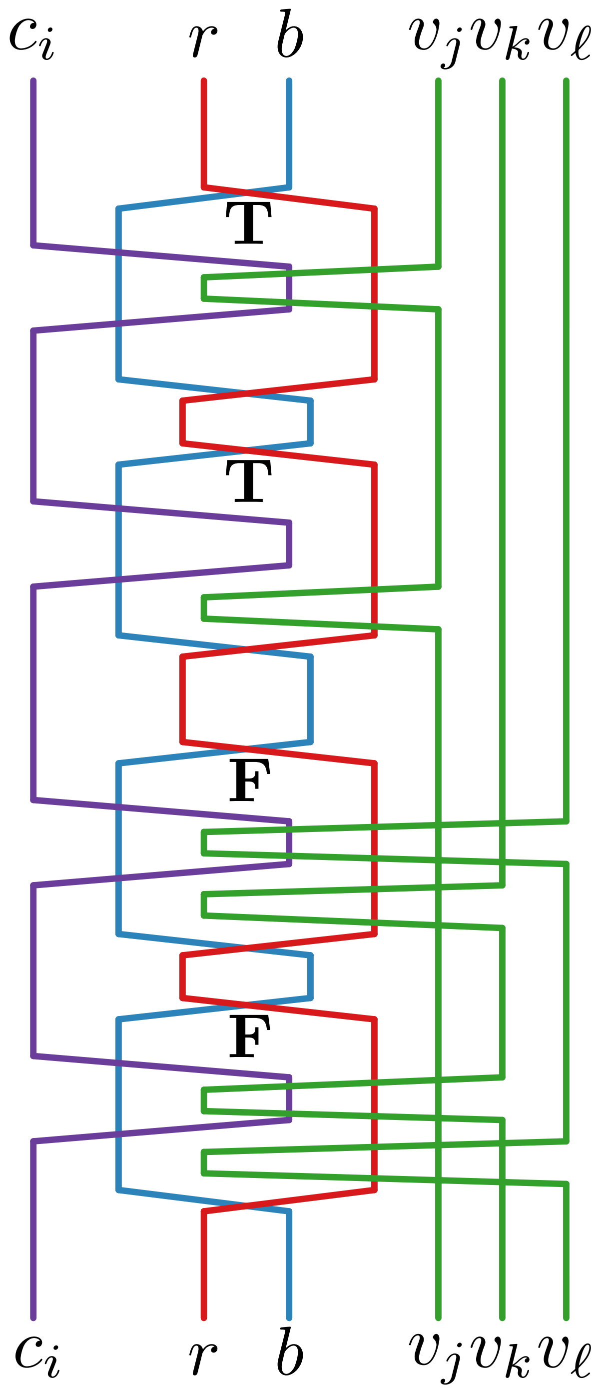
Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*, the other two *false*.
- For each clause, there is a wire with an *arm* in each of the 4 loops.
- For each variable, there is a wire entering either both *true* or both *false* loops.
- Each clause wire meets precisely its three corresponding variable wires – each one in a different loop.

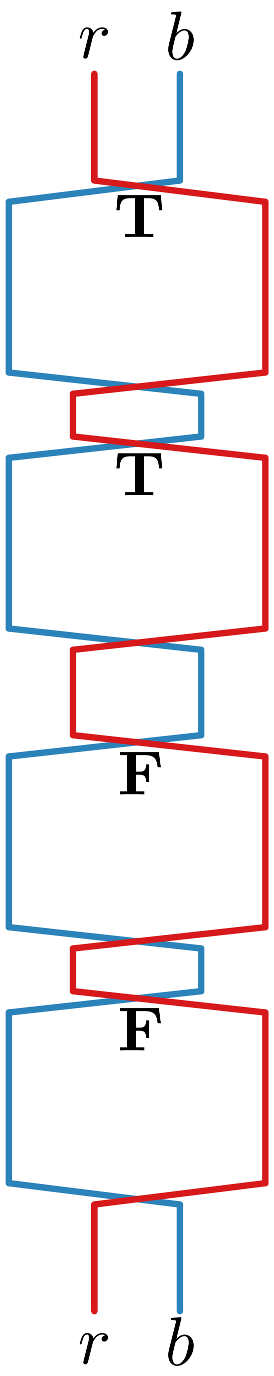


Idea

- Two wires build 4 *loops* that we consider.
- Two loops represent *true*, the other two *false*.
- For each clause, there is a wire with an *arm* in each of the 4 loops.
- For each variable, there is a wire entering either both *true* or both *false* loops.
- Each clause wire meets precisely its three corresponding variable wires – each one in a different loop.
- Only 2 *true* loops and 2 *false* loops $\Rightarrow$ clause wires meet all their variable wires iff POSITIVE NOT-ALL-EQUAL 3-SAT DIFF formula is satisfiable.

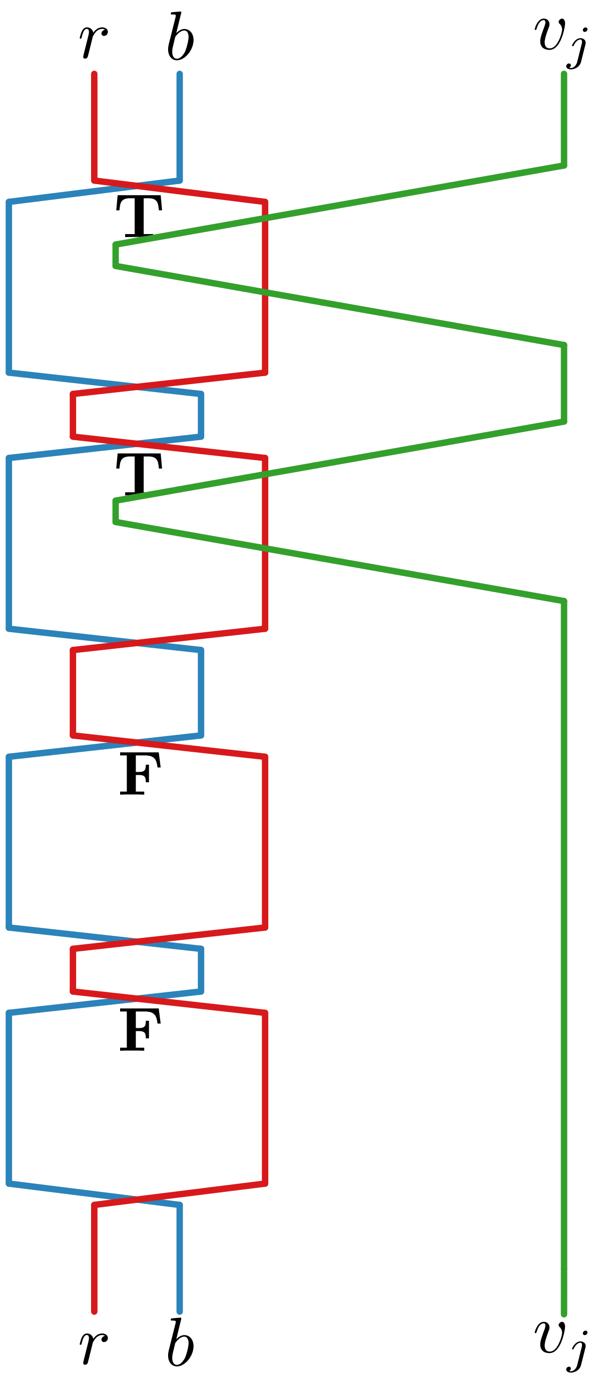


Variable Gadget



r, b: central loop structure

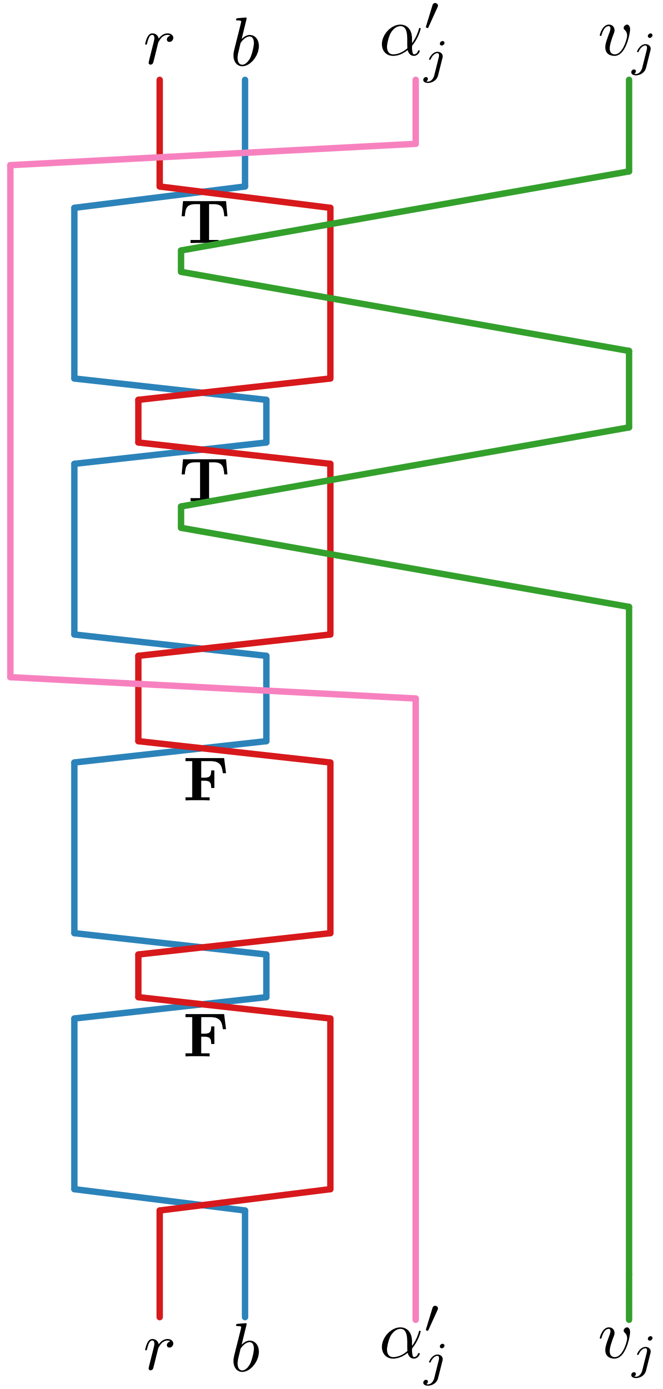
Variable Gadget



r, b: central loop structure

v_j: variable wire of j-th variable

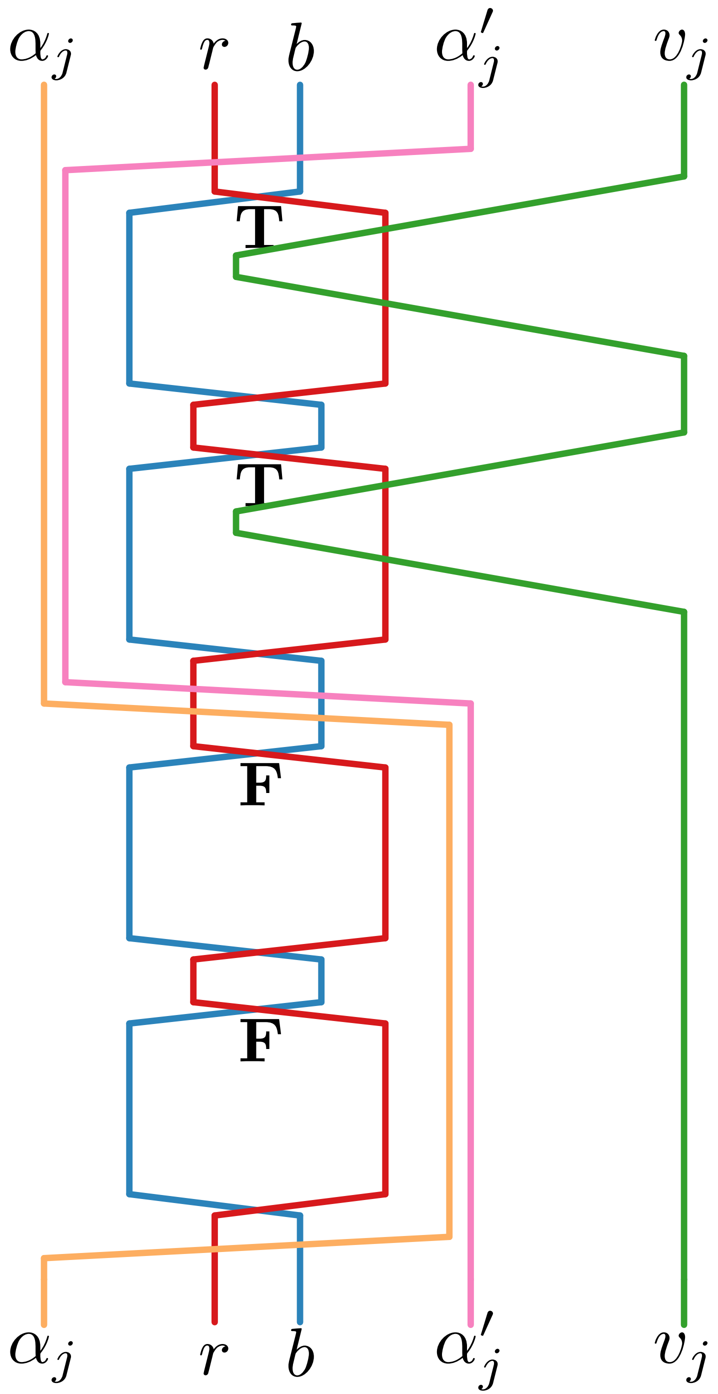
Variable Gadget



r, b: central loop structure

v_j: variable wire of j-th variable

Variable Gadget

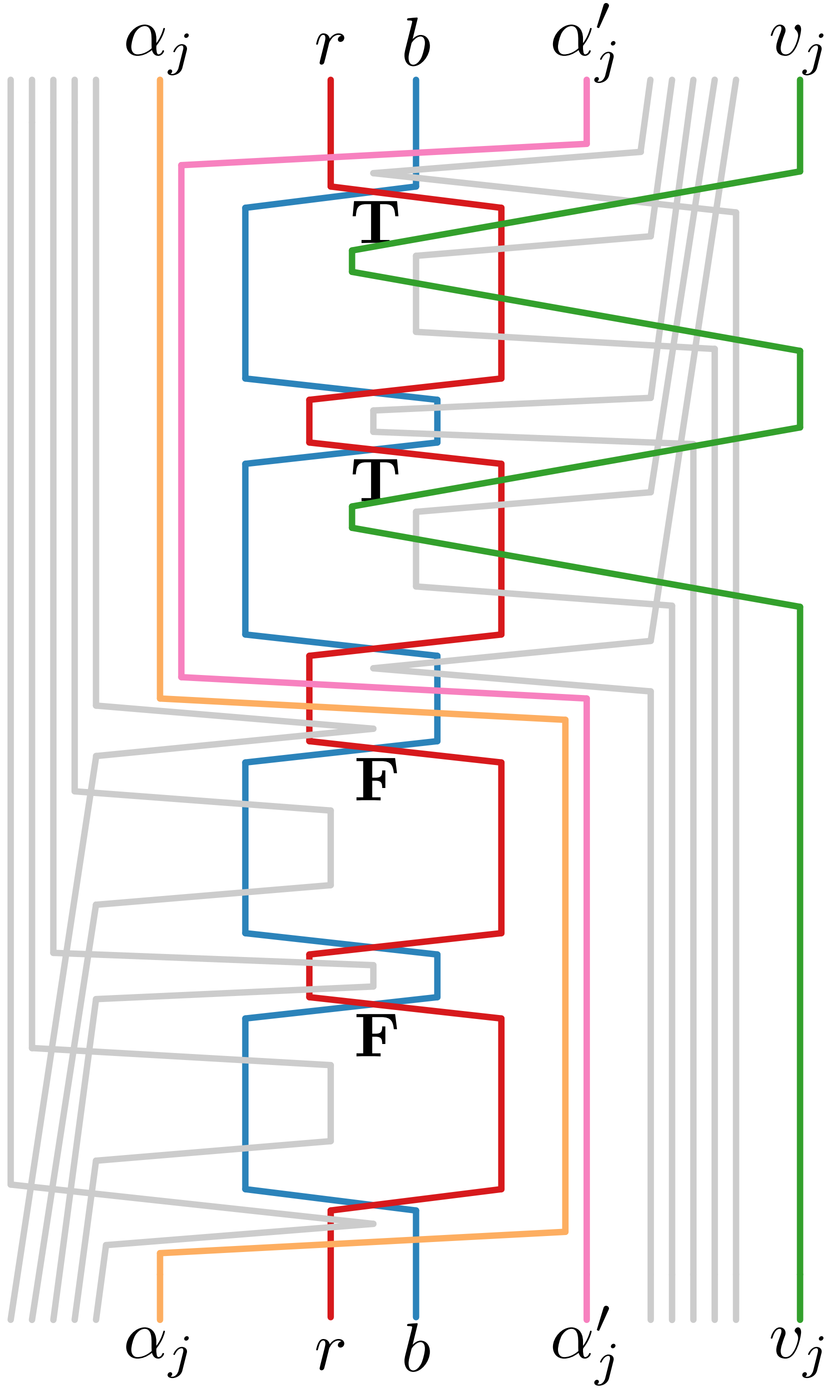


r, b: central loop structure

v_j: variable wire of j-th variable

α_j, α'_j: make v_j appear only in *true* or in *false* loops

Variable Gadget

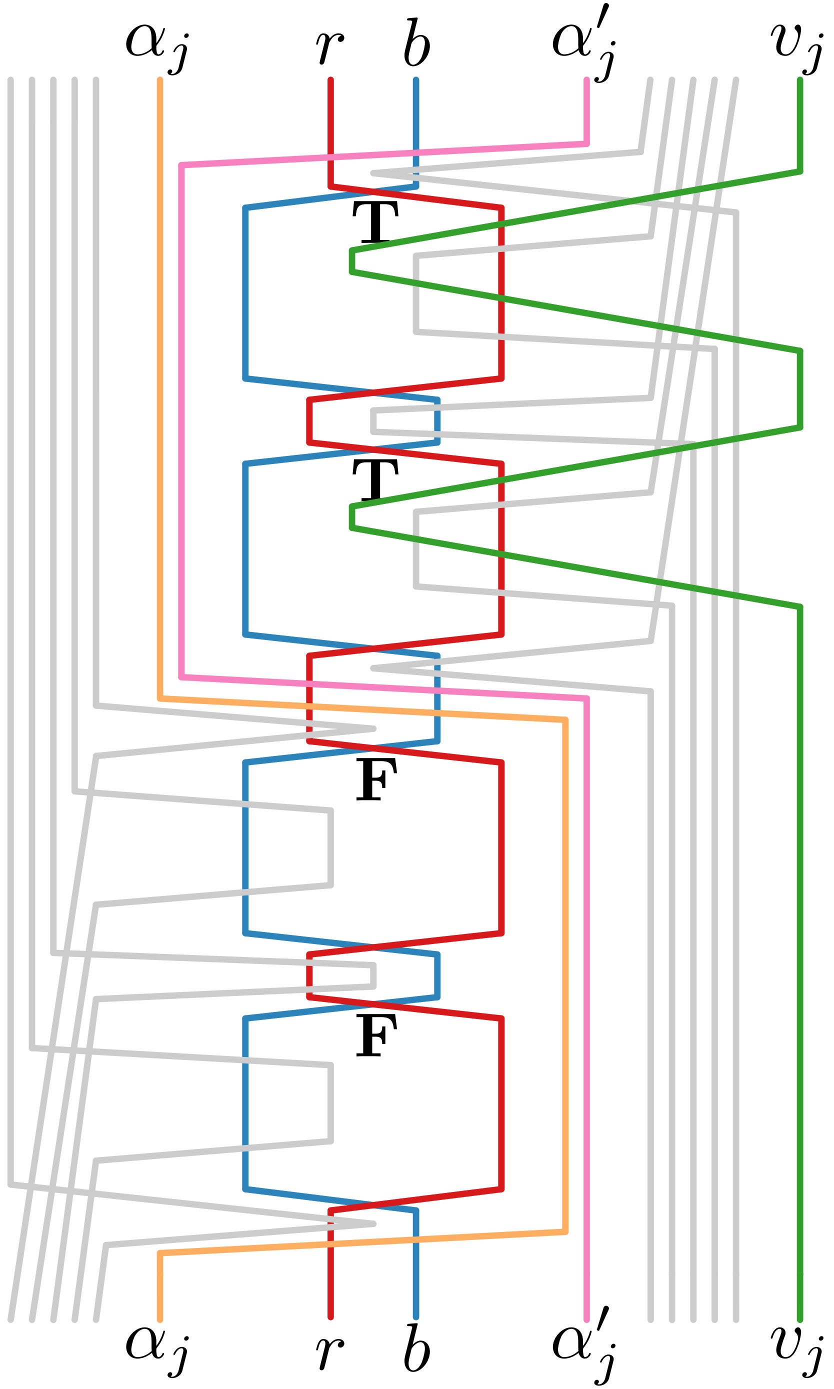


r, b: central loop structure

v_j: variable wire of j-th variable

α_j, α'_j: make v_j appear only in *true* or in *false* loops

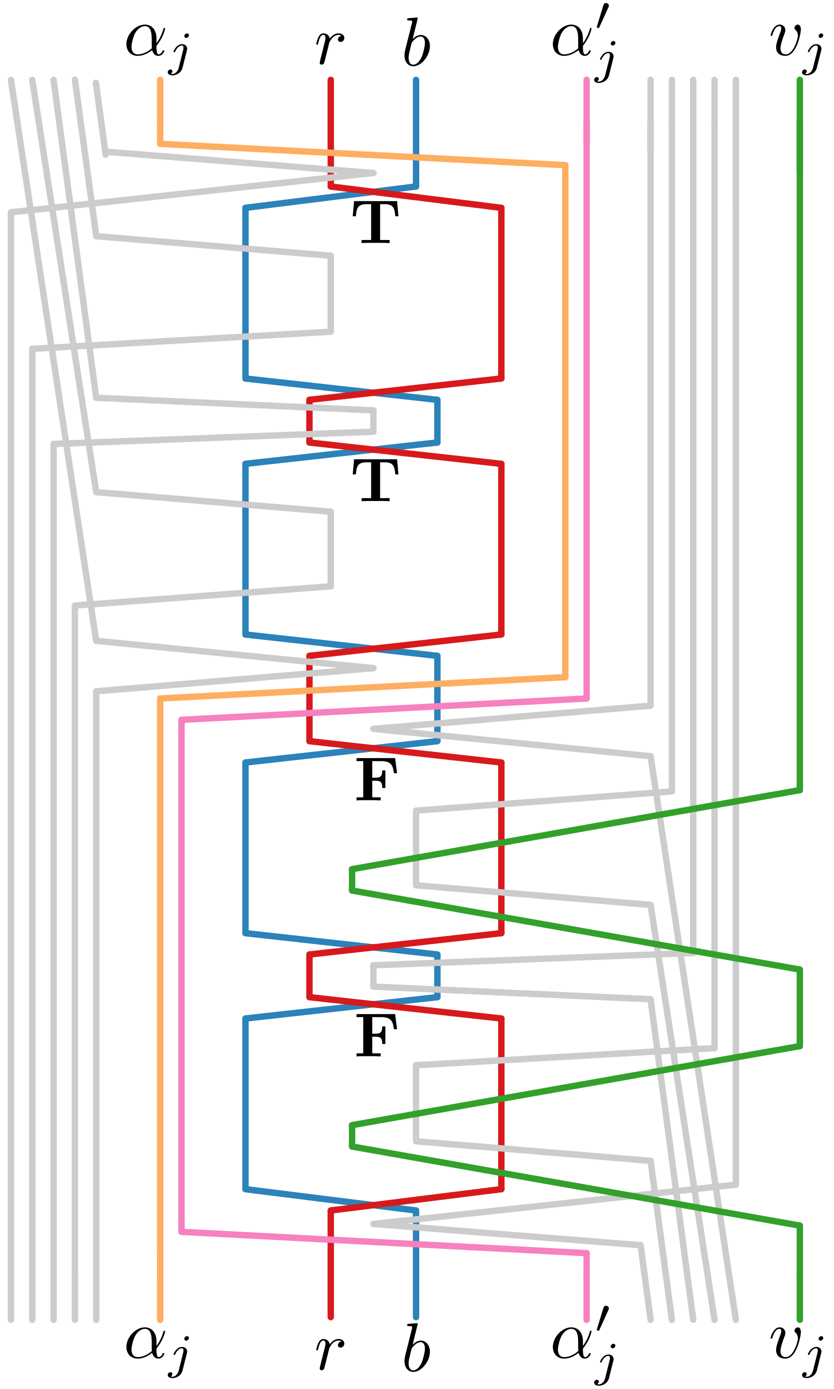
Variable Gadget



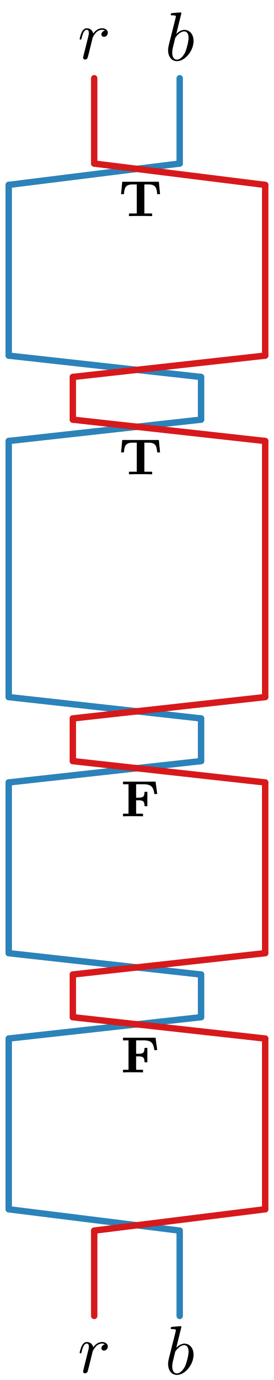
r, b: central loop structure

v_j: variable wire of j-th variable

α_j, α'_j: make v_j appear only in *true* or in *false* loops

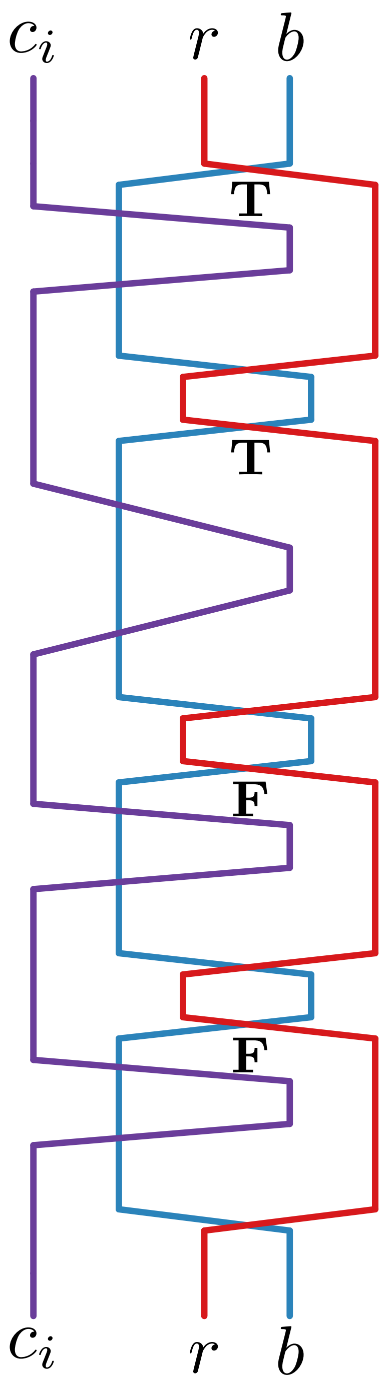


Clause Gadget



r, b: central loop structure

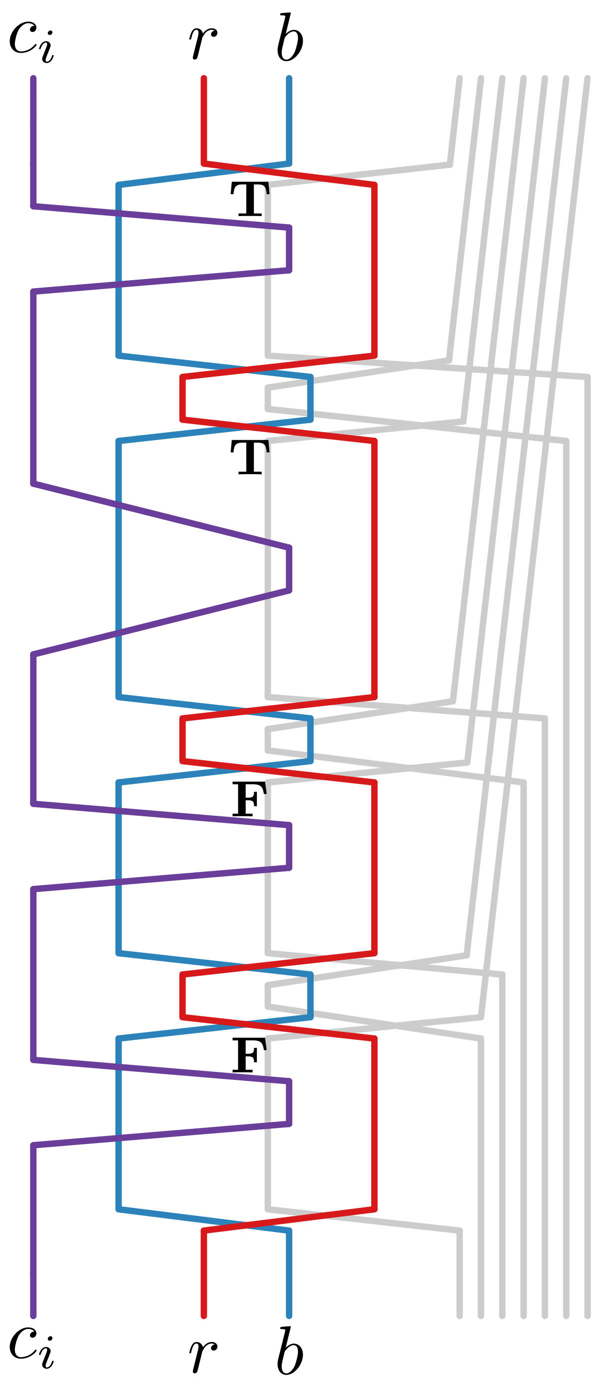
Clause Gadget



r, b: central loop structure

c_i: clause wire of i-th clause

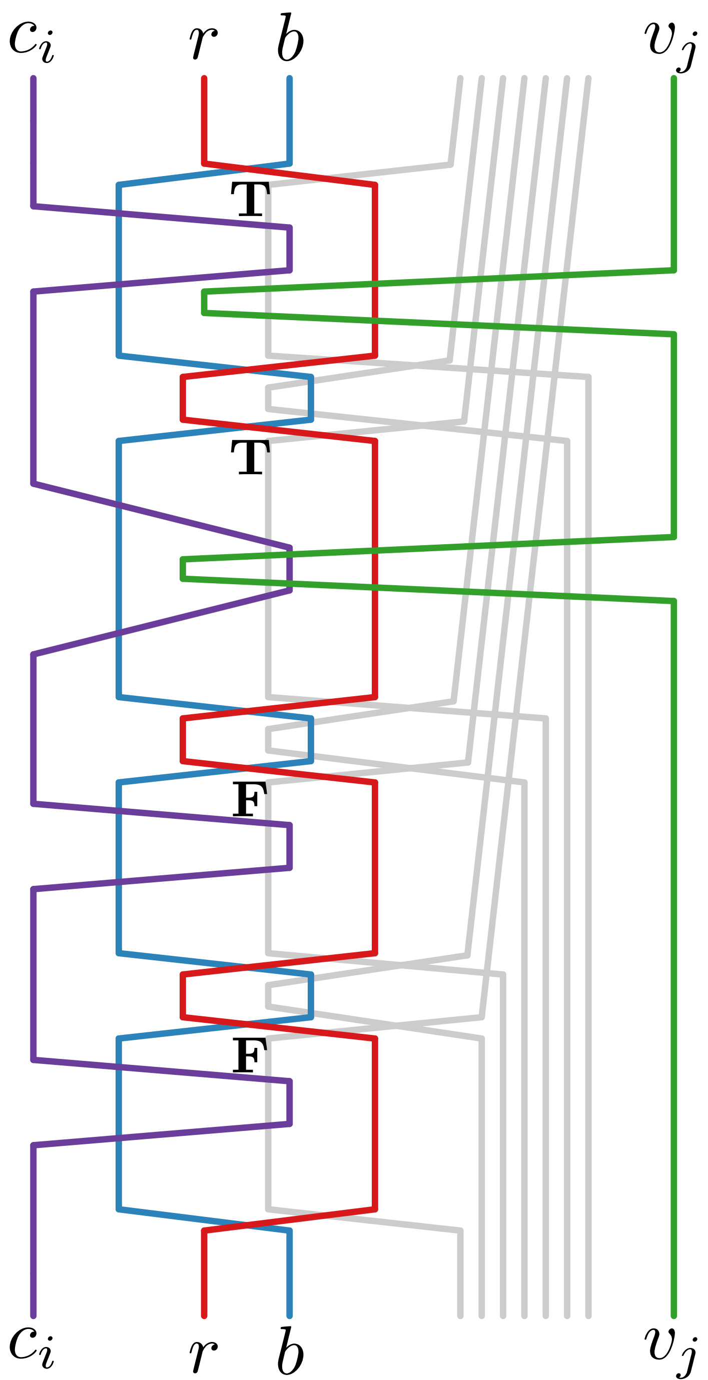
Clause Gadget



r, b: central loop structure

c_i: clause wire of i-th clause

Clause Gadget

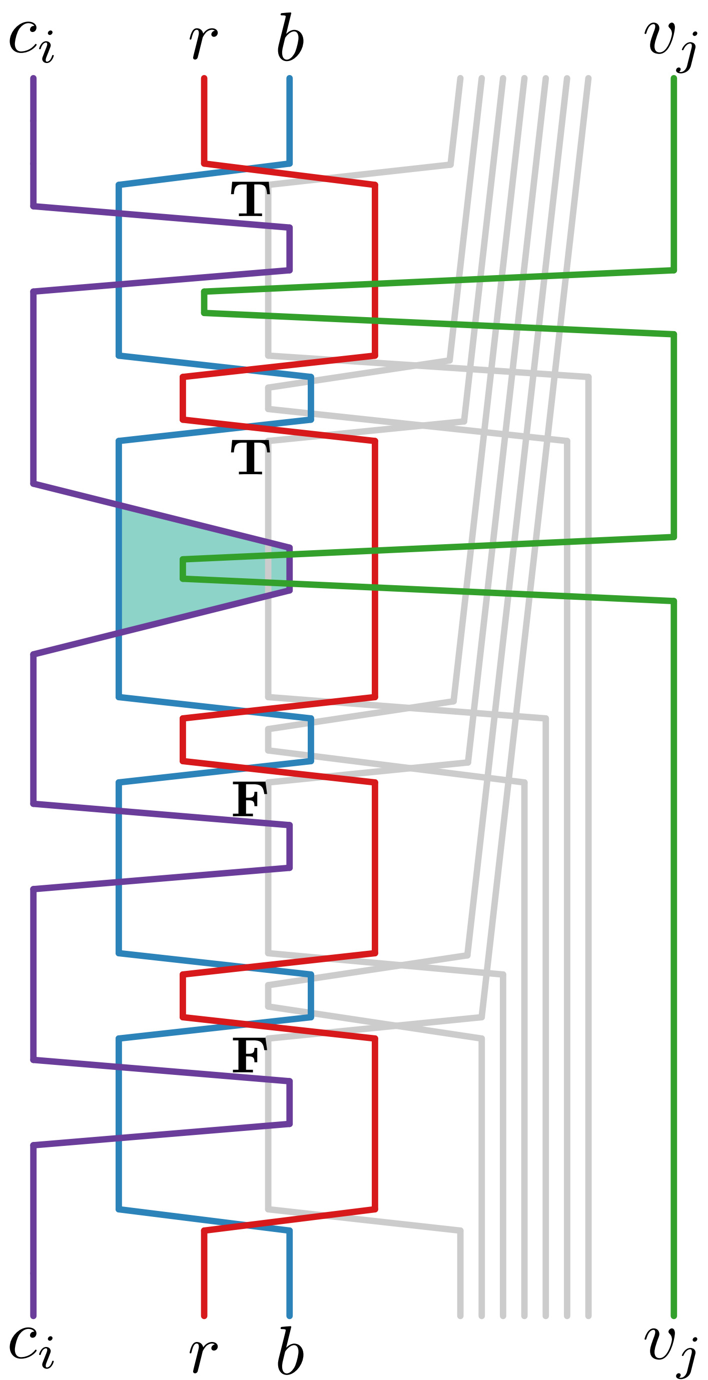


r, b: central loop structure

c_i: clause wire of i-th clause

v_j: variable wire of j-th variable

Clause Gadget

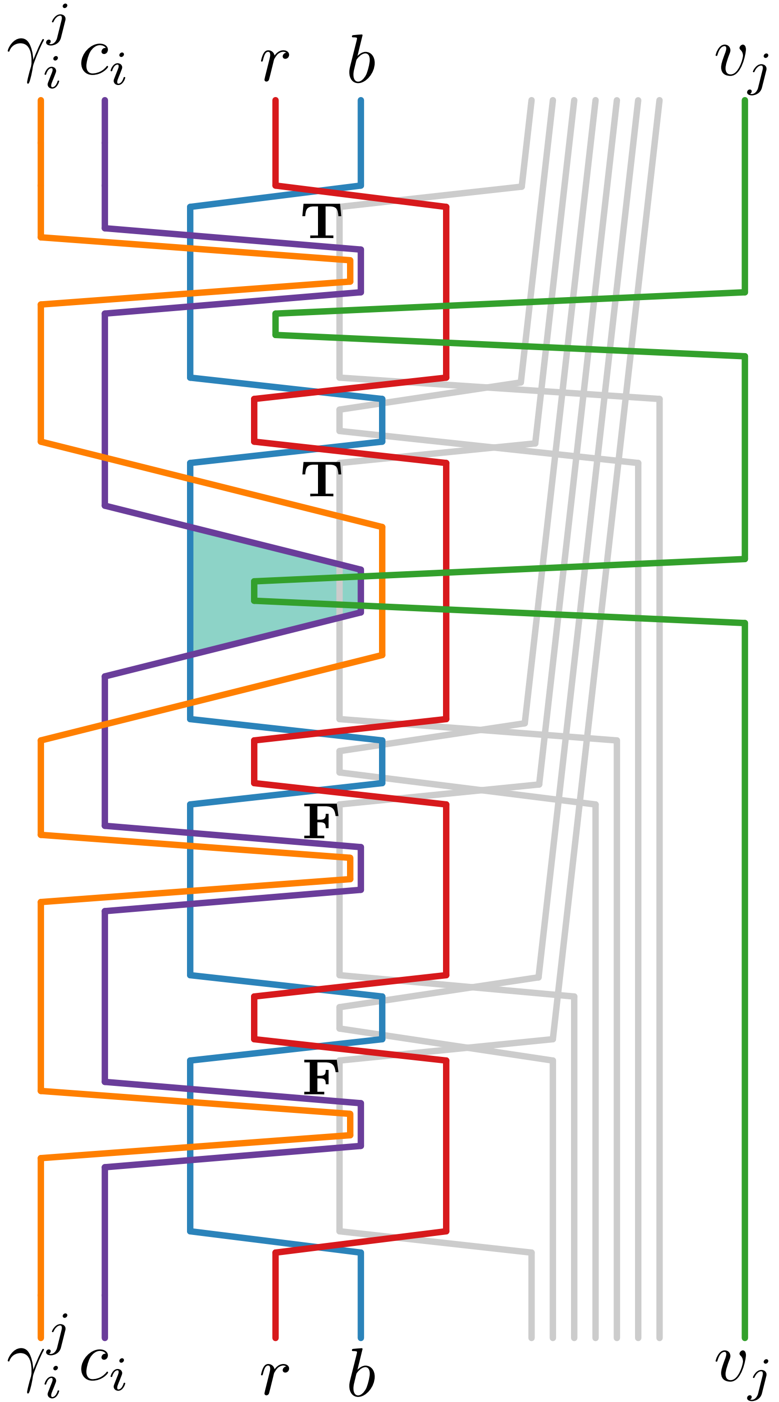


r, b: central loop structure

c_i: clause wire of i-th clause

v_j: variable wire of j-th variable

Clause Gadget



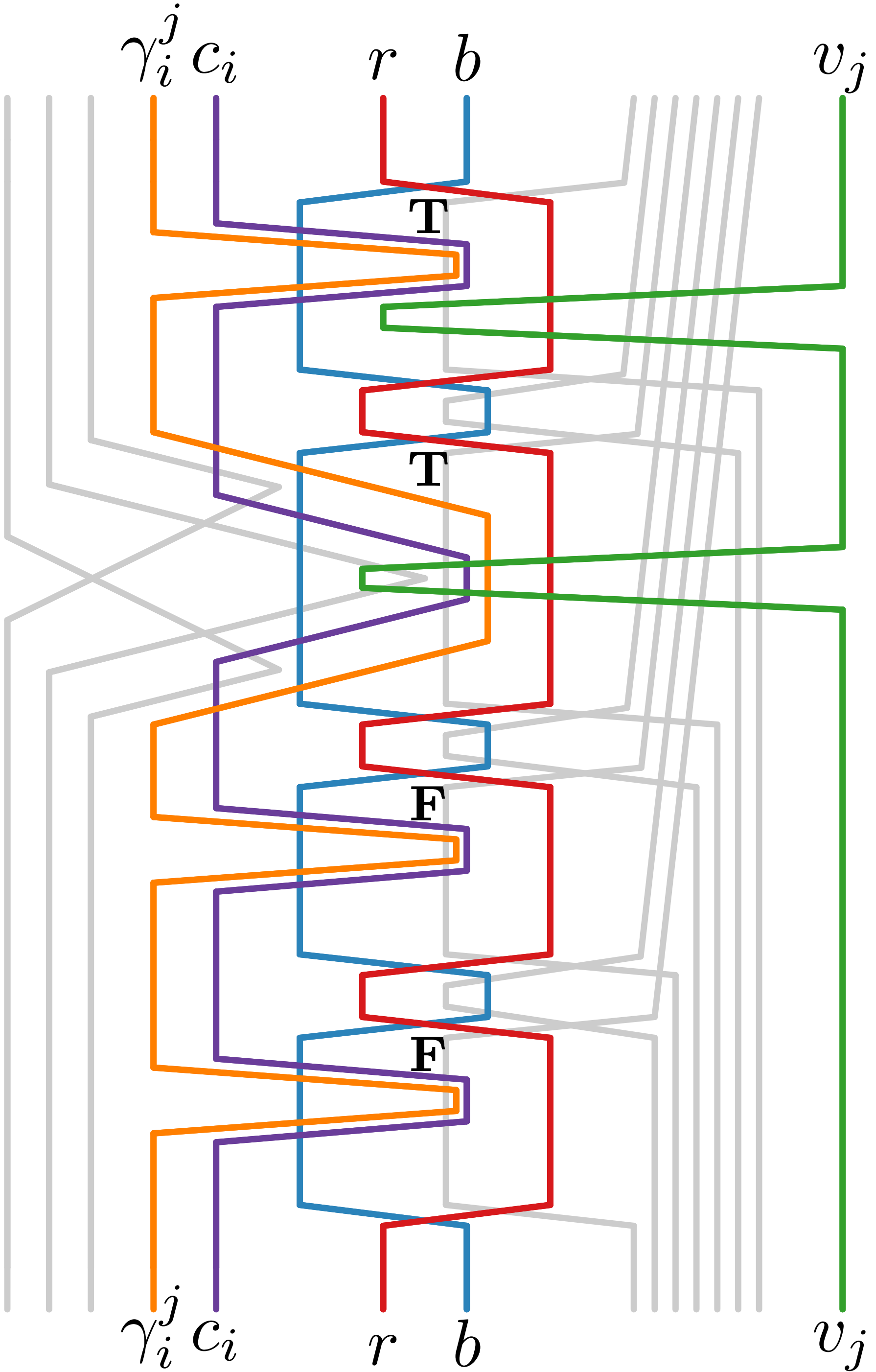
r, b: central loop structure

c_i: clause wire of i-th clause

v_j: variable wire of j-th variable

γ_i^j: protects the arm of c_i that intersects v_j from other variable wires

Clause Gadget



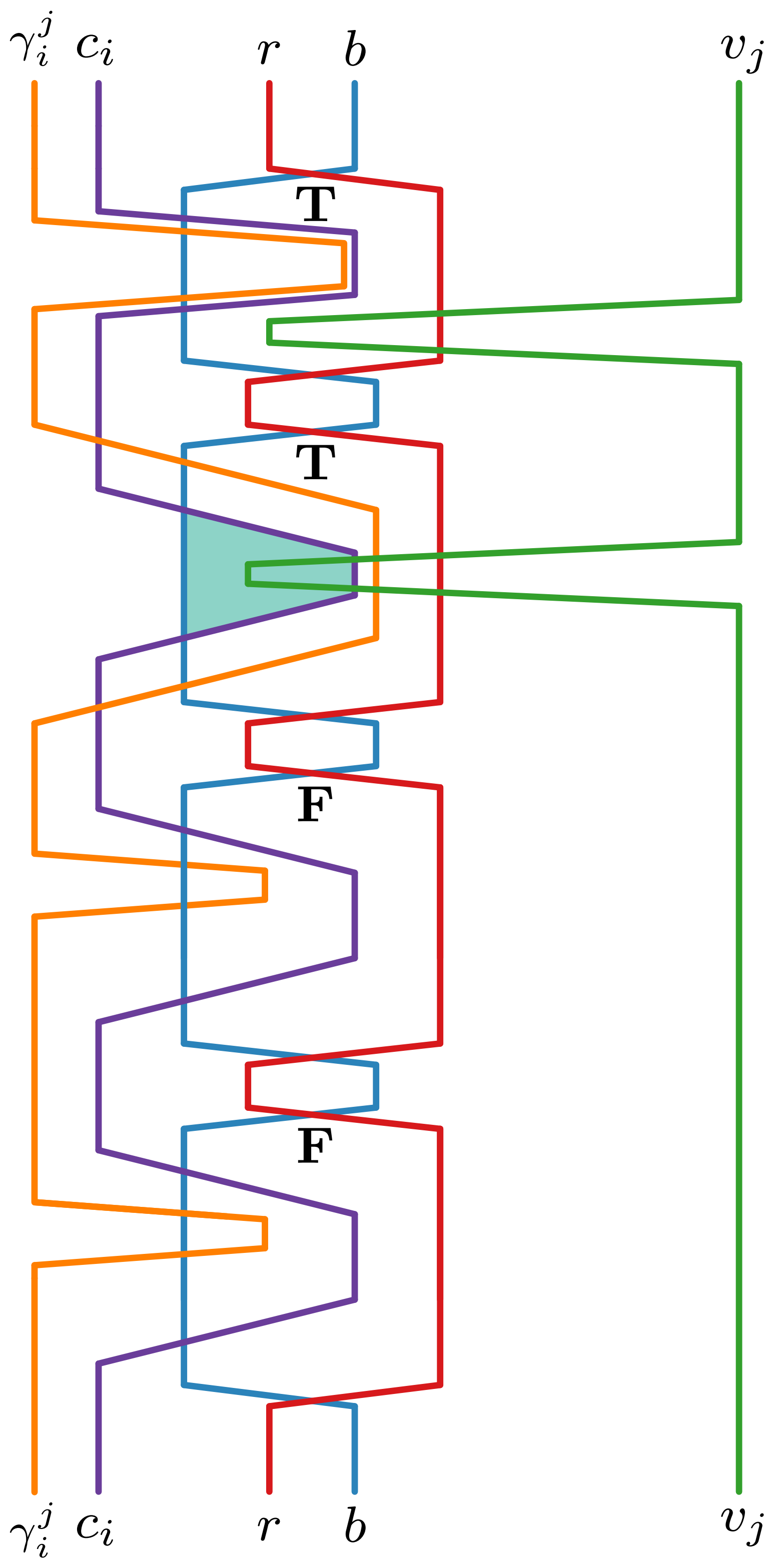
r, b: central loop structure

c_i: clause wire of i-th clause

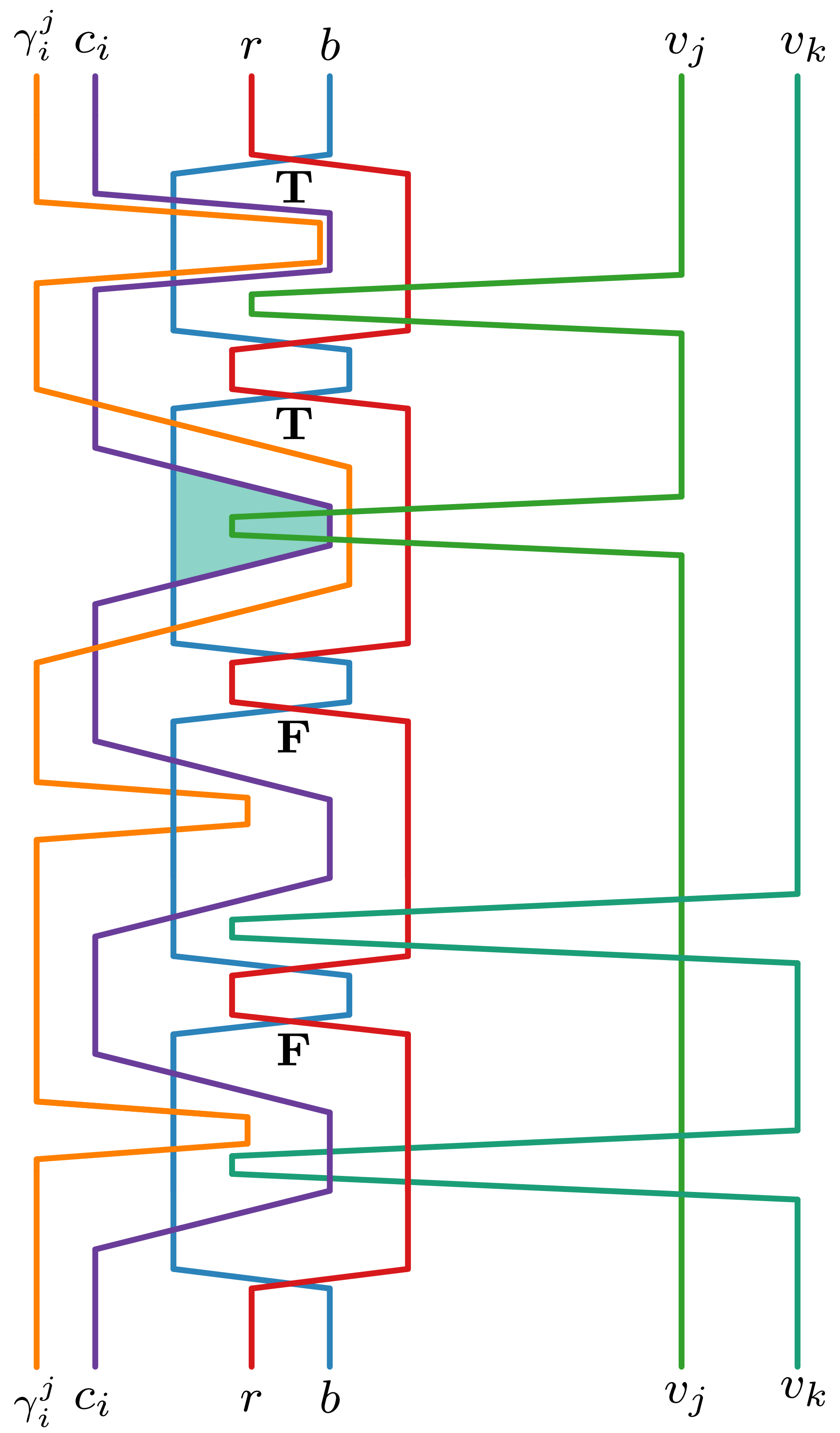
v_j: variable wire of j-th variable

γ_i^j: protects the arm of c_i that intersects v_j from other variable wires

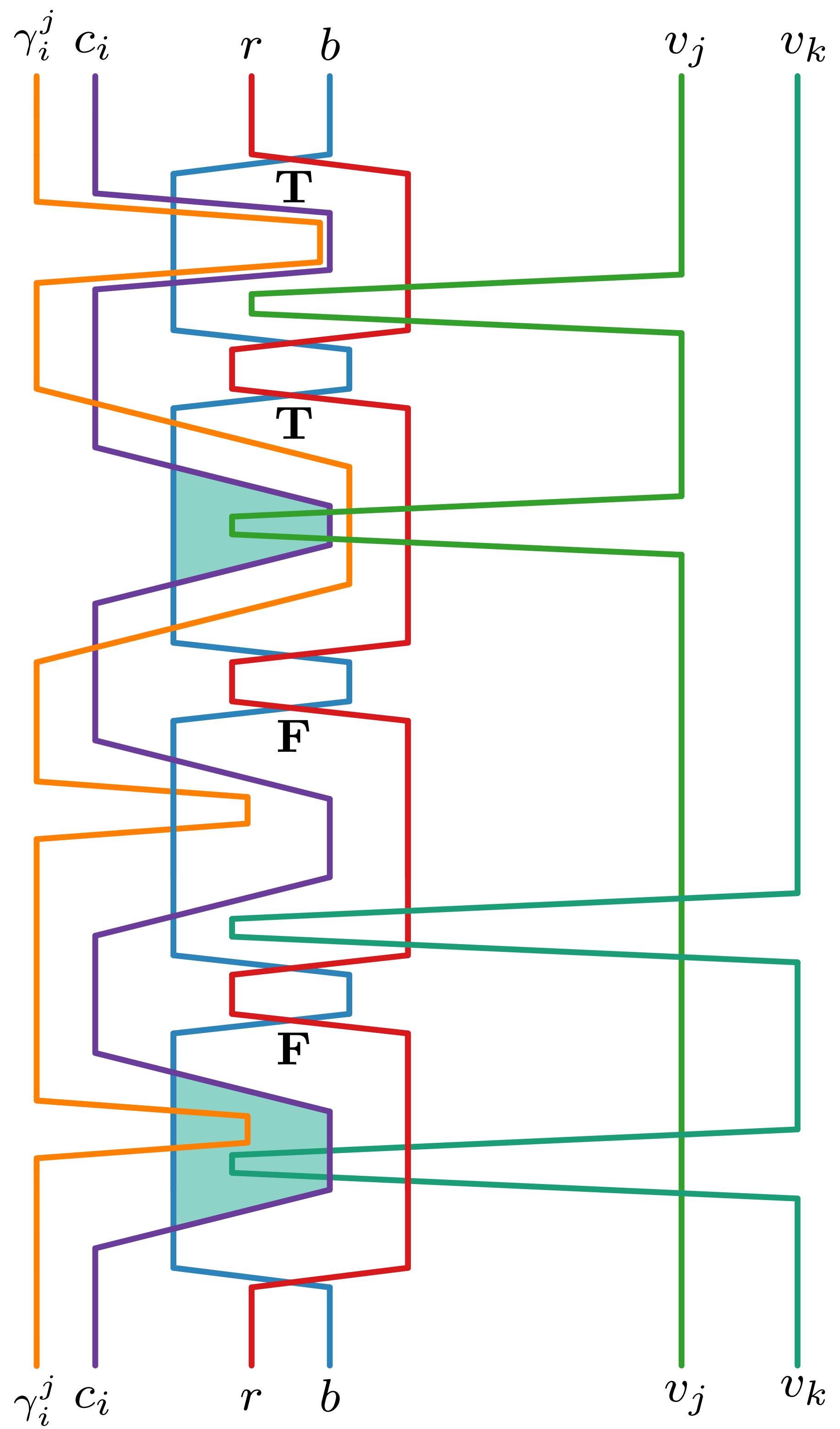
Clause Gadget



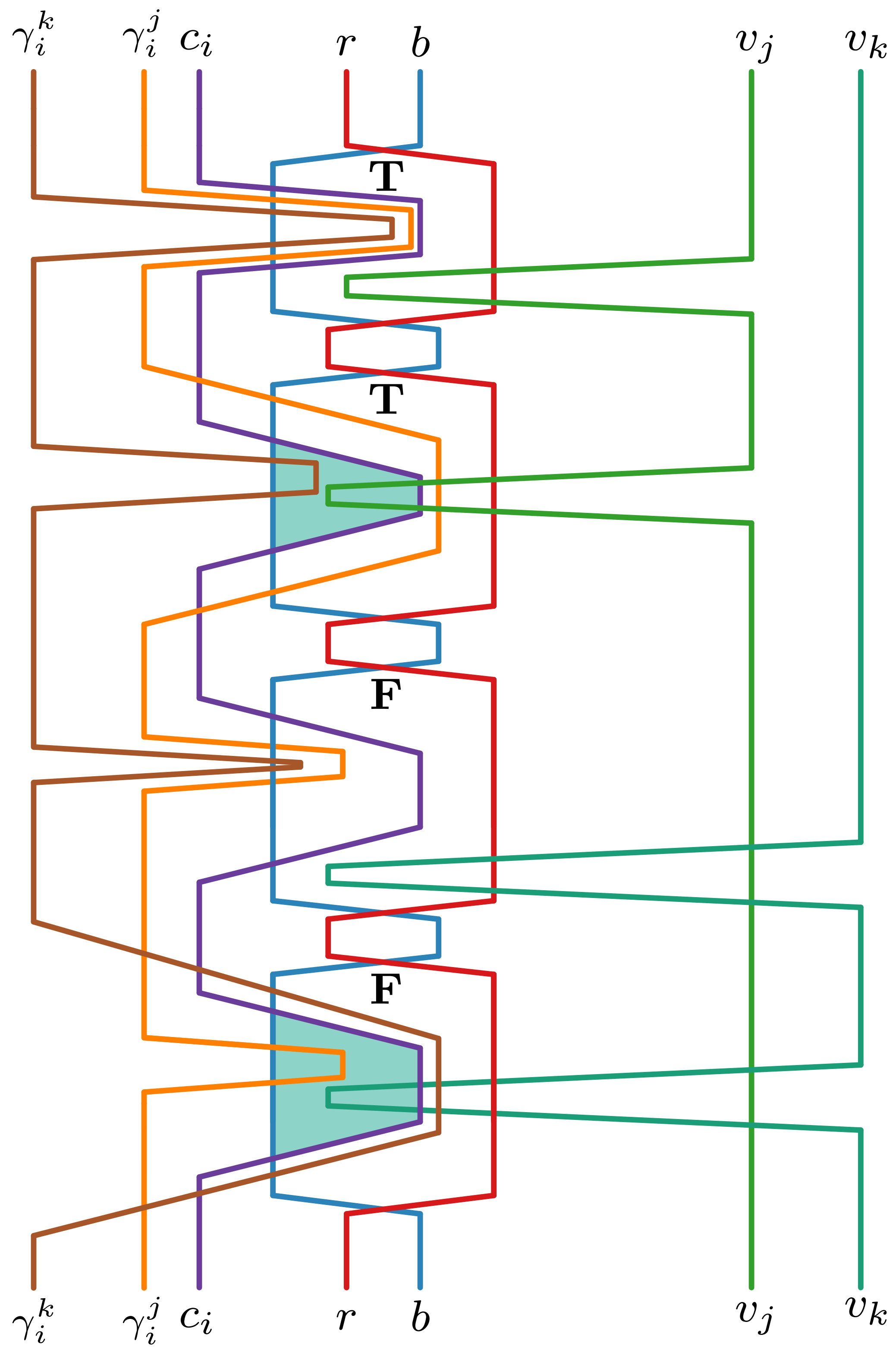
Clause Gadget



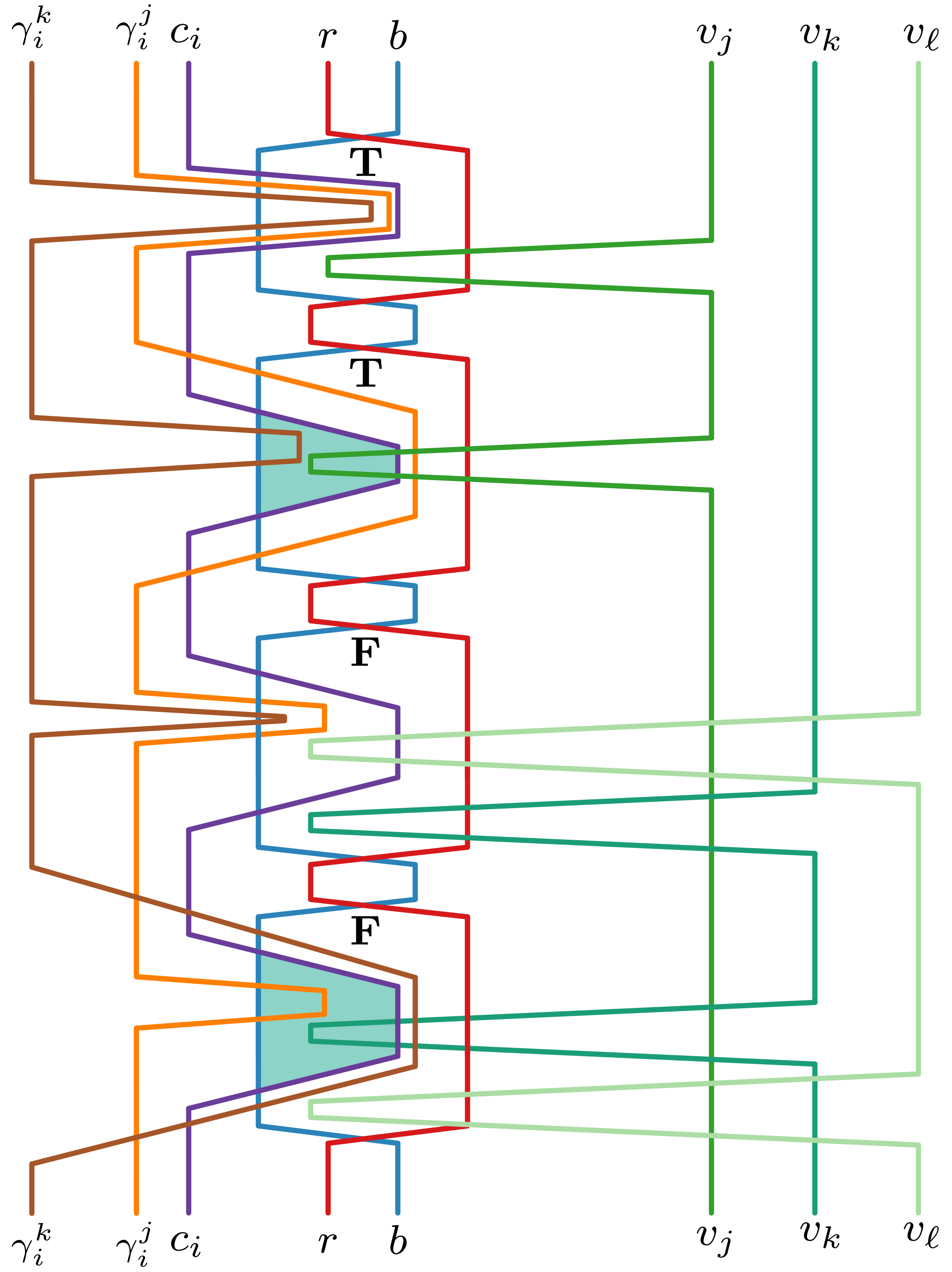
Clause Gadget



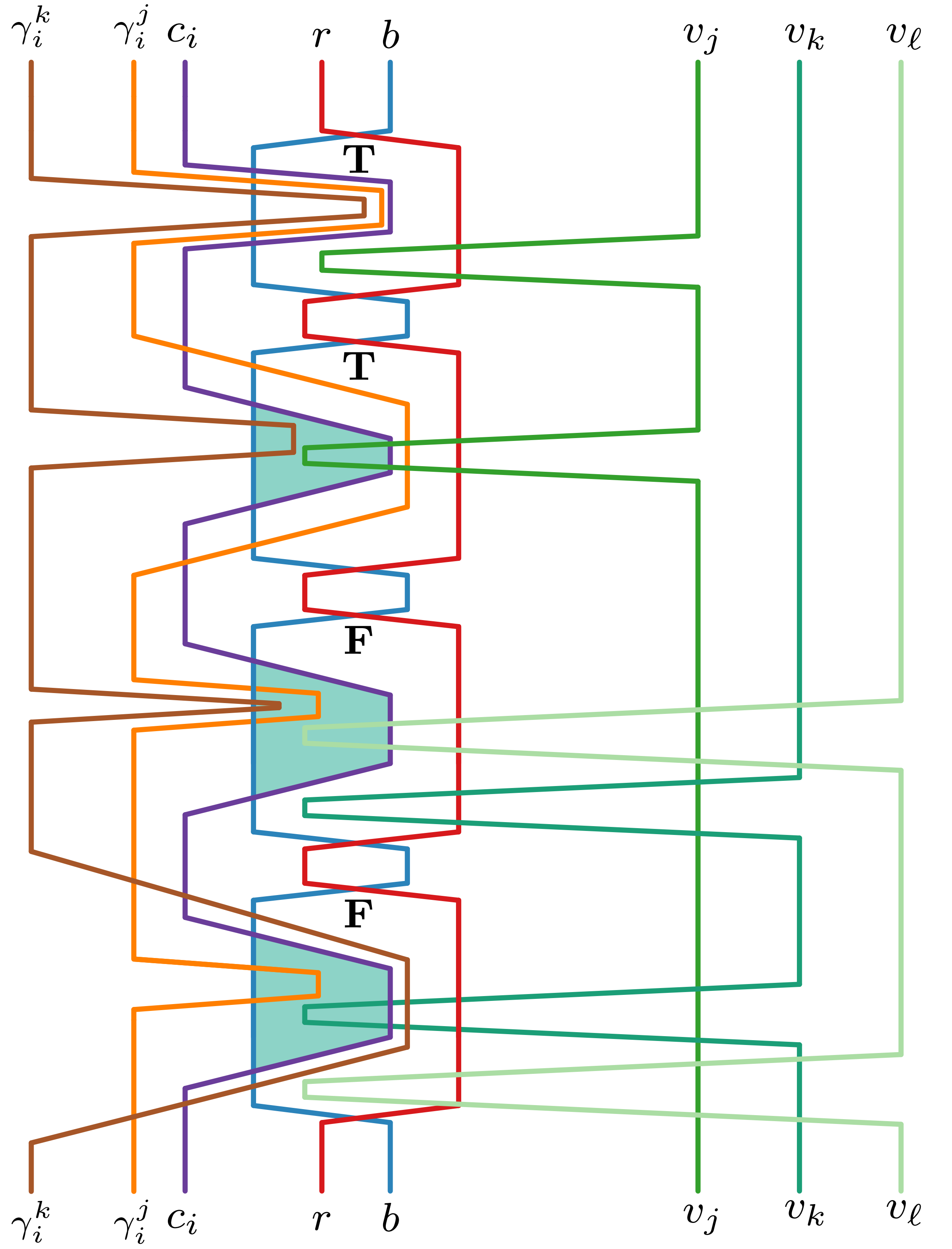
Clause Gadget



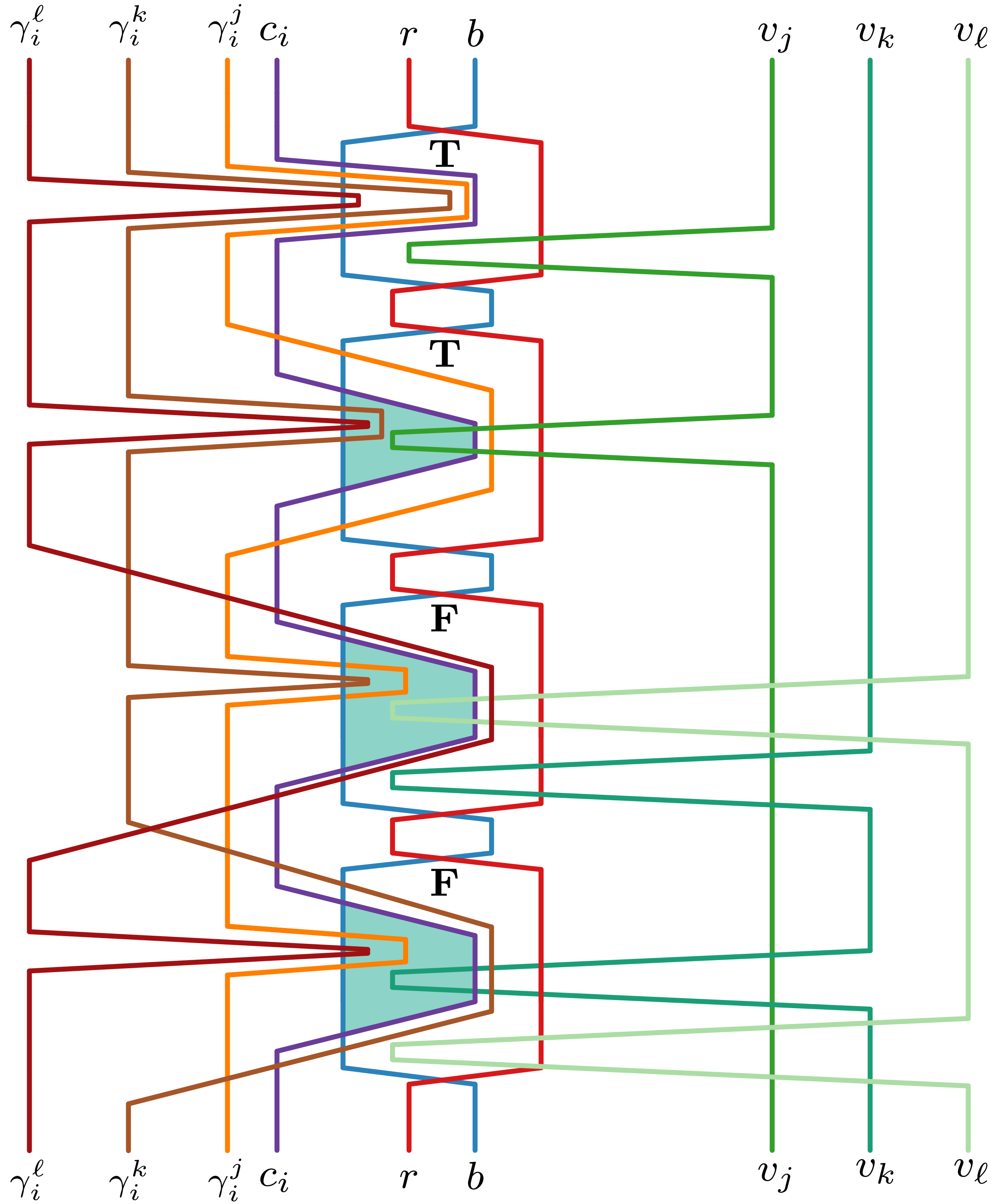
Clause Gadget



Clause Gadget



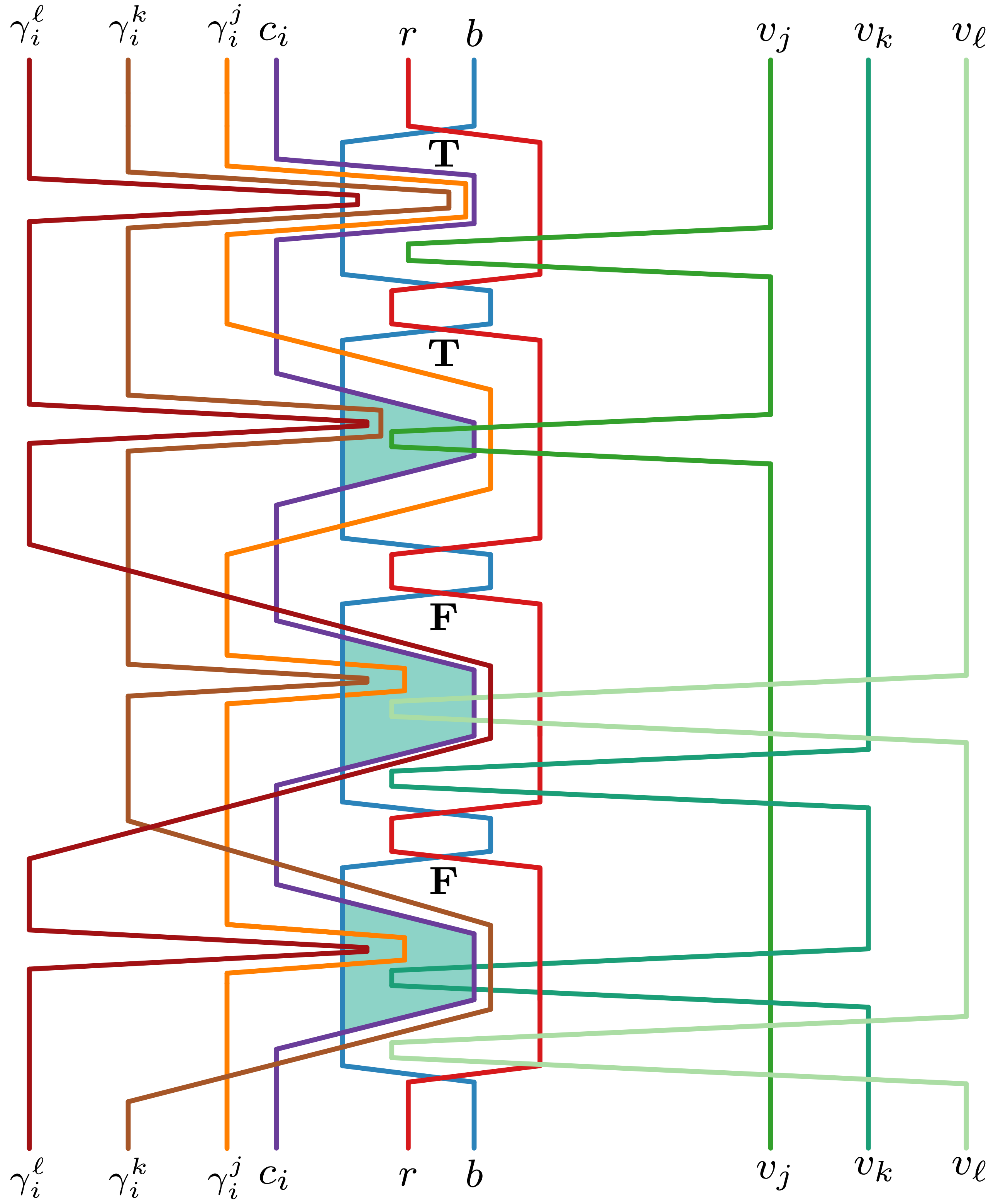
Clause Gadget



Clause Gadget

Hence:

Only 2 *true* loops and 2 *false* loops
$\Rightarrow$ clause wires meet all their
variable wires iff
Positive Not-All-Equal 3-SAT Diff
formula is satisfiable.



Our contribution

We consider the **feasibility** problem – whether a given list has a tangle.

- **Complexity:** improve the NP-hardness result of [Yamanaka et al., CCCG'18]

  Given a list of swaps, it is NP-complete to decide if it is feasible *even if every pair of wires has* $O(1)$ *(eight) swaps*.

- **Exp.-Time Algorithm:** can check feasibility faster than finding optimal-height tangles via [FKWRZ, GD'19]

- **FPT Algorithm** parametrized by the number of wires

Our contribution

We consider the **feasibility** problem – whether a given list has a tangle.

- **Complexity:** improve the NP-hardness result of [Yamanaka et al., CCCG'18]

  Given a list of swaps, it is NP-complete to decide if it is feasible *even if every pair of wires has* $O(1)$ *(eight) swaps.*

- **Exp.-Time Algorithm:** can check feasibility faster than finding optimal-height tangles via [FKWRZ, GD'19]

- **FPT Algorithm** parametrized by the number of wires

Dynamic Programming Algorithm

$O\Big(\lambda \varphi^n n \log |L|\Big)$ $\longrightarrow$ $O\Big(\lambda n^3 \log |L|\Big)$

[FKWRZ, GD'19]

Dynamic Programming Algorithm

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$O\Big(\lambda\varphi^n n \log|L|\Big) \longrightarrow O\Big(\lambda n^3 \log|L|\Big)$

[FKWRZ, GD'19]

Dynamic Programming Algorithm

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$\lambda = \#$ of **distinct sublists** of L.

$O\Big(\lambda\varphi^n n \log|L|\Big) \longrightarrow O\Big(\lambda n^3 \log|L|\Big)$

[FKWRZ, GD'19]

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

Dynamic Programming Algorithm

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$\lambda = \#$ of **distinct sublists** of L.

φ is the golden ration (≈ 1.618).

$O\Big(\lambda \varphi^n n \log |L|\Big) \longrightarrow O\Big(\lambda n^3 \log |L|\Big)$

[FKWRZ, GD'19]

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

Dynamic Programming Algorithm

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$\lambda = \#$ of **distinct sublists** of L.

φ is the golden ration (≈ 1.618).

$|L| = \sum l_{ij}$.

$O\Big(\lambda \varphi^n n \log |L|\Big) \longrightarrow O\Big(\lambda n^3 \log |L|\Big)$

[FKWRZ, GD'19]

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

Dynamic Programming Algorithm

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$O\Big(\lambda n^3 \log |L|\Big)$

$\lambda = \#$ of **distinct sublists** of L.

Consider them in order of **increasing length**.

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

Dynamic Programming Algorithm

$O\left(\lambda n^3 \log |L|\right)$

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$\lambda = \#$ of **distinct sublists** of L.

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

Consider them in order of **increasing length**.

Create a **Boolean table** with one entry for each sublist L':
`true` iff L' is feasible.

Dynamic Programming Algorithm

$O\Big(\lambda n^3 \log |L|\Big)$

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$\lambda = \#$ of **distinct sublists** of L.

Consider them in order of **increasing length**.

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

Create a **Boolean table** with one entry for each sublist L': `true` iff L' is feasible.

Base case: the empty list is feasible.

Dynamic Programming Algorithm

$O\left(\lambda n^3 \log |L|\right)$

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$\lambda = \#$ of **distinct sublists** of L.

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

Consider them in order of **increasing length**.

Create a **Boolean table** with one entry for each sublist L':
`true` iff L' is feasible.

Base case: the empty list is feasible.

Let L' be the next list to consider.

Dynamic Programming Algorithm

$O\left(\lambda n^3 \log |L|\right)$

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$\lambda = \#$ of **distinct sublists** of L.

Consider them in order of **increasing length**.

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

Create a **Boolean table** with one entry for each sublist L': `true` iff L' is feasible.

Note that each layer has exactly one swap.

Base case: the empty list is feasible.

Let L' be the next list to consider.

For each swap ij in L', check if there is a tangle realizing L' such that ij is **the last swap**.

Dynamic Programming Algorithm

$O\left(\lambda n^3 \log |L|\right)$

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$\lambda = \#$ of **distinct sublists** of L.

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

Consider them in order of **increasing length**.

Create a **Boolean table** with one entry for each sublist L': `true` iff L' is feasible.

Note that each layer has exactly one swap.

Base case: the empty list is feasible.

Let L' be the next list to consider.

For each swap ij in L', check if there is a tangle realizing L' such that ij is **the last swap**.
yes – `true`, no – `false`

Dynamic Programming Algorithm

Let $L = (\ell_{ij})$ be the given list of swaps
with n wires.
$\lambda = \#$ of **distinct sublists** of L.

$O\Big(\lambda n^3 \log |L|\Big)$

Consider them in order of **increasing length**.

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

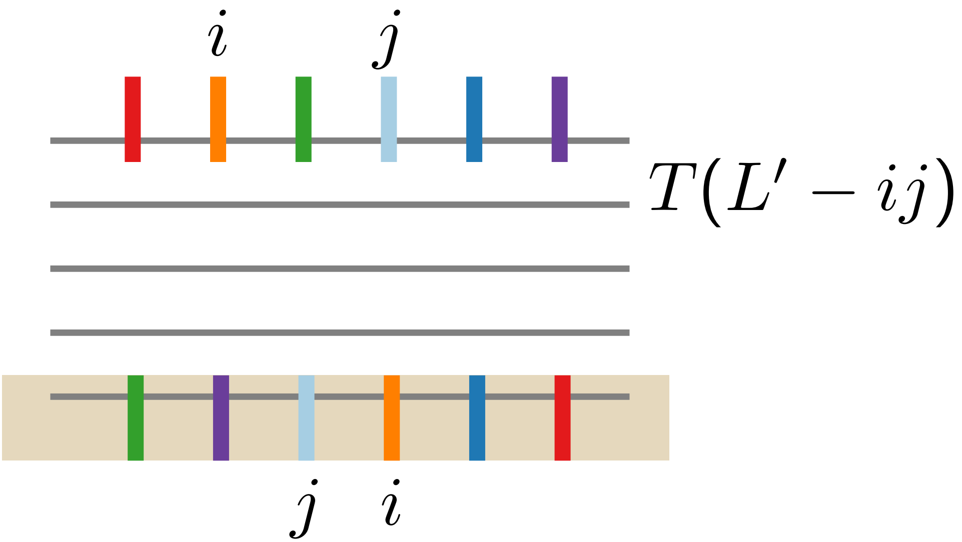
Create a **Boolean table** with one entry for each sublist L':
`true` iff L' is feasible.

Note that each layer has exactly one swap.

Base case: the empty list is feasible.

Let L' be the next list to consider.

For each swap ij in L', check if there is a tangle realizing L' such that ij is **the last swap**.
yes – `true`, no – `false`



Dynamic Programming Algorithm

$O\left(\lambda n^3 \log |L|\right)$

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$\lambda = \#$ of **distinct sublists** of L.

Consider them in order of **increasing length**.

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

Create a **Boolean table** with one entry for each sublist L': `true` iff L' is feasible.

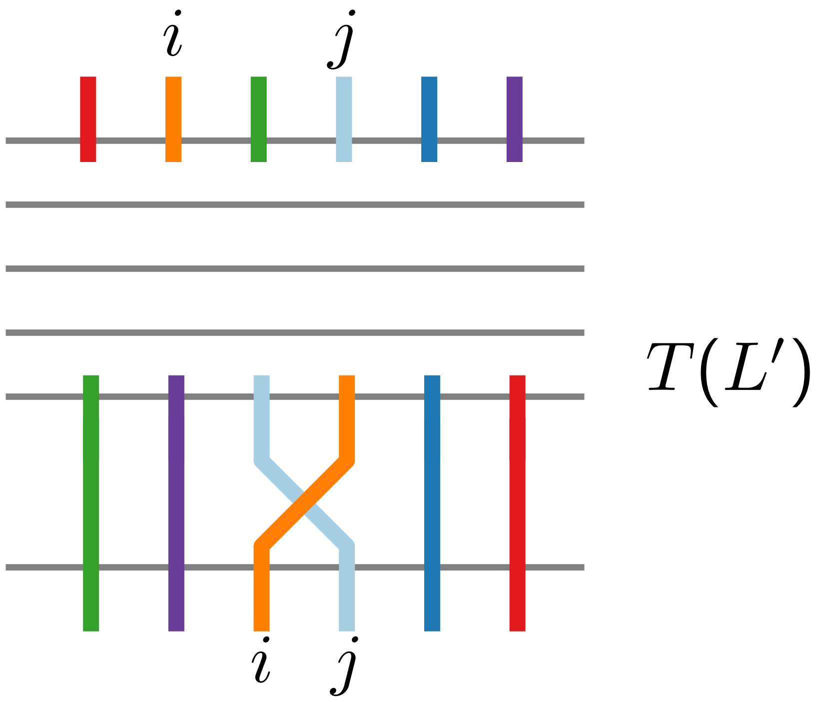
Note that each layer has exactly one swap.

Base case: the empty list is feasible.

Let L' be the next list to consider.

For each swap ij in L', check if there is a tangle realizing L' such that ij is **the last swap**.

yes – `true`, no – `false`



Dynamic Programming Algorithm

$O\left(\lambda n^3 \log |L|\right)$

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$\lambda = \#$ of **distinct sublists** of L.

Consider them in order of **increasing length**.

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

Create a **Boolean table** with one entry for each sublist L': `true` iff L' is feasible.

Note that each layer has exactly one swap.

Base case: the empty list is feasible.

Let L' be the next list to consider.

For each swap ij in L', check if there is a tangle realizing L' such that ij is **the last swap**.

Running time

$O($

Dynamic Programming Algorithm

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$O\left(\lambda n^3 \log |L|\right)$

$\lambda = \#$ of **distinct sublists** of L.

Consider them in order of **increasing length**.

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

Create a **Boolean table** with one entry for each sublist L': `true` iff L' is feasible.

Note that each layer has exactly one swap.

Base case: the empty list is feasible.

Let L' be the next list to consider.

For each swap ij in L', check if there is a tangle realizing L' such that ij is **the last swap**.

Running time

$O(\lambda)$

Dynamic Programming Algorithm

$O\left(\lambda n^3 \log |L|\right)$

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$\lambda = \#$ of **distinct sublists** of L.

Consider them in order of **increasing length**.

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

Create a **Boolean table** with one entry for each sublist L': `true` iff L' is feasible.

Note that each layer has exactly one swap.

Base case: the empty list is feasible.

Let L' be the next list to consider.

For each swap ij in L', check if there is a tangle realizing L' such that ij is **the last swap**.

Running time

$O(\lambda) \cdot O(n^2)$

Dynamic Programming Algorithm

$O\left(\lambda n^3 \log |L|\right)$

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$\lambda = \#$ of **distinct sublists** of L.

Consider them in order of **increasing length**.

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

Create a **Boolean table** with one entry for each sublist L': `true` iff L' is feasible.

Note that each layer has exactly one swap.

Base case: the empty list is feasible.

Let L' be the next list to consider.

For each swap ij in L', check if there is a tangle realizing L' such that ij is **the last swap**.

Running time

$O(\lambda) \cdot O(n^2) \cdot O(n \log |L|)$

Dynamic Programming Algorithm

$O\left(\lambda n^3 \log |L|\right)$

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$\lambda = \#$ of **distinct sublists** of L.

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

Consider them in order of **increasing length**.

Create a **Boolean table** with one entry for each sublist L': `true` iff L' is feasible.

Note that each layer has exactly one swap.

Base case: the empty list is feasible.

Let L' be the next list to consider.

For each swap ij in L', check if there is a tangle realizing L' such that ij is **the last swap**.

Running time

$$\lambda = \prod_{i<j} (\ell_{ij} + 1) \leq \left(\frac{2|L|}{n^2} + 1\right)^{n^2/2}$$

$O(\lambda) \cdot O(n^2) \cdot O(n \log |L|)$

Dynamic Programming Algorithm

Let $L = (\ell_{ij})$ be the given list of swaps with n wires.

$\lambda = \#$ of **distinct sublists** of L.

Consider them in order of **increasing length**.

Create a **Boolean table** with one entry for each sublist L': `true` iff L' is feasible.

Base case: the empty list is feasible.

Let L' be the next list to consider.

For each swap ij in L', check if there is a tangle realizing L' such that ij is **the last swap**.

Running time

$O(\lambda) \cdot O(n^2) \cdot O(n \log |L|)$

$$O\Big(\big(\tfrac{2|L|}{n^2}+1\big)^{n^2/2} n^3 \log |L|\Big)$$

$L' = (\ell'_{ij})$ is a *sublist* of L if $\ell'_{ij} \leq \ell_{ij}$

Note that each layer has exactly one swap.

$$\lambda = \prod_{i<j} (\ell_{ij}+1) \leq \left(\frac{2|L|}{n^2}+1\right)^{n^2/2}$$

Our contribution

We consider the **feasibility** problem – whether a given list has a tangle.

- **Complexity:** improve the NP-hardness result of [Yamanaka et al., CCCG'18]

  Given a list of swaps, it is NP-complete to decide if it is feasible
  even if every pair of wires has $O(1)$ *(eight) swaps.*

- **Exp.-Time Algorithm:** can check feasibility faster than finding optimal-height tangles via [FKWRZ, GD'19]

- **FPT Algorithm** parametrized by the number of wires

Our contribution

We consider the **feasibility** problem – whether a given list has a tangle.

- **Complexity:** improve the NP-hardness result of [Yamanaka et al., CCCG'18]

  Given a list of swaps, it is NP-complete to decide if it is feasible
  even if every pair of wires has $O(1)$ (eight) swaps.

- **Exp.-Time Algorithm:** can check feasibility faster than finding optimal-height tangles via [FKWRZ, GD'19]

- **FPT Algorithm** parametrized by the number of wires

FPT Algorithm Parametrized by the Number of Wires

$$O\left(\left(\frac{2|L|}{n^2}+1\right)^{n^2/2} n^3 \log |L|\right)$$

FPT Algorithm Parametrized by the Number of Wires

$O\Big(\big(\frac{2|L|}{n^2}+1\big)^{n^2/2} n^3 \log |L|\Big)$

$|L| =$ size of input.

FPT Algorithm Parametrized by the Number of Wires

$$O\left(\left(\frac{2|L|}{n^2}+1\right)^{n^2/2} n^3 \log |L|\right)$$

$|L|$ = size of input. Note that $|L| \geq n/2$.

FPT Algorithm Parametrized by the Number of Wires

$O\Big(\big(\frac{2|L|}{n^2}+1\big)^{n^2/2} n^3 \log |L|\Big)$

$|L|$ = size of input. Note that $|L| \geq n/2$.

We now consider n as our parameter.

FPT Algorithm Parametrized by the Number of Wires

$$O\left(\left(\frac{2|L|}{n^2}+1\right)^{n^2/2} n^3 \log |L|\right)$$

$|L| =$ size of input. Note that $|L| \geq n/2$.

We now consider n as our parameter. Need to separate n and $|L|$ in runtime!

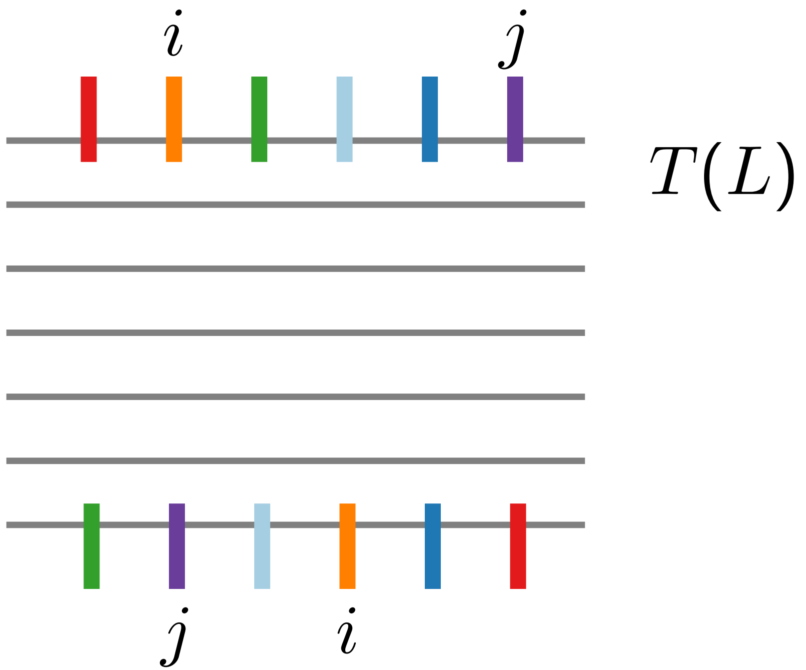
FPT Algorithm Parametrized by the Number of Wires

$O\left(\left(\frac{2|L|}{n^2}+1\right)^{n^2/2} n^3 \log |L|\right)$

$|L|$ = size of input. Note that $|L| \geq n/2$.

We now consider n as our parameter. Need to separate n and $|L|$ in runtime!

Let L be a feasible list.



FPT Algorithm Parametrized by the Number of Wires

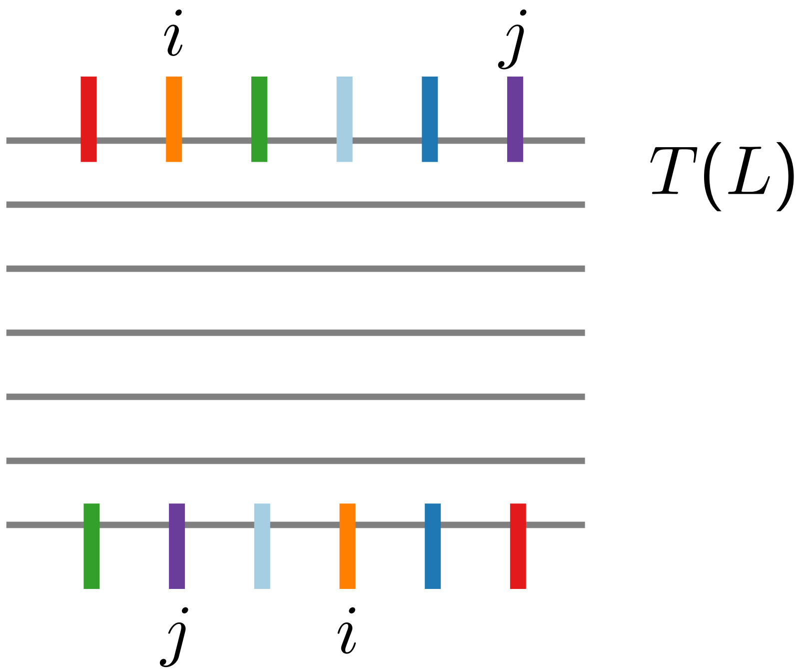
$O\Big(\big(\frac{2|L|}{n^2}+1\big)^{n^2/2} n^3 \log |L|\Big)$

$|L|$ = size of input. Note that $|L| \geq n/2$.

We now consider n as our parameter. Need to separate n and $|L|$ in runtime!

Let L be a feasible list.

Let L' be identical to L, but with two additional ij swaps.



FPT Algorithm Parametrized by the Number of Wires

$|L|$ = size of input. Note that $|L| \geq n/2$.

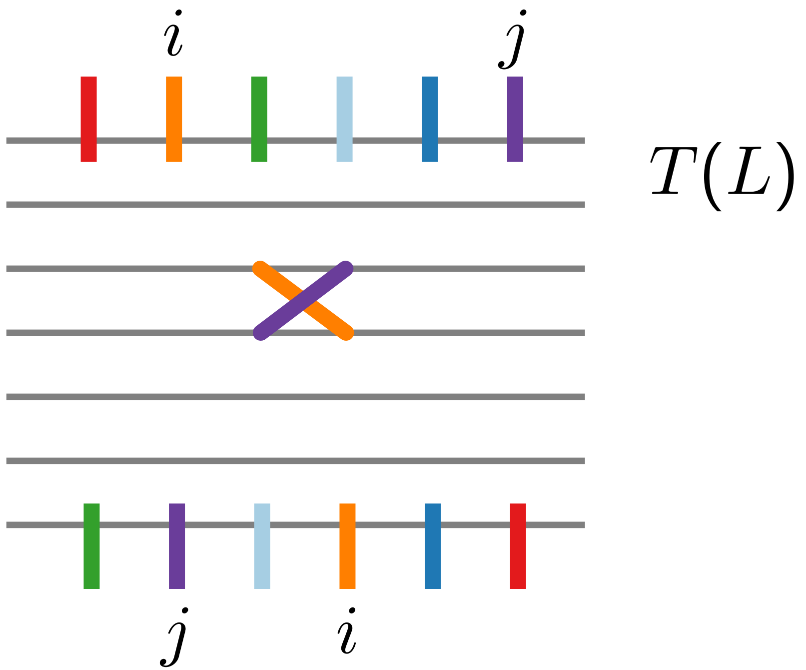
$O\left(\left(\frac{2|L|}{n^2}+1\right)^{n^2/2} n^3 \log |L|\right)$

We now consider n as our parameter. Need to separate n and $|L|$ in runtime!

Let L be a feasible list.

Let L' be identical to L, but with two additional ij swaps.

If L contains a swap ij



FPT Algorithm Parametrized by the Number of Wires

$|L| =$ size of input. Note that $|L| \geq n/2$.

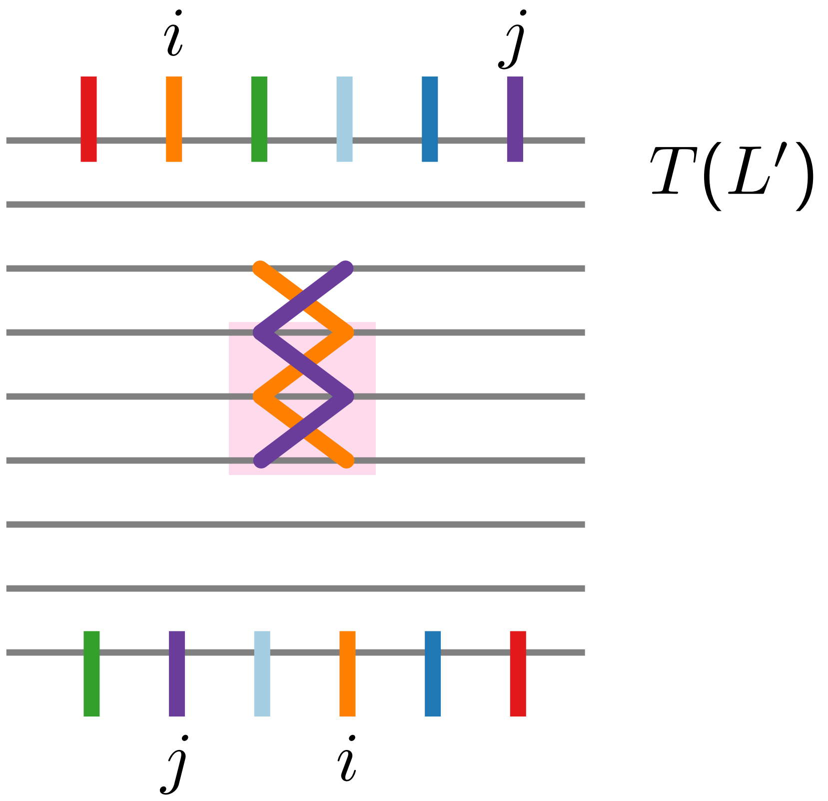
$O\Big(\big(\frac{2|L|}{n^2}+1\big)^{n^2/2} n^3 \log |L|\Big)$

We now consider n as our parameter. Need to separate n and $|L|$ in runtime!

Let L be a feasible list.

Let L' be identical to L, but with two additional ij swaps.

If L contains a swap ij



FPT Algorithm Parametrized by the Number of Wires

$|L|$ = size of input. Note that $|L| \geq n/2$.

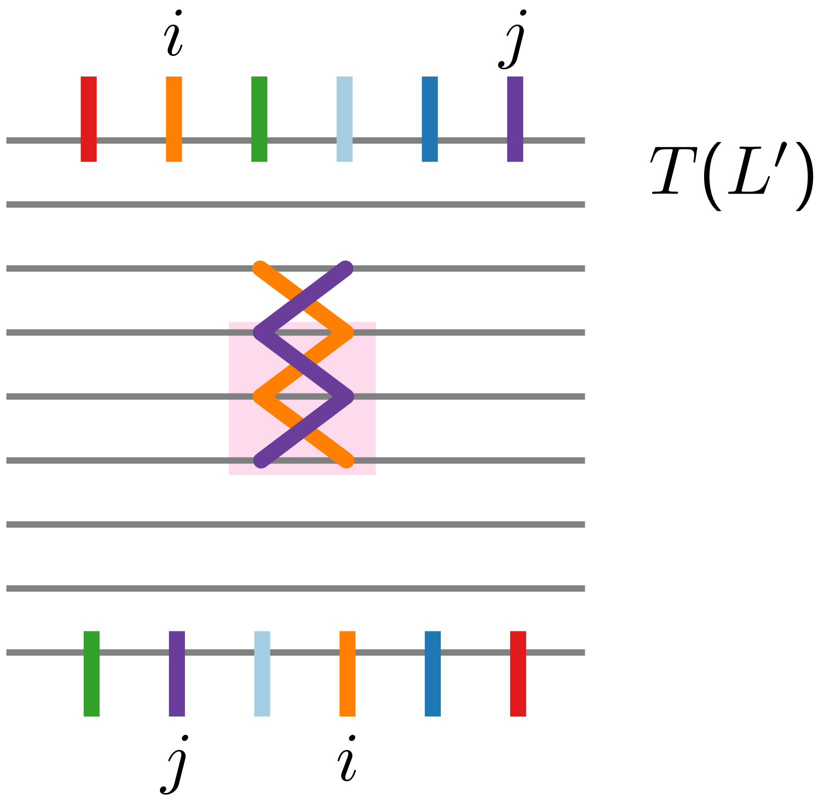
$O\left(\left(\frac{2|L|}{n^2}+1\right)^{n^2/2} n^3 \log |L|\right)$

We now consider n as our parameter. Need to separate n and $|L|$ in runtime!

Let L be a feasible list.

Let L' be identical to L, but with two additional ij swaps.

If L contains a swap ij, L' is feasible.



FPT Algorithm Parametrized by the Number of Wires

$O\left(\left(\frac{2|L|}{n^2}+1\right)^{n^2/2} n^3 \log |L|\right)$

$|L| =$ size of input. Note that $|L| \geq n/2$.

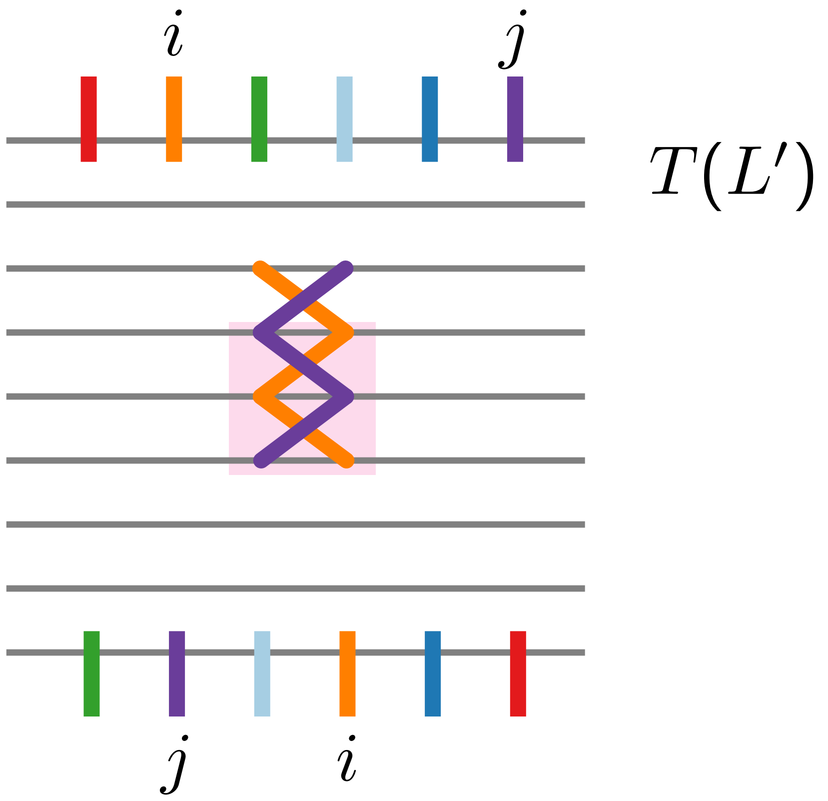
We now consider n as our parameter. Need to separate n and $|L|$ in runtime!

Let L be a feasible list.

Let L' be identical to L, but with two additional ij swaps.

If L contains a swap ij, L' is feasible.

We say that L can be *extended* to L'.



FPT Algorithm Parametrized by the Number of Wires

$O\left(\left(\frac{2|L|}{n^2}+1\right)^{n^2/2} n^3 \log |L|\right)$

$|L| =$ size of input. Note that $|L| \geq n/2$.

We now consider n as our parameter. Need to separate n and $|L|$ in runtime!

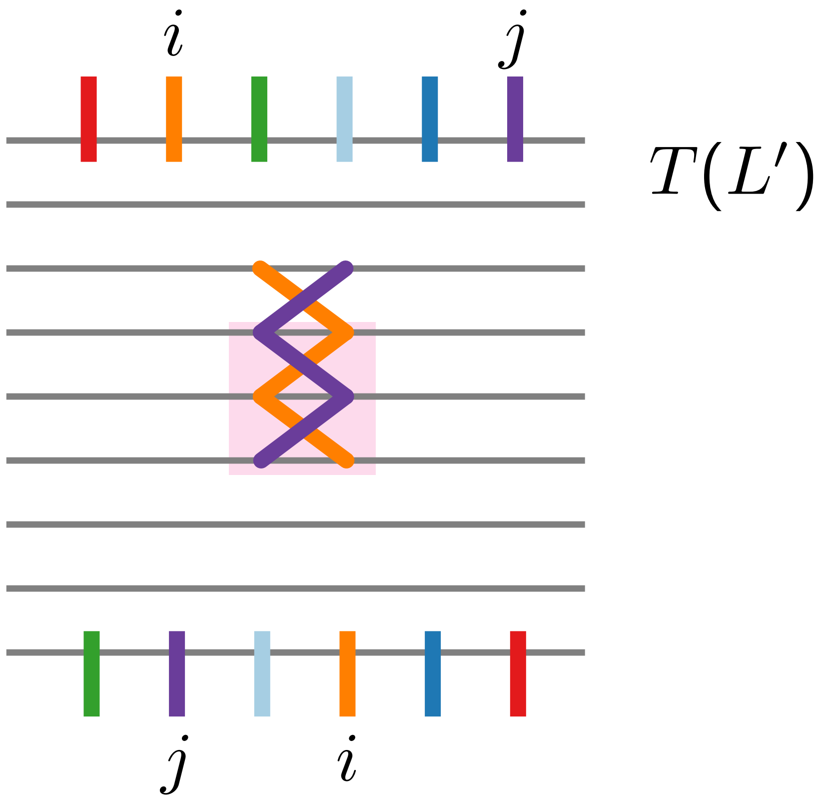
Let L be a feasible list.

Let L' be identical to L, but with two additional ij swaps.

If L contains a swap ij, L' is feasible.

We say that L can be extended to L'.

We say that a feasible list L_{min} is *minimal*
if there is no feasible list L that can be extended to L_{min}.



FPT Algorithm Parametrized by the Number of Wires

$O\Big(\big(\frac{2|L|}{n^2}+1\big)^{n^2/2} n^3 \log |L|\Big)$

$|L| =$ size of input. Note that $|L| \geq n/2$.

We now consider n as our parameter. Need to separate n and $|L|$ in runtime!

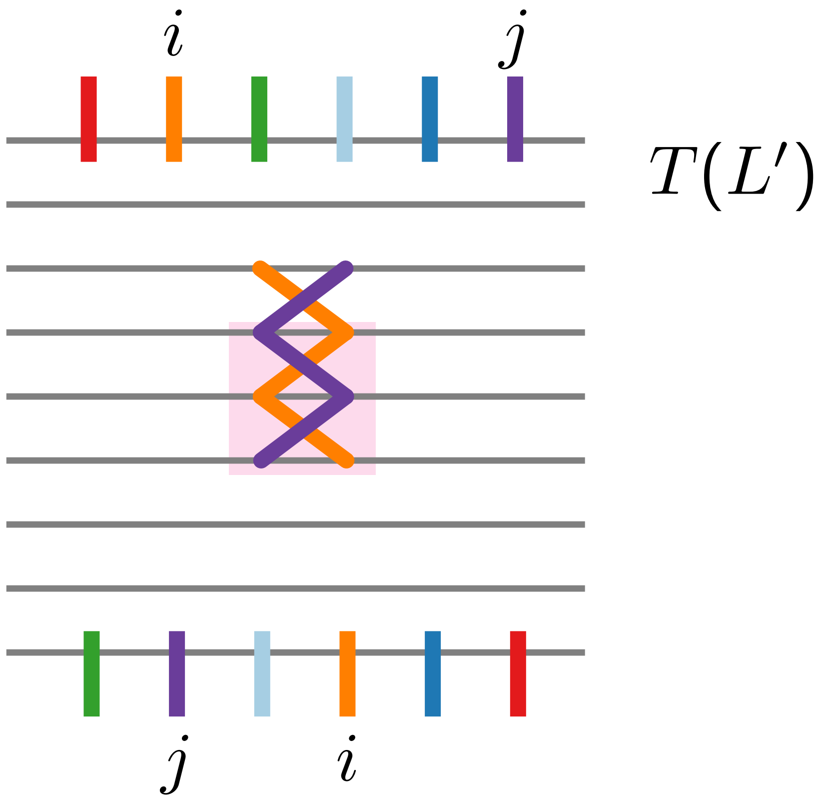
Let L be a feasible list.

Let L' be identical to L, but with two additional ij swaps.

If L contains a swap ij, L' is feasible.

We say that L can be extended to L'.

We say that a feasible list L_{min} is *minimal*
if there is no feasible list L that can be extended to L_{min}.



Lemma.
If $L = (l_{ij})$ is a minimal feasible list with n wires,
then $l_{ij} \leq n^2/4 + 1$ for each $i, j \in [n]$.

FPT Algorithm Parametrized by the Number of Wires

Theorem.
There is an FPT algorithm for LIST-FEASIBILITY w.r.t. n.

$$O\Big(\big(\tfrac{2|L|}{n^2}+1\big)^{n^2/2} n^3 \log |L|\Big)$$

$$\downarrow$$

$$O\Big(\big(\tfrac{n}{2}\big)^{n^2} \cdot n^3 \log n + n^2 \log |L|\Big)$$

FPT Algorithm Parametrized by the Number of Wires

$$O\Big(\big(\tfrac{2|L|}{n^2}+1\big)^{n^2/2} n^3 \log |L|\Big)$$

Theorem.
There is an FPT algorithm for LIST-FEASIBILITY w.r.t. n.

$$\downarrow$$

Proof idea.
Create a list $L' = (l'_{ij})$ from $L = (l_{ij})$ by setting $l'_{ij} = \min\{l_{ij}, n^2/4+1\}$.

$$O\Big(\big(\tfrac{n}{2}\big)^{n^2} \cdot n^3 \log n + n^2 \log |L|\Big)$$

FPT Algorithm Parametrized by the Number of Wires

$$O\left(\left(\frac{2|L|}{n^2}+1\right)^{n^2/2} n^3 \log |L|\right)$$

$$\downarrow$$

$$O\left(\left(\frac{n}{2}\right)^{n^2} \cdot n^3 \log n + n^2 \log |L|\right)$$

Theorem.
There is an FPT algorithm for List-Feasibility w.r.t. n.

Proof idea.
Create a list $L' = (l'_{ij})$ from $L = (l_{ij})$ by setting $l'_{ij} = \min\{l_{ij}, n^2/4 + 1\}$.

Use the algorithm described before for L'.

FPT Algorithm Parametrized by the Number of Wires

$O\left(\left(\frac{2|L|}{n^2}+1\right)^{n^2/2} n^3 \log |L|\right)$

$\downarrow$

$O\left(\left(\frac{n}{2}\right)^{n^2} \cdot n^3 \log n + n^2 \log |L|\right)$

Theorem.
There is an FPT algorithm for List-Feasibility w.r.t. n.

Proof idea.
Create a list $L' = (l'_{ij})$ from $L = (l_{ij})$ by setting $l'_{ij} = \min\{l_{ij}, n^2/4 + 1\}$.

Use the algorithm described before for L'.

Check if there is a feasible sublist of L' that can be extended to L.

FPT Algorithm Parametrized by the Number of Wires

$O\left(\left(\frac{2|L|}{n^2}+1\right)^{n^2/2} n^3 \log |L|\right)$

$\downarrow$

$O\left(\left(\frac{n}{2}\right)^{n^2} \cdot n^3 \log n + n^2 \log |L|\right)$

Theorem.
There is an FPT algorithm for LIST-FEASIBILITY w.r.t. n.

Proof idea.
Create a list $L' = (l'_{ij})$ from $L = (l_{ij})$ by setting
$l'_{ij} = \min\{l_{ij}, n^2/4 + 1\}$.

Use the algorithm described before for L'.

Check if there is a feasible sublist of L' that can be extended to L.

Running time

FPT Algorithm Parametrized by the Number of Wires

$$O\Big(\big(\tfrac{2|L|}{n^2}+1\big)^{n^2/2} n^3 \log |L|\Big)$$

$$O\Big(\big(\tfrac{n}{2}\big)^{n^2} \cdot n^3 \log n + n^2 \log |L|\Big)$$

Theorem.

There is an FPT algorithm for List-Feasibility w.r.t. n.

Proof idea.

Create a list $L' = (l'_{ij})$ from $L = (l_{ij})$ by setting $l'_{ij} = \min\{l_{ij}, n^2/4 + 1\}$.

Use the algorithm described before for L'.

Check if there is a feasible sublist of L' that can be extended to L.

Running time

$$O\Big(\big(\tfrac{2|L'|}{n^2}+1\big)^{n^2/2} n^3 \log |L'| + n^2 \log |L|\Big)$$

FPT Algorithm Parametrized by the Number of Wires

$$O\left(\left(\frac{2|L|}{n^2}+1\right)^{n^2/2} n^3 \log |L|\right)$$

Theorem.
There is an FPT algorithm for List-Feasibility w.r.t. n.

$$O\left(\left(\frac{n}{2}\right)^{n^2} \cdot n^3 \log n + n^2 \log |L|\right)$$

Proof idea.
Create a list $L' = (l'_{ij})$ from $L = (l_{ij})$ by setting $l'_{ij} = \min\{l_{ij}, n^2/4 + 1\}$.

Use the algorithm described before for L'.

Check if there is a feasible sublist of L' that can be extended to L.

Running time

$$O\left(\left(\frac{2|L'|}{n^2}+1\right)^{n^2/2} n^3 \log |L'| + n^2 \log |L|\right)$$

FPT Algorithm Parametrized by the Number of Wires

$O\Big(\big(\frac{2|L|}{n^2}+1\big)^{n^2/2} n^3 \log |L|\Big)$

Theorem.
There is an FPT algorithm for $\textsc{List-Feasibility}$ w.r.t. n.

$O\Big(\big(\frac{n}{2}\big)^{n^2} \cdot n^3 \log n + n^2 \log |L|\Big)$

Proof idea.
Create a list $L' = (l'_{ij})$ from $L = (l_{ij})$ by setting $l'_{ij} = \min\{l_{ij}, n^2/4 + 1\}$.

Use the algorithm described before for L'.

Check if there is a feasible sublist of L' that can be extended to L.

Running time

$O\Big(\big(\frac{2|L'|}{n^2}+1\big)^{n^2/2} n^3 \log |L'| + n^2 \log |L|\Big)$

FPT Algorithm Parametrized by the Number of Wires

$$O\Big(\big(\tfrac{2|L|}{n^2}+1\big)^{n^2/2} n^3 \log |L|\Big)$$

Theorem.
There is an FPT algorithm for LIST-FEASIBILITY w.r.t. n.

$$O\Big(\big(\tfrac{n}{2}\big)^{n^2} \cdot n^3 \log n + n^2 \log |L|\Big)$$

Proof idea.
Create a list $L' = (l'_{ij})$ from $L = (l_{ij})$ by setting $l'_{ij} = \min\{l_{ij}, n^2/4+1\}$.

Use the algorithm described before for L'.

Check if there is a feasible sublist of L' that can be extended to L.

Running time

$$O\Big(\big(\tfrac{2|L'|}{n^2}+1\big)^{n^2/2} n^3 \log |L'| + n^2 \log |L|\Big)$$

Lemma: $|L'| \le \binom{n}{2}\big(\frac{n^2}{4}+1\big)$

Open Problems

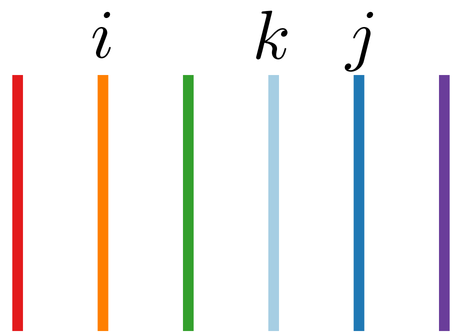
- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

Open Problems

- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

Open Problems

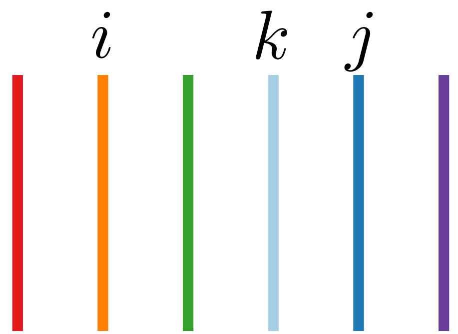


- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

Open Problems



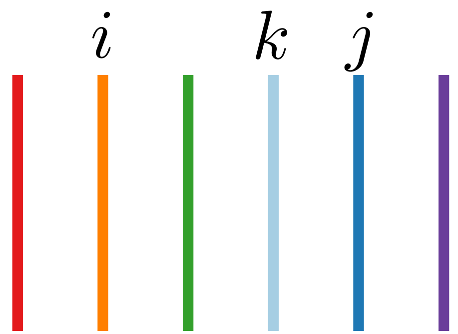
- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\left(\ell_{ik} = \ell_{kj} = 0 \text{ implies } \ell_{ij} = 0\right)$.

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

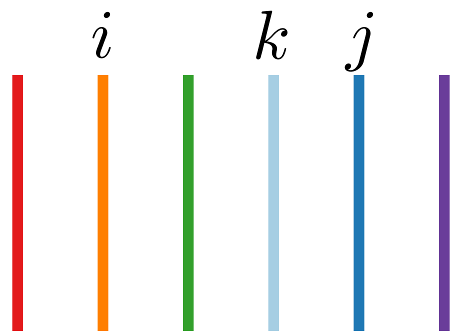
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\ 1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

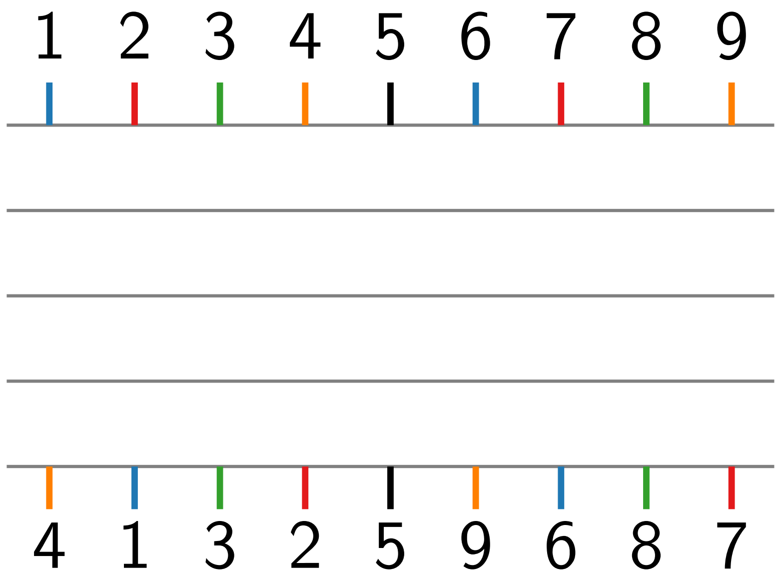
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable* if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

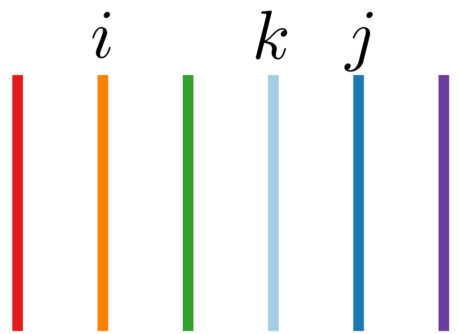
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\,1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4 6–9
2–3 7–8
2–4 7–9
3–4 8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

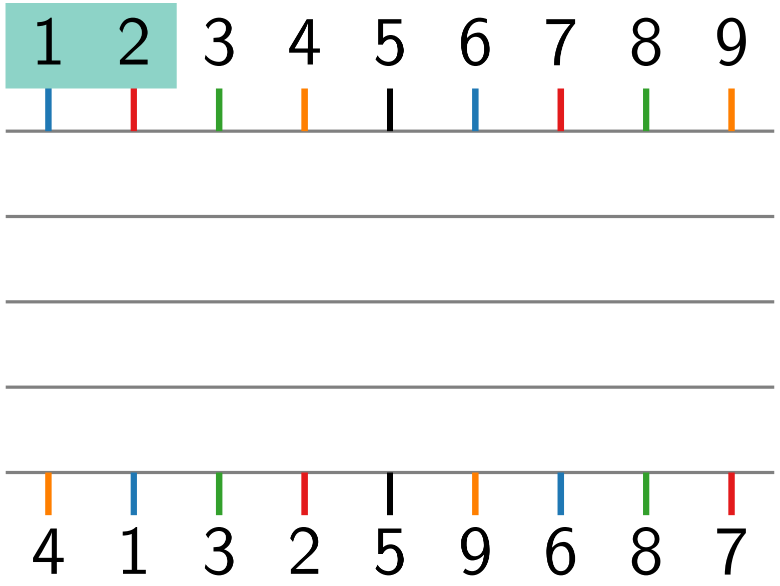
No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i{<}k{<}j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

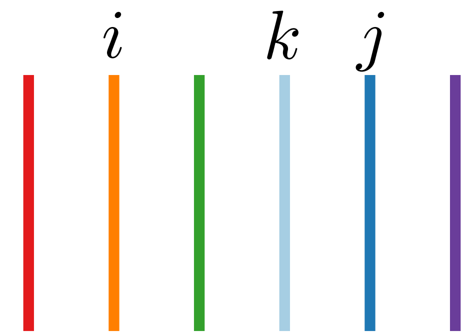
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\,1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

odd-even sort

1–4 6–9
2–3 7–8
2–4 7–9
3–4 8–9



Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

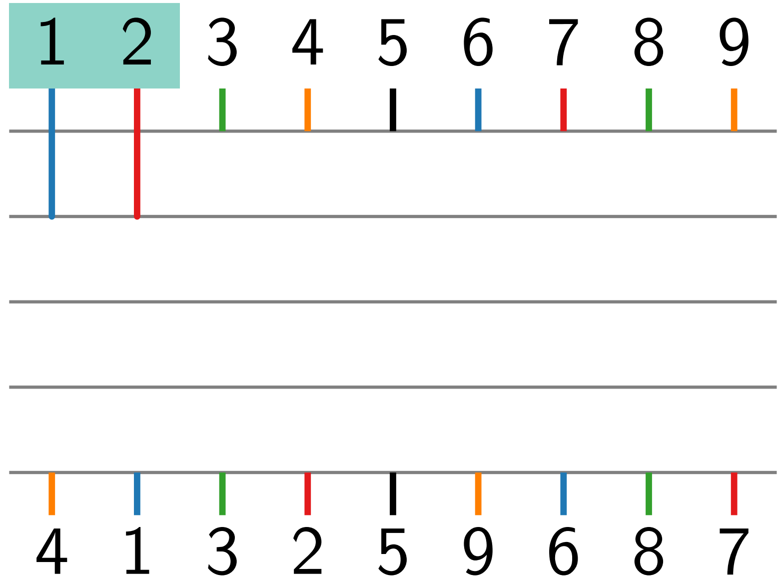
No!

A list (ℓ_{ij}) is *non-separable* if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

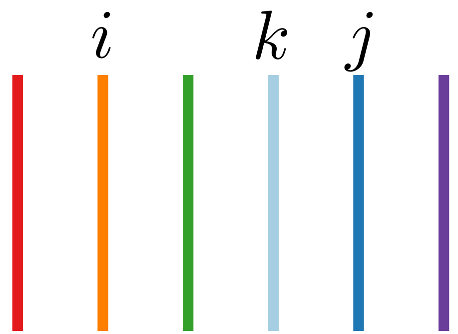
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\ 1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

odd-even sort

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

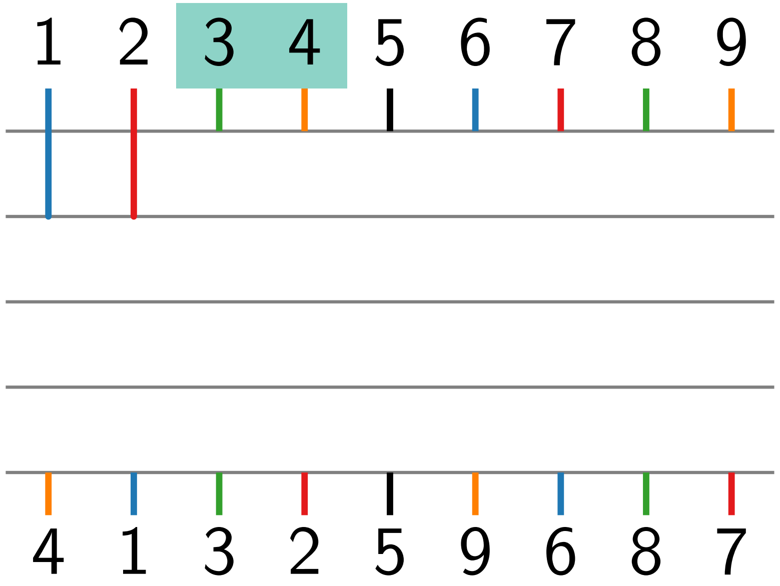
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

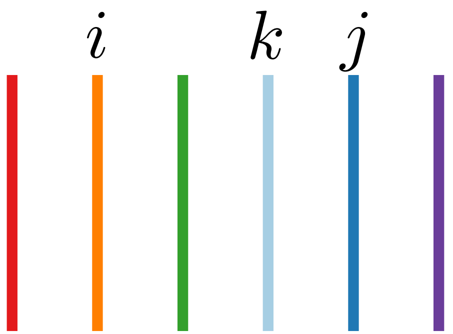
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+$ 1 in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

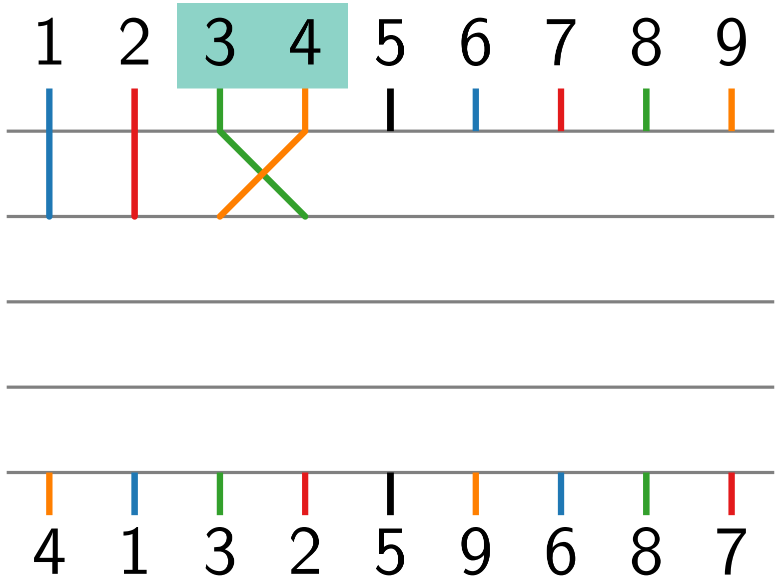
No!

A list (ℓ_{ij}) is *non-separable* if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

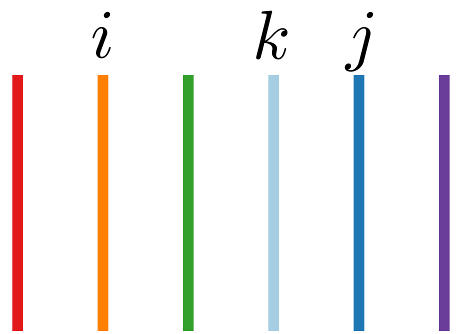
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\,1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

odd-even sort

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

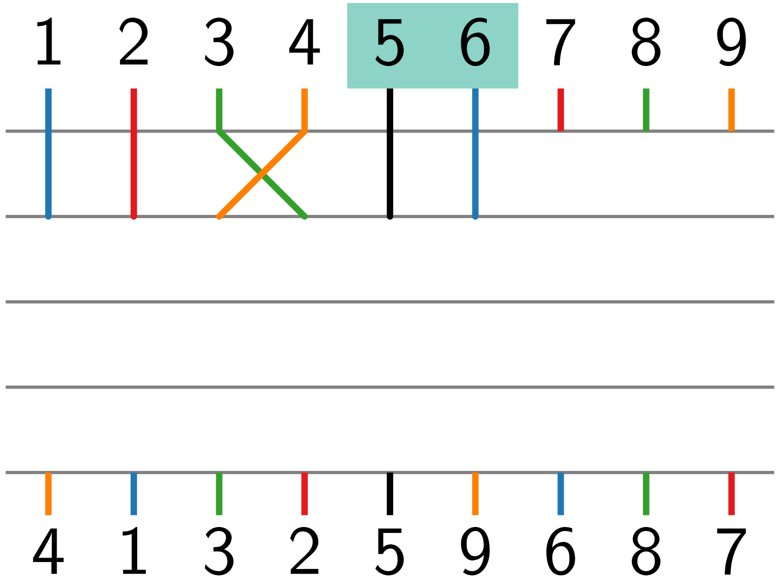
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable* if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

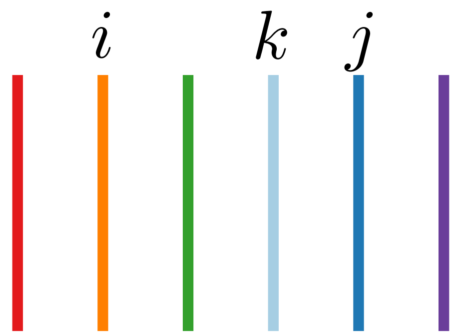
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+$ 1 in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

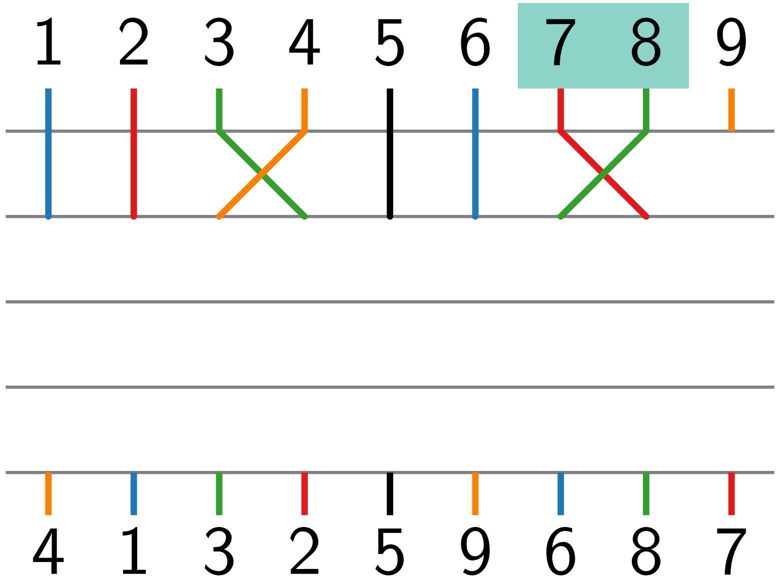
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable* if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

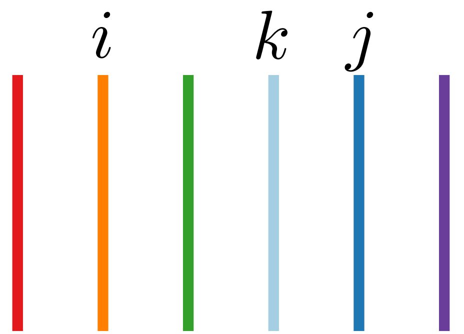
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\,1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

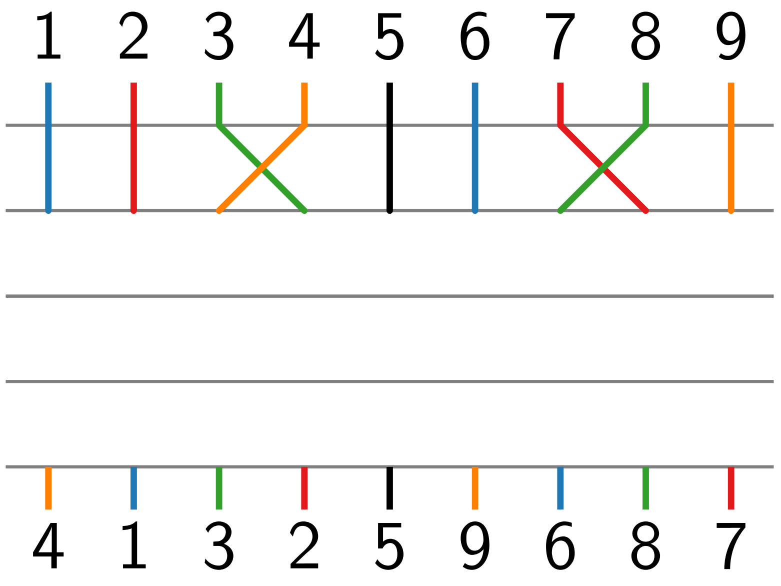
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable* if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

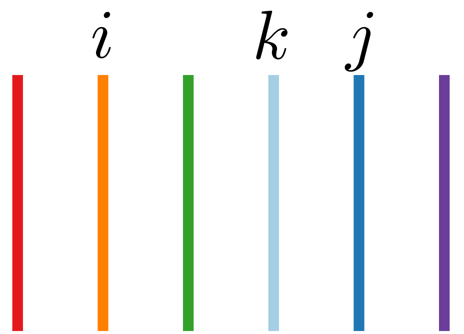
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\ 1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

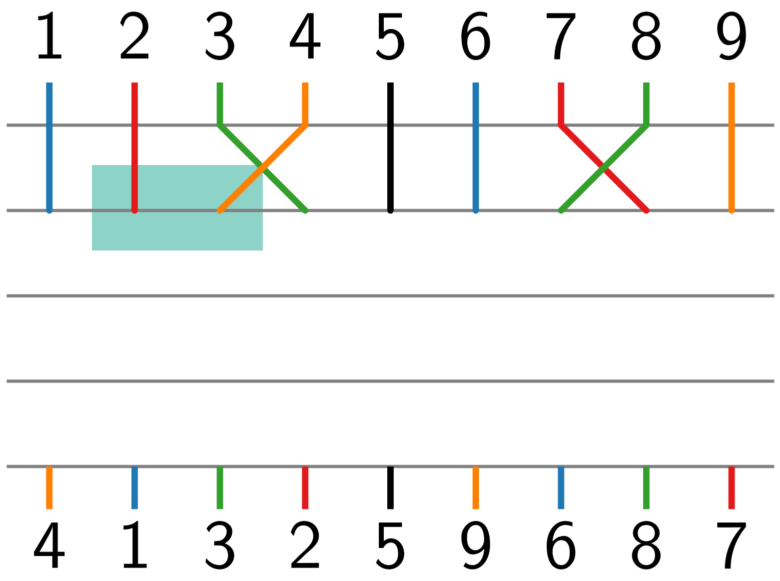
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\left(\ell_{ik} = \ell_{kj} = 0 \text{ implies } \ell_{ij} = 0\right)$.

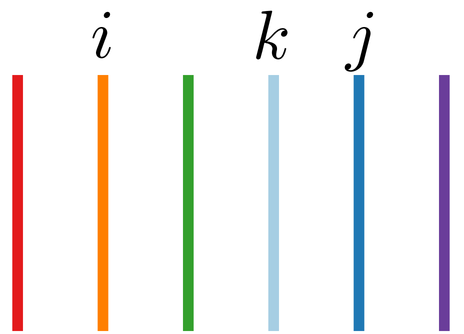
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\,1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

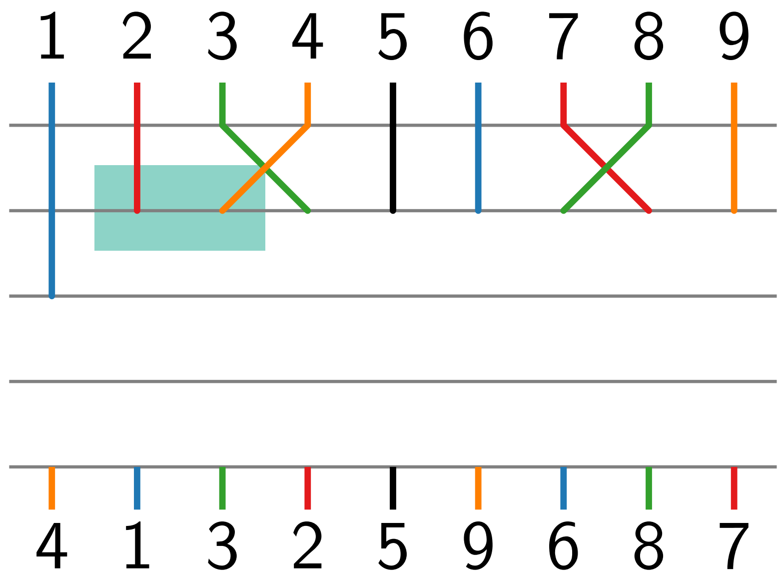
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

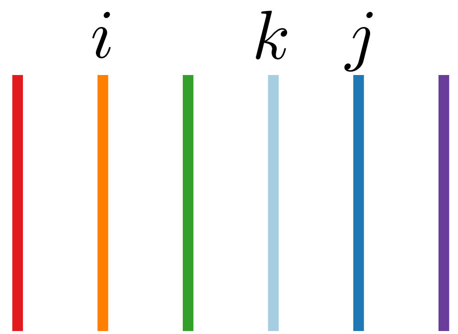
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\ 1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times. How difficult is it to check feasibility of L?

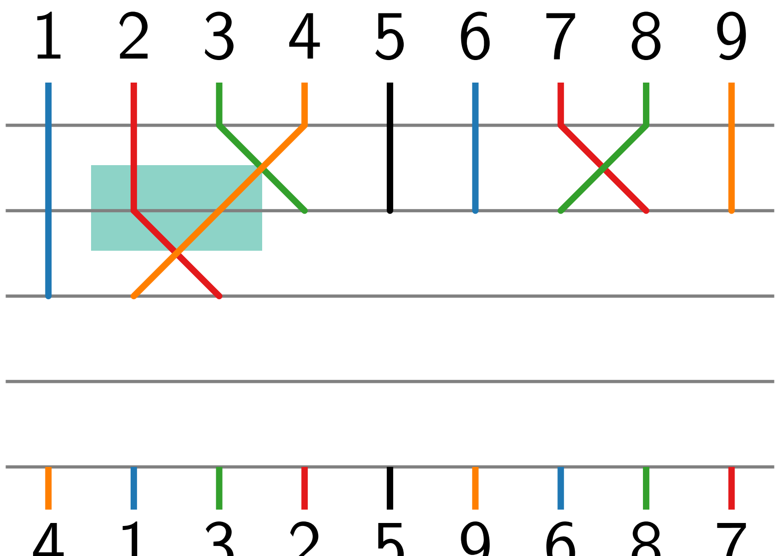
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable* if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

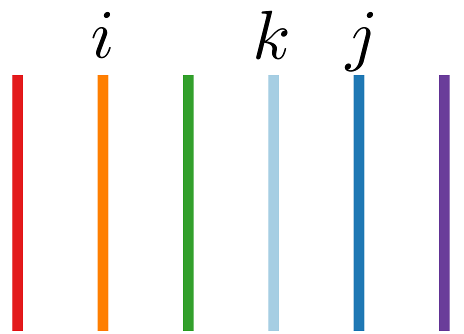
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87] We can find a tangle that has height at most OPT$+\,1$ in polynomial time. Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

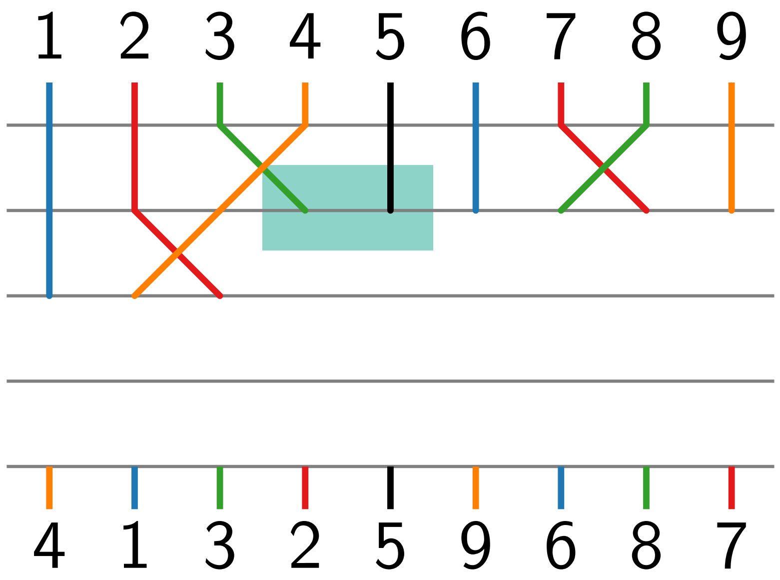
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

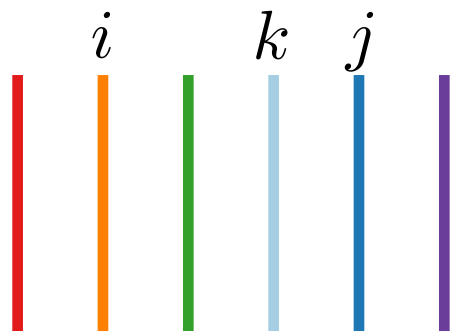
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\ 1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



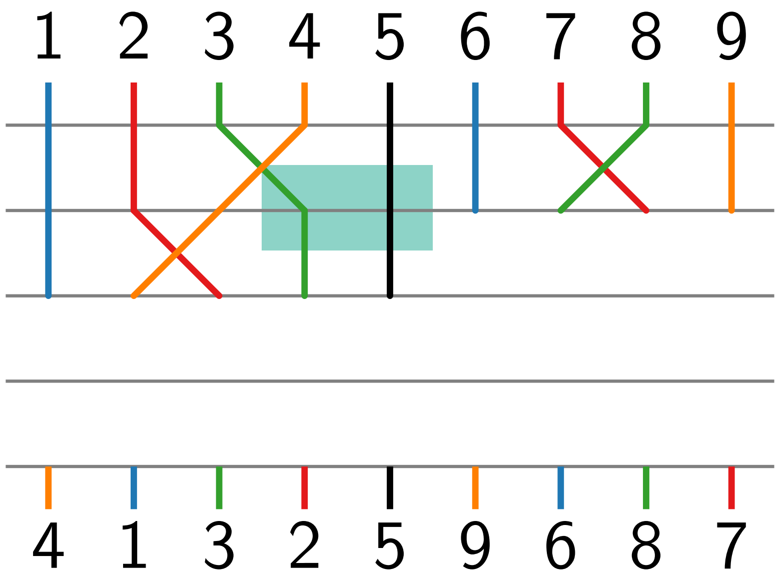
- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.
No!

A list (ℓ_{ij}) is *non-separable* if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

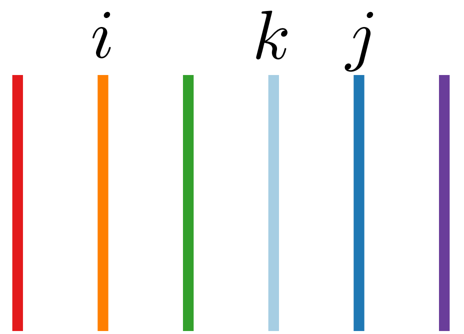
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\,1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

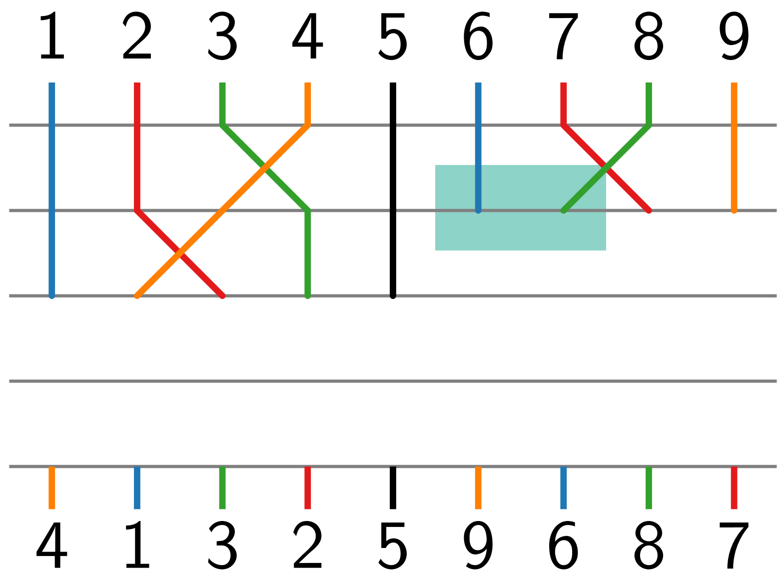
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i{<}k{<}j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

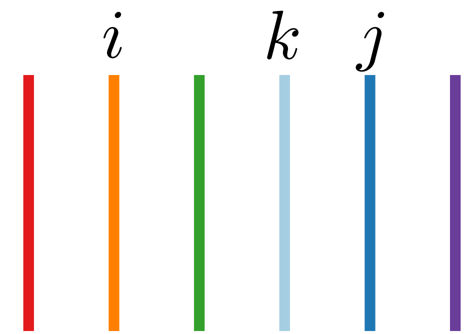
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\ 1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

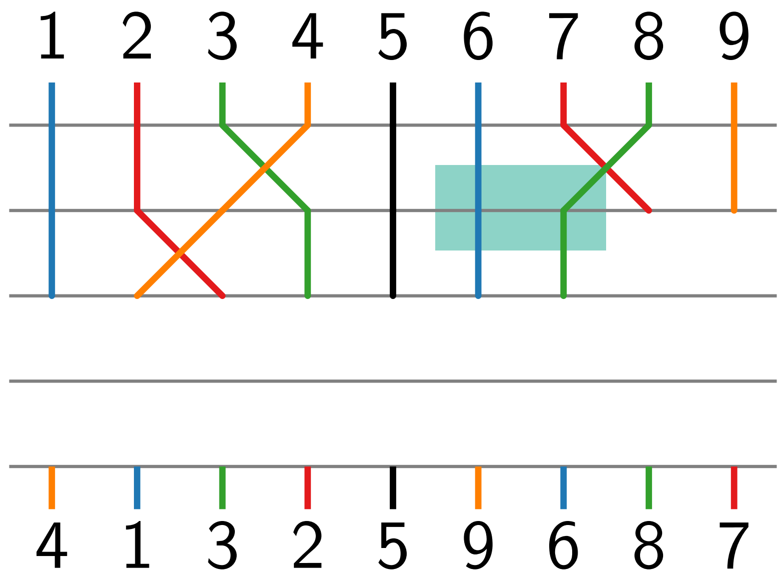
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

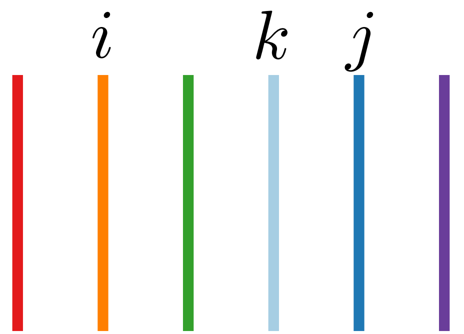
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\ 1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

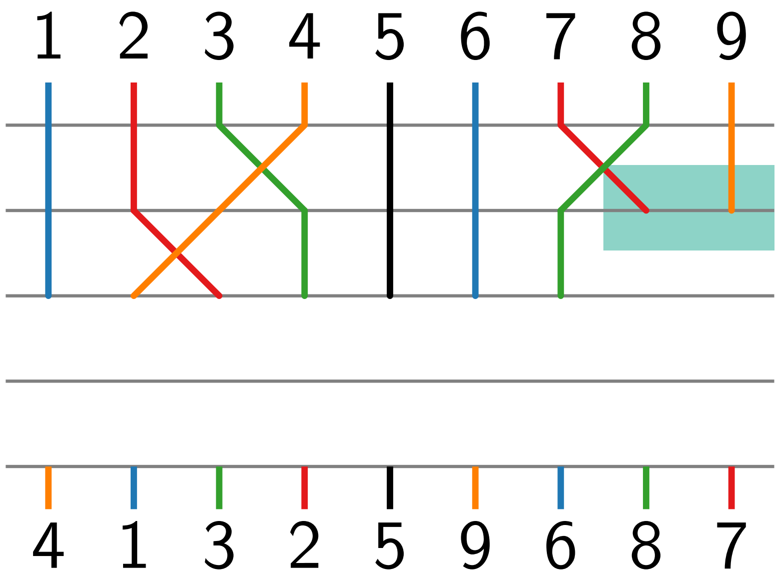
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

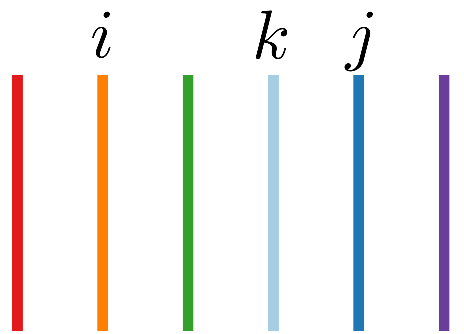
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\ 1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

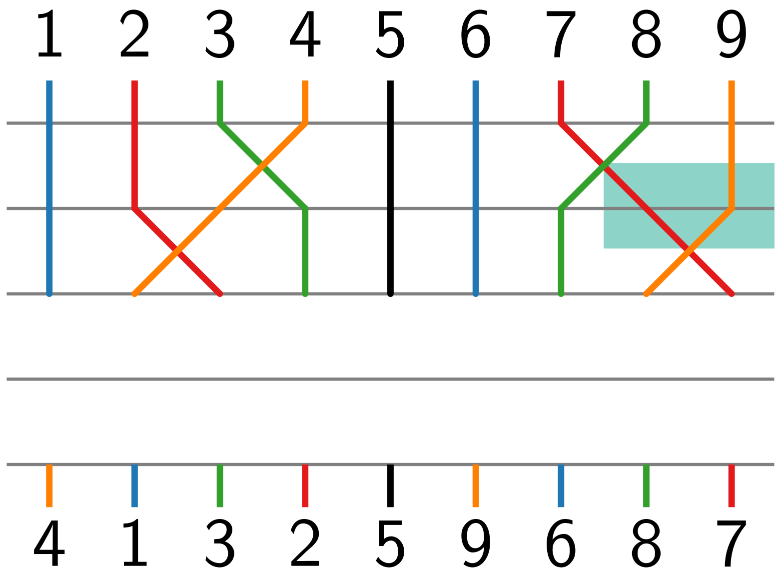
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

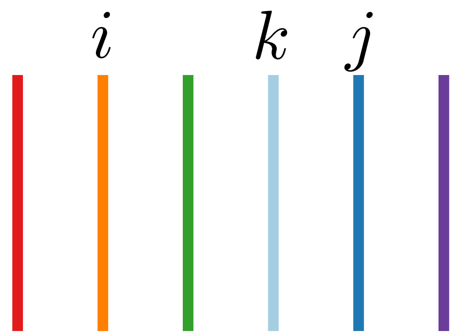
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\,1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

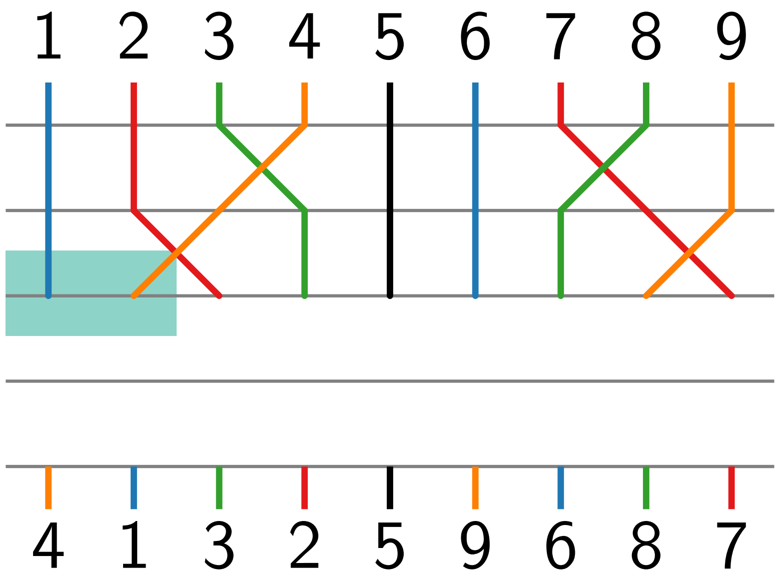
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable* if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

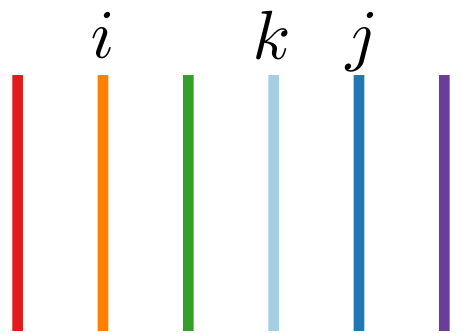
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\,1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

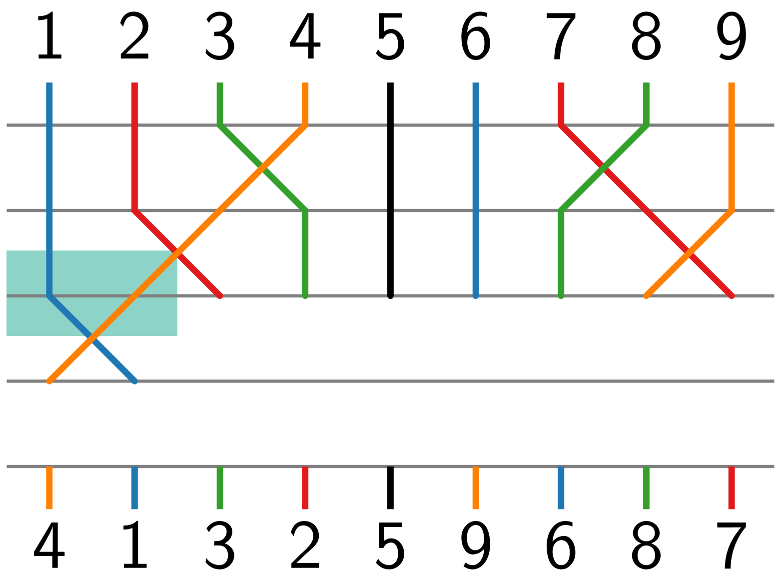
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable* if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

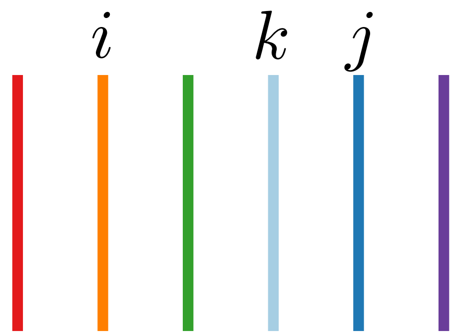
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+$ 1 in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

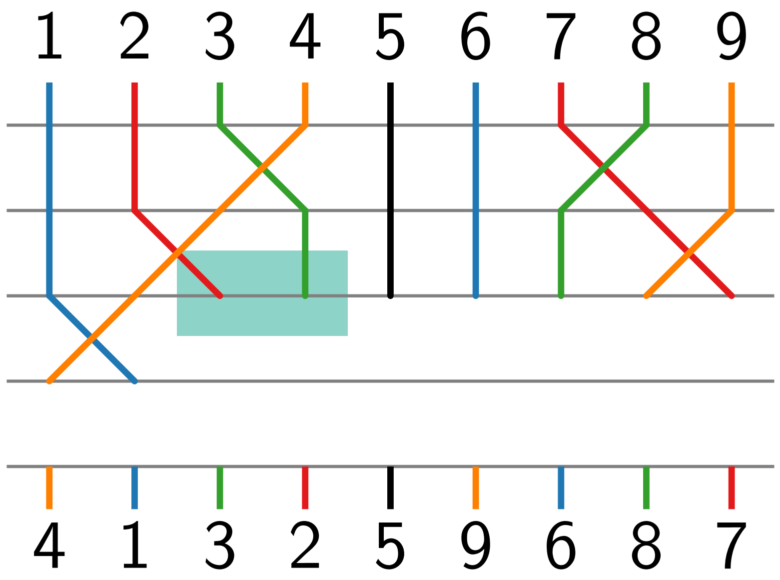
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable* if $\forall i<k<j$: $(\ell_{ik} = \ell_{kj} = 0 \text{ implies } \ell_{ij} = 0)$.

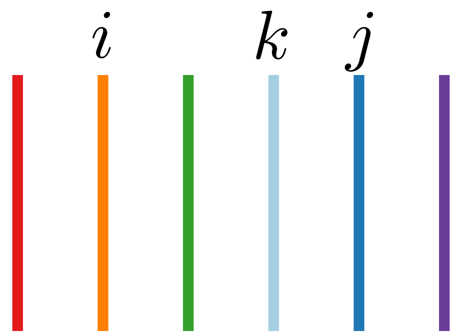
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+$ 1 in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

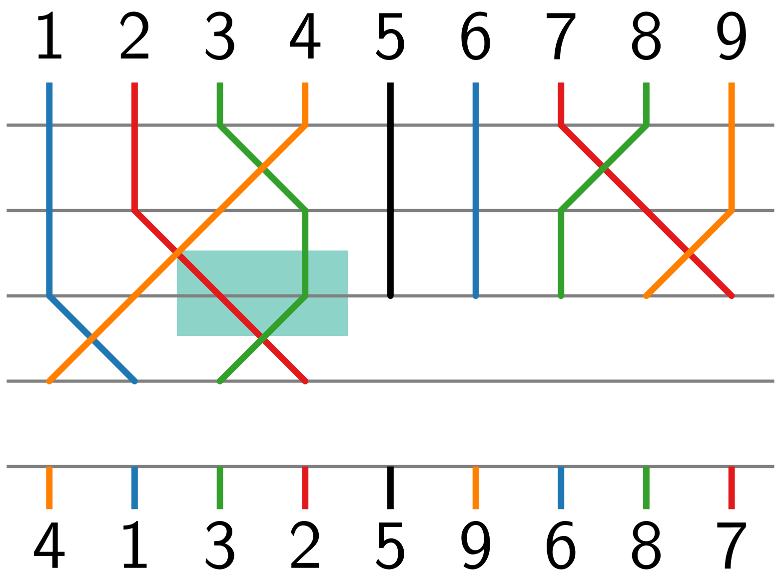
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

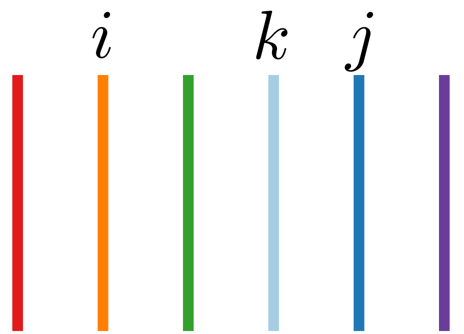
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+ 1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

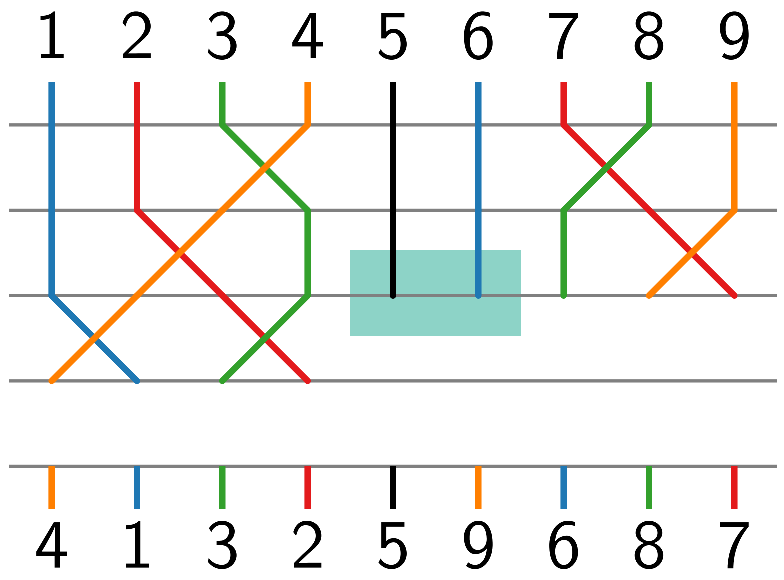
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

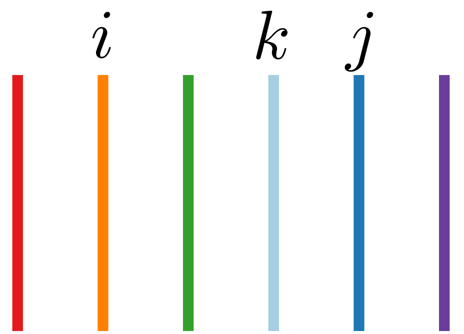
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\,1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

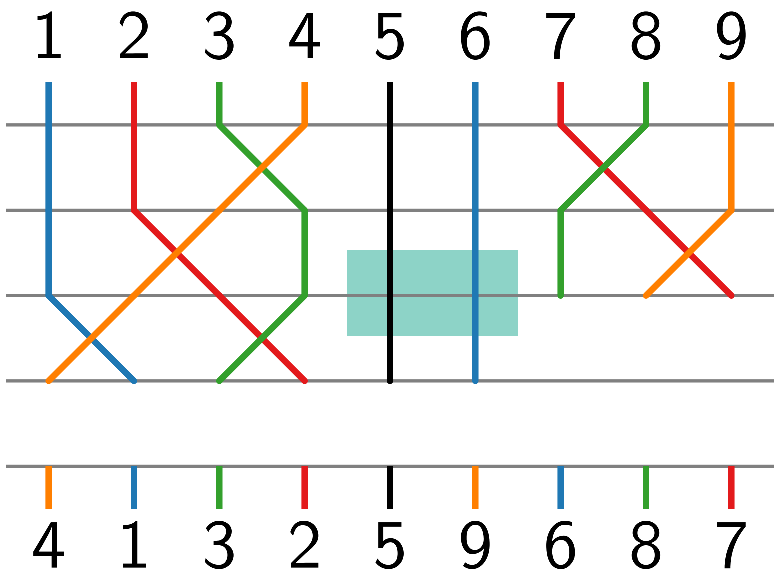
No!

A list (ℓ_{ij}) is *non-separable* if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\,1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

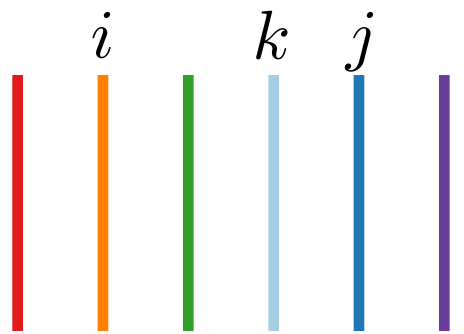
1–4
2–3
2–4
3–4

6–9
7–8
7–9
8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

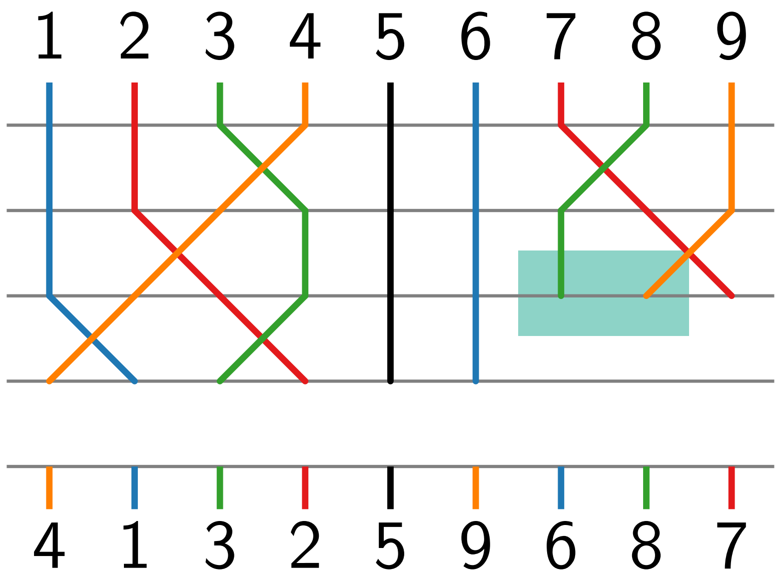
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable* if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

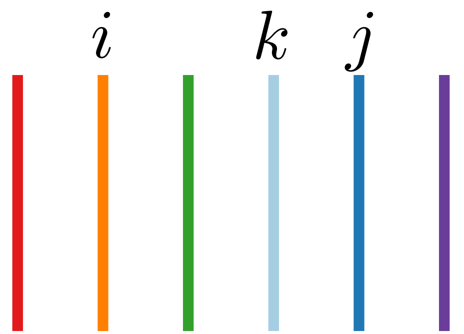
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+ 1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



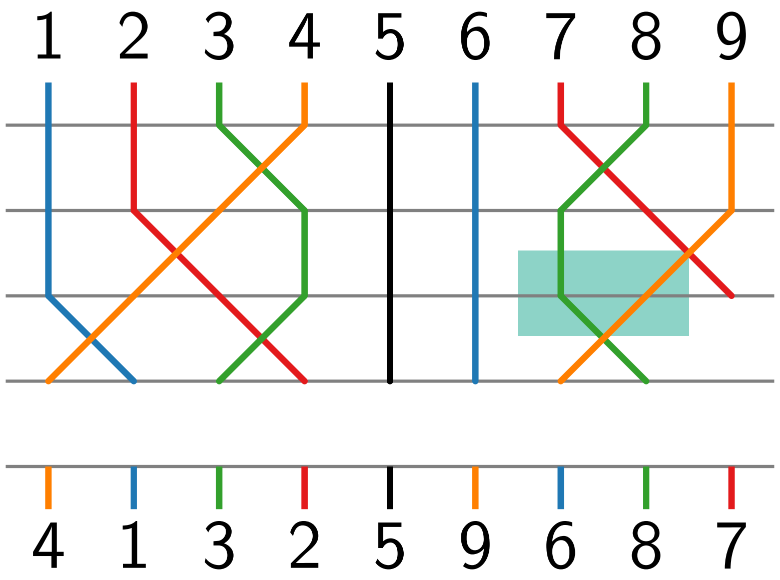
- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.
No!

A list (ℓ_{ij}) is *non-separable* if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

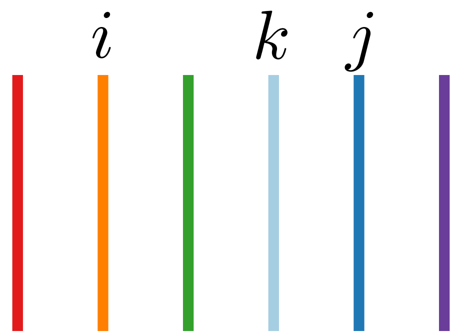
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+ 1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

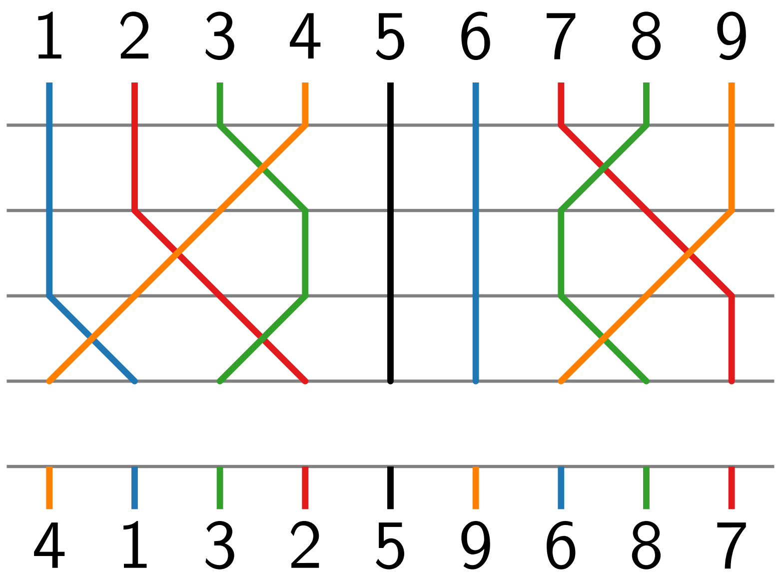
No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+$ 1 in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

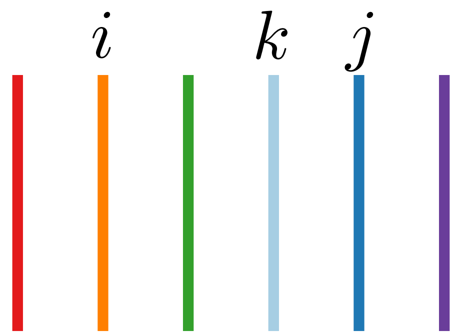
1–4
2–3
2–4
3–4

6–9
7–8
7–9
8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

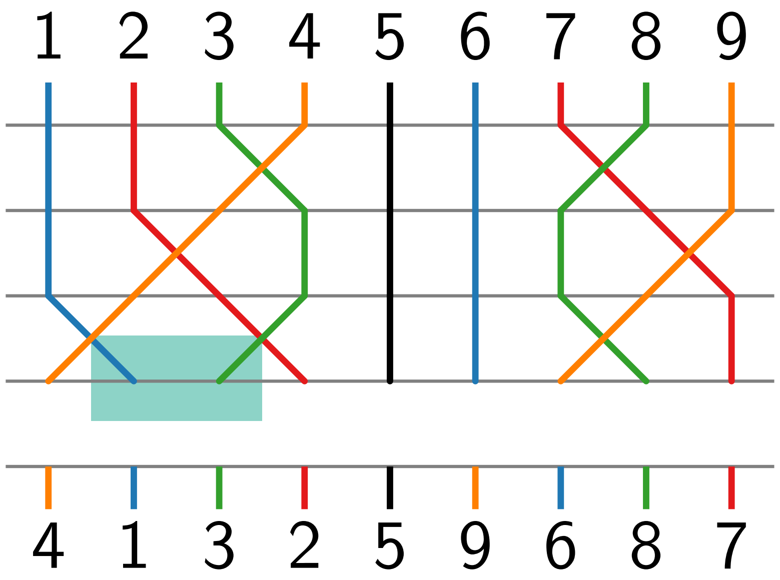
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

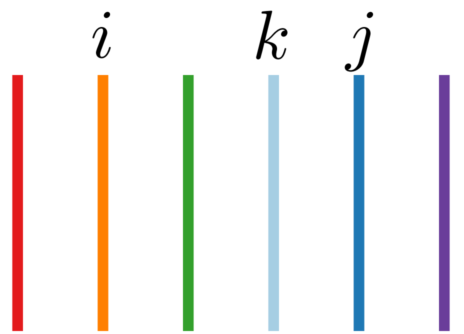
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\ 1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

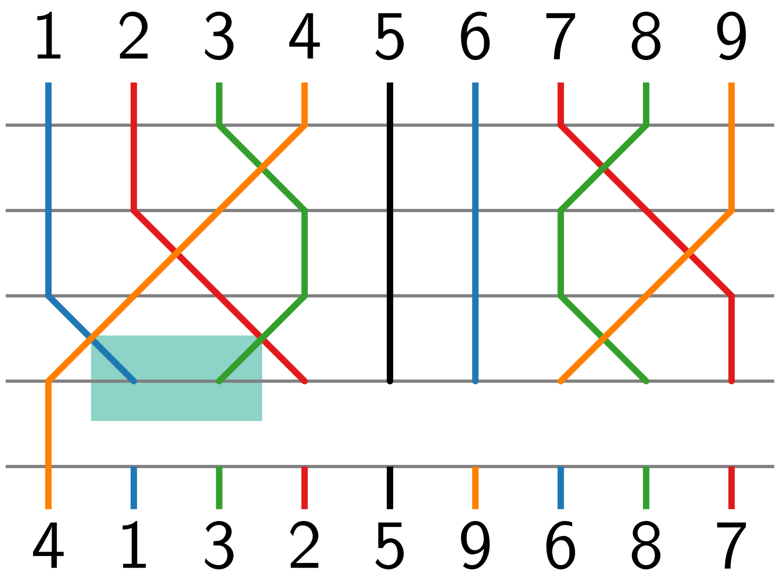
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

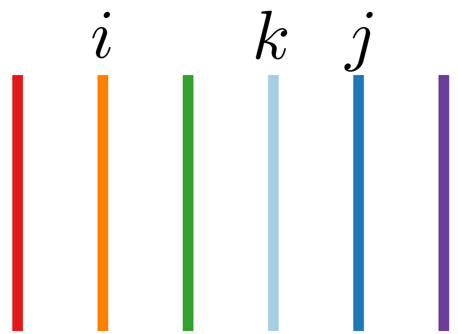
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+$ 1 in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

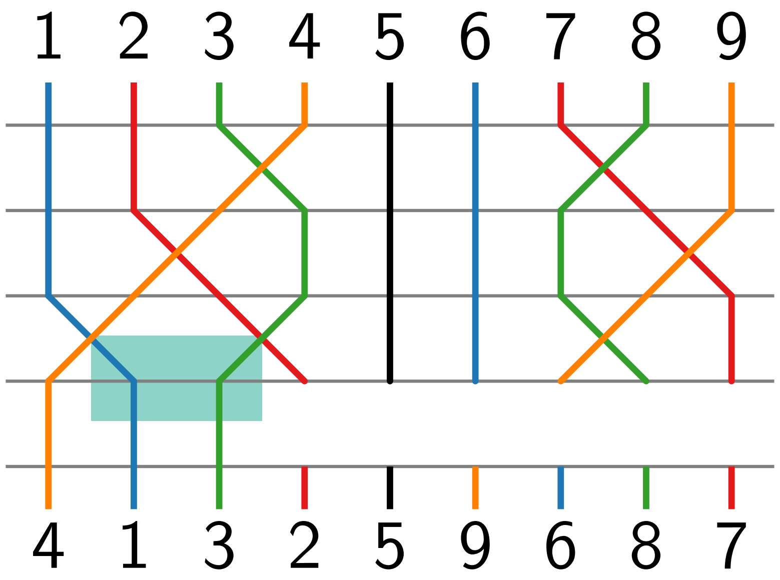
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable* if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

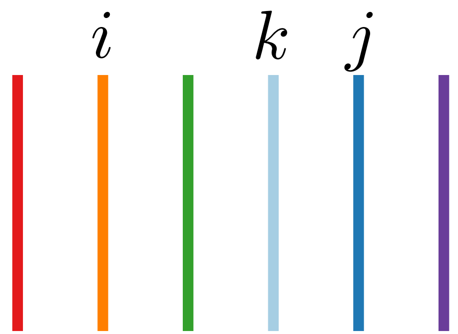
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\,1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



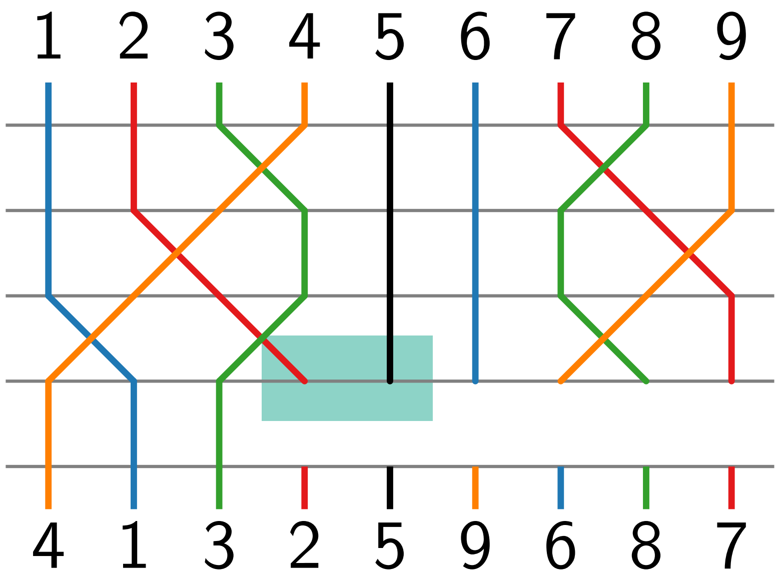
- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.
No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

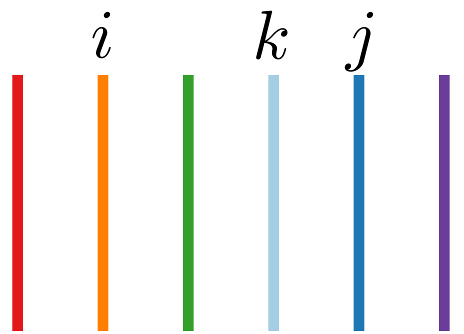
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+$ 1 in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

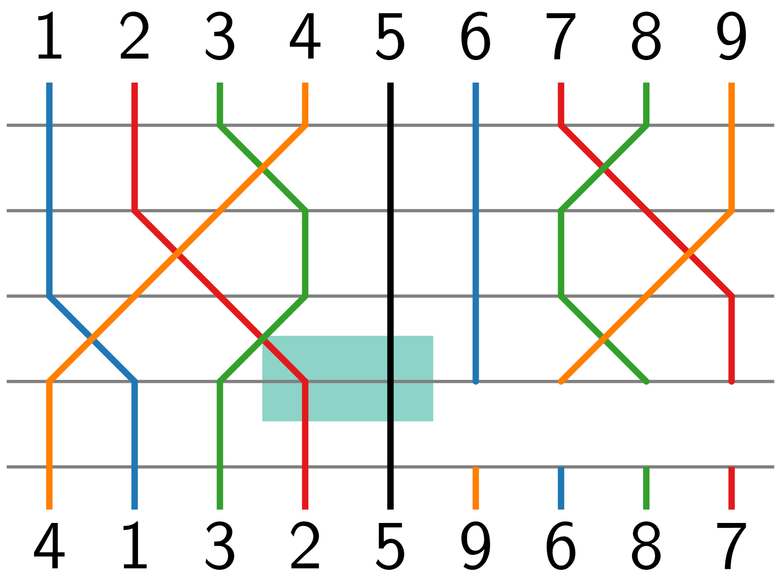
No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+ 1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

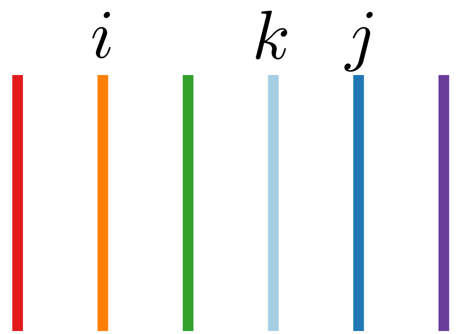
1–4
2–3
2–4
3–4

6–9
7–8
7–9
8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

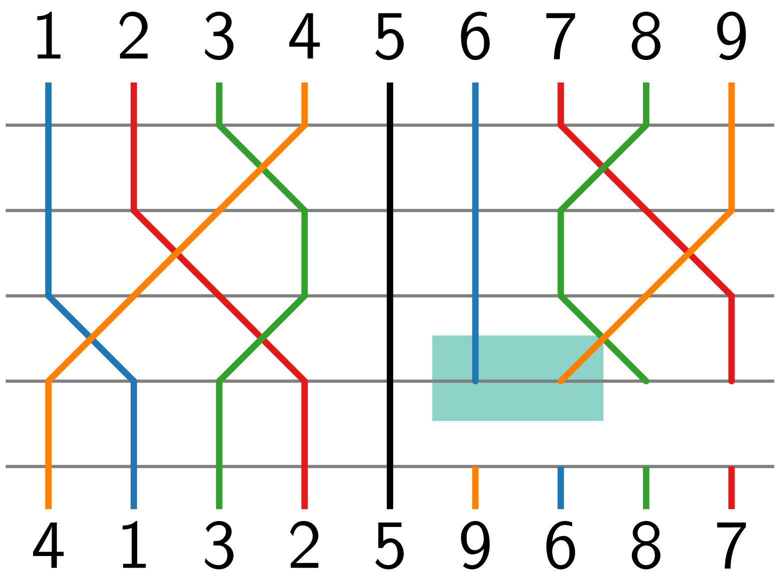
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i{<}k{<}j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

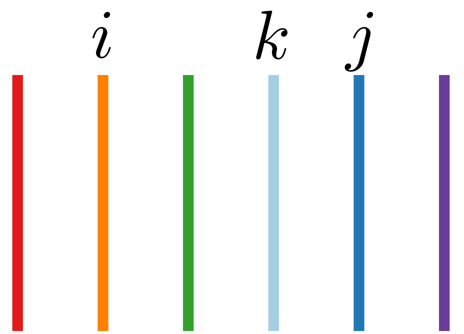
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+$ 1 in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

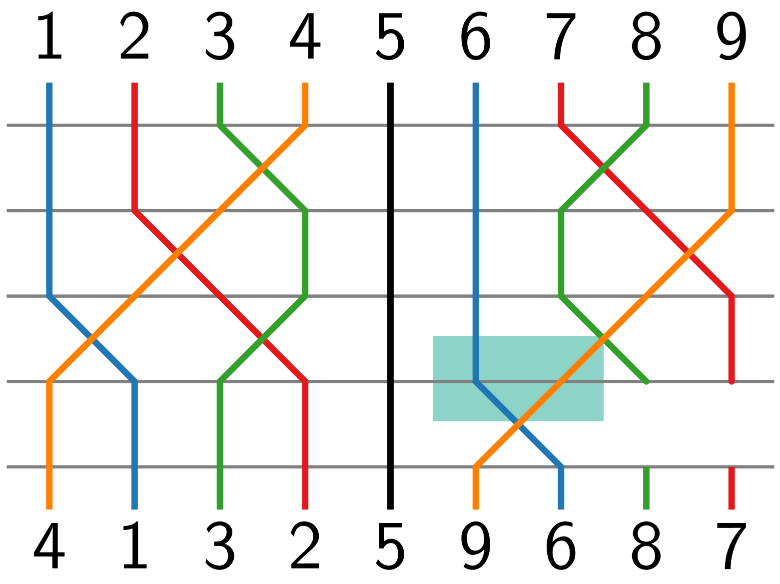
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

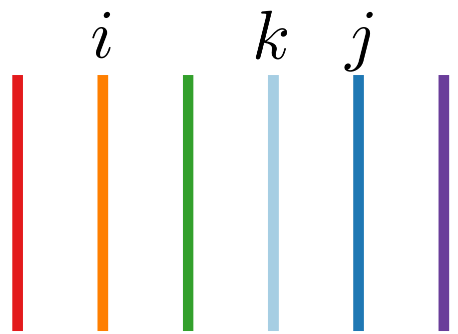
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\ 1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

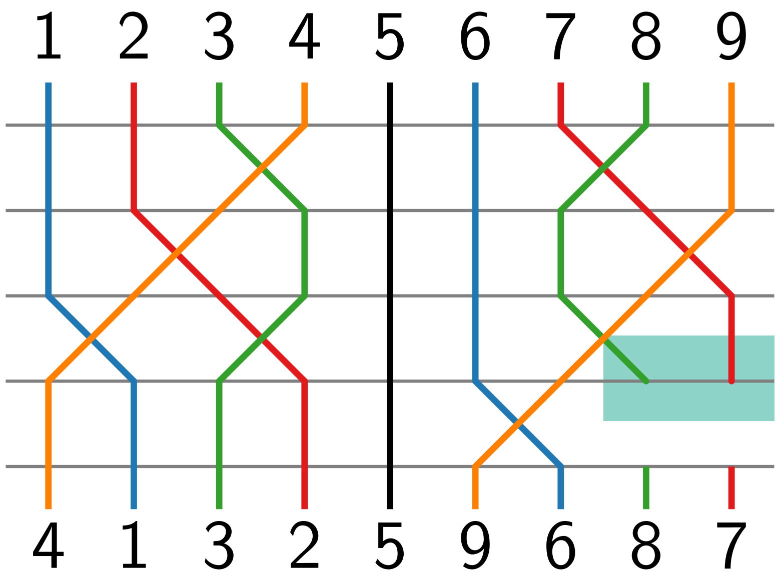
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

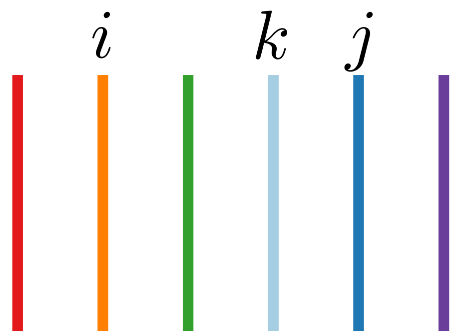
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\ 1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

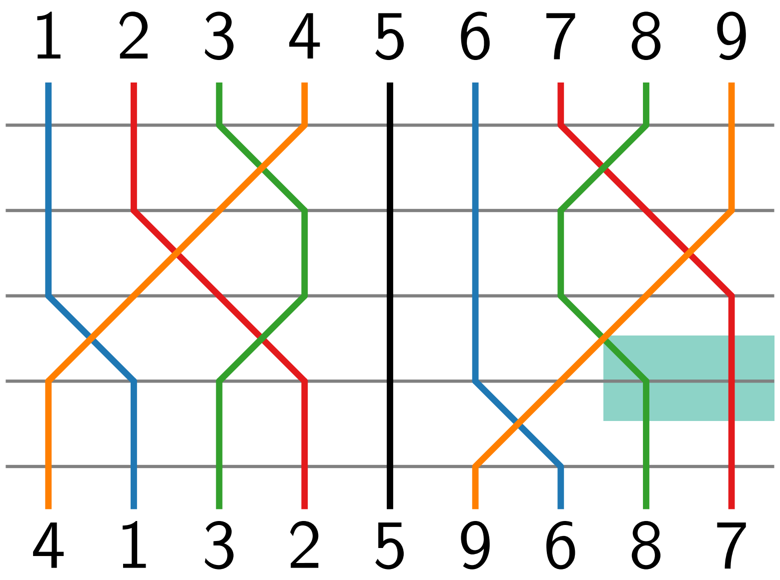
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

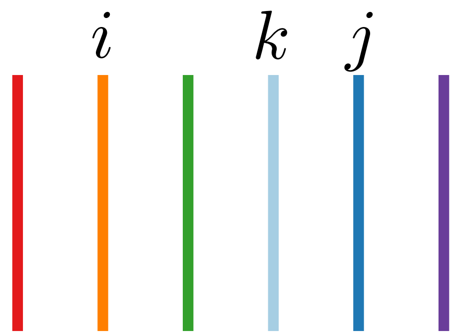
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\,1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

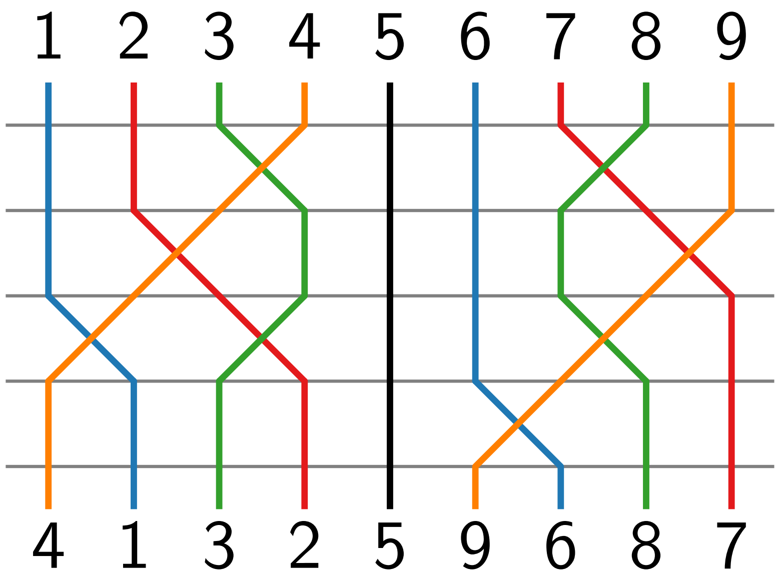
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i{<}k{<}j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

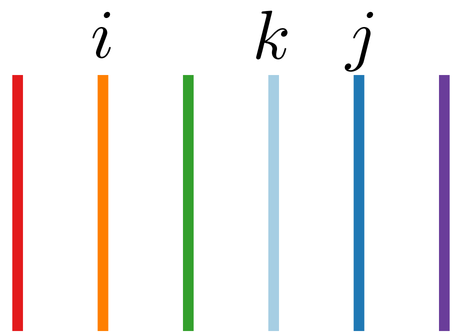
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+\,1$ in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9



odd-even sort

Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

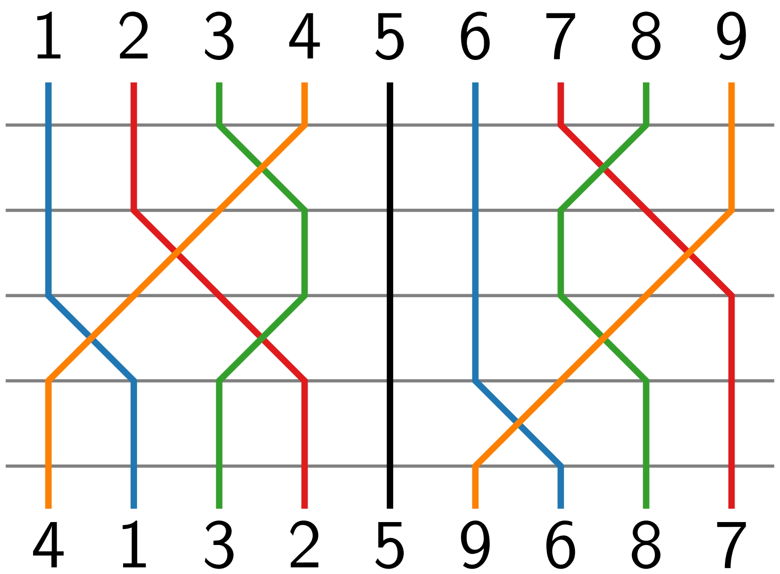
Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

No!

A list (ℓ_{ij}) is *non-separable* if $\forall i<k<j$: $\big(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0\big)$.

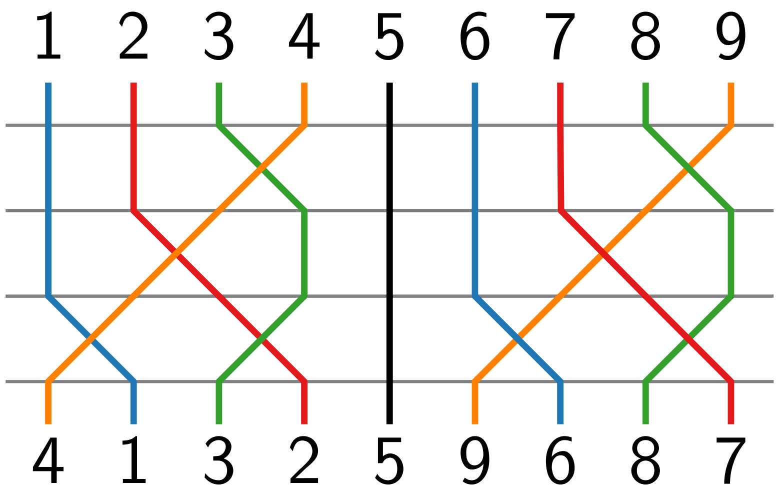
- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+$ 1 in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9

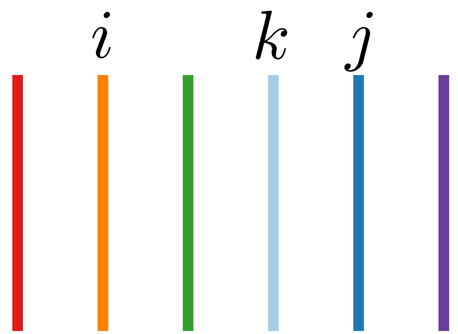


odd-even sort

OPT



Open Problems



- Let L be a list where each swap occurs even number of times.
  How difficult is it to check feasibility of L?

Conjecture. [FKWRZ, GD'19]
Every non-separable even list L is feasible.

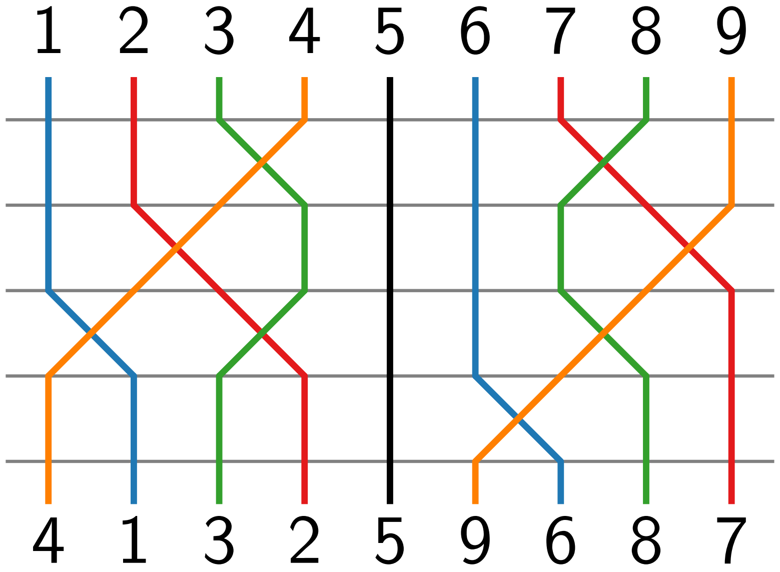
No!

A list (ℓ_{ij}) is *non-separable*
if $\forall i<k<j$: $(\ell_{ik} = \ell_{kj} = 0$ implies $\ell_{ij} = 0)$.

- Let L be a list where each swap occurs at most once. [Sado and Igarashi, TSC'87]
  We can find a tangle that has height at most OPT$+$ 1 in polynomial time.
  Can we also always find a tangle of height OPT efficiently?

1–4	6–9
2–3	7–8
2–4	7–9
3–4	8–9

odd-even sort



Thank you!

OPT