Building Efficient and Compact Data Structures for Simplicial Complexes

Karthik C. S.

Weizmann Institute of Science.

Joint work with Jean-Daniel Boissonnat (INRIA) and Sébastien Tavenas (MPI).

Simplicial Complex (definition)

$V = \{1, 2, 3, 4, 5, 6\}$

Simplicial Complex (definition)

$V = \{1, 2, 3, 4, 5, 6\}$

A simplicial complex $K \subseteq \mathcal{P}(V)$:

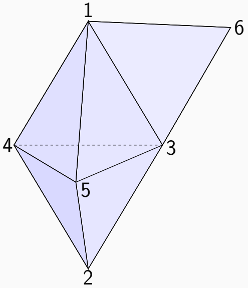
- $p \in V \Rightarrow \{p\} \in K$.
- $\sigma \in K, \tau \subseteq \sigma \Rightarrow \tau \in K$.

Simplicial Complex (definition)

$V = \{1, 2, 3, 4, 5, 6\}$

A simplicial complex $K \subseteq \mathcal{P}(V)$:

- $p \in V \Rightarrow \{p\} \in K$.
- $\sigma \in K, \tau \subseteq \sigma \Rightarrow \tau \in K$.

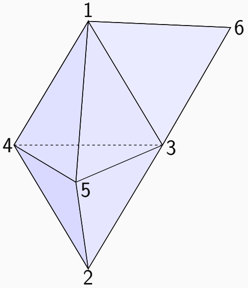


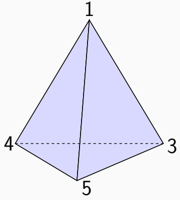
Simplicial Complex (definition)

$V = \{1, 2, 3, 4, 5, 6\}$

A simplicial complex $K \subseteq \mathcal{P}(V)$:

- ◇ $p \in V \Rightarrow \{p\} \in K$.
- ◇ $\sigma \in K, \tau \subseteq \sigma \Rightarrow \tau \in K$.





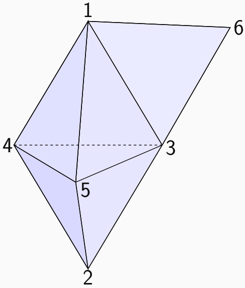
Simplicial Complex (definition)

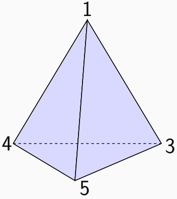
$V = \{1, 2, 3, 4, 5, 6\}$

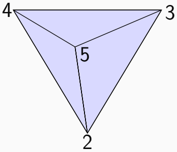
A simplicial complex $K \subseteq \mathcal{P}(V)$:

◇ $p \in V \Rightarrow \{p\} \in K$.

◇ $\sigma \in K, \tau \subseteq \sigma \Rightarrow \tau \in K$.





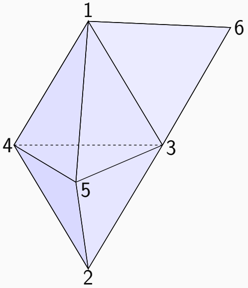


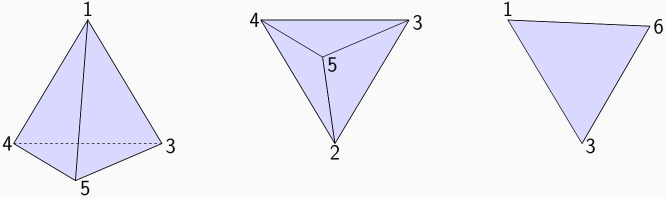
Simplicial Complex (definition)

$V = \{1, 2, 3, 4, 5, 6\}$

A simplicial complex $K \subseteq \mathcal{P}(V)$:

- ◇ $p \in V \Rightarrow \{p\} \in K$.
- ◇ $\sigma \in K, \tau \subseteq \sigma \Rightarrow \tau \in K$.



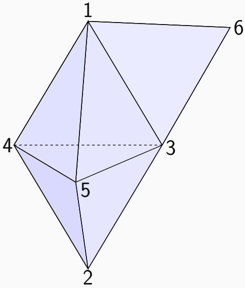


Simplicial Complex (definition)

$V = \{1, 2, 3, 4, 5, 6\}$

A simplicial complex $K \subseteq \mathcal{P}(V)$:

- ◇ $p \in V \Rightarrow \{p\} \in K$.
- ◇ $\sigma \in K, \tau \subseteq \sigma \Rightarrow \tau \in K$.

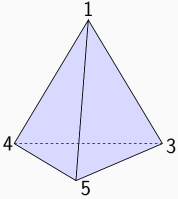


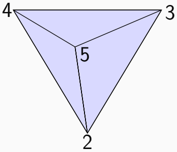
6 vertices

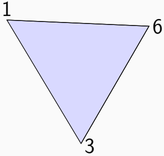
3 dimensional

3 maximal simplices

28 simplices







Agenda

Find a representation for simplicial complexes:

Find a representation for simplicial complexes:

⋆ Small size.

Find a representation for simplicial complexes:

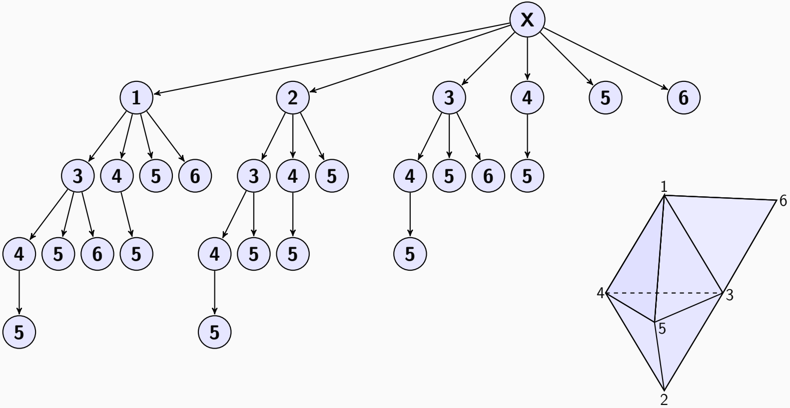
⋆ Small size.

⋆ Perform queries quickly:

- Simplex Membership.
- Simplex Insertion.
- Simplex Removal.

Simplex Tree

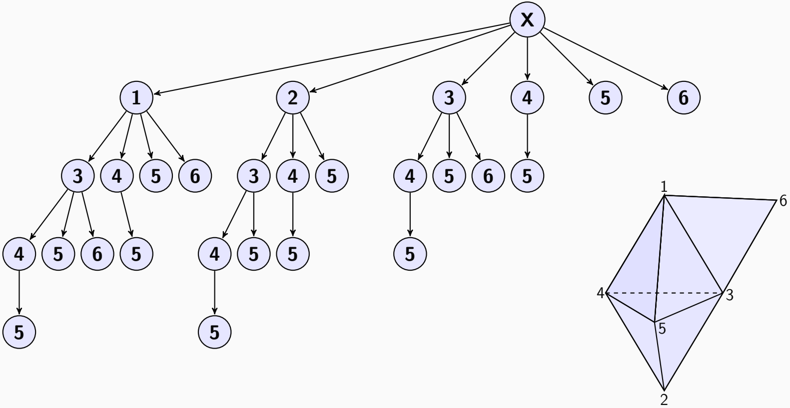
Introduced by Boissonnat and Maria [ESA '12, Algorithmica '14].



Simplex Tree

Storage: $\Theta(m \log n)$

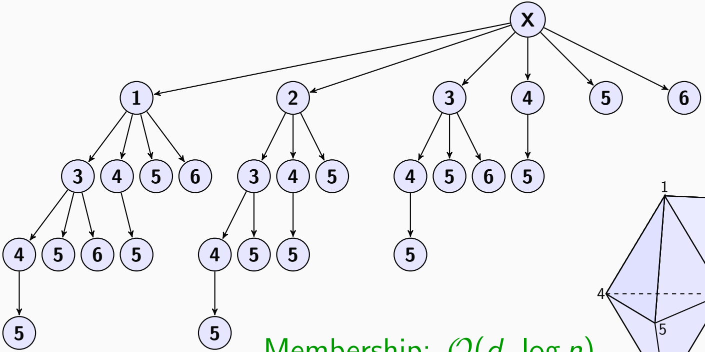
m: # of simplices
n: # of vertices

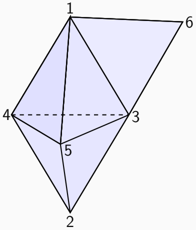


Simplex Tree

Storage: $\Theta(m \log n)$

m: # of simplices
n: # of vertices





Membership: $\mathcal{O}(d_\sigma \log n)$

Insertion: $\mathcal{O}(2^{d_\sigma} d_\sigma \log n)$

Removal: $\mathcal{O}(m \log n)$

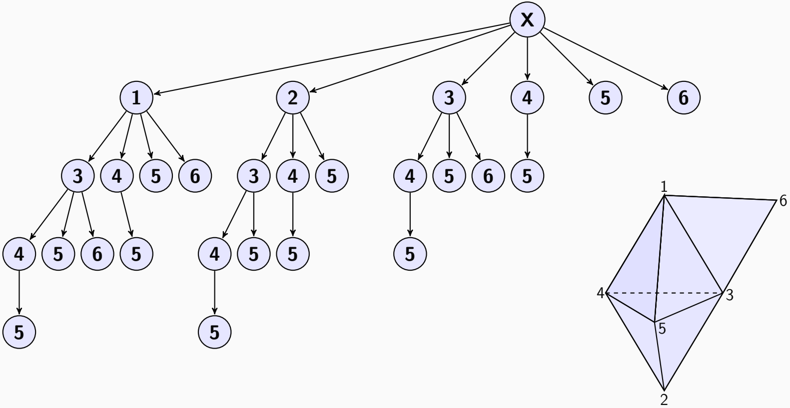
σ: a simplex

d_σ: dimension of σ

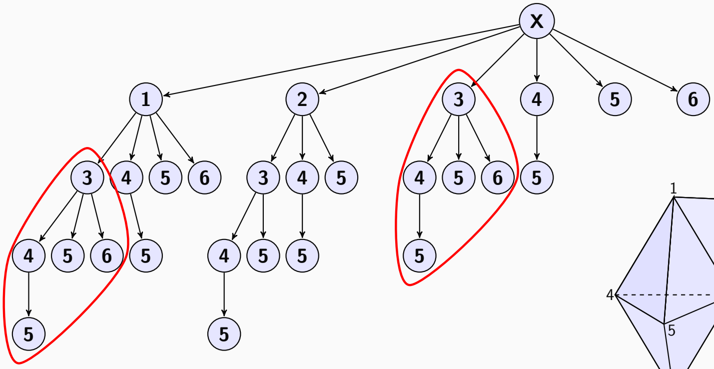
Our Results:

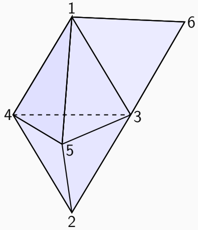
- Compression of Simplex Tree.
- New Data Structure:
  - ⋆ Compact.
  - ⋆ Better performance.

Simplex Tree: Let's store less!



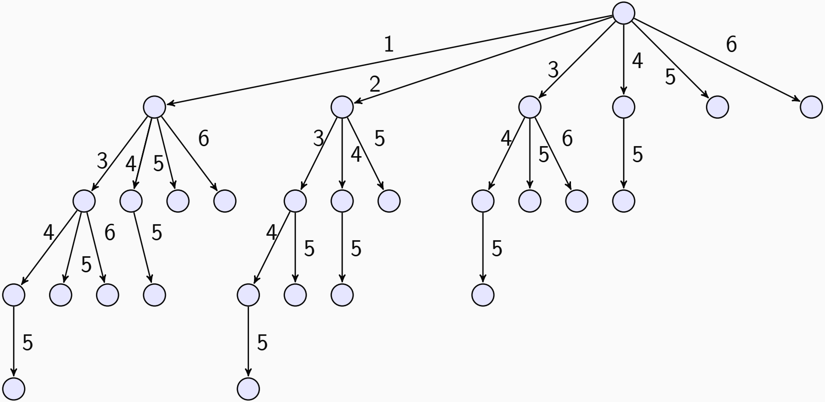
Simplex Tree: Let's store less!



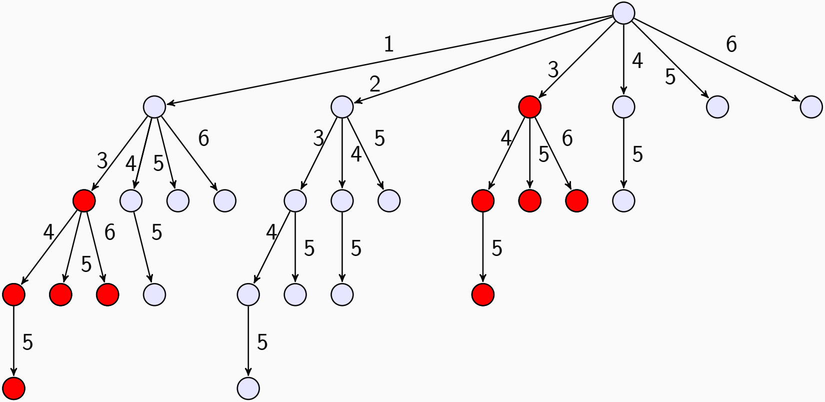


We can compress!

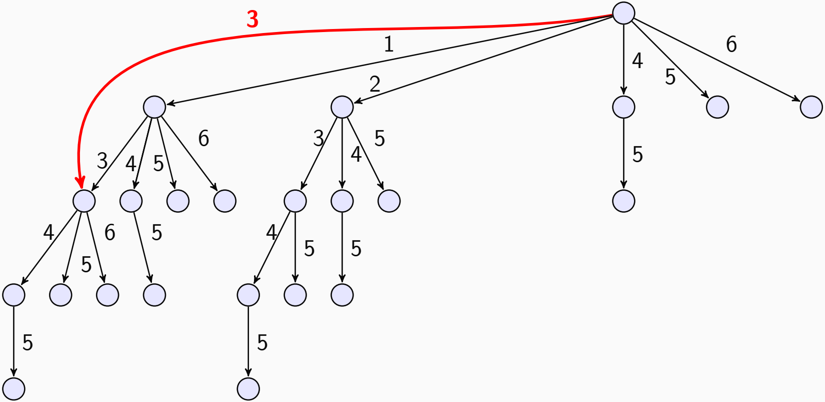
Simplex Automaton



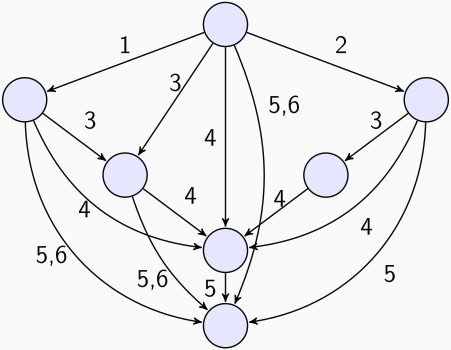
Simplex Automaton



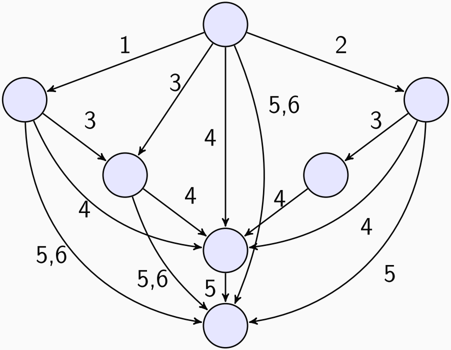
Simplex Automaton



Minimal Simplex Automaton



Minimal Simplex Automaton



Hopcroft's Algorithm: $O(m \log m \log n)$ time.

Minimal Simplex Automaton: A Discussion

- ⋆ Simplex Tree: Simplex $\leftrightarrow$ Node.

  Minimal Simplex Automaton: Simplex $\leftrightarrow$ Path.

Minimal Simplex Automaton: A Discussion

- ⋆ Simplex Tree: Simplex $\leftrightarrow$ Node.

  Minimal Simplex Automaton: Simplex $\leftrightarrow$ Path.

- ⋆ Answering static queries remains unchanged.

Minimal Simplex Automaton: A Discussion

- ⋆ Simplex Tree: Simplex $\leftrightarrow$ Node.

  Minimal Simplex Automaton: Simplex $\leftrightarrow$ Path.

- ⋆ Answering static queries remains unchanged.

- ⋆ Dynamic queries: more complex.

Data Set 1: Rips Complex from sampling of Klein bottle in $\mathbb{R}^5$.

n	α	d	k	m	Size After Compression	Compression Ratio
10,000	0.15	10	24,970	604,573	218,452	2.77
10,000	0.16	13	25,410	1,387,023	292,974	4.73
10,000	0.17	15	27,086	3,543,583	400,426	8.85
10,000	0.18	17	27,286	10,508,486	524,730	20.03

Experiments

Data Set 1: Rips Complex from sampling of Klein bottle in $\mathbb{R}^5$.

n	α	d	k	m	Size After Compression	Compression Ratio
10,000	0.15	10	24,970	604,573	218,452	2.77
10,000	0.16	13	25,410	1,387,023	292,974	4.73
10,000	0.17	15	27,086	3,543,583	400,426	8.85
10,000	0.18	17	27,286	10,508,486	524,730	20.03

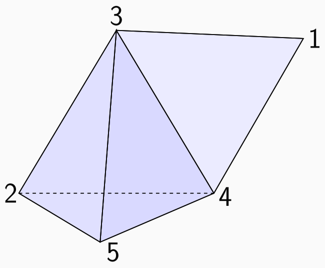
Data Set 2: Flag complexes generated from random graph $G_{n,p}$.

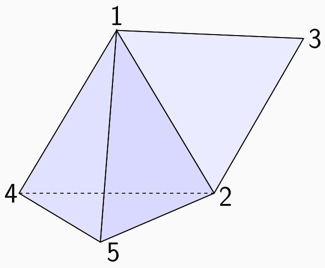
n	p	d	k	m	Size After Compression	Compression Ratio
25	0.8	17	77	315,370	467	537.3
30	0.75	18	83	4,438,559	627	7,079.0
35	0.7	17	181	3,841,591	779	4,931.4
40	0.6	19	204	9,471,220	896	10,570.6
50	0.5	20	306	25,784,504	1,163	22,170.7

Labeling of vertices

- Size of ST is invariant over labeling.
- Size of MSA is dependent on labeling.

- Size of ST is invariant over labeling.
- Size of MSA is dependent on labeling.





Theorem

The task of finding a labeling to minimize size of MSA is NP-Complete.

Build a data structure which has:

- Slightly worse performance on membership query.
- Smaller size.
- Quicker insertion and removal.

Build a data structure which has:

- Slightly worse performance on membership query.
- Smaller size.
- Quicker insertion and removal.

Inspiration:

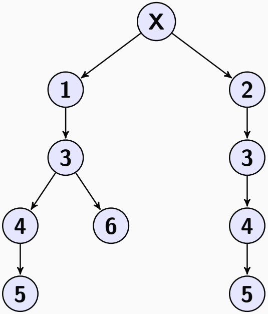
Deterministic Finite state Automaton(DFA)

vs

Non-deterministic Finite state Automaton(NFA).

Maximal Simplex Tree

- Store only maximal simplices in Trie.
- Size: $\mathcal{O}(kd \log n)$.

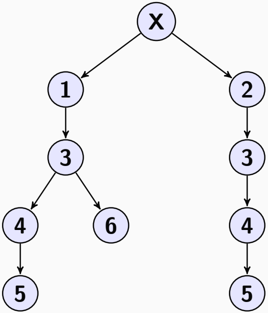


k : # of maximal simplices

Maximal Simplex Tree

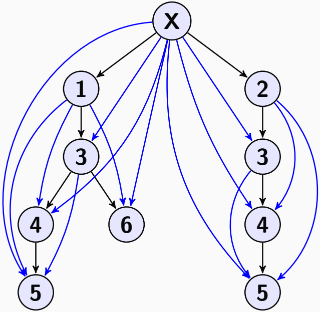
- Store only maximal simplices in Trie.
- Size: $\mathcal{O}(kd \log n)$.

We have a matching lower bound.



Maximal Simplex Tree

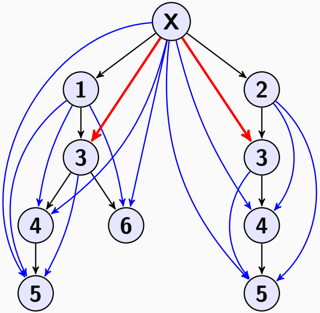
- Store only maximal simplices in Trie.
- Size: $\mathcal{O}(kd \log n)$.



- Transitive Closure.
- Path(s) $\leftrightarrow$ Simplex.

Maximal Simplex Tree

- Store only maximal simplices in Trie.
- Size: $\mathcal{O}(kd \log n)$.

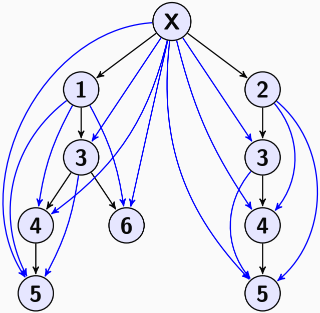


- Transitive Closure.
- Path(s) $\leftrightarrow$ Simplex.

NFA recognizing all simplex words.

Maximal Simplex Tree

- Store only maximal simplices in Trie.
- Size: $\mathcal{O}(kd \log n)$.

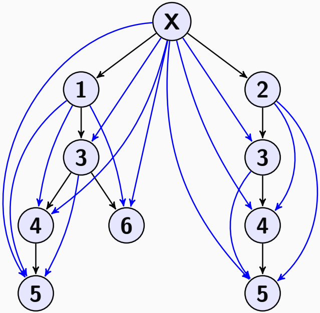


- Transitive Closure.
- Path(s) $\leftrightarrow$ Simplex.

Insertion and Removal are quicker(?).

Maximal Simplex Tree

- Store only maximal simplices in Trie.
- Size: $\mathcal{O}(kd \log n)$.

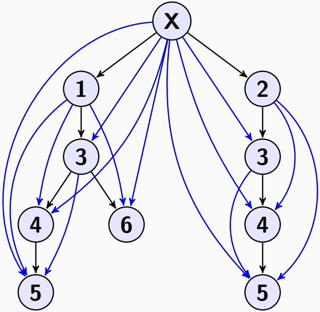


- Transitive Closure.
- Path(s) $\leftrightarrow$ Simplex.

Membership is still not efficient!

Maximal Simplex Tree

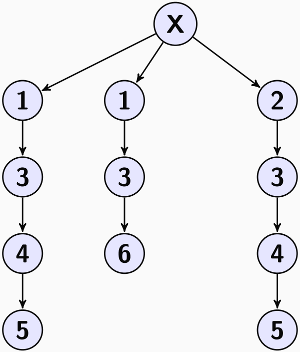
- Store only maximal simplices in Trie.
- Size: $\mathcal{O}(kd \log n)$.



- Transitive Closure.
- Path(s) $\leftrightarrow$ Simplex.

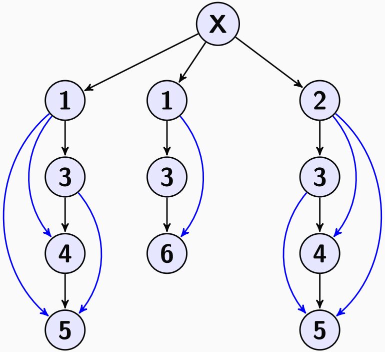
We will fix this and build NFA: Simplex Array List.

Unprefixed Maximal Simplex Tree



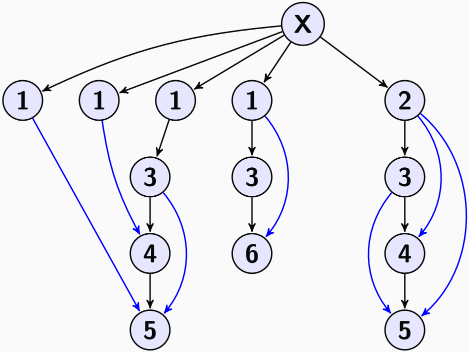
Common prefixes are not merged.

Unprefixed Maximal Simplex Tree

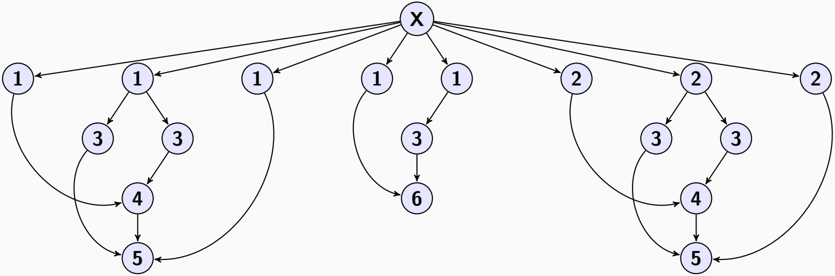


Transitive Closure.

Unprefixed Maximal Simplex Tree

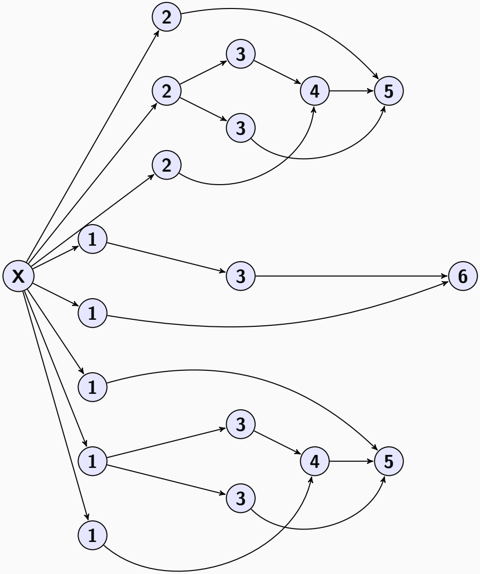


Ensure all children are of same label.

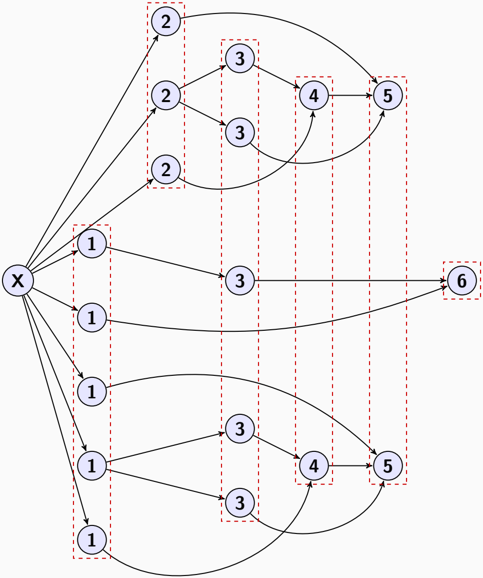


Duplicate from top to bottom.

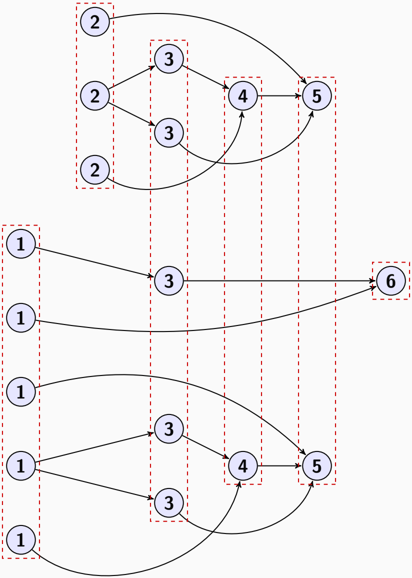
Simplex Array List



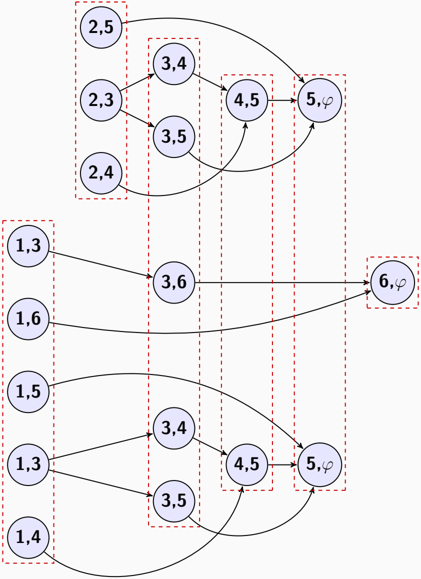
Simplex Array List



Simplex Array List

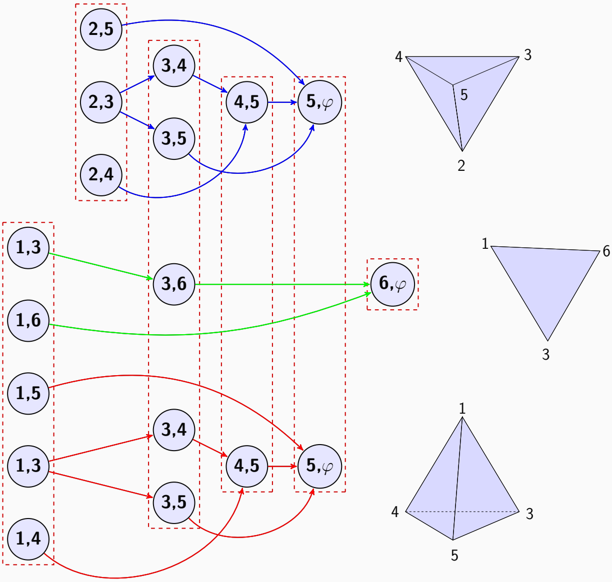


Simplex Array List

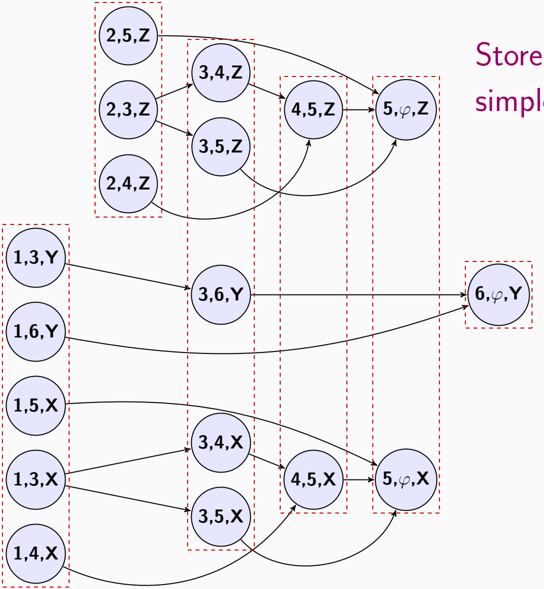


Store label of children.

Simplex Array List

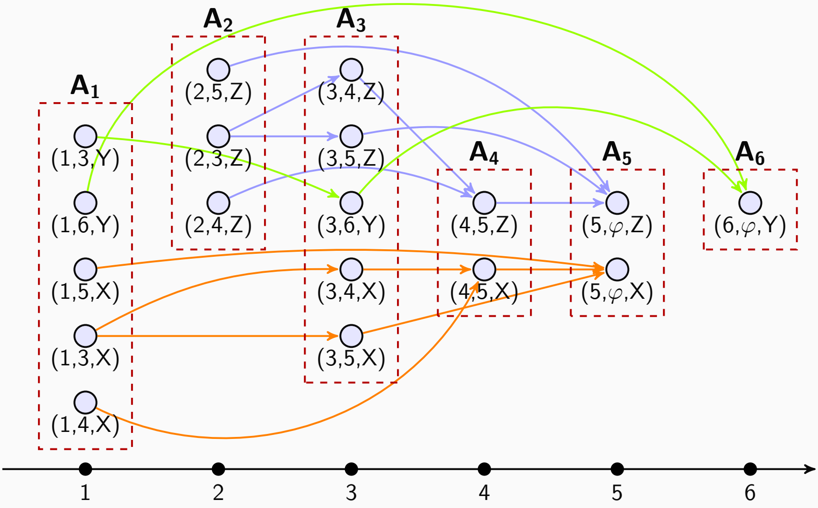


Simplex Array List



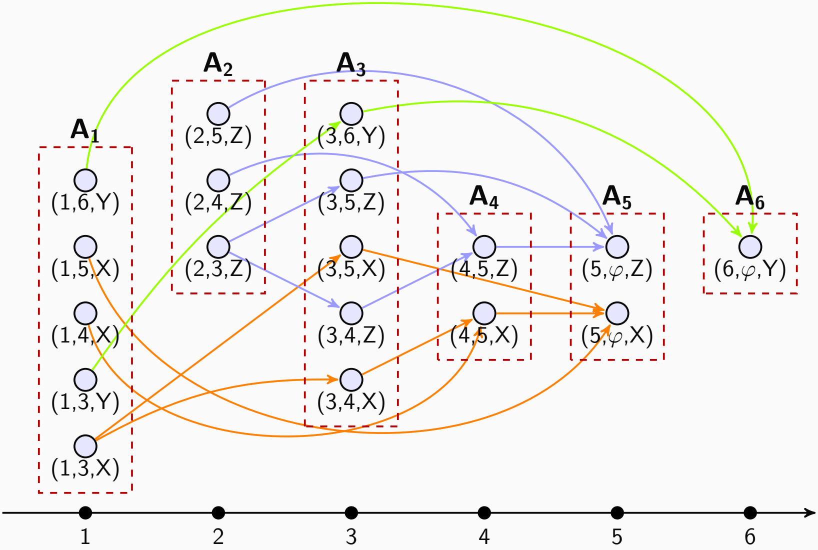
Store the maximal simplex information.

Simplex Array List

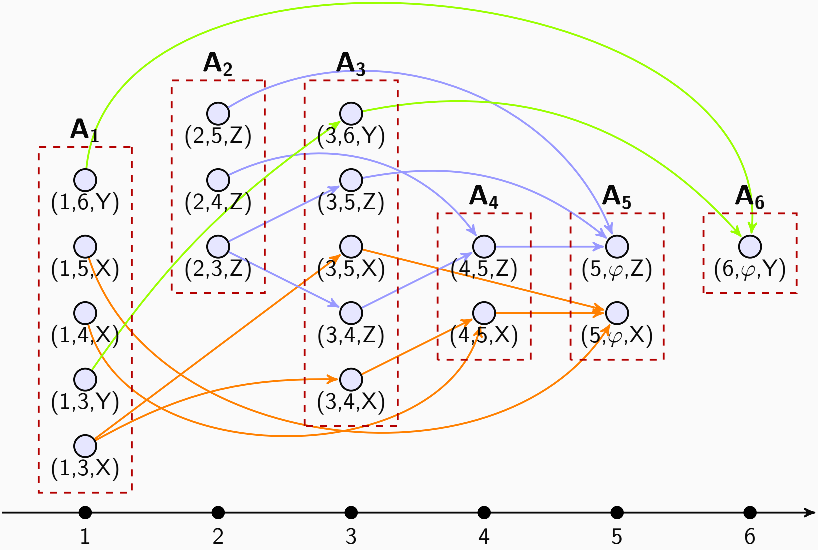


Sort according to second coordinate.

Simplex Array List (SAL)

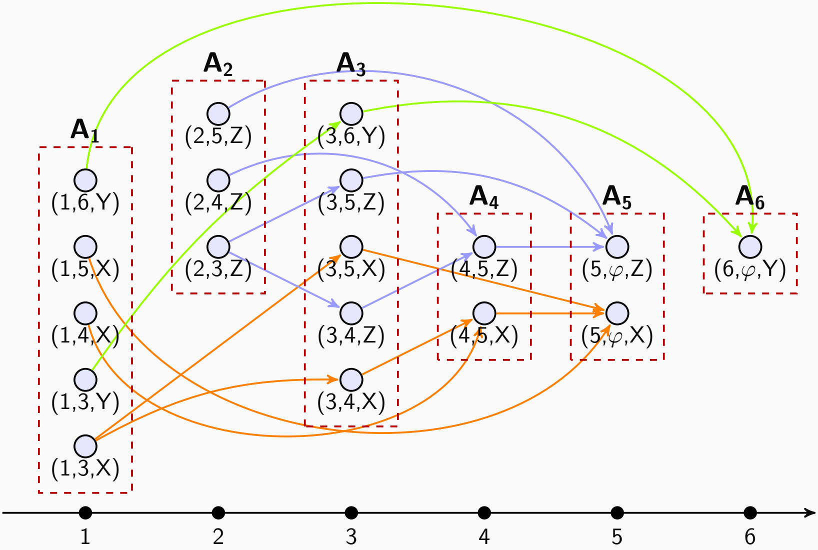


Simplex Array List (SAL)



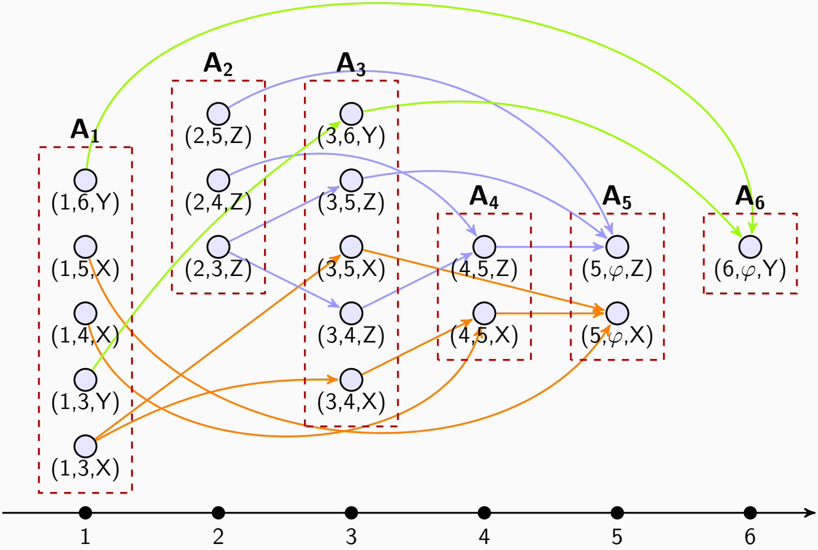
Storage: $\mathcal{O}\big((\log k + \log n)\cdot kd \cdot d \cdot d\big) = \mathcal{O}\big(kd^3(\log k + \log n)\big)$

Simplex Array List (SAL)



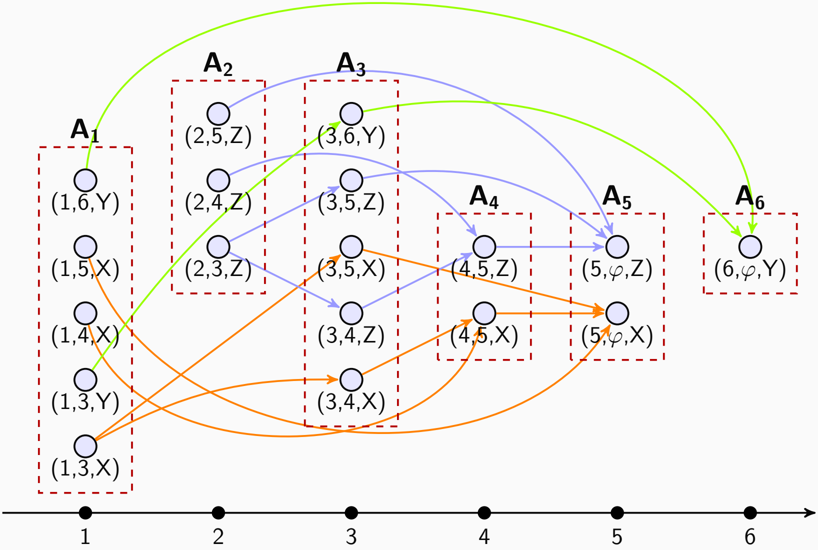
Storage: $\mathcal{O}((\log k + \log n)\cdot kd\cdot d\cdot d) = \mathcal{O}(kd^3(\log k + \log n))$

Simplex Array List (SAL)



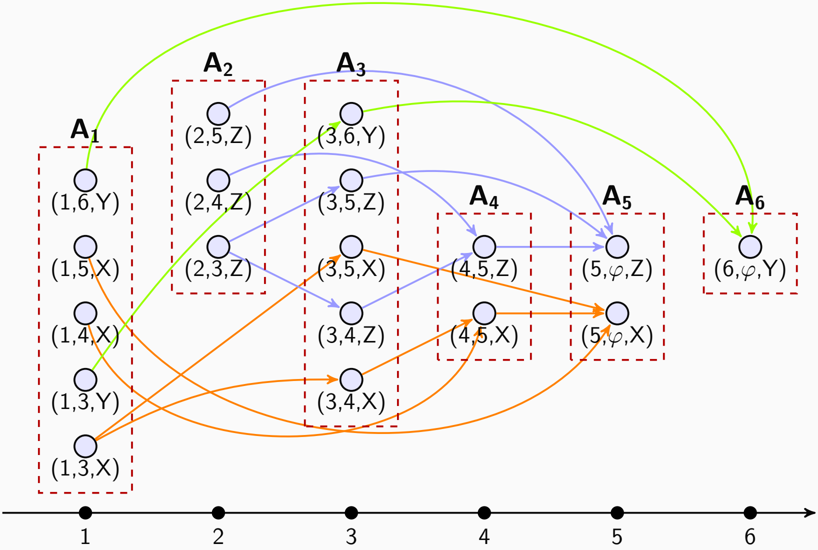
Storage: $\mathcal{O}((\log k + \log n) \cdot kd \cdot d \cdot d) = \mathcal{O}(kd^3(\log k + \log n))$

Simplex Array List (SAL)



Storage: $\mathcal{O}((\log k + \log n)\cdot kd\cdot d\cdot\ ) = \mathcal{O}(kd^3(\log k + \log n))$

Simplex Array List (SAL)



Storage: $\mathcal{O}((\log k + \log n) \cdot kd \cdot d \cdot d) = \mathcal{O}(kd^3(\log k + \log n))$

One parameter to rule them all: λ.

One parameter to rule them all: λ.

$\lambda(e)$: # of maximal simplices that contain an edge e of K.

$$\lambda = \max_{e \in K} \lambda(e).$$

One parameter to rule them all: λ.

$\lambda(e)$: # of maximal simplices that contain an edge e of K.

$$\lambda = \max_{e \in K} \lambda(e).$$

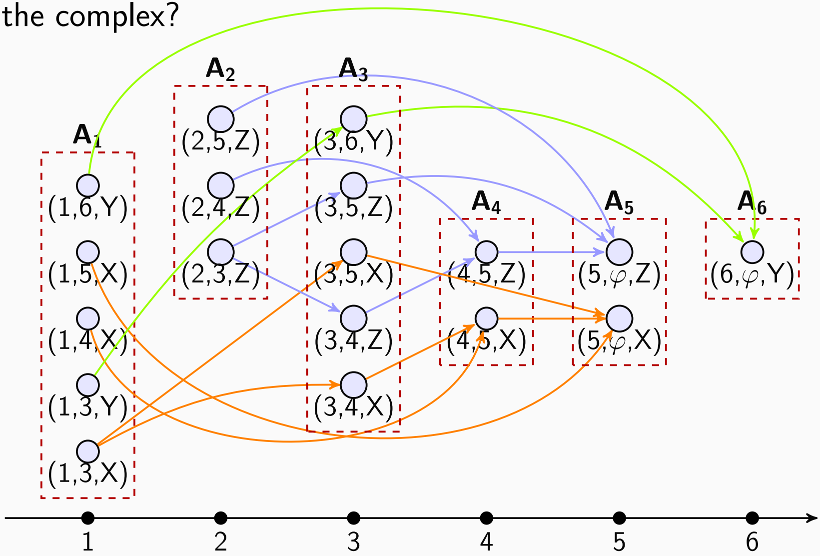
λ is at most k.

Value of λ

Data Set: Rips Complex from sampling of Klein bottle in $\mathbb{R}^5$.

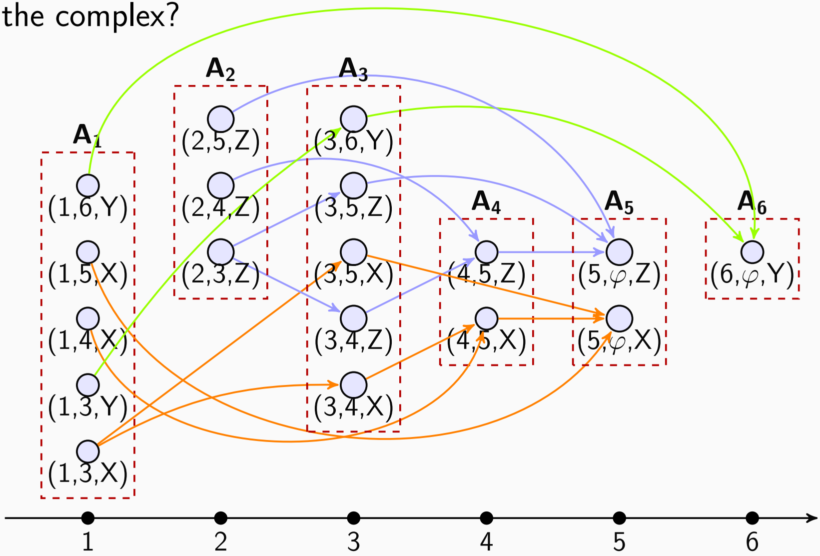
No	n	α	d	k	m	λ	$\lvert SAL \rvert$
1	10,000	0.15	10	24,970	604,573	53	424,440
2	10,000	0.16	13	25,410	1,387,023	61	623,238
3	10,000	0.17	15	27,086	3,543,583	67	968,766
4	10,000	0.18	17	27,286	10,508,486	91	1,412,310

Is the triangle 2−3−5
in the complex?



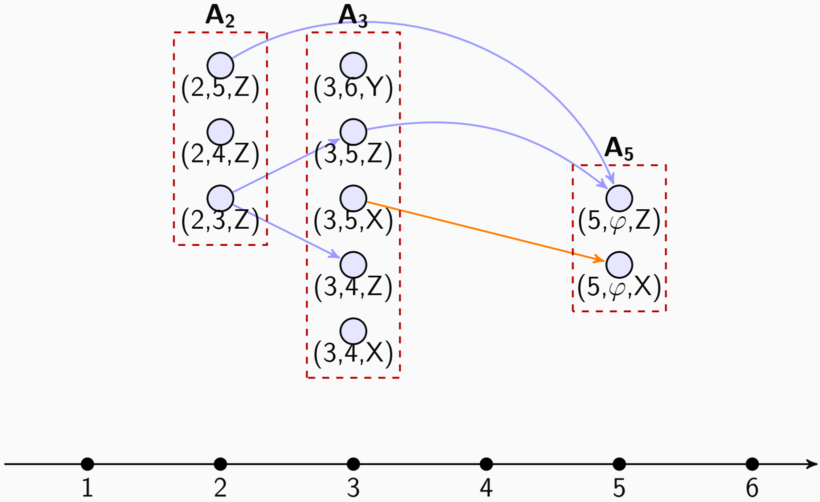
Is the triangle 2−3−5 in the complex?

Step 1: Find A_2, A_3, and A_5.



Is the triangle 2−3−5 in the complex?

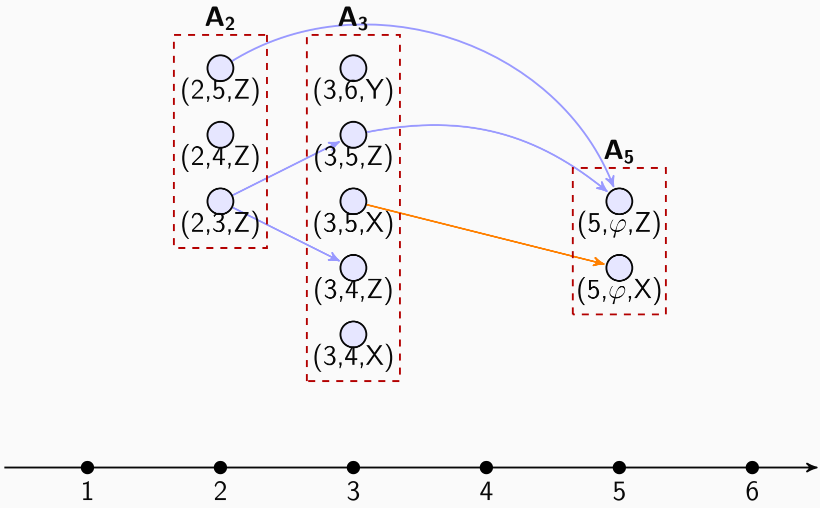
Step 1: Find A_2, A_3, and A_5.



Operations on Simplex Array List

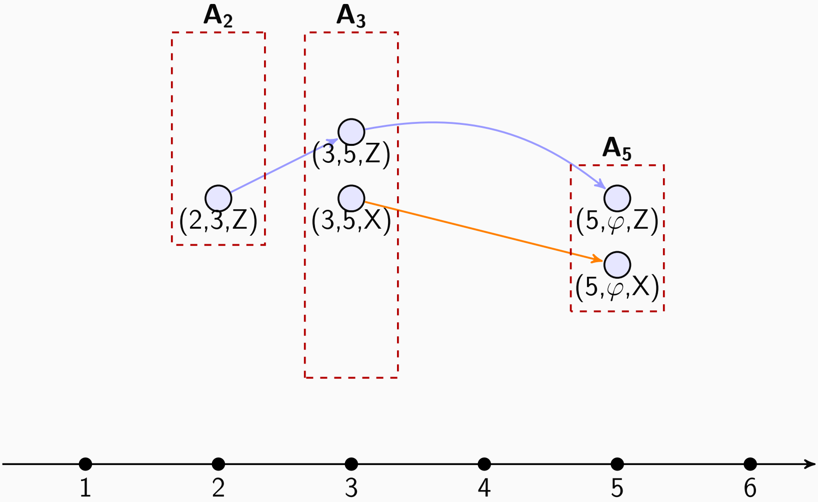
Is the triangle $2-3-5$ in the complex?

Step 2: Find labels $(2, 3, \star)$, $(3, 5, \star)$, and $(5, \star, \star)$.



Is the triangle 2−3−5 in the complex?

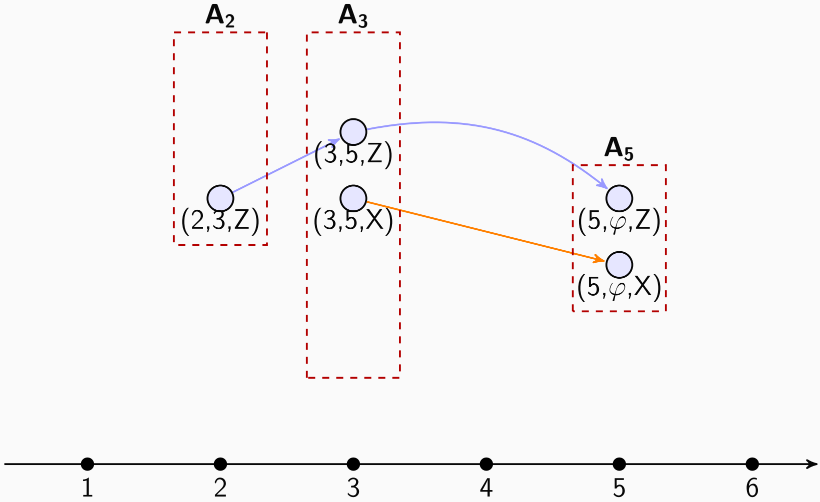
Step 2: Find labels $(2, 3, \star)$, $(3, 5, \star)$, and $(5, \star, \star)$.



Operations on Simplex Array List

Is the triangle 2−3−5 in the complex?

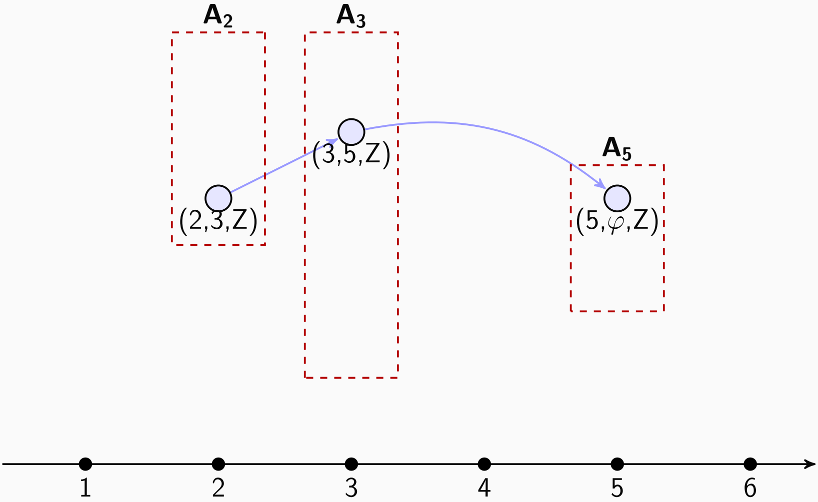
Step 3: Is there a directed path hitting A_2, A_3, and A_5.



Operations on Simplex Array List

Is the triangle 2−3−5 in the complex?

Step 3: Is there a directed path hitting A_2, A_3, and A_5.



Membership on SAL

Input: A simplex $\sigma = v_{\ell_0} \cdots v_{\ell_{d_\sigma}}$.

Task: Check if σ is in K.

1. Find $A_{\ell_0}, \ldots, A_{\ell_{d_\sigma}}$.
2. Determine B_{ℓ_i} (contiguous subarray of A_{ℓ_i}) such that it contains all nodes of the form (ℓ_i, ℓ_{i+1}, z).
3. Let $\mathcal{P}$ be projection onto third coordinate.

$$\sigma \in K \iff \bigcap_{0 \leq i \leq d_\sigma} \mathcal{P}(B_{\ell_i}) \neq \emptyset.$$

Insertion on SAL

Input: A maximal simplex $\sigma = v_{\ell_0} \cdots v_{\ell_{d_\sigma}}$.

Task: Insert σ in K.

1. Remove all $\tau \in K$ which were maximal but are now contained in K.
   1.1 For every edge $e \subseteq \sigma$ compute Z_e:
   $$Z_e = \{\tau \in K \mid \tau \text{is maximal, } e \subseteq \tau\}.$$
   1.2 Check if any simplex $\underset{e\in\sigma}{\cup} Z_e$ is in σ. If yes then, remove them.
2. Build connected component for σ in SAL.
3. Updating the arrays A_{ℓ_i}.

Removal on SAL

Input: A simplex $\sigma = v_{\ell_0} \cdots v_{\ell_{d_\sigma}}$.

Task: Remove σ from K.

1. Obtain the set Z_σ of maximal simplices in K which contain σ.
2. For every $\tau \in Z_\sigma$, remove τ from K and insert the facets of τ which do not contain σ.

SAL Operations Summary

	Simplex Tree	Simplex Array List
Storage	$\mathcal{O}(k2^d \log n)$	$\mathcal{O}(kd^3(\log n + \log k))$
Membership	$\mathcal{O}(d_\sigma \log n)$	$\mathcal{O}(d_\sigma \lambda \log(kd))$
Insertion	$\mathcal{O}(2^{d_\sigma} d_\sigma \log n)$	$\mathcal{O}(d_\sigma^3 \lambda(d_\sigma + \log(kd)))$
Removal	$\mathcal{O}(k2^d \log n)$	$\mathcal{O}(d_\sigma d^3 \lambda \log(kd))$

SAL Operations Summary

	Simplex Tree	Simplex Array List
Storage	$\mathcal{O}(k2^d \log n)$	$\mathcal{O}(kd^3(\log n + \log k))$
Membership	$\mathcal{O}(d_\sigma \log n)$	$\mathcal{O}(d_\sigma \lambda \log(kd))$
Insertion	$\mathcal{O}(2^{d_\sigma} d_\sigma \log n)$	$\mathcal{O}(d_\sigma^3 \lambda(d_\sigma + \log(kd)))$
Removal	$\mathcal{O}(k2^d \log n)$	$\mathcal{O}(d_\sigma d^3 \lambda \log(kd))$

SAL Operations Summary

	Simplex Tree	Simplex Array List
Storage	$\mathcal{O}(k2^d \log n)$	$\mathcal{O}(kd^3(\log n + \log k))$
Membership	$\mathcal{O}(d_\sigma \log n)$	$\mathcal{O}(d_\sigma \lambda \log(kd))$
Insertion	$\mathcal{O}(2^{d_\sigma} d_\sigma \log n)$	$\mathcal{O}(d_\sigma^3 \lambda(d_\sigma + \log(kd)))$
Removal	$\mathcal{O}(k2^d \log n)$	$\mathcal{O}(d_\sigma d^3 \lambda \log(kd))$

SAL Operations Summary

	Simplex Tree	Simplex Array List
Storage	$\mathcal{O}(k2^d \log n)$	$\mathcal{O}(kd^3(\log n + \log k))$
Membership	$\mathcal{O}(d_\sigma \log n)$	$\mathcal{O}(d_\sigma \lambda \log(kd))$
Insertion	$\mathcal{O}(2^{d_\sigma} d_\sigma \log n)$	$\mathcal{O}(d_\sigma^3 \lambda (d_\sigma + \log(kd)))$
Removal	$\mathcal{O}(k2^d \log n)$	$\mathcal{O}(d_\sigma d^3 \lambda \log(kd))$

SAL Operations Summary

	Simplex Tree	Simplex Array List
Storage	$\mathcal{O}(k2^{d} \log n)$	$\mathcal{O}(kd^{3}(\log n + \log k))$
Membership	$\mathcal{O}(d_\sigma \log n)$	$\mathcal{O}(d_\sigma \lambda \log(kd))$
Insertion	$\mathcal{O}(2^{d_\sigma} d_\sigma \log n)$	$\mathcal{O}(d_\sigma^3 \lambda(d_\sigma + \log(kd)))$
Removal	$\mathcal{O}(k2^{d} \log n)$	$\mathcal{O}(d_\sigma d^3 \lambda \log(kd))$

SAL vs ST: Experiments

- Marc Glisse and Sivaprasad implemented SAL.
- Data Set: Rips Complex from sampling of Klein bottle in $\mathbb{R}^5$.
- Operations: Insertion and removal of random simplices, and contraction of randomly chosen edges.

SAL vs ST: Experiments

- Marc Glisse and Sivaprasad implemented SAL.
- Data Set: Rips Complex from sampling of Klein bottle in $\mathbb{R}^5$.
- Operations: Insertion and removal of random simplices, and contraction of randomly chosen edges.

No	n	α	Average d_σ	k	ST Time (s)	SAL Time (s)
1	1,000	0.3	11.78	4,299	72	34
2	2,500	0.3	13.77	15,605	Killed	76
3	10,000	0.2	6.9	29,676	Killed	52

Thank you!

1-SAL vs 0-SAL: Experiments

- Marc Glisse and Sivaprasad implemented SAL.
- Data Set: Rips Complex from sampling of Klein bottle in $\mathbb{R}^5$.
- Operations: Insertion and removal of random simplices, and contraction of randomly chosen edges.

No	n	α	Average d_σ	k	ST Time (s)	0-SAL Time (s)	1-SAL Time (s)
1	1,000	0.3	11.78	4,299	72	5	34
2	2,500	0.3	13.77	15,605	Killed	66	76
3	10,000	0.2	6.9	29,676	Killed	114	52

- average $\lambda_1 = 2.17$; average $\lambda_0 = 23.25$.