

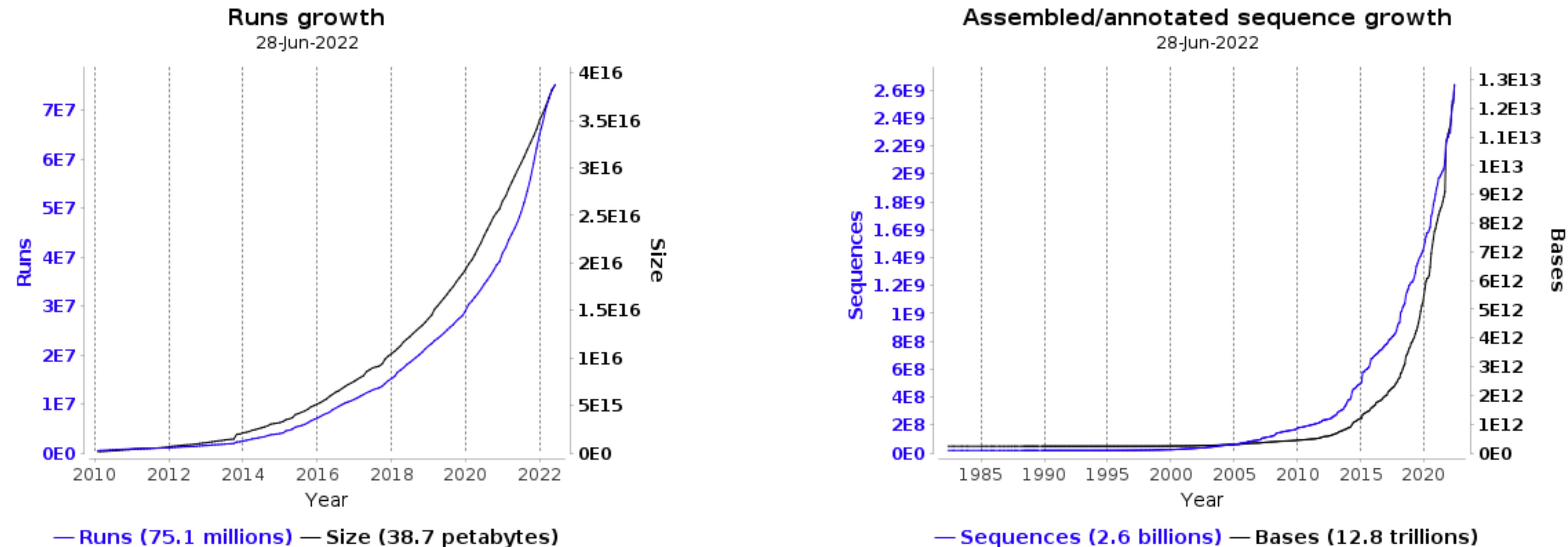
Scalable Indexing of Sequence Data in Annotated Genome Graphs

Mikhail Karasikov

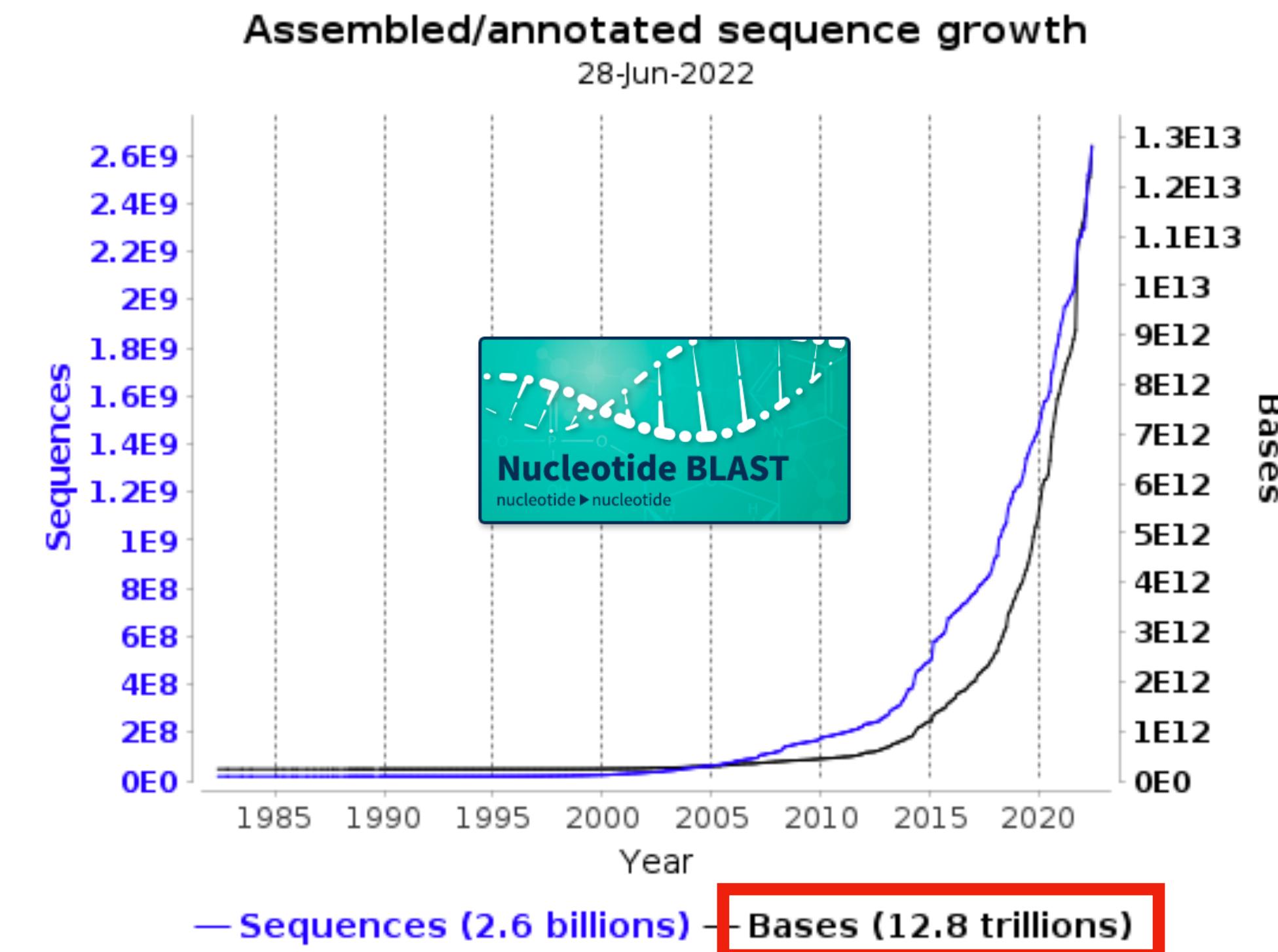
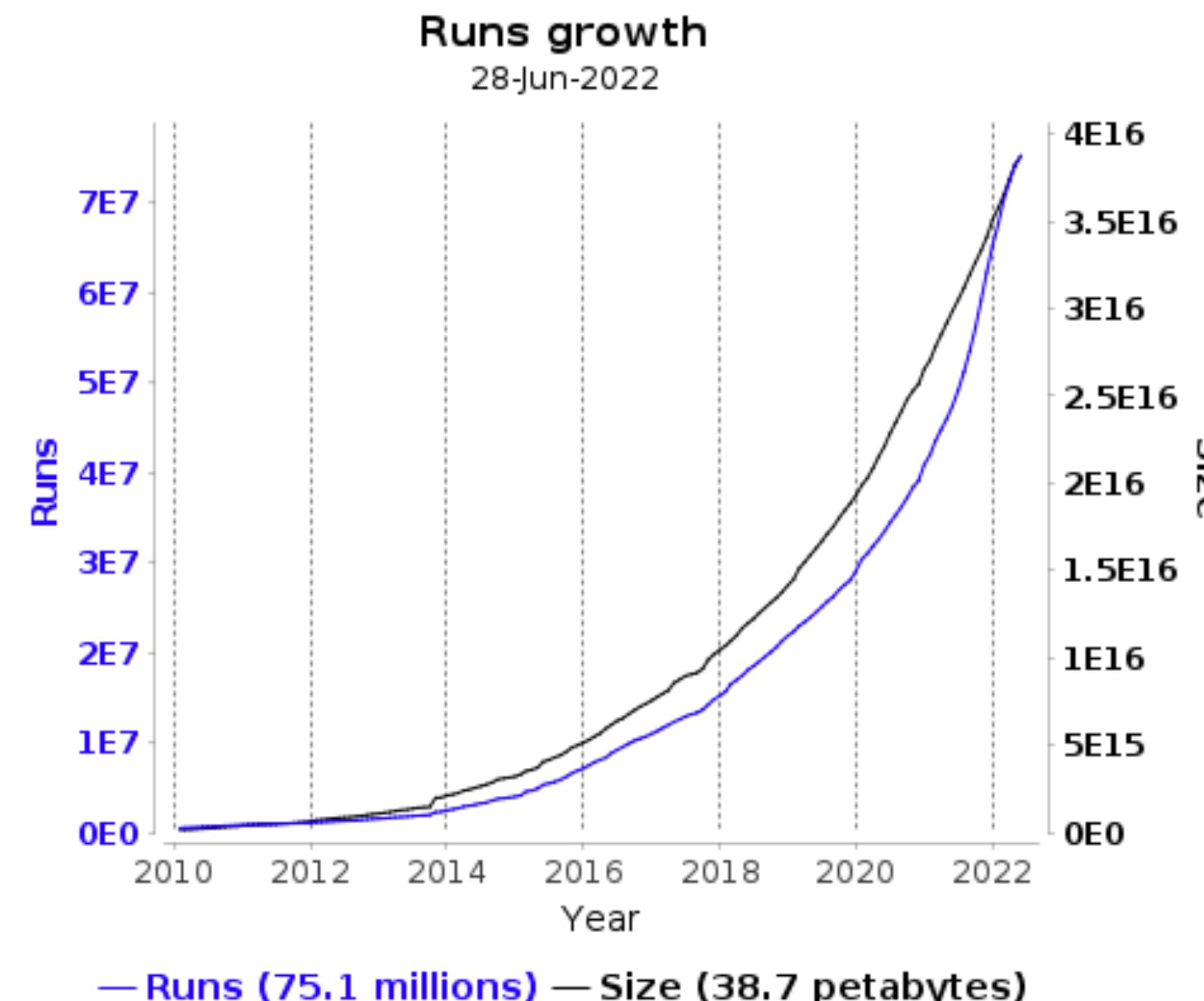
JOBIM 2022

7 July 2022 (Rennes, France)

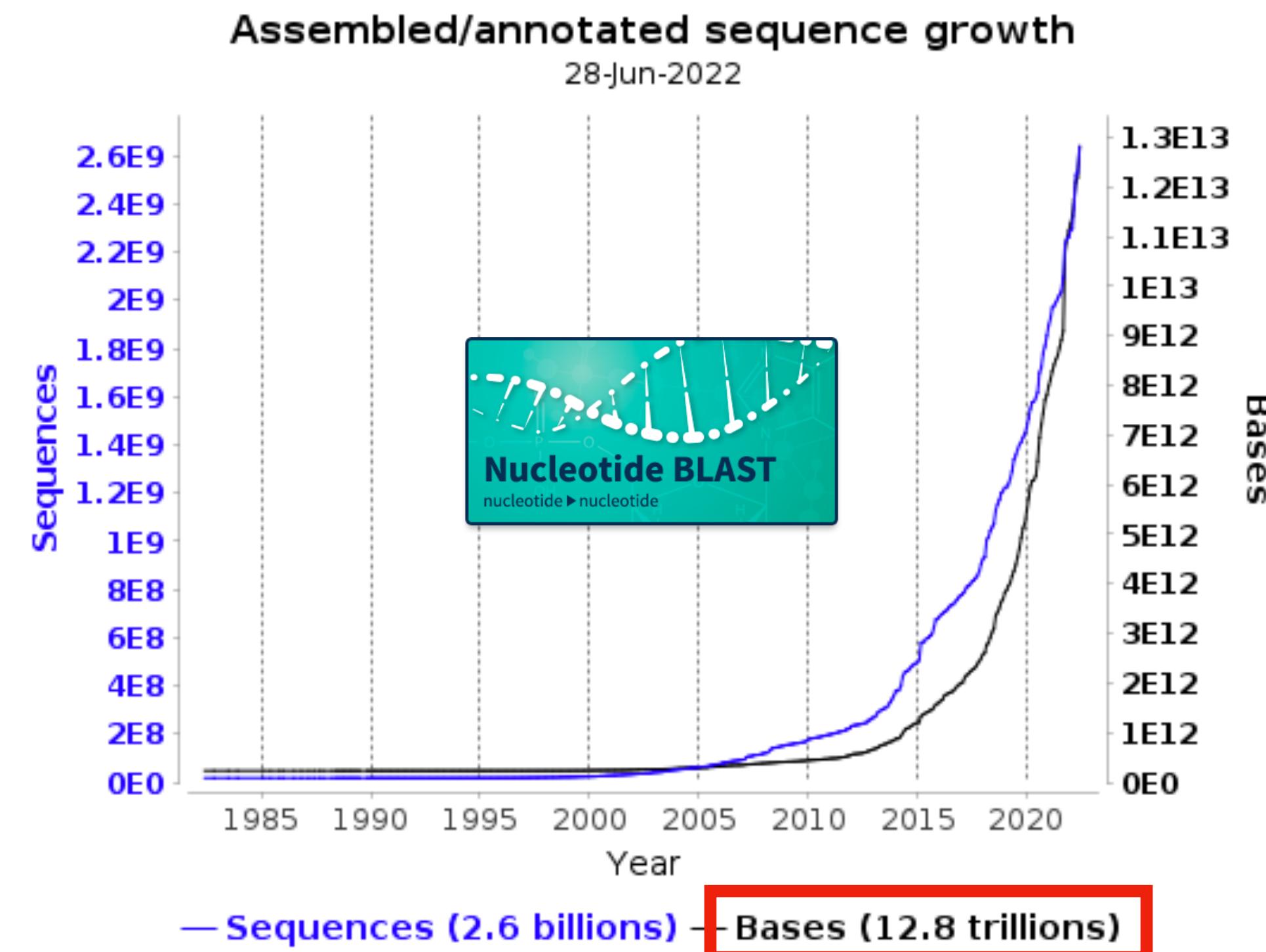
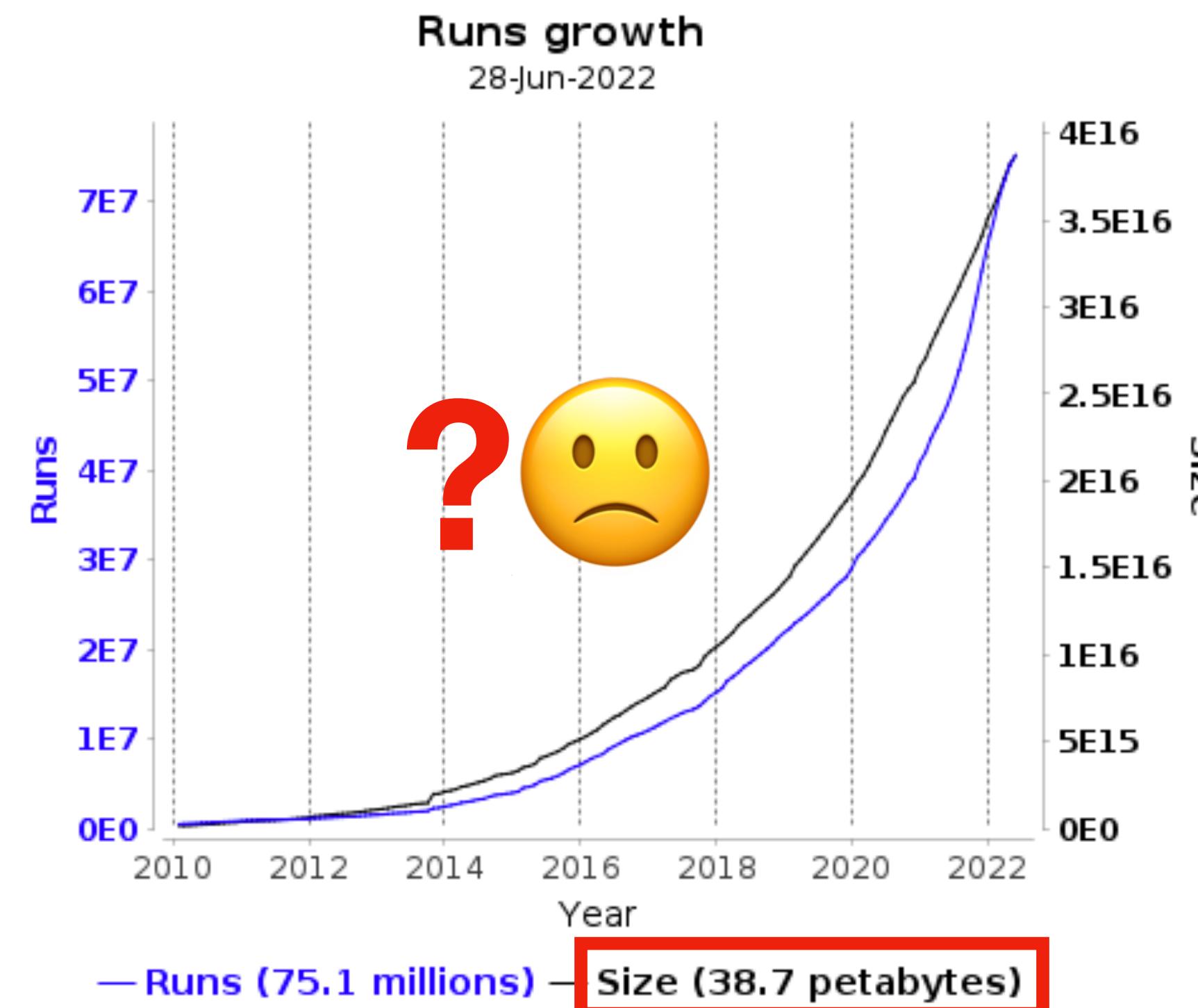
Growth of sequence archives



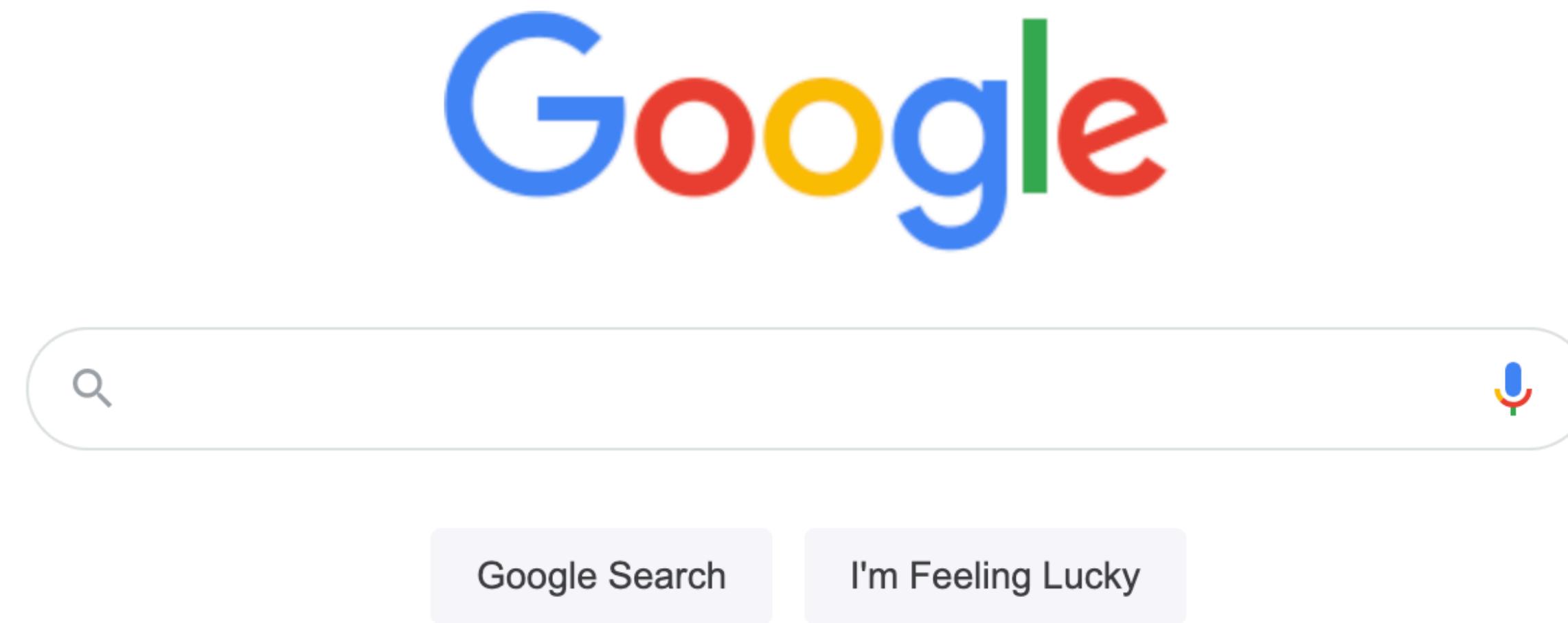
Growth of sequence archives



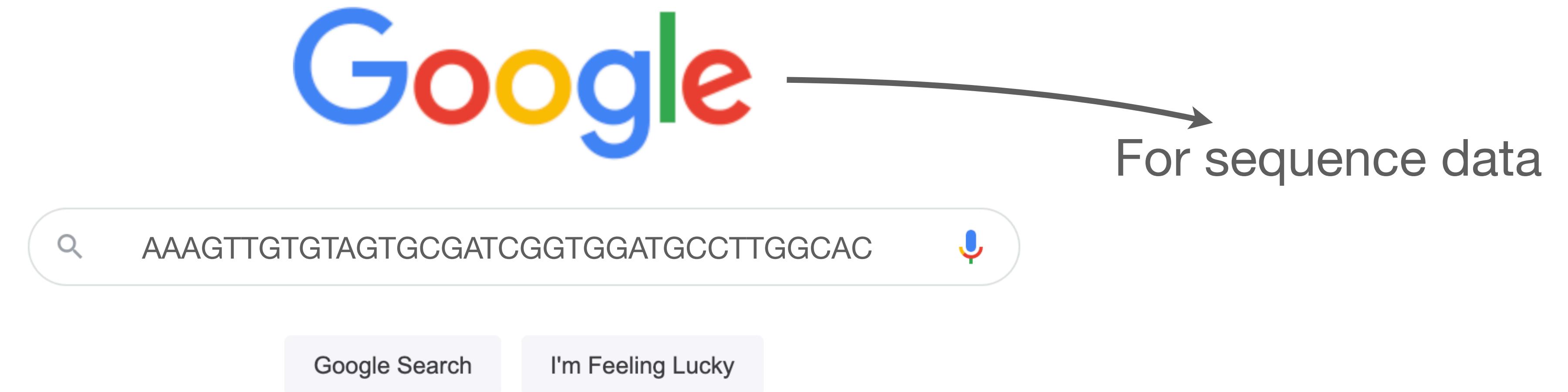
Growth of sequence archives



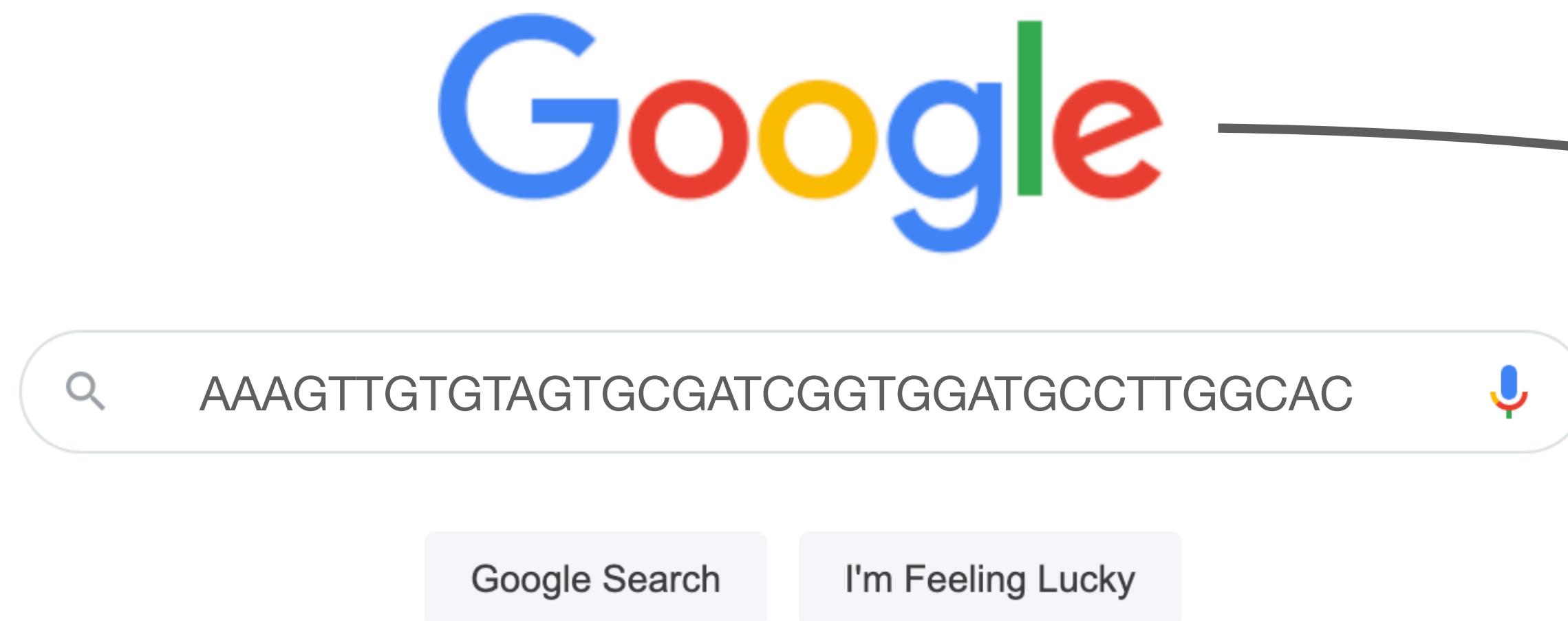
What we want



What we want

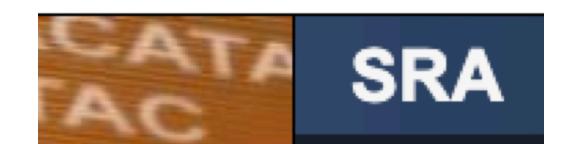


What we want



→

For sequence data



...

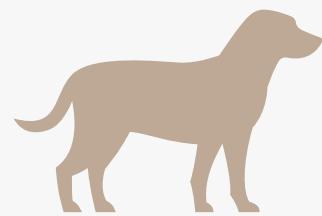
Indexing workflow



>smp_1
ACGTAC
ACGC
CGTAC



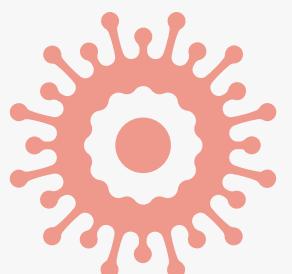
>smp_2
ACGAA
ACGTAC
ACG



>smp_3
ACGTA
GTACT
ACGAT



...

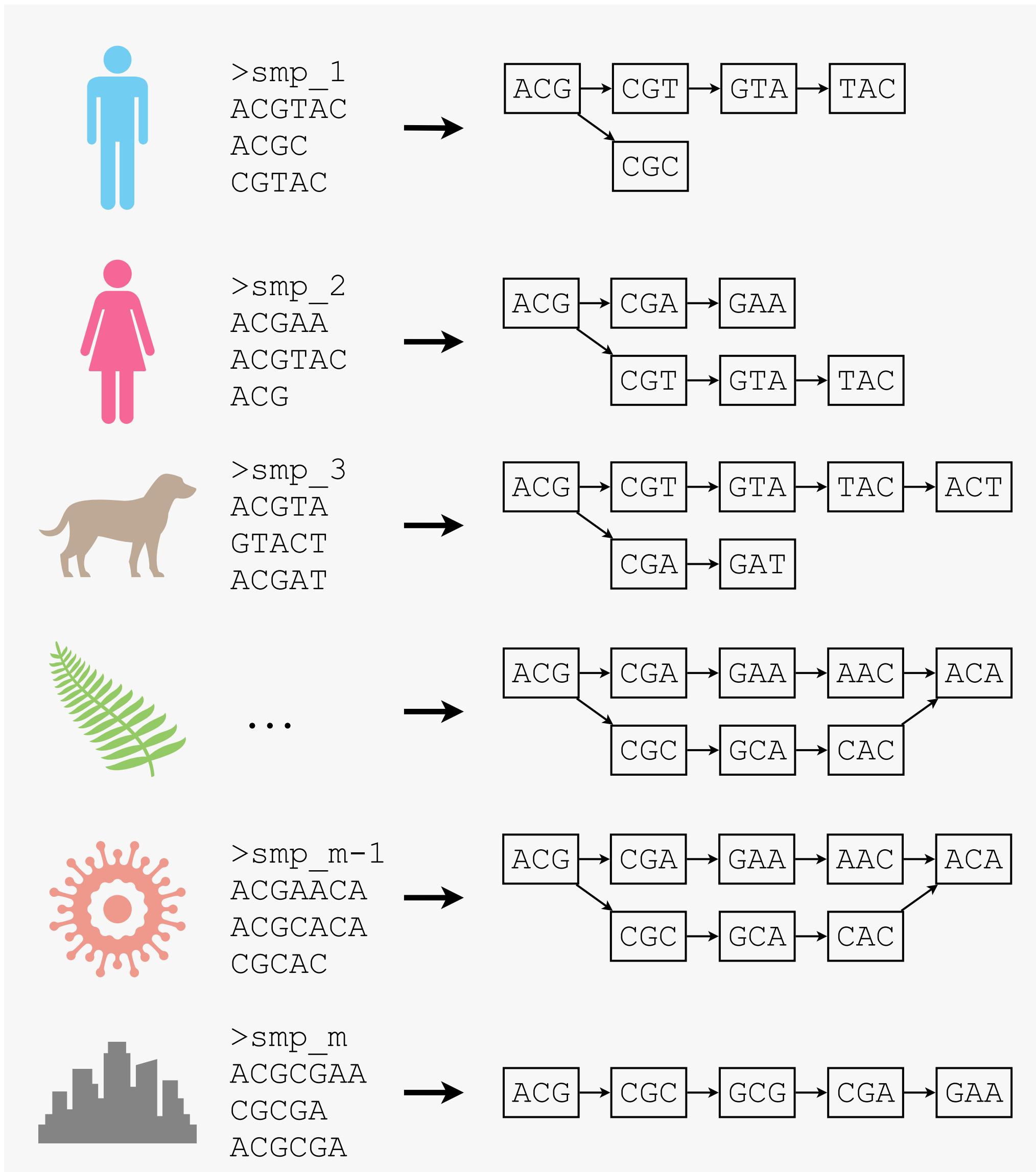


>smp_m-1
ACGAACA
ACGCACA
CGCAC

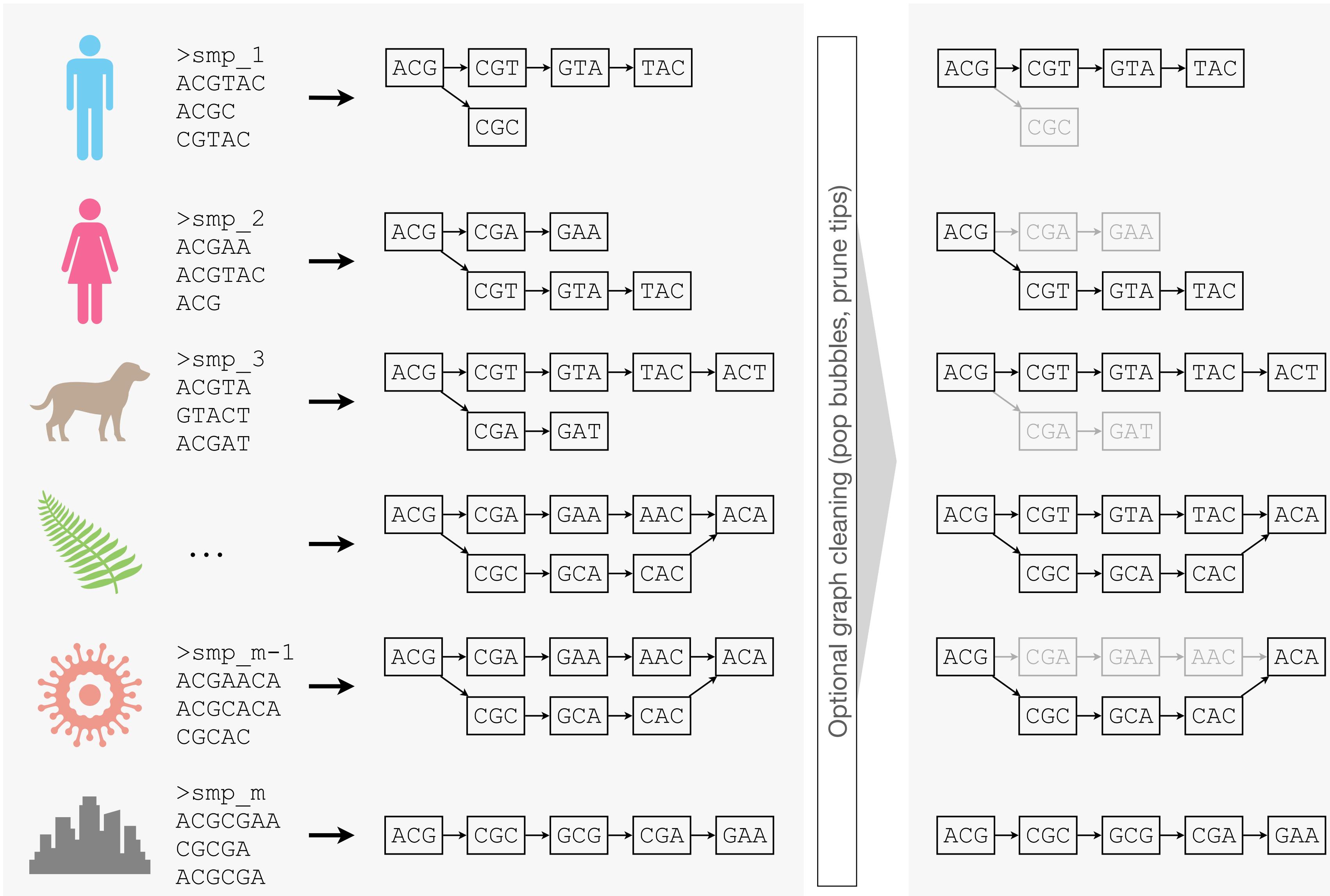


>smp_m
ACGCGAA
CGCGA
ACGCGA

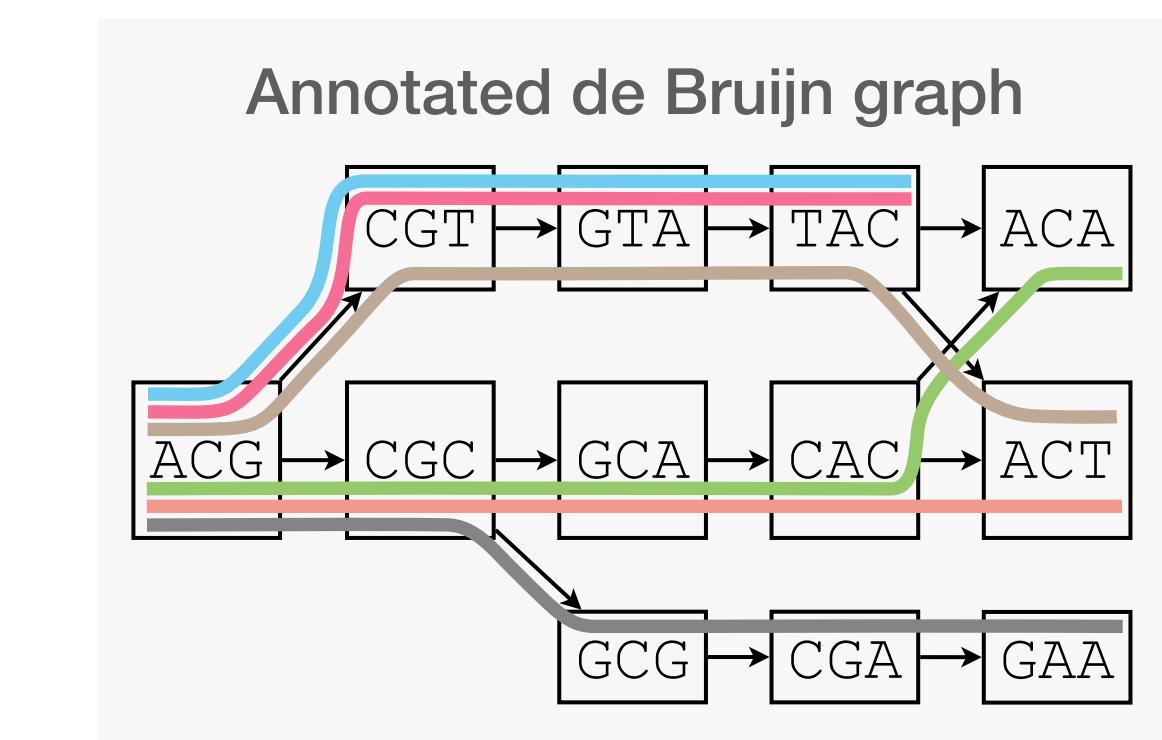
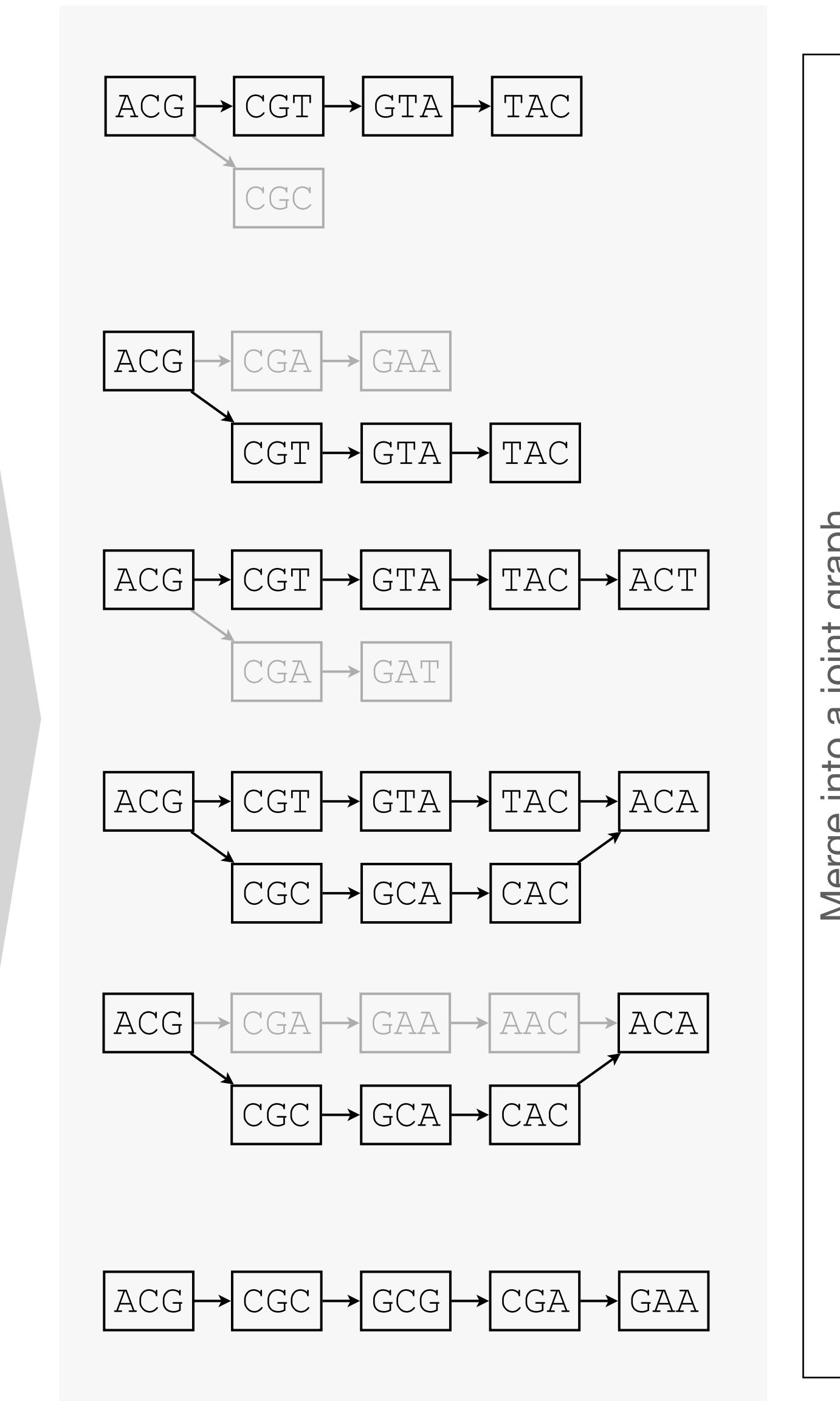
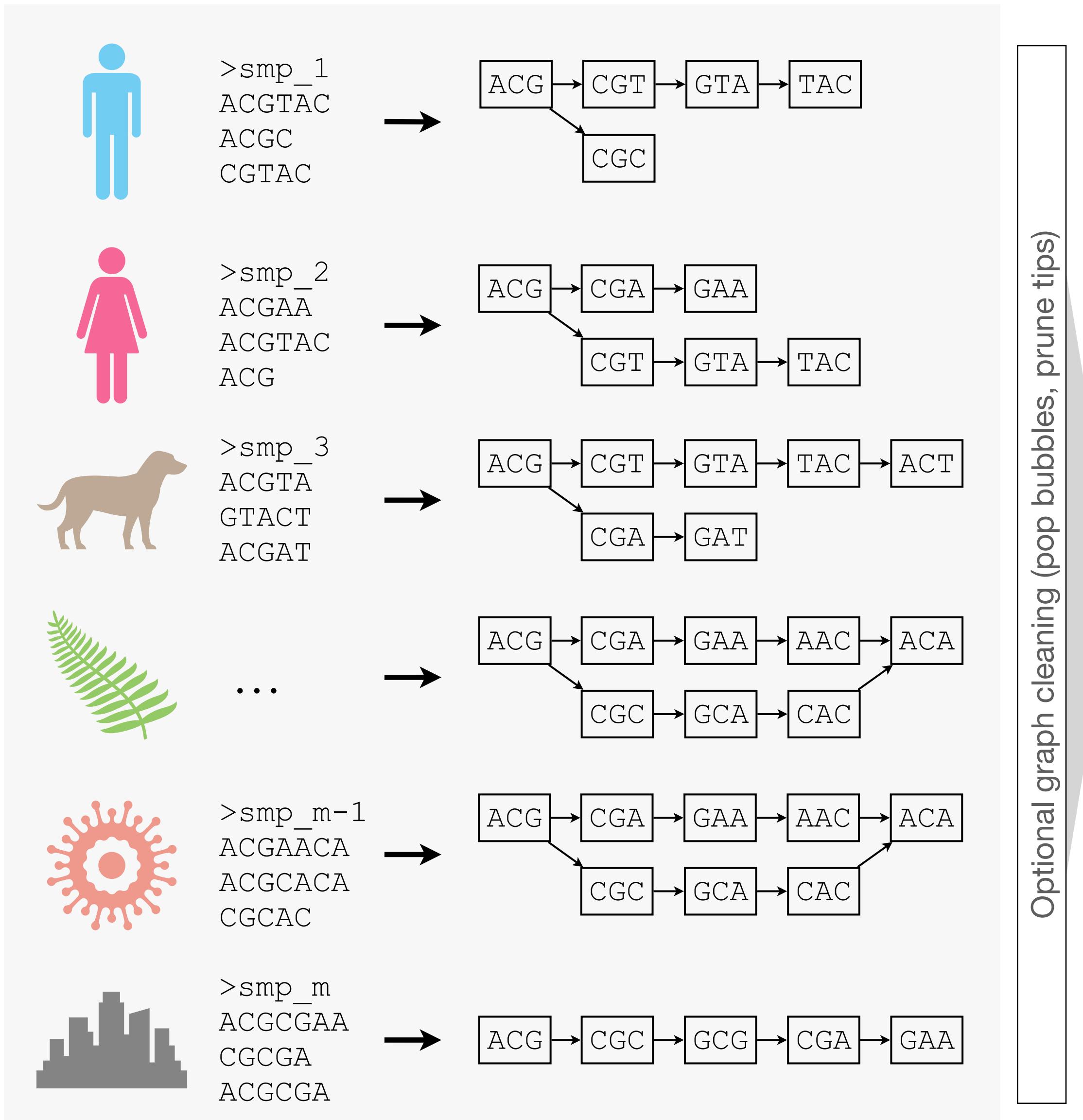
Indexing workflow



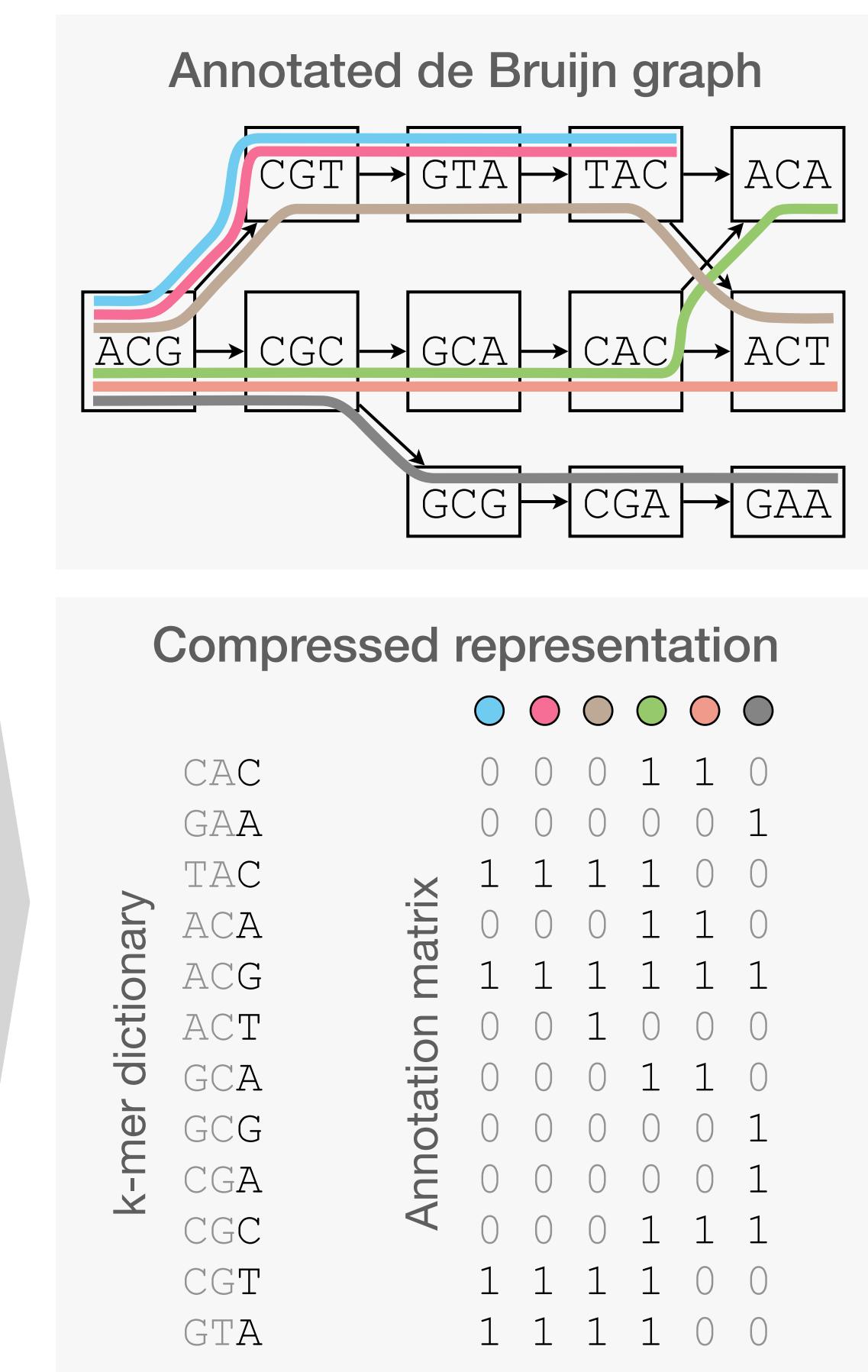
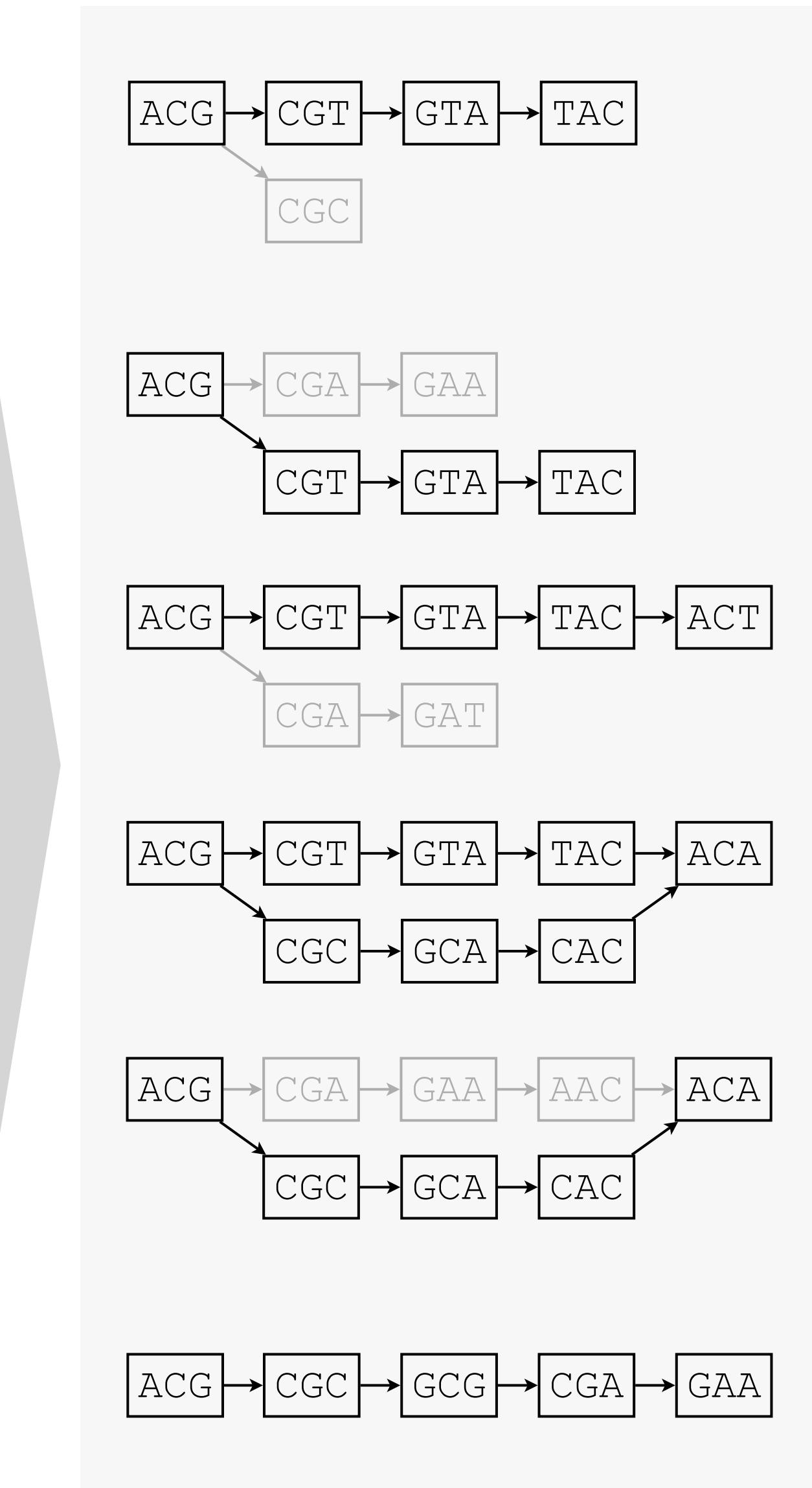
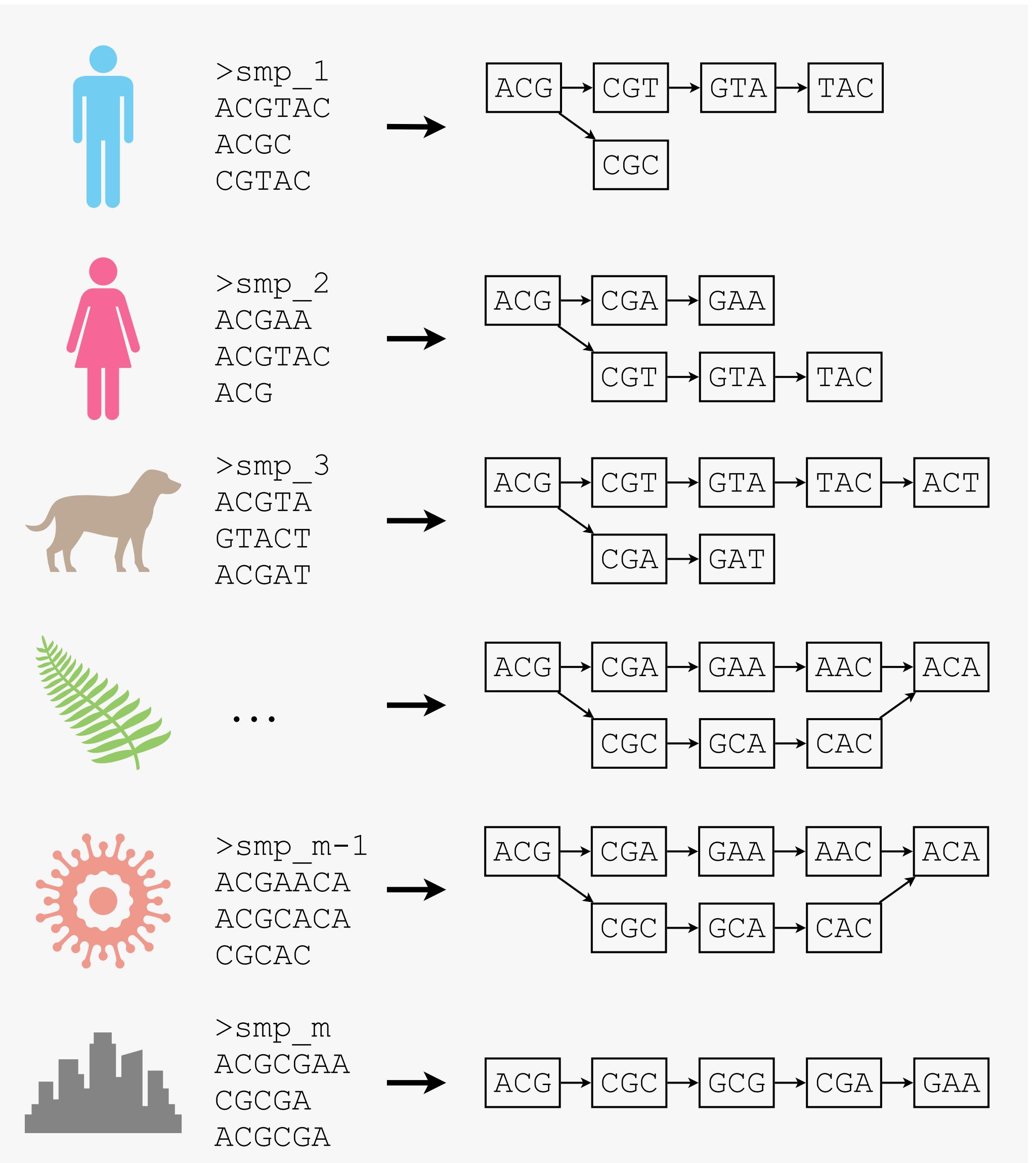
Indexing workflow



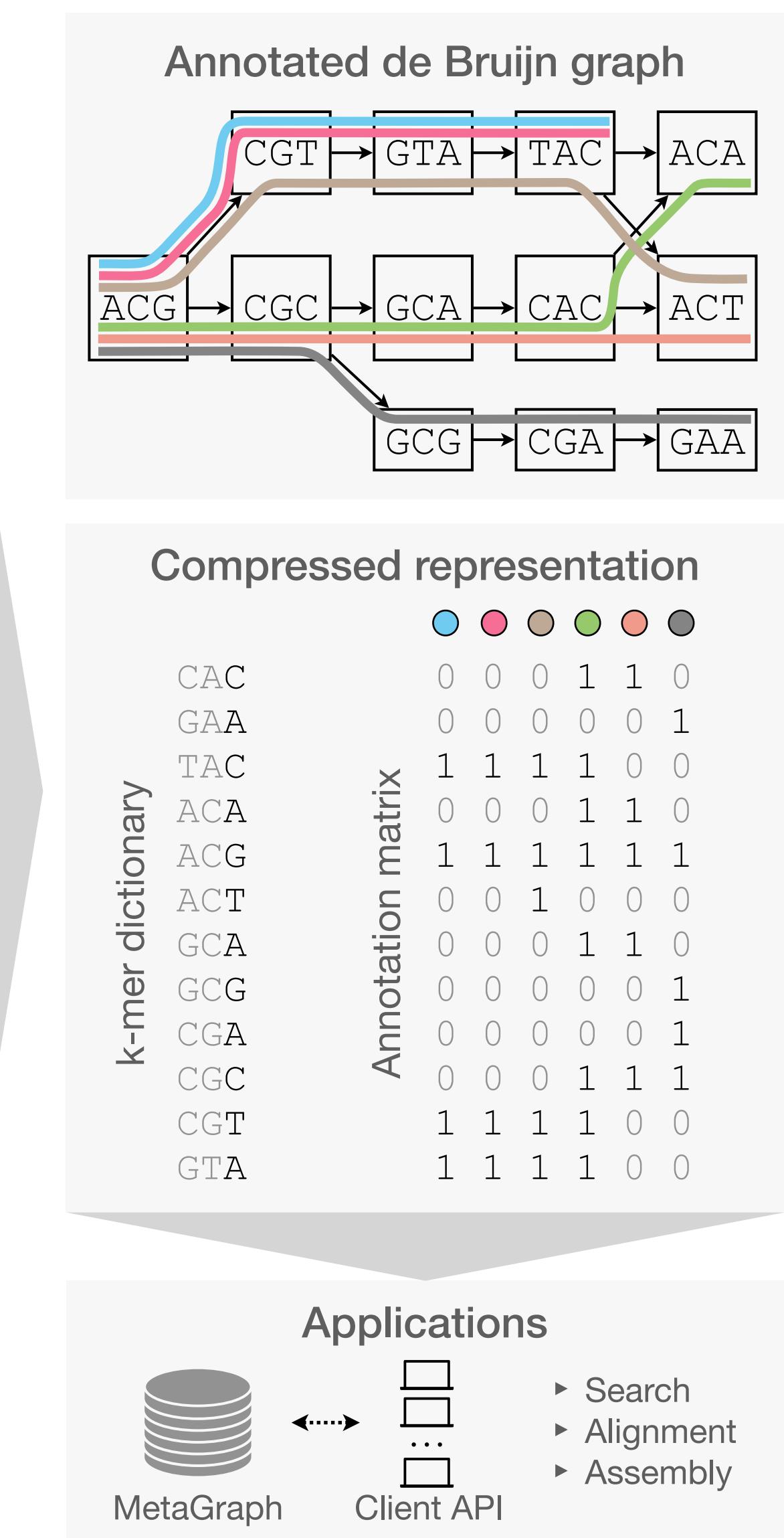
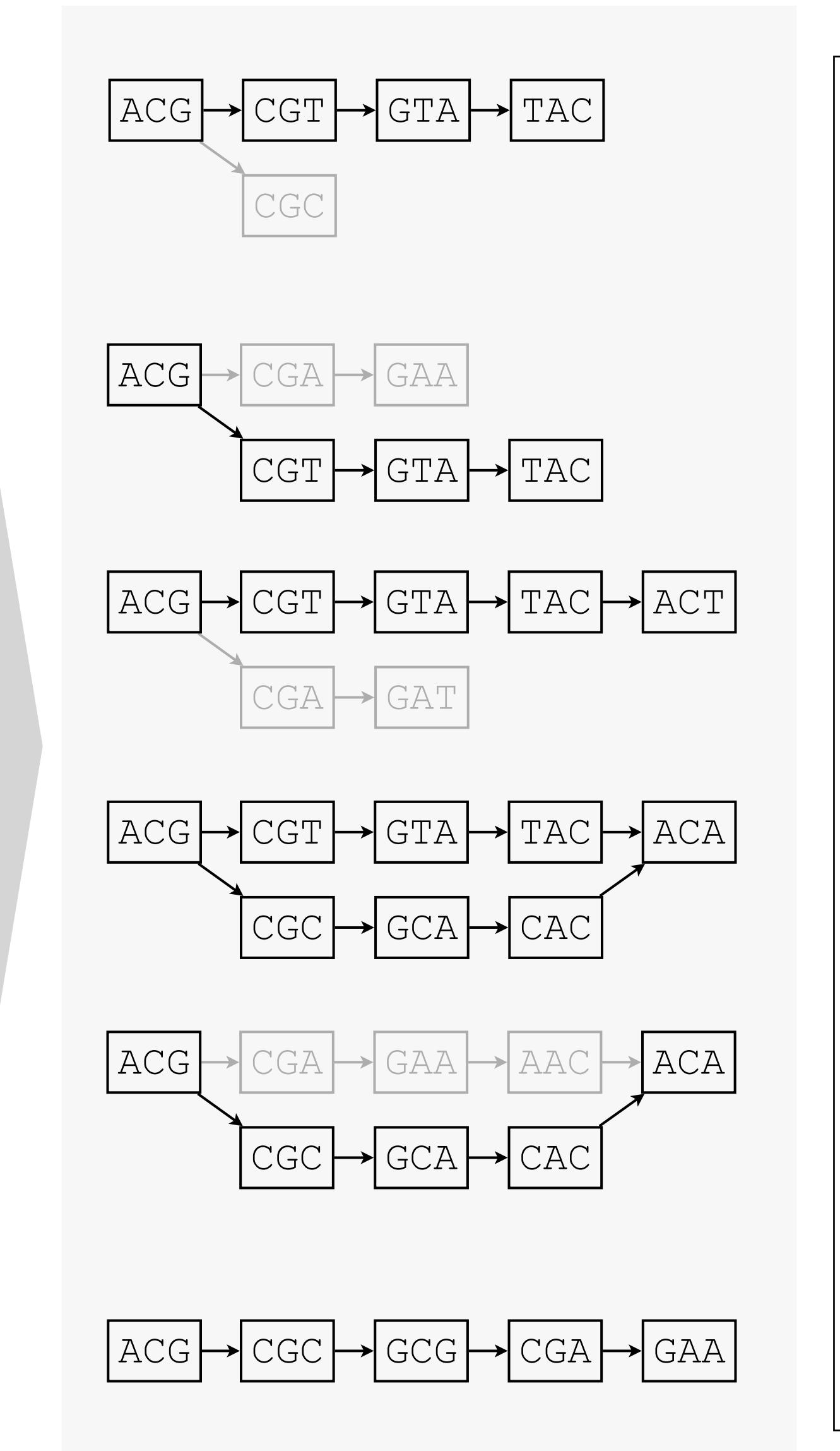
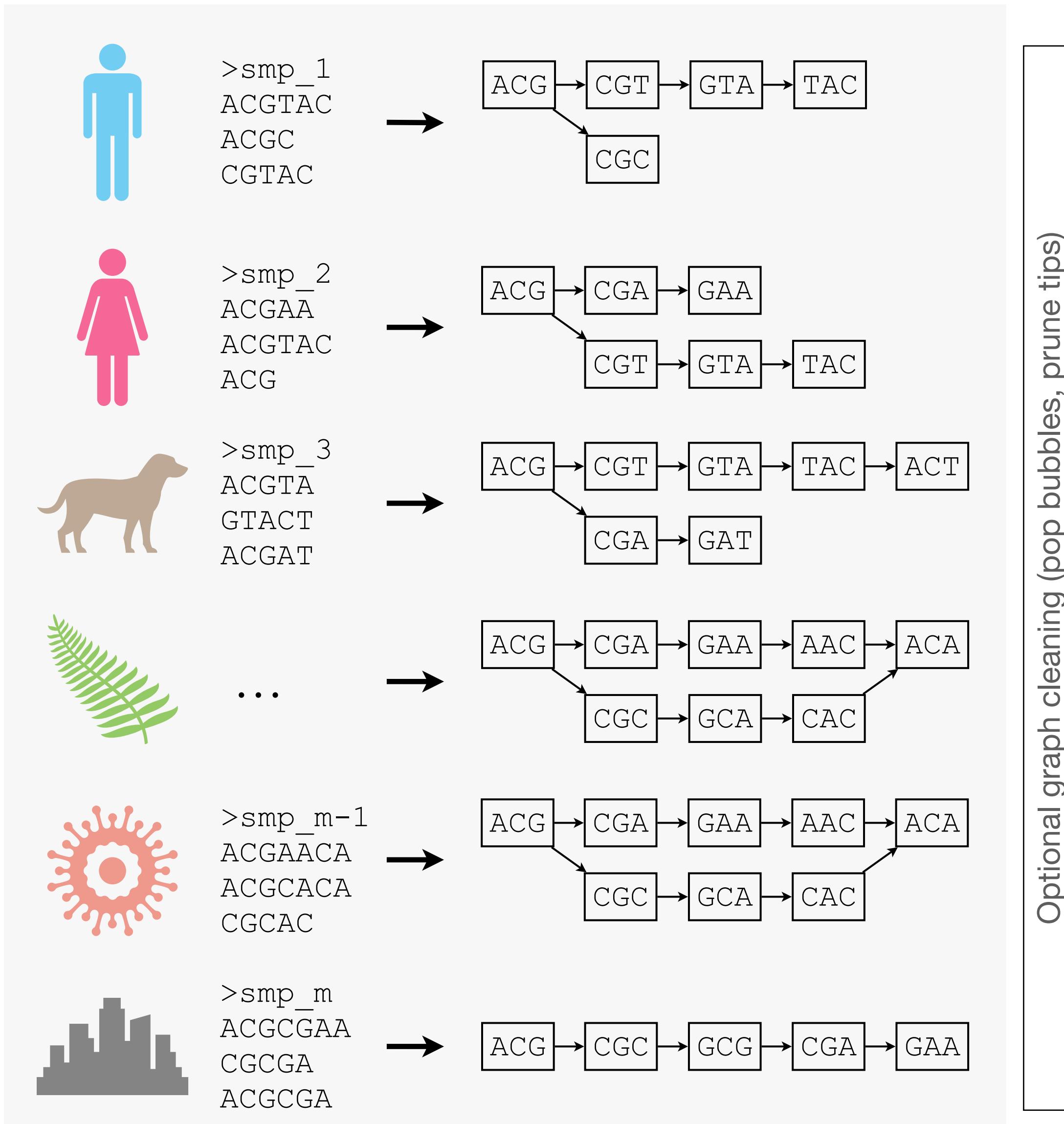
Indexing workflow



Indexing workflow



Indexing workflow



Why MetaGraph

Home Search Align Graphs BioRxiv

MetaGraph: Efficient search for DNA sequences

AAAGTTGTAGTGCATCGTGATGCCCTGGCACCAAGAGCCATGAAGGACGTTGTGACCTGCGATAAGCCTGGGAGTTGGTGACCGAGCTGTGATCCGGGG
TGTCCGAATGGGAAACCTGGAATGTCGGAGTAGTGTCCGGTGGCCCTGCCCTGAATGTATAGGGTGTGGTGGTAACCGGGGAAGTGAAACATCTTAGTACCCG

Select graph:
SRA-Microbe

Minimum k-mer matches: 100%

Search with alignment ?

Search SRA-Microbe

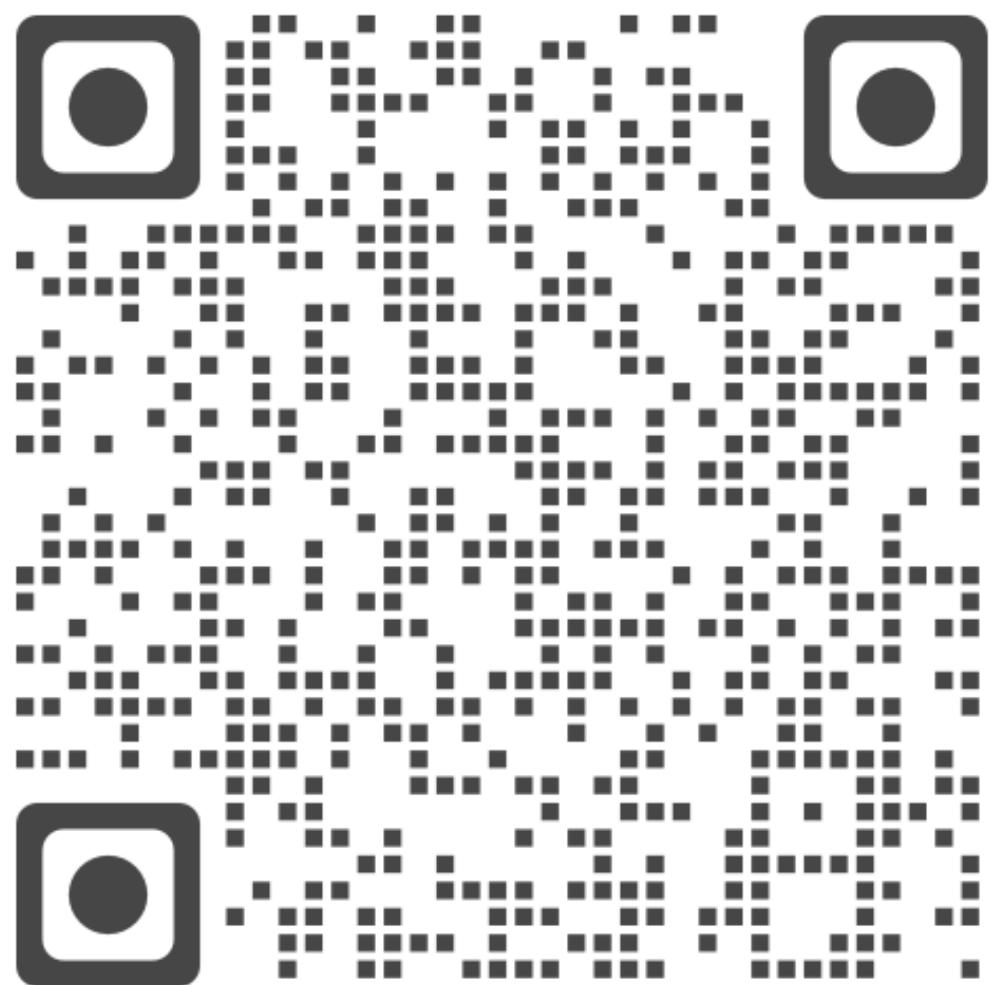
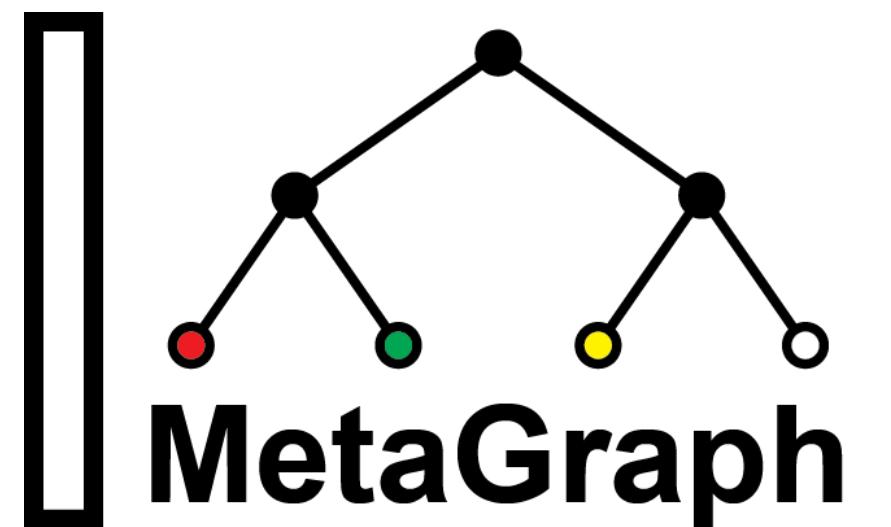
Search results

Show 10 entries

#	sample	k-mer matches
1	SRR849181	450
2	SRR849180	450
3	SRR849179	450
4	SRR039941	450
5	SRR037532	450

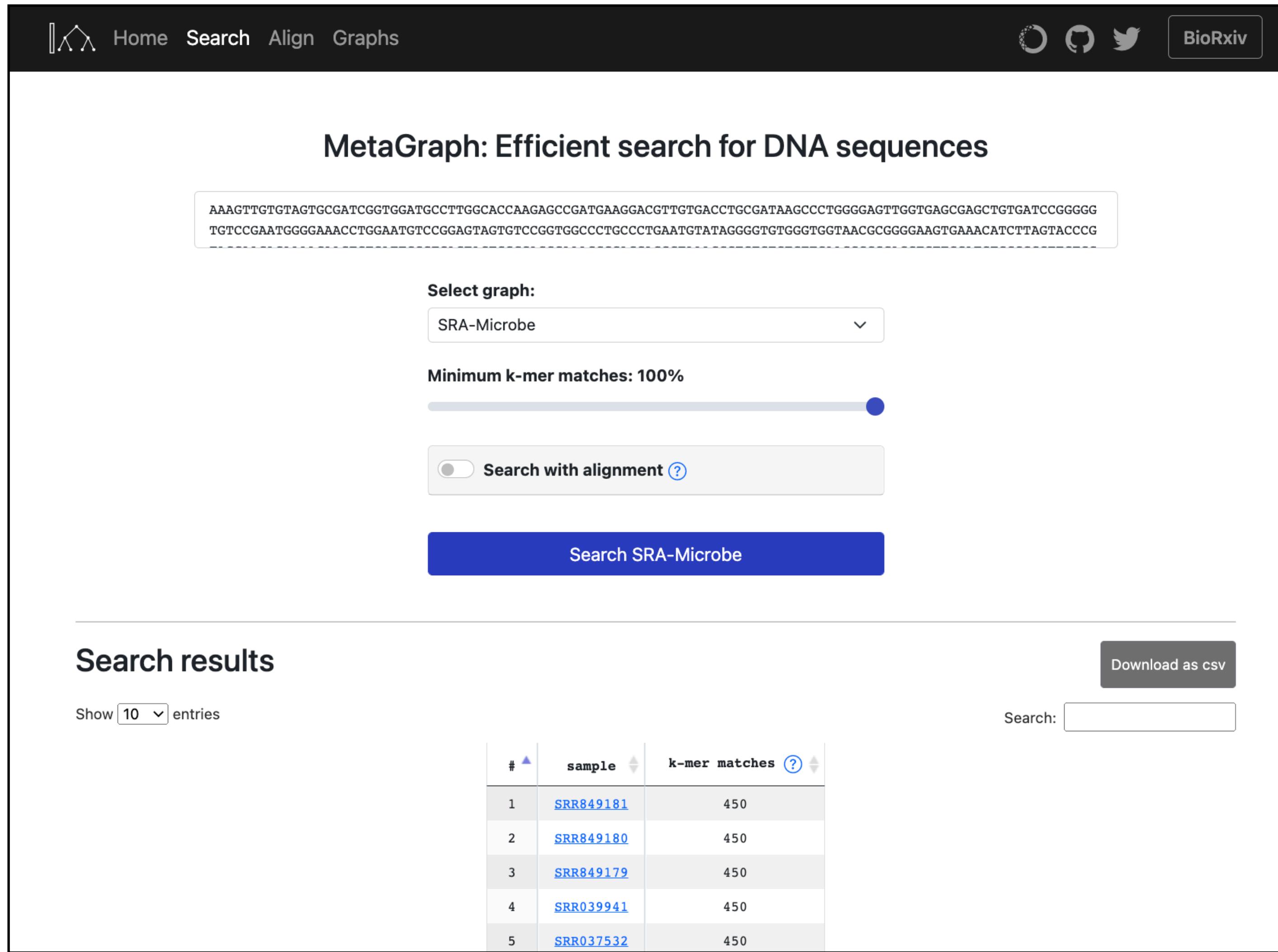
Download as csv

Search:



metagraph.ethz.ch/search

Why MetaGraph



The screenshot shows the MetaGraph search interface. At the top, there's a navigation bar with icons for Home, Search, Align, Graphs, and social media links. A BioRxiv button is also present. Below the navigation, the title "MetaGraph: Efficient search for DNA sequences" is displayed. A sequence input field contains:
AAAGTTGTAGTGCATCGTGATGCCCTGGCACCAAGAGCCATGAAGGACGTTGTGACCTGCGATAAGCCTGGGAGTTGGTGACCGAGCTGTGATCCGGGG
TGTCCGAATGGGAAACCTGGAATGTCGGAGTAGTGTCCGGTGGCCCTGCCCTGAATGTATAGGGTGTGGTAACCGGGAGTGAAACATCTTAGTACCCG

Below the sequence, a "Select graph:" dropdown is set to "SRA-Microbe". A slider indicates "Minimum k-mer matches: 100%". A checkbox for "Search with alignment" is checked. A large blue "Search SRA-Microbe" button is centered.

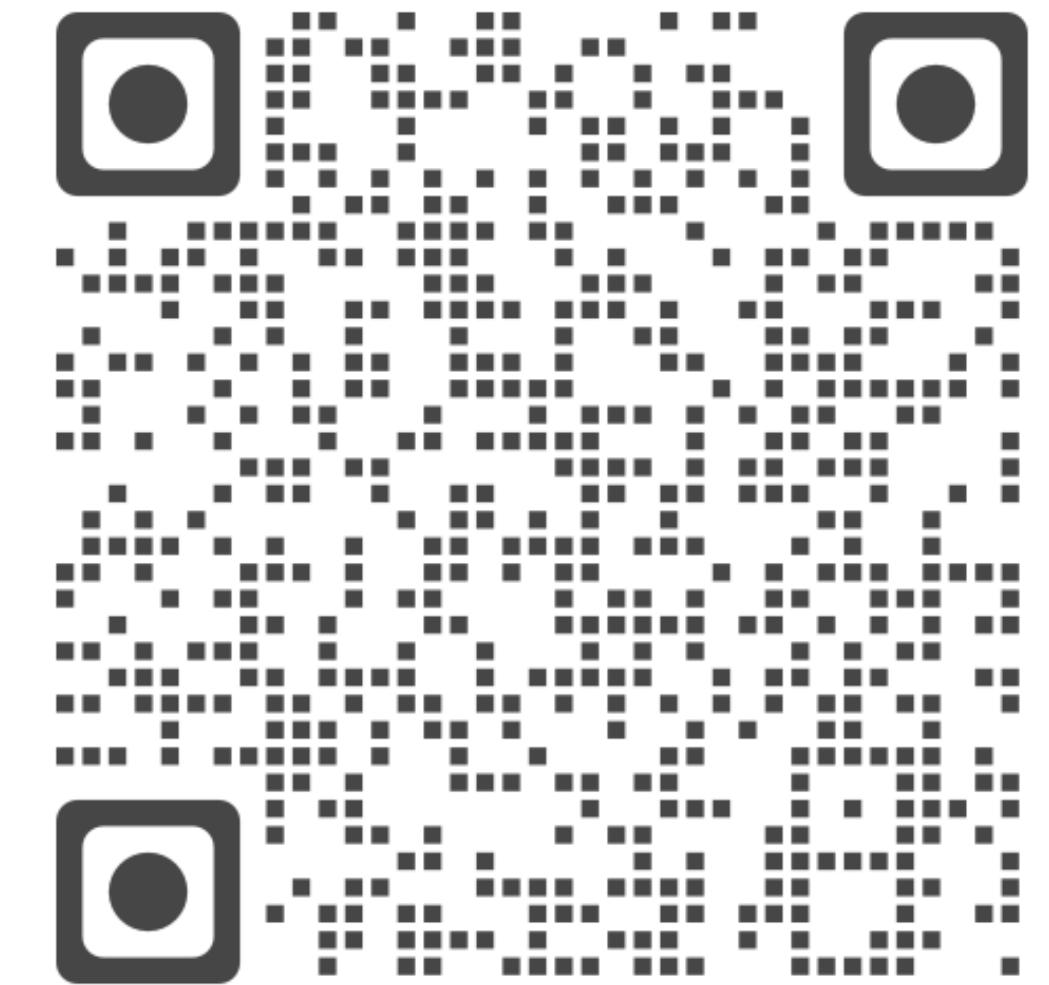
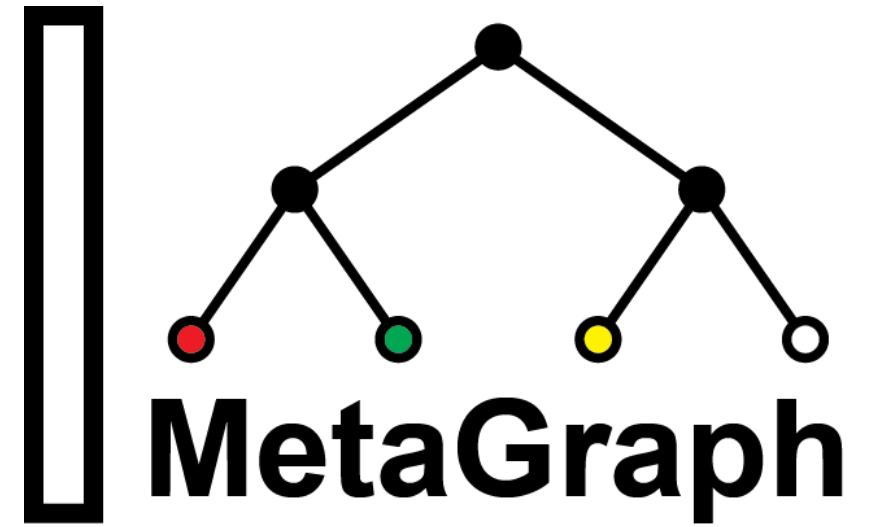
Search results

Show 10 entries

#	sample	k-mer matches
1	SRR849181	450
2	SRR849180	450
3	SRR849179	450
4	SRR039941	450
5	SRR037532	450

Download as csv

Search:



metagraph.ethz.ch/search

Python Client API

```
In [1]: from metagraph.client import GraphClient
SRV = "metagraph.ethz.ch"
PORT = 12345
g1 = GraphClient(SRV, PORT, api_path="/metasub")
g2 = GraphClient(SRV, PORT, api_path="/refseq")
```



```
In [2]: query = "GGCTAACTACGTGCCAGCAGCCGGTAATAC"
g1.search(query, align=True)
```

```
Out [2]:   sample      sequence      score
0  SRR2201245  GGCTAACTACGTGCCAGCAGCCGGTAATAC  64
1  ERR1732568  GGCTAACTACGTGCCAGCAGCCGGTAATAC  64
2  ERR847096   GGCTAACTACGTGCCAGCAGCCGGTAATAC  64
...       ...          ...          ...
```

Why MetaGraph

MetaGraph: Indexing and Analysing Nucleotide Archives at Petabase-scale

Mikhail Karasikov, Harun Mustafa, Daniel Danciu, Marc Zimmermann, Christopher Barber, Gunnar Rätsch, André Kahles
doi: <https://doi.org/10.1101/2020.10.01.322164>

Home Search Align Graphs BioRxiv

MetaGraph: Efficient search for DNA sequences

AAAGTTGTAGTGCATCGTGATGCCCTGGCACCAAGAGCCATGAAGGACGTTGTGACCTGCGATAAGCCTGGGAGTTGGTGACCGAGCTGTGATCCGGGG
TGTCCGAATGGGAAACCTGGAATGTCGGAGTAGTGTCGGTGGCCCTGCCCTGAATGTAGGGTGTTGGTAACCGGGAGTGAAACATCTTAGTACCCG

Select graph:
SRA-Microbe

Minimum k-mer matches: 100%

Search with alignment ?

Search SRA-Microbe

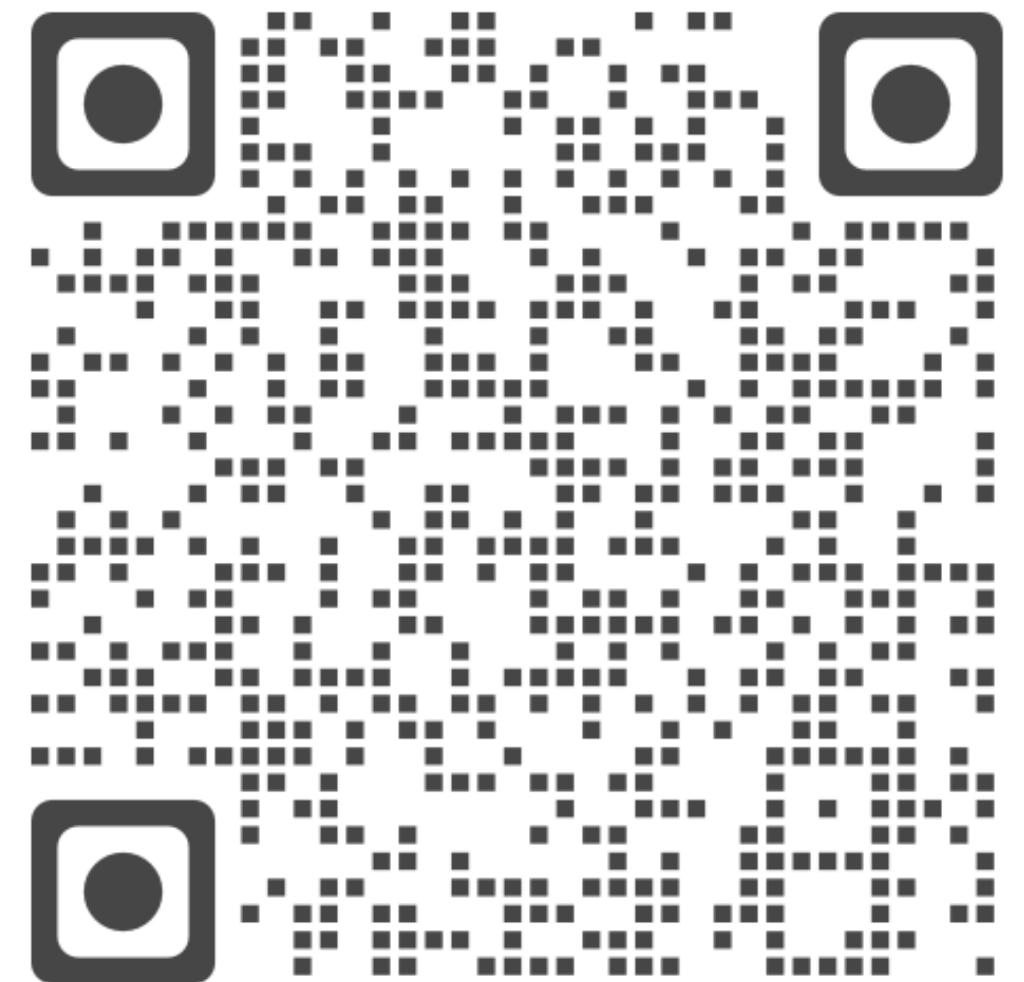
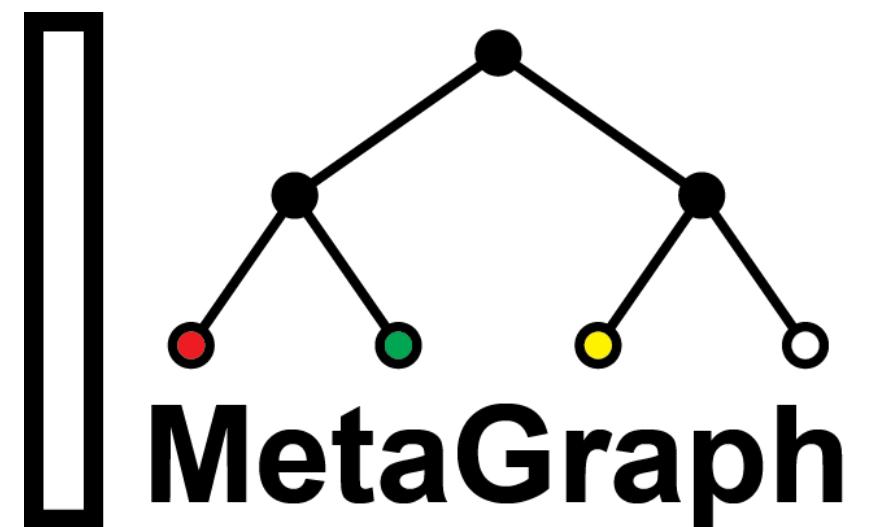
Search results

Show 10 entries

#	sample	k-mer matches
1	SRR849181	450
2	SRR849180	450
3	SRR849179	450
4	SRR039941	450
5	SRR037532	450

Download as csv

Search:



metagraph.ethz.ch/search

Python Client API

```
In [1]: from metagraph.client import GraphClient
SRV = "metagraph.ethz.ch"
PORT = 12345
g1 = GraphClient(SRV, PORT, api_path="/metasub")
g2 = GraphClient(SRV, PORT, api_path="/refseq")
```

```
In [2]: query = "GGCTAACTACGTGCCAGCAGCCGGTAATAC"
g1.search(query, align=True)
```

	sample	sequence	score
0	SRR2201245	GGCTAACTACGTGCCAGCAGCCGGTAATAC	64
1	ERR1732568	GGCTAACTACGTGCCAGCAGCCGGTAATAC	64
2	ERR847096	GGCTAACTACGTGCCAGCAGCCGGTAATAC	64
...

Why MetaGraph

MetaGraph: Indexing and Analysing Nucleotide Archives at Petabase-scale

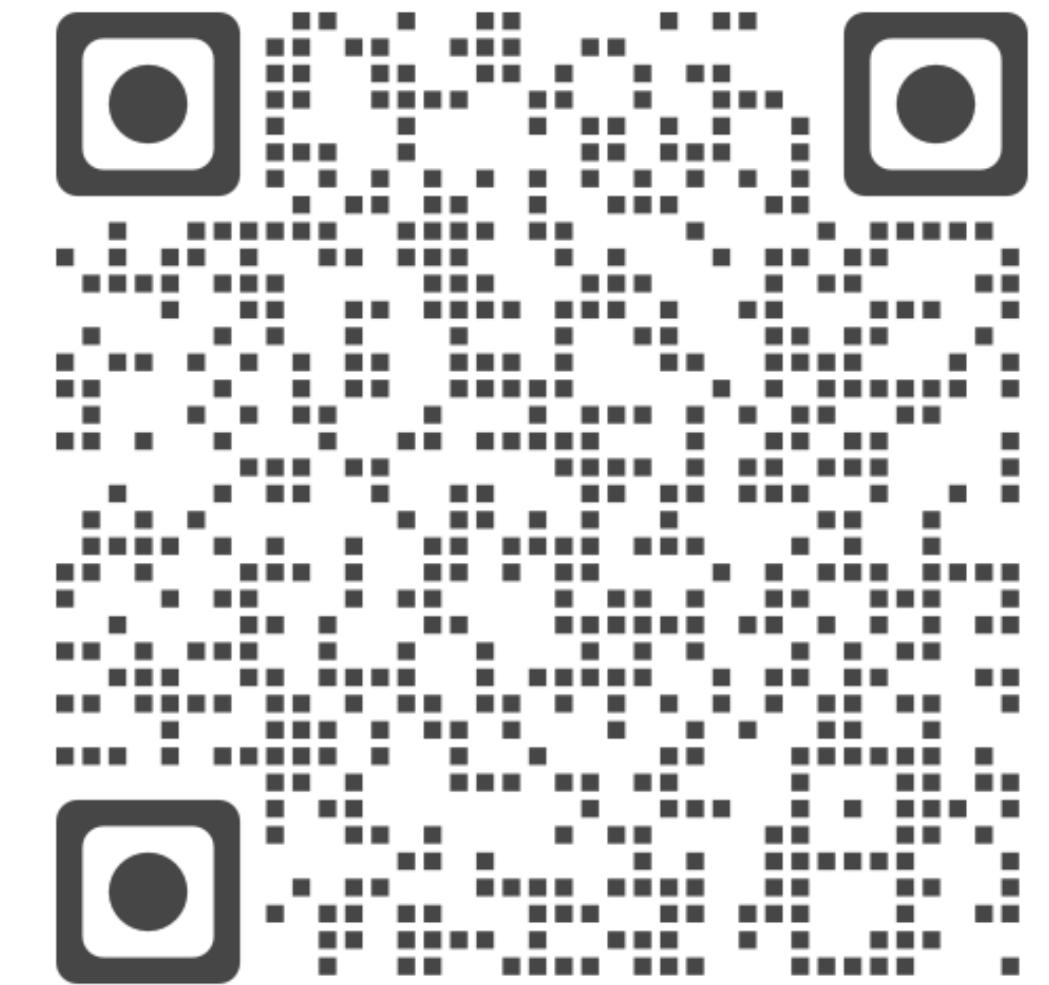
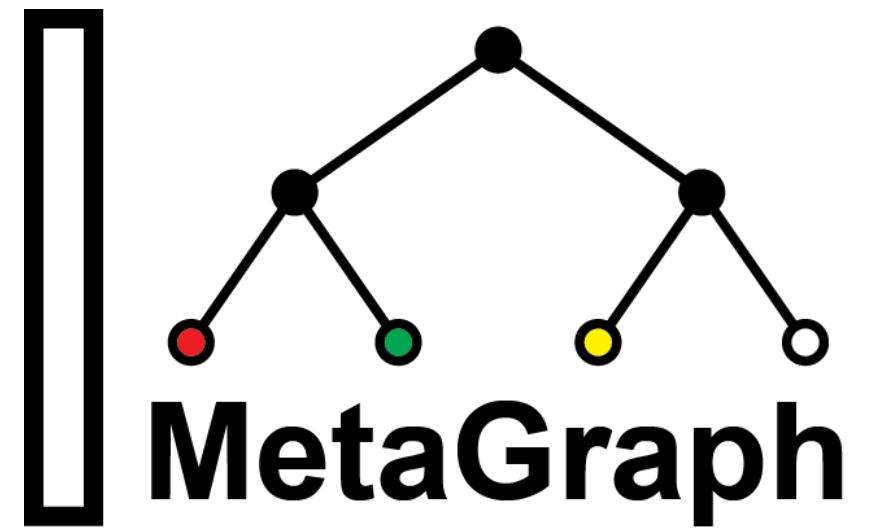
 Mikhail Karasikov,  Harun Mustafa, Daniel Danciu, Marc Zimmermann, Christopher Barber,

 Gunnar Rätsch,  André Kahles

doi: <https://doi.org/10.1101/2020.10.01.322164>

- ▶ MetaGraph is an open source modular framework

- various (succinct) graph representations (Bowe *et al.*, 2012; Conway *et al.*, 2011)
- different schemes for annotation representation (Almodaresi *et al.*, 2017, Muggli *et al.*, 2017, Karasikov *et al.*, 2019, Danciu *et al.*, 2021, Karasikov *et al.*, 2022, ...)
- also available as an Anaconda package or Docker container



metagraph.ethz.ch/search

Why MetaGraph

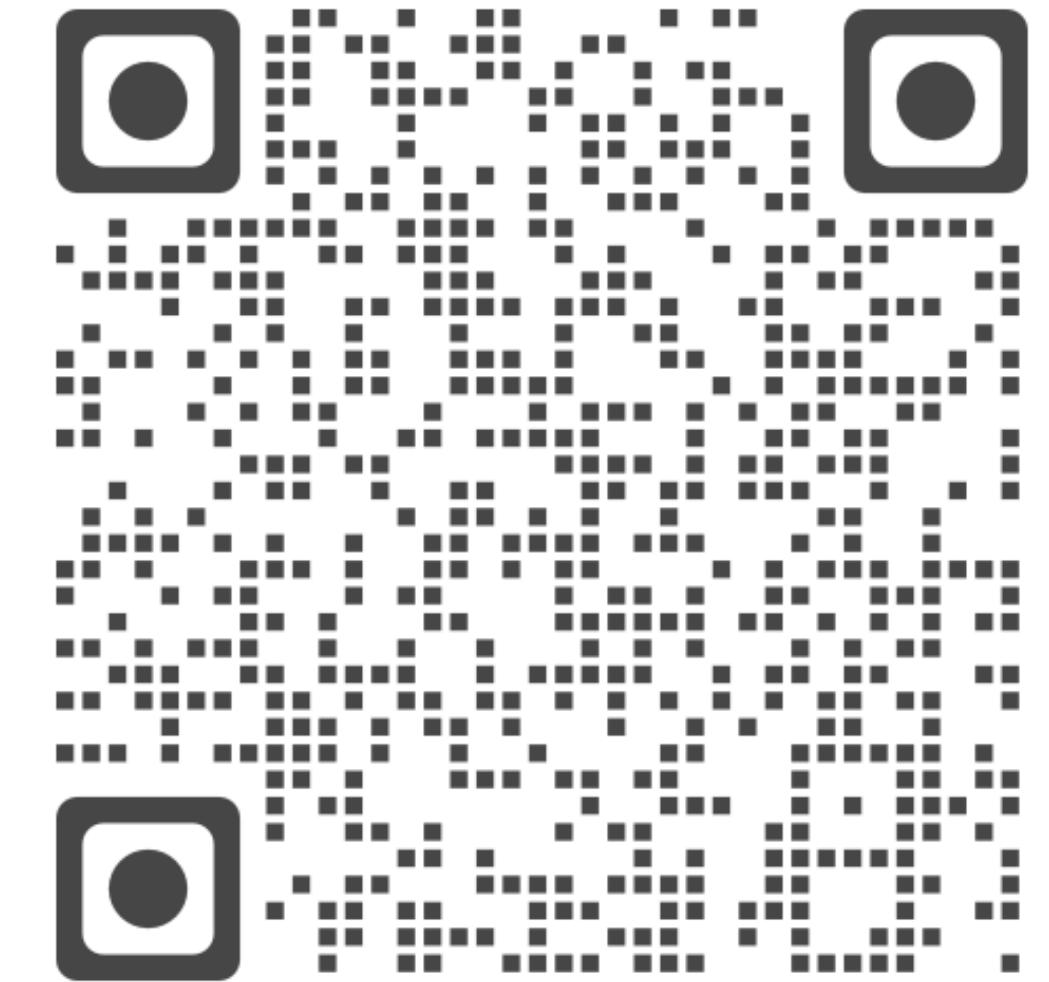
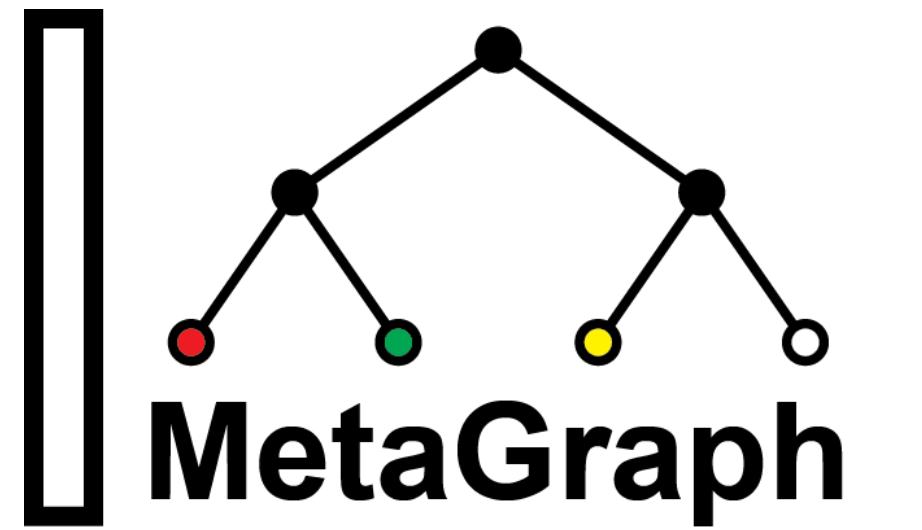
MetaGraph: Indexing and Analysing Nucleotide Archives at Petabase-scale

 Mikhail Karasikov,  Harun Mustafa, Daniel Danciu, Marc Zimmermann, Christopher Barber,

 Gunnar Rätsch,  André Kahles

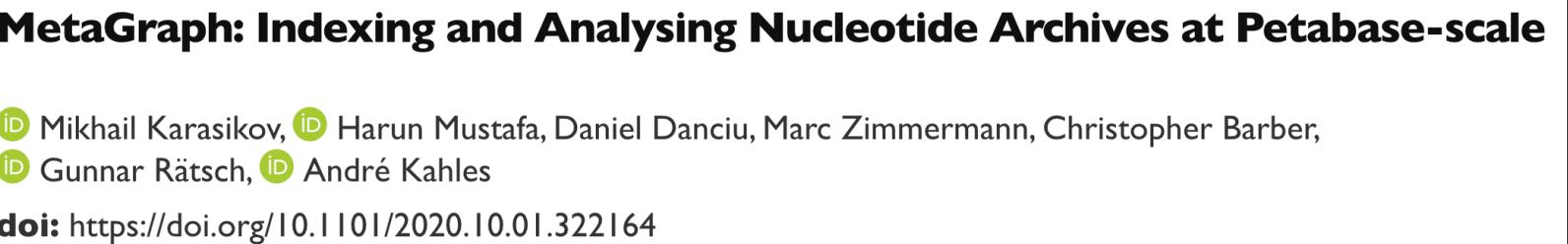
doi: <https://doi.org/10.1101/2020.10.01.322164>

- ▶ MetaGraph is an open source modular framework
 - various (succinct) graph representations (Bowe *et al.*, 2012; Conway *et al.*, 2011)
 - different schemes for annotation representation (Almodaresi *et al.*, 2017, Muggli *et al.*, 2017, Karasikov *et al.*, 2019, Danciu *et al.*, 2021, Karasikov *et al.*, 2022, ...)
 - also available as an Anaconda package or Docker container
- ▶ MetaGraph is **highly scalable**
 - scalable construction ⇒ works on Petabase size inputs
 - fast query ⇒ can query millions of sequences per hour
 - distributed representation ⇒ flexible API for client / server setup

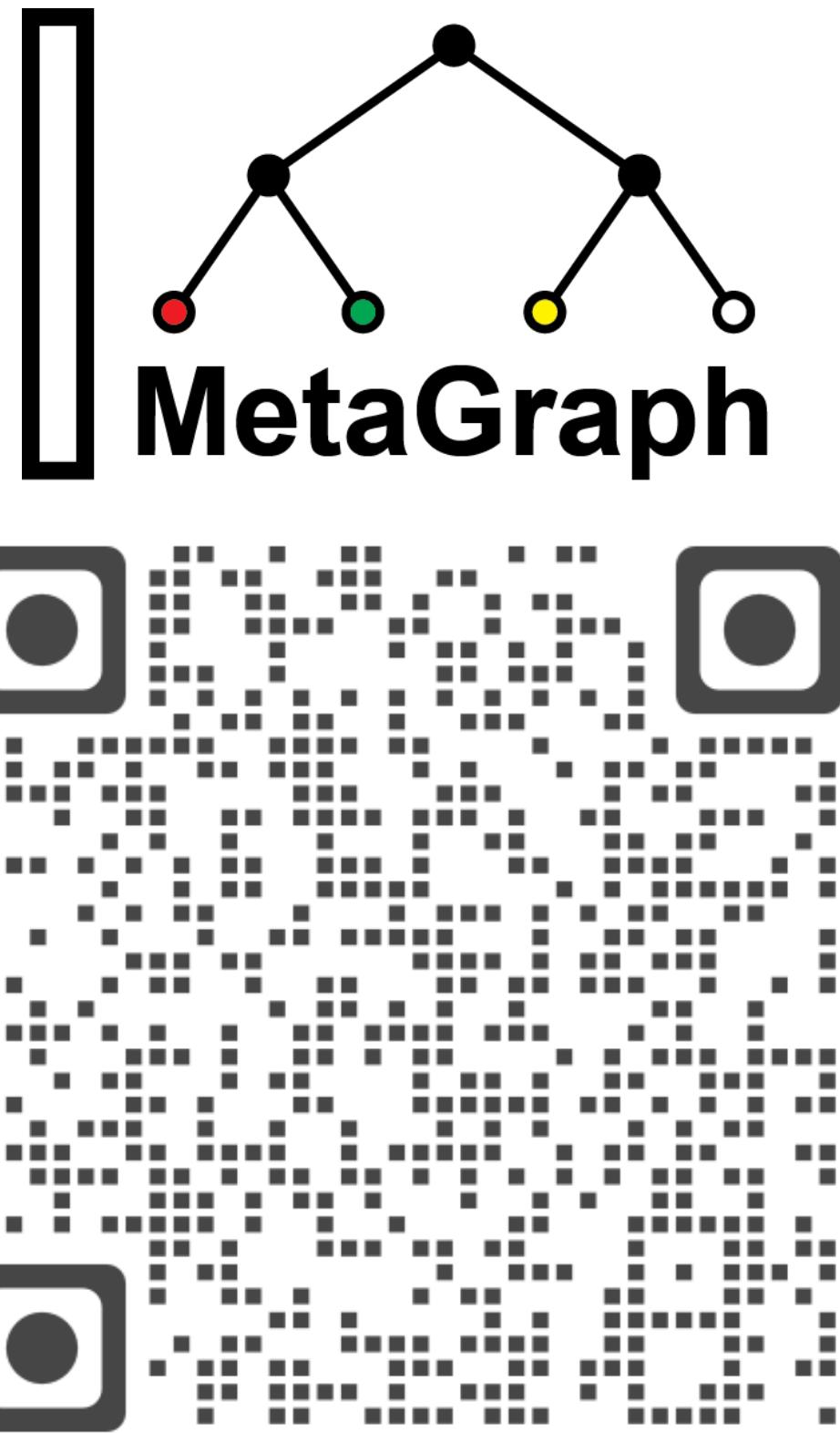


metagraph.ethz.ch/search

Why MetaGraph

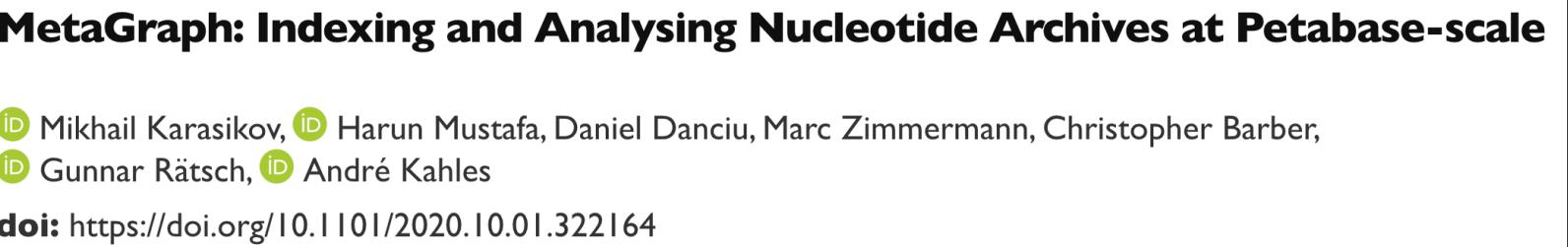


- ▶ MetaGraph is an open source modular framework
 - various (succinct) graph representations (Bowe *et al.*, 2012; Conway *et al.*, 2011)
 - different schemes for annotation representation (Almodaresi *et al.*, 2017, Muggli *et al.*, 2017, Karasikov *et al.*, 2019, Danciu *et al.*, 2021, Karasikov *et al.*, 2022, ...)
 - also available as an Anaconda package or Docker container
- ▶ MetaGraph is **highly scalable**
 - scalable construction ⇒ works on Petabase size inputs
 - fast query ⇒ can query millions of sequences per hour
 - distributed representation ⇒ flexible API for client / server setup
- ▶ Compressed and **lossless k-mer set representation**
 - up to 8,000x compression on transcriptome data

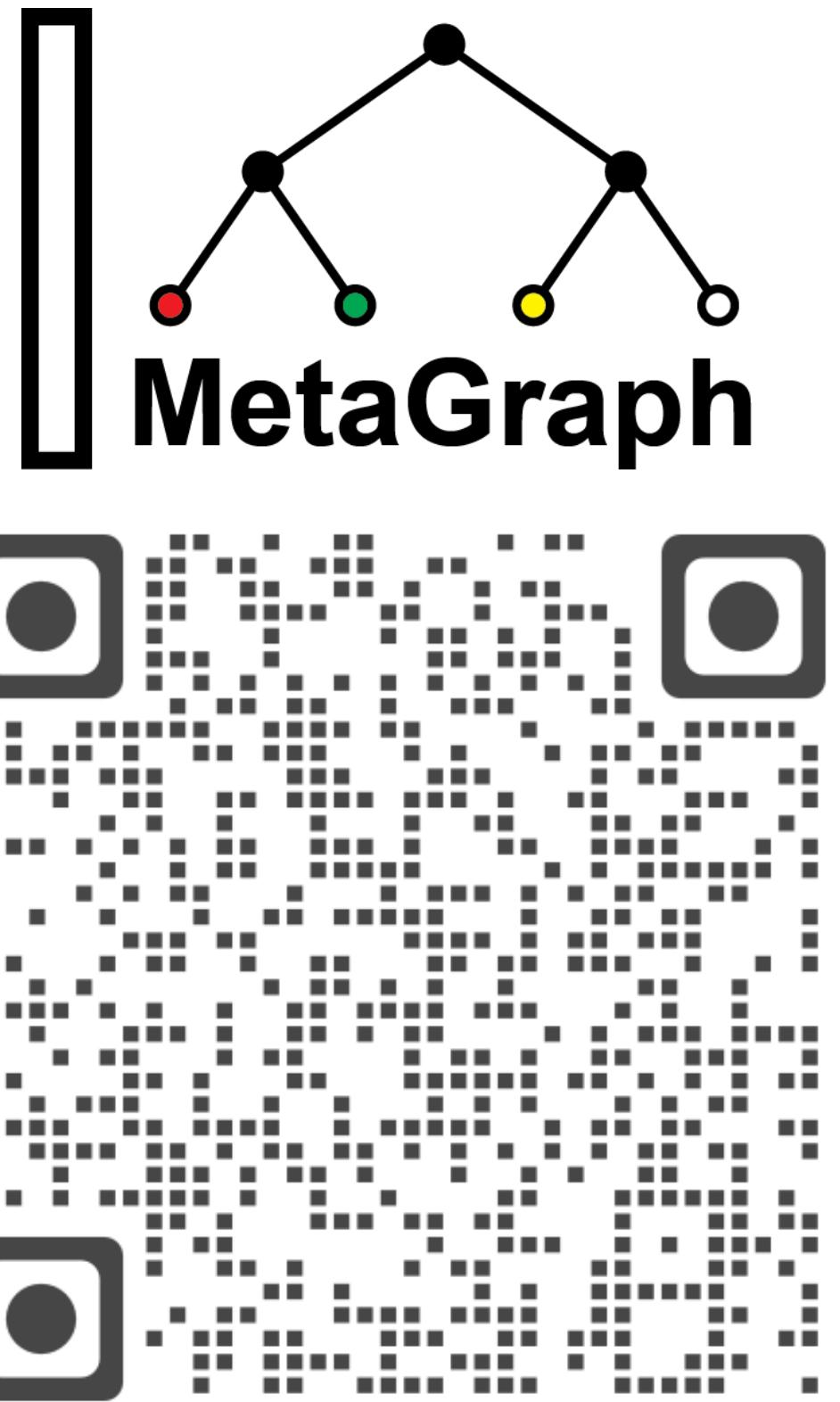


metagraph.ethz.ch/search

Why MetaGraph

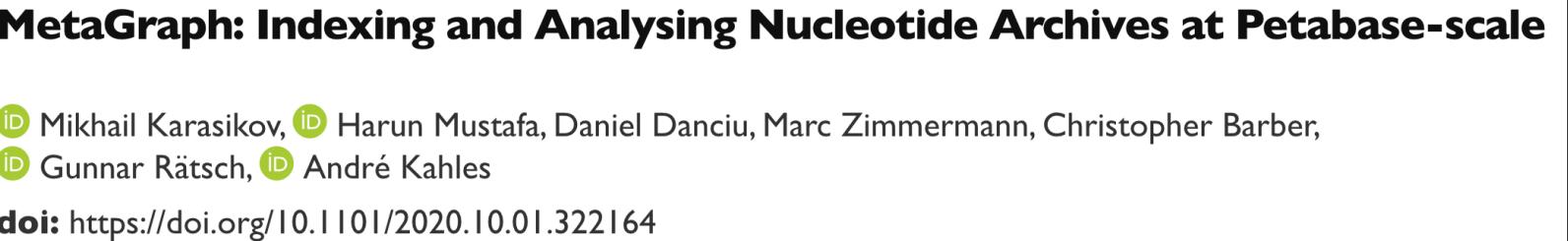


- ▶ MetaGraph is an open source modular framework
 - various (succinct) graph representations (Bowe *et al.*, 2012; Conway *et al.*, 2011)
 - different schemes for annotation representation (Almodaresi *et al.*, 2017, Muggli *et al.*, 2017, Karasikov *et al.*, 2019, Danciu *et al.*, 2021, Karasikov *et al.*, 2022, ...)
 - also available as an Anaconda package or Docker container
- ▶ MetaGraph is **highly scalable**
 - scalable construction ⇒ works on Petabase size inputs
 - fast query ⇒ can query millions of sequences per hour
 - distributed representation ⇒ flexible API for client / server setup
- ▶ Compressed and **lossless k-mer set representation**
 - up to 8,000x compression on transcriptome data
- ▶ Sensitive **sequence-to-graph alignment**
 - efficient sub-k seeding and performant chaining

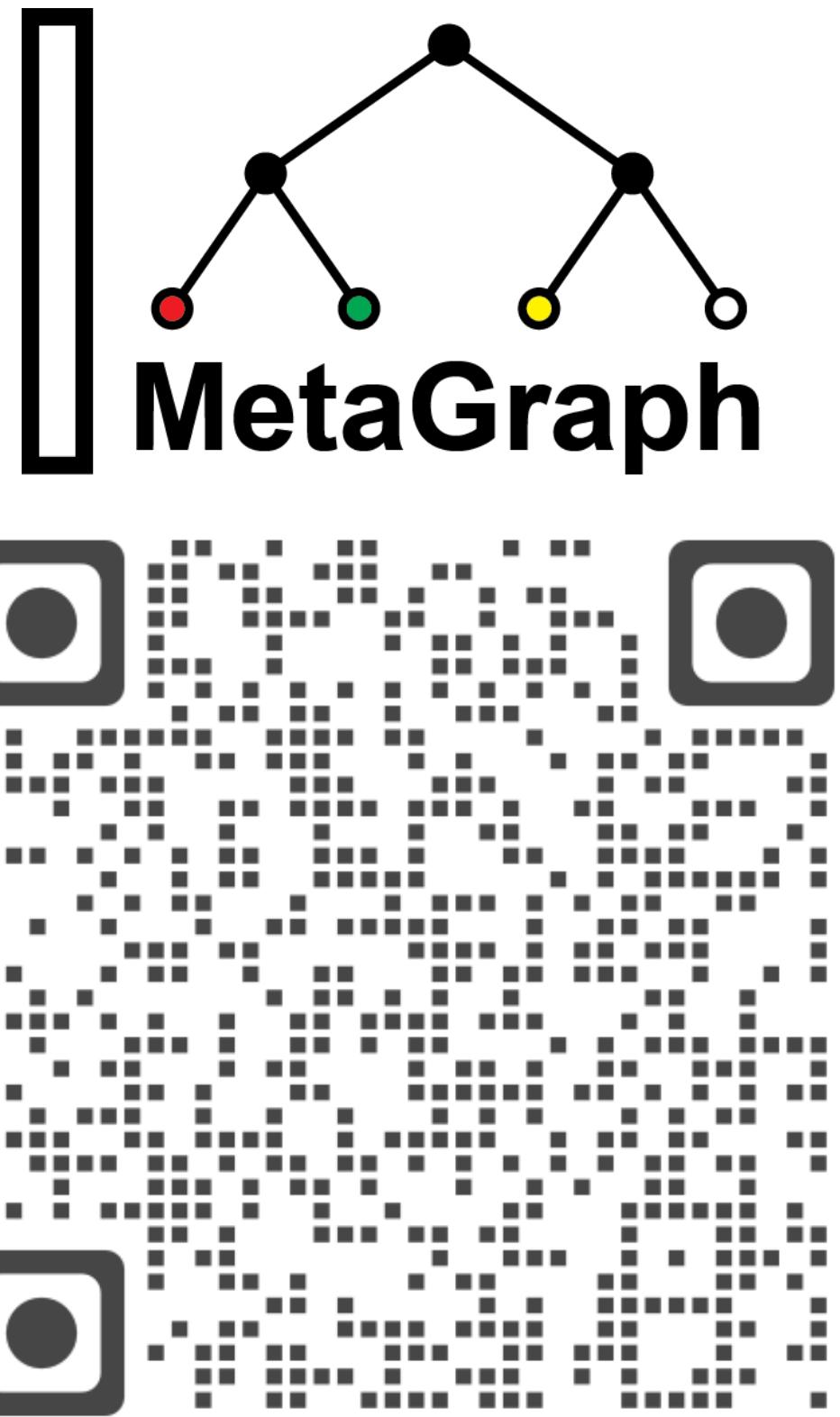


metagraph.ethz.ch/search

Why MetaGraph



- ▶ MetaGraph is an open source modular framework
 - various (succinct) graph representations (Bowe *et al.*, 2012; Conway *et al.*, 2011)
 - different schemes for annotation representation (Almodaresi *et al.*, 2017, Muggli *et al.*, 2017, Karasikov *et al.*, 2019, Danciu *et al.*, 2021, Karasikov *et al.*, 2022, ...)
 - also available as an Anaconda package or Docker container
- ▶ MetaGraph is **highly scalable**
 - scalable construction ⇒ works on Petabase size inputs
 - fast query ⇒ can query millions of sequences per hour
 - distributed representation ⇒ flexible API for client / server setup
- ▶ Compressed and **lossless k-mer set representation**
 - up to 8,000x compression on transcriptome data
- ▶ Sensitive **sequence-to-graph alignment**
 - efficient sub-k seeding and performant chaining
- ▶ Support for **integrative analysis**
 - novel concept of differential assembly



Why MetaGraph

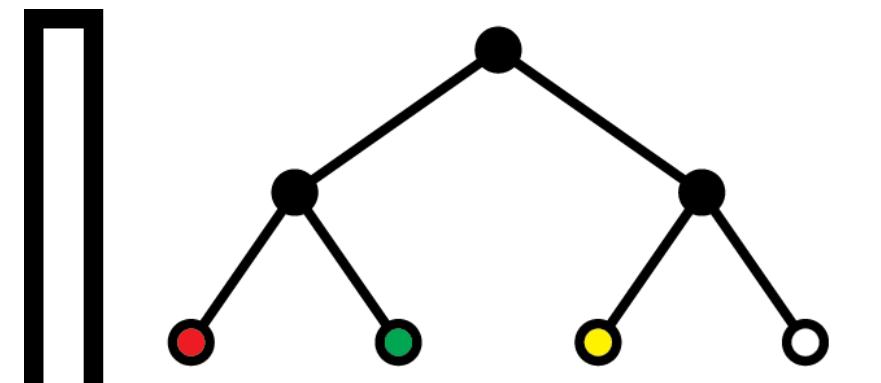
MetaGraph: Indexing and Analysing Nucleotide Archives at Petabase-scale

Mikhail Karasikov, Harun Mustafa, Daniel Danciu, Marc Zimmermann, Christopher Barber,

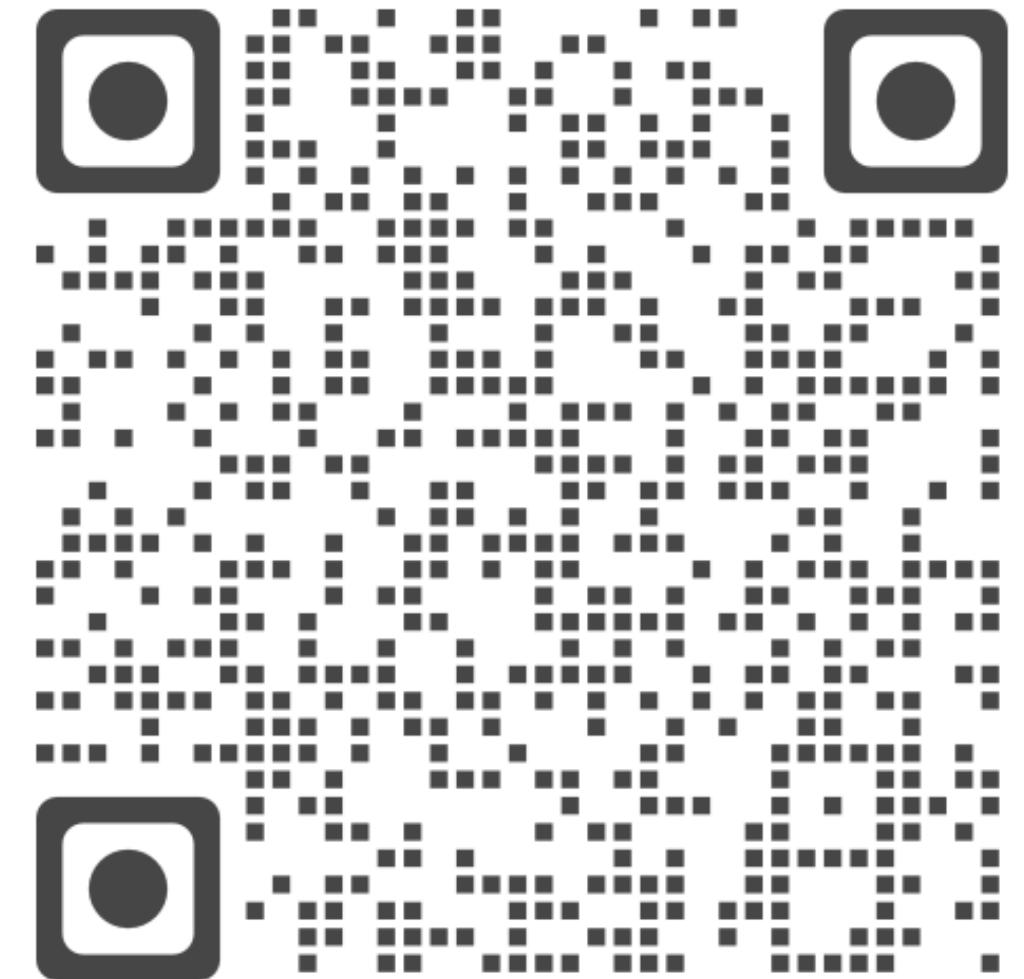
Gunnar Rätsch, André Kahles

doi: <https://doi.org/10.1101/2020.10.01.322164>

- ▶ MetaGraph is an open source modular framework
 - various (succinct) graph representations (Bowe *et al.*, 2012; Conway *et al.*, 2011)
 - different schemes for annotation representation (Almodaresi *et al.*, 2017, Muggli *et al.*, 2017, Karasikov *et al.*, 2019, Danciu *et al.*, 2021, Karasikov *et al.*, 2022, ...)
 - also available as an Anaconda package or Docker container
- ▶ MetaGraph is **highly scalable**
 - scalable construction ⇒ works on Petabase size inputs
 - fast query ⇒ can query millions of sequences per hour
 - distributed representation ⇒ flexible API for client / server setup
- ▶ Compressed and **lossless k-mer set representation**
 - up to 8,000x compression on transcriptome data
- ▶ Sensitive **sequence-to-graph alignment**
 - efficient sub-k seeding and performant chaining
- ▶ Support for **integrative analysis**
 - novel concept of differential assembly



MetaGraph



metagraph.ethz.ch/search

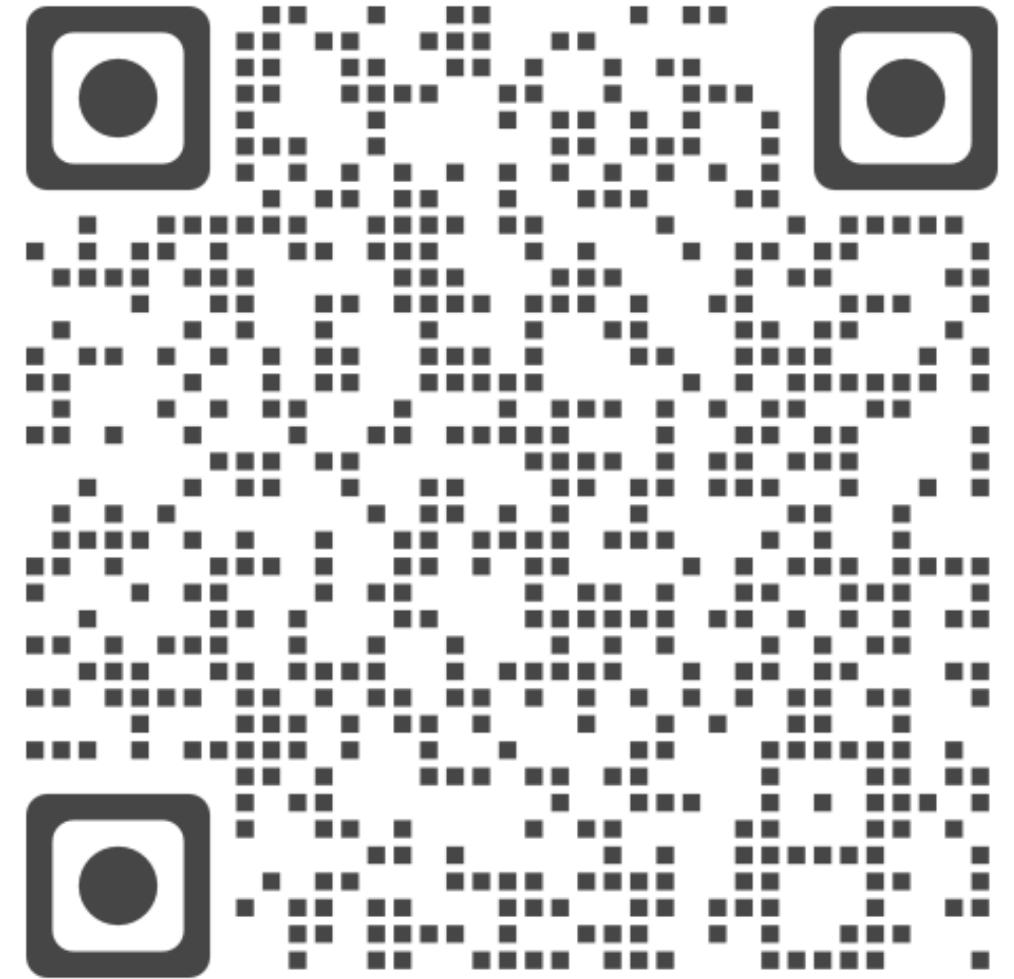
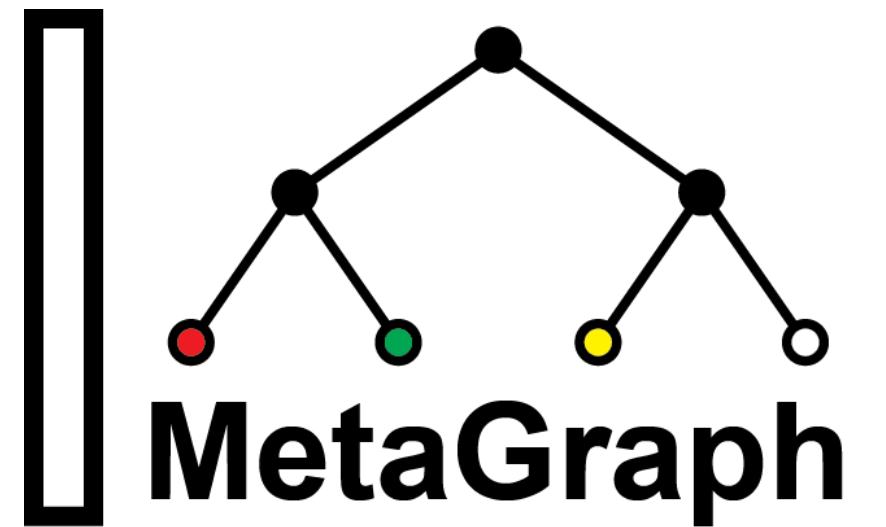
Now also

- ▶ Indexing quantitative data
 - **k-mer abundances**
 - **k-mer coordinates** (positions in genomes)
 - fully lossless representation

Why MetaGraph

MetaGraph: Indexing and Analysing Nucleotide Archives at Petabase-scale
ID Mikhail Karasikov, ID Harun Mustafa, Daniel Danciu, Marc Zimmermann, Christopher Barber,
ID Gunnar Rätsch, ID André Kahles
doi: <https://doi.org/10.1101/2020.10.01.322164>

- ▶ MetaGraph is an open source modular framework
 - various (succinct) graph representations (Bowe *et al.*, 2012; Conway *et al.*, 2011)
 - different schemes for annotation representation (Almodaresi *et al.*, 2017, Muggli *et al.*, 2017, Karasikov *et al.*, 2019, Danciu *et al.*, 2021, Karasikov *et al.*, 2022, ...)
 - also available as an Anaconda package or Docker container
- ▶ MetaGraph is **highly scalable**
 - scalable construction ⇒ works on Petabase size inputs
 - fast query ⇒ can query millions of sequences per hour
 - distributed representation ⇒ flexible API for client / server setup
- ▶ Compressed and **lossless k-mer set representation**
 - up to 8,000x compression on transcriptome data
- ▶ Sensitive **sequence-to-graph alignment**
 - efficient sub-k seeding and performant chaining
- ▶ Support for **integrative analysis**
 - novel concept of differential assembly



metagraph.ethz.ch/search

Now also

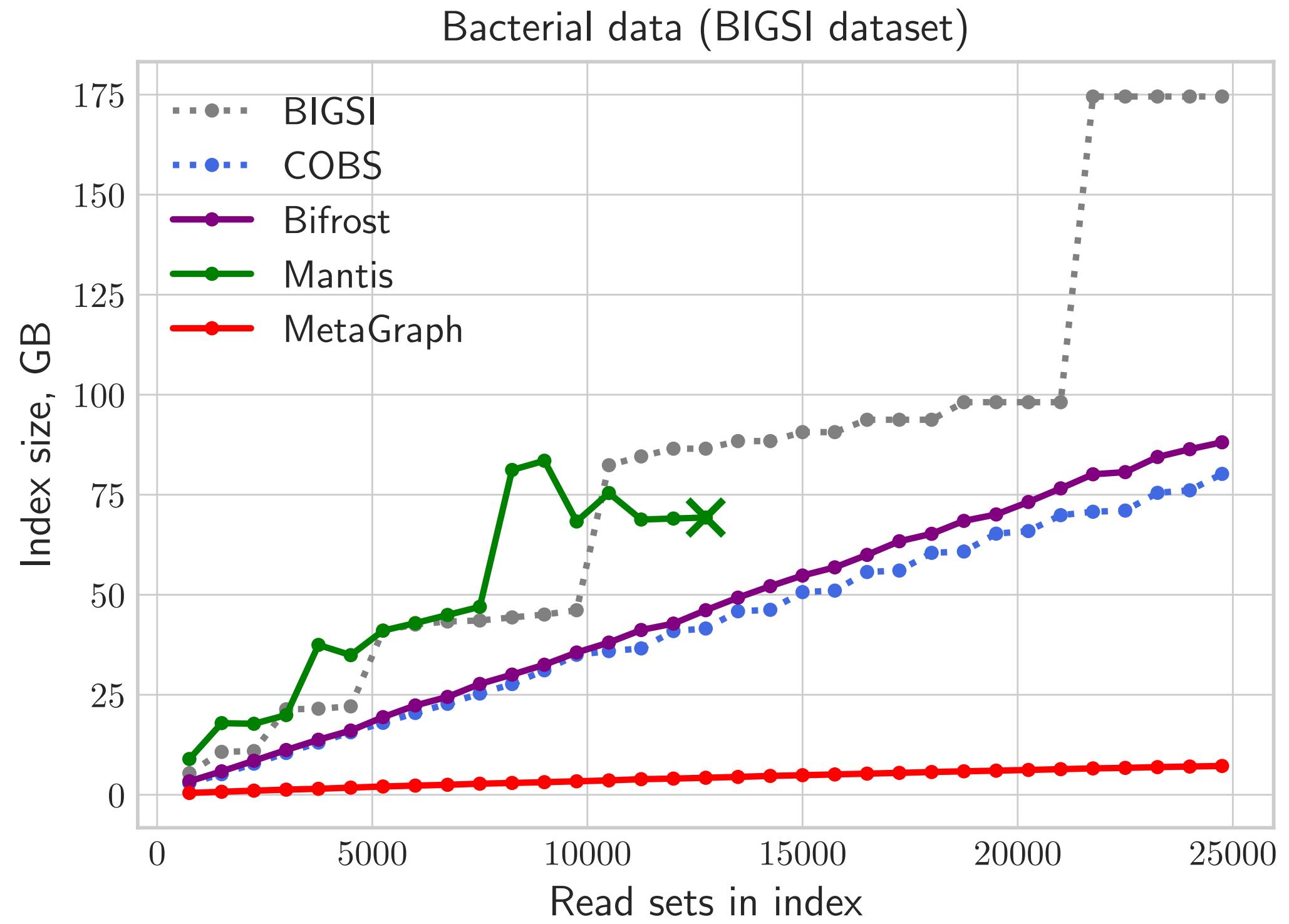
- ▶ Indexing quantitative data
 - **k-mer abundances**
 - **k-mer coordinates** (positions in genomes)
 - fully lossless representation

Soon: memory mapping (by Marek Kokot)

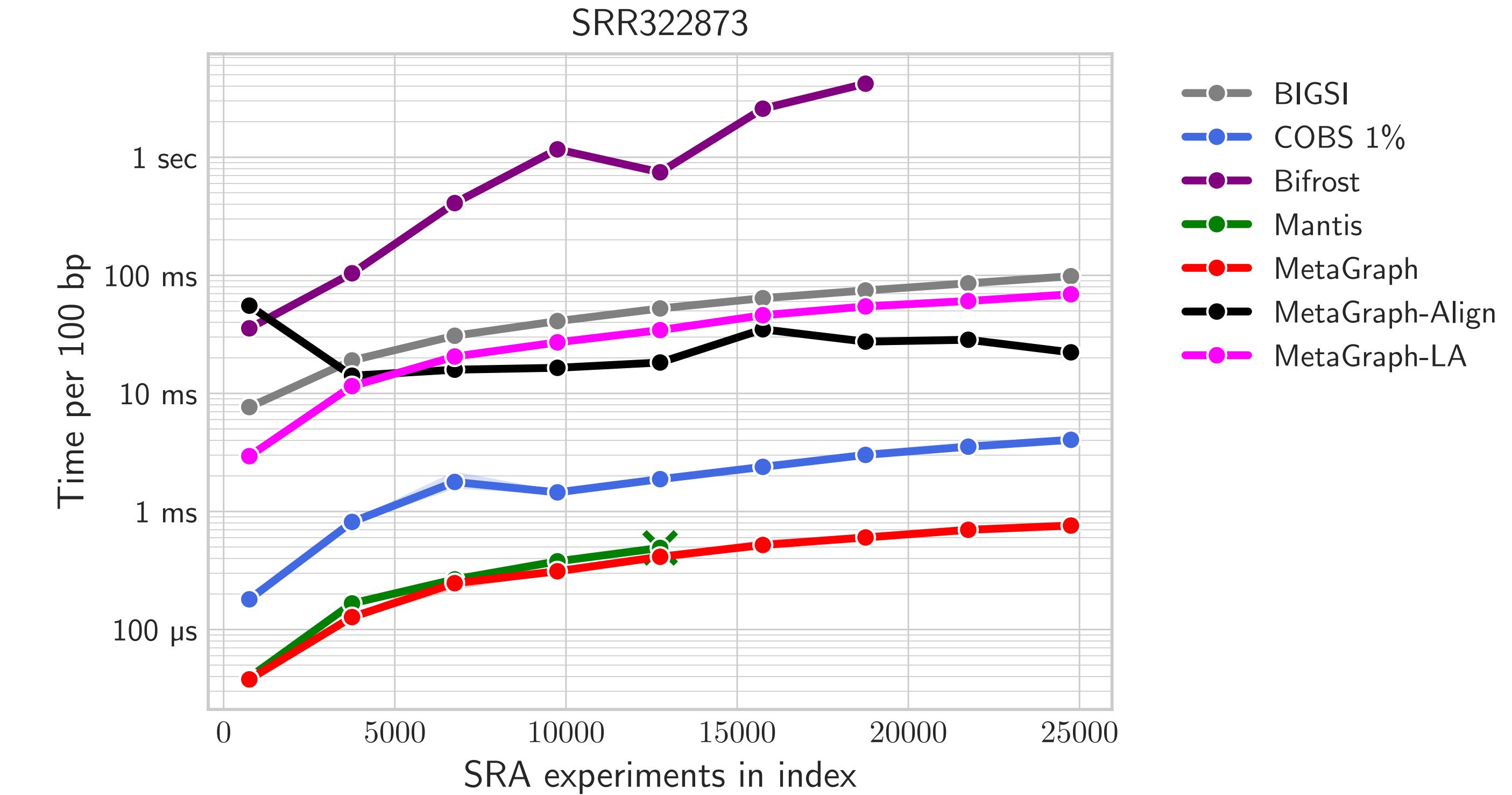
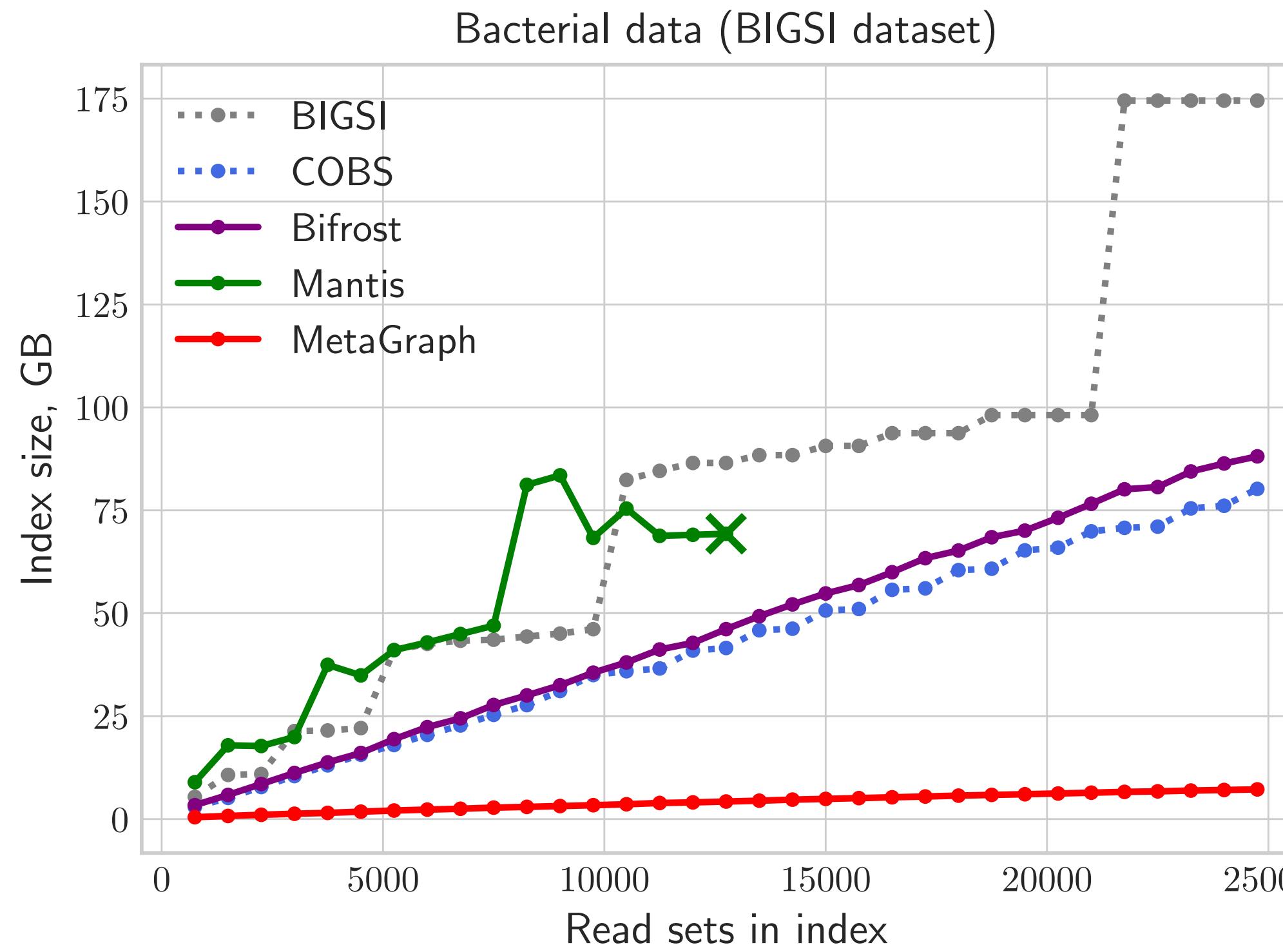
- host terabyte-size indexes with **less RAM**



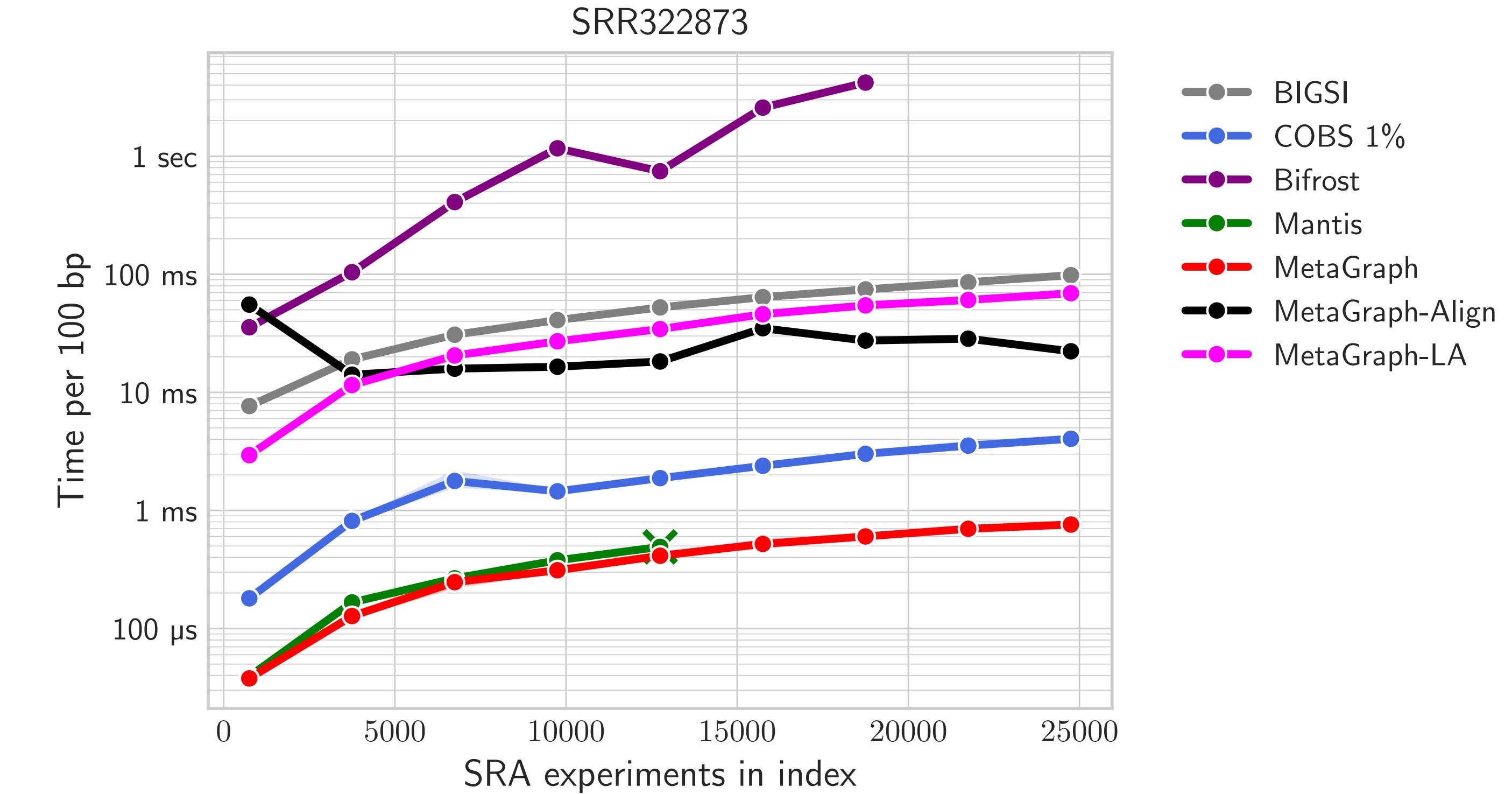
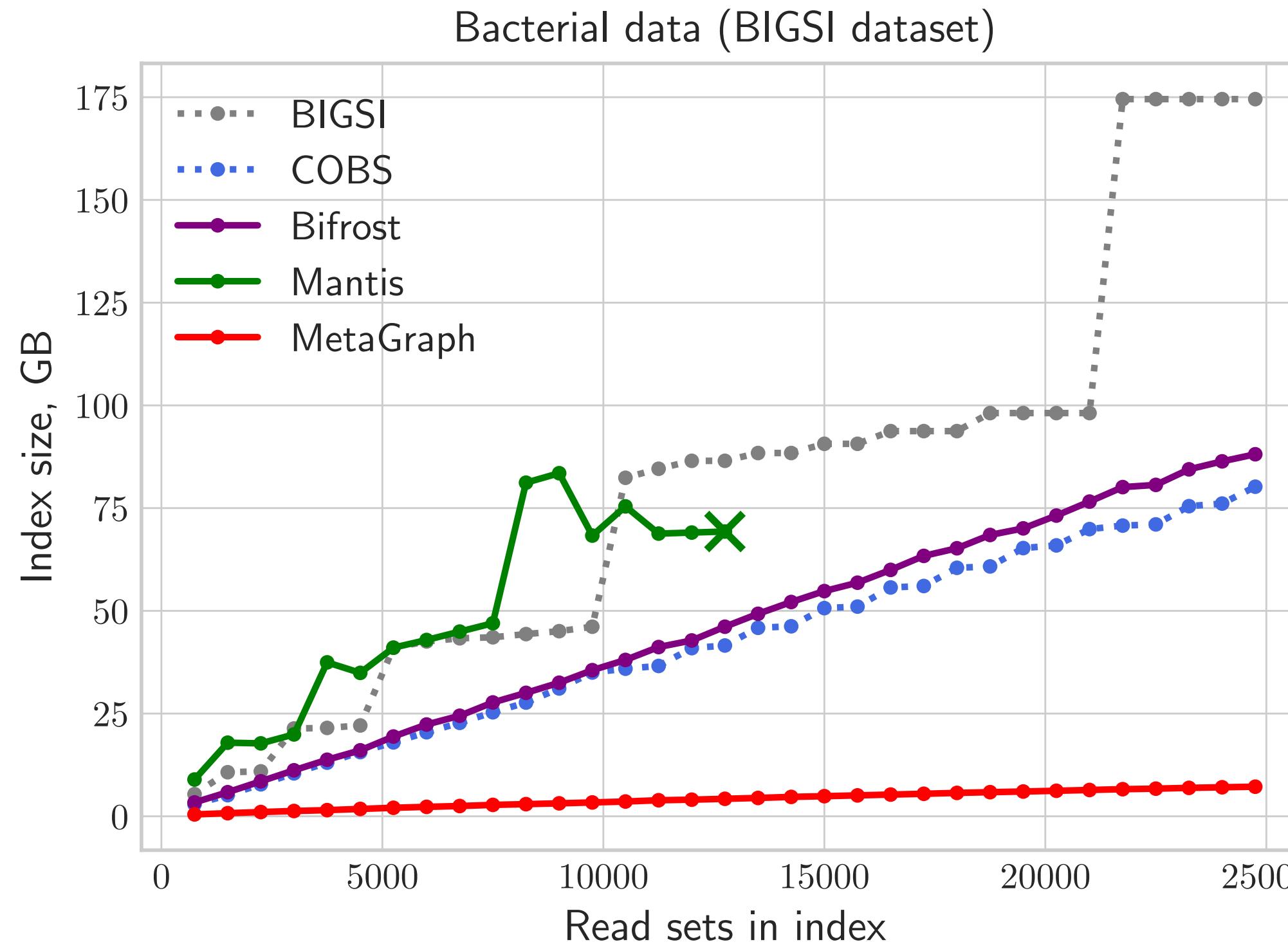
Scalability of MetaGraph



Scalability of MetaGraph

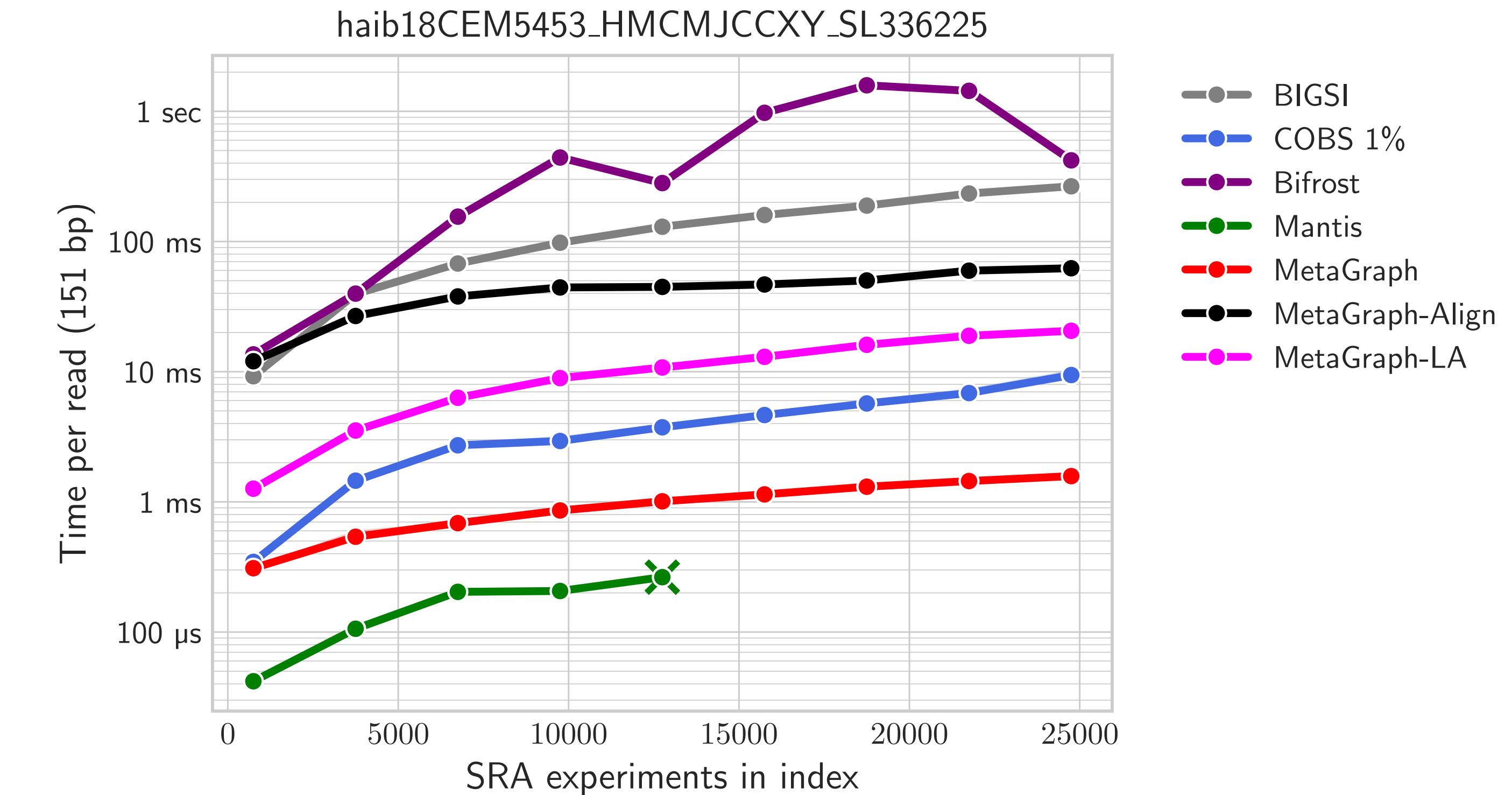
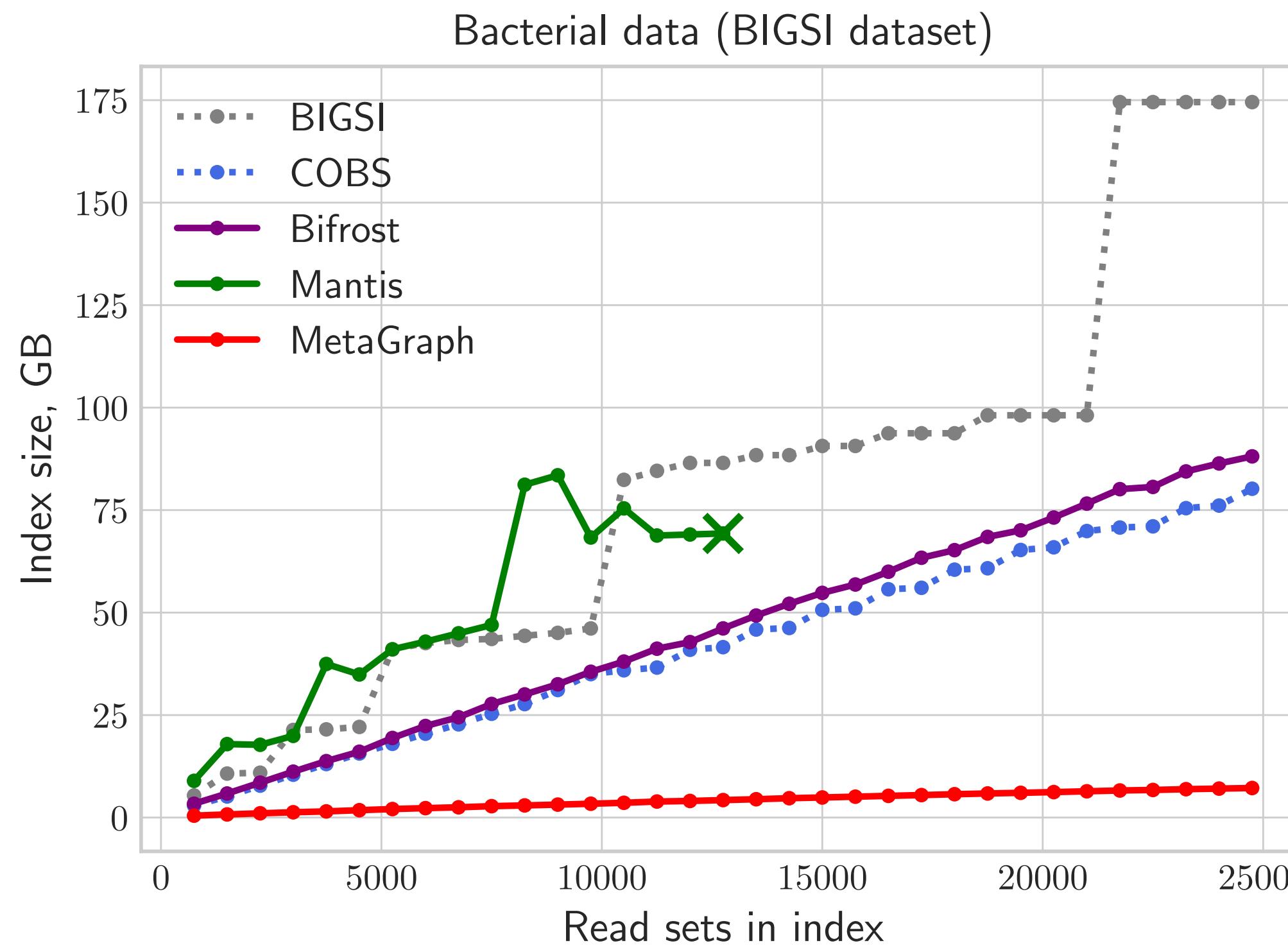


Scalability of MetaGraph



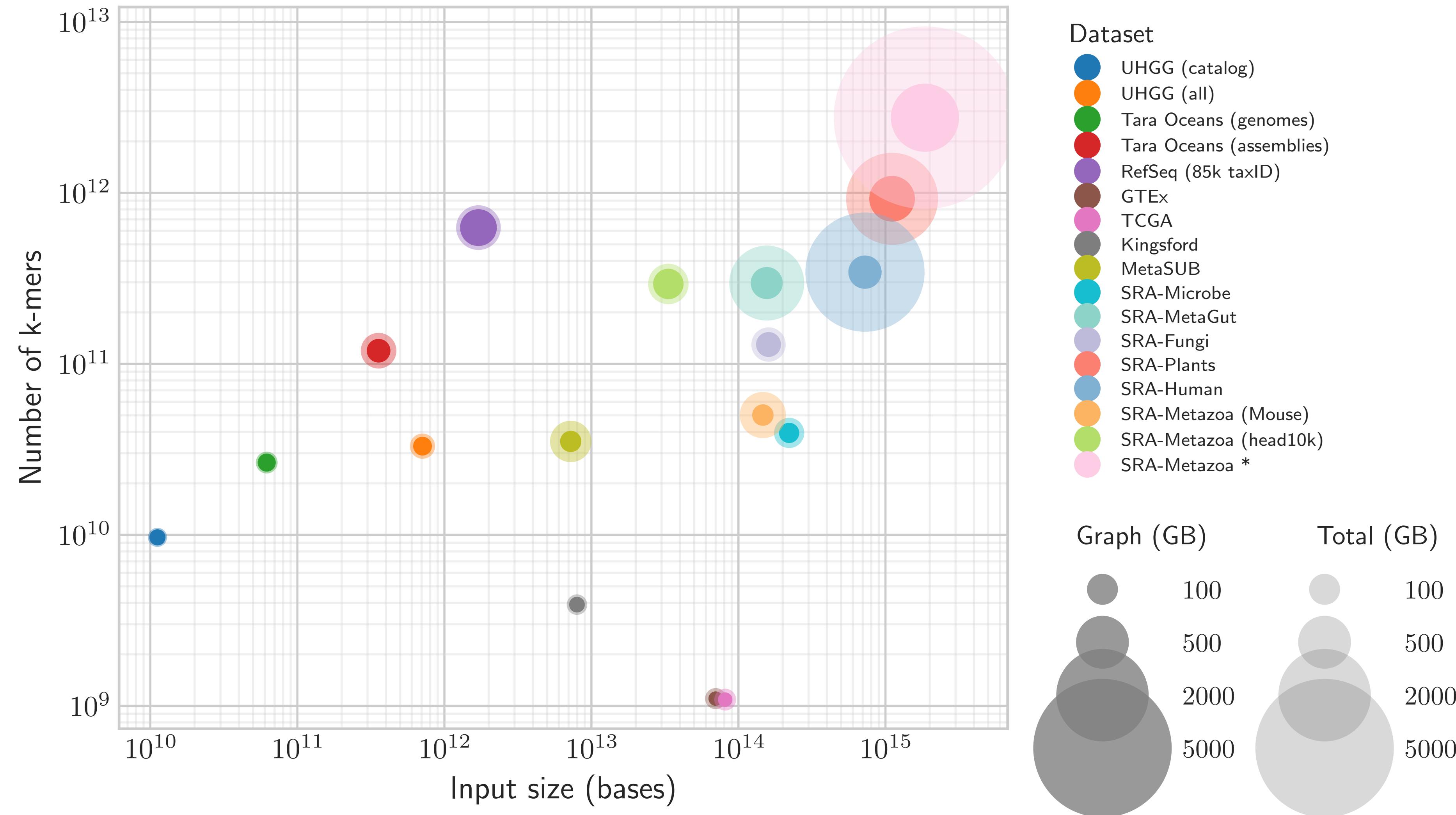
- ▶ Use of **succinct data structures** and **efficient representation schemes**
 - algorithmic choices that work efficiently with succinct data structures (e.g., batch operations)

Scalability of MetaGraph



- ▶ Use of **succinct data structures** and **efficient representation schemes**
 - algorithmic choices that work efficiently with succinct data structures (e.g., batch operations)

Indexing petabase-scale inputs



Datasets: UHGG ([Almeida et al., 2020](#)), Tara Oceans ([Paoli, 2022](#)), GTEX ([Lonsdale et al., 2013](#)), TCGA ([Kaiser, 2005](#)), MetaSUB ([Danko et al., 2019](#)), SRA-Microbe ([Bradley et al., 2019](#))

Indexing petabase-scale inputs

Dataset	Tbp	Input (gz)	k	# k-mers	# labels	Graph	Annot.	Index	Ratio
UHGG (catalog)	0.01	3.3 GB	31	$9.7 \cdot 10^9$	4,644	2.6 GB	0.6 GB	3.2 GB	1.0×
UHGG (all)	0.71	206.0 GB	31	$33.0 \cdot 10^9$	286,997	9.6 GB	17.7 GB	27.3 GB	7.6×
Tara Oceans (genomes)	0.06	17.9 GB	31	$26.5 \cdot 10^9$	6,414,647	7.3 GB	3.8 GB	11.1 GB	1.6×
Tara Oceans (assemblies)	0.36	106.8 GB	31	$119.4 \cdot 10^9$	318,205,057	34.4 GB	89.7 GB	124.1 GB	0.9×
RefSeq (85k taxID)	1.70	502.4 GB	31	$626.2 \cdot 10^9$	85,375	176.0 GB	87.5 GB	263.6 GB	1.9×
RefSeq (33M accessions)	1.70	502.5 GB	31	$626.2 \cdot 10^9$	32,881,348	176.0 GB	287.0 GB	463.1 GB	1.1×
Kingsford	8.0	2.9 TB	21	$3.9 \cdot 10^9$	2,652	1.8 GB	4.8 GB	6.6 GB	437×
GTEEx	70.0	40.0 TB	41	$1.1 \cdot 10^9$	9,759	0.4 GB	8.0 GB	8.4 GB	4,742×
TCGA	81.2	65.0 TB	31	$1.1 \cdot 10^9$	11,095	0.4 GB	10.7 GB	11.1 GB	5,831×
MetaSUB	7.2	5.5 TB	19	$35.2 \cdot 10^9$	4,220	20.5 GB	185.9 GB	206.4 GB	27×
SRA-MetaGut	155.8	86.0 TB	31	$296.9 \cdot 10^9$	242,619	112.2 GB	999.0 GB	1,111.3 GB	77×
SRA-Microbe	221.1	170.0 TB	31	$39.5 \cdot 10^9$	446,506	15.4 GB	50.1 GB	65.5 GB	2,595×
SRA-Fungi	160.2	80.0 TB	31	$129.7 \cdot 10^9$	121,900	43.5 GB	64.7 GB	108.1 GB	740×
SRA-Plants	1,109.2	575.9 TB	31	$923.4 \cdot 10^9$	531,736	333.8 GB	1,510.3 GB	1,844.1 GB	312×
SRA-Human	725.4	345.7 TB	31	$343.9 \cdot 10^9$	436,502	127.6 GB	3,274.5 GB	3,402.1 GB	102×
SRA-Metazoa (Mouse)	146.6	61.3 TB	31	$50.2 \cdot 10^9$	57,938	21.1 GB	270.5 GB	291.6 GB	210×
SRA-Metazoa (10k)	33.4	16.5 TB	31	$293.3 \cdot 10^9$	10,000	94.0 GB	98.7 GB	192.7 GB	86×
SRA-Metazoa *	1,856.8	925.3 TB	31	$2749.8 \cdot 10^9$	797,883	960.4 GB	7,898.4 GB	8,858.8 GB	104×

Indexing petabase-scale inputs

Dataset	Tbp	Input (gz)	k	# k-mers	# labels	Graph	Annot.	Index	Ratio	
UHGG (catalog)	Can we index k-mer positions as well?	0.01	3.3 GB	31	$9.7 \cdot 10^9$	4,644	2.6 GB	0.6 GB	3.2 GB	1.0×
UHGG (all)		0.71	206.0 GB	31	$33.0 \cdot 10^9$	286,997	9.6 GB	17.7 GB	27.3 GB	7.6×
Tara Oceans (genomes)		0.56	106.8 GB	31	$119.4 \cdot 10^9$	518,205,057	34.4 GB	89.7 GB	124.1 GB	0.9×
Tara Oceans (assemblies)		1.70	502.4 GB	31	$626.2 \cdot 10^9$	85,375	176.0 GB	87.5 GB	263.6 GB	1.9×
RefSeq (85k taxID)		1.70	502.5 GB	31	$626.2 \cdot 10^9$	32,881,348	176.0 GB	287.0 GB	463.1 GB	1.1×
RefSeq (33M accessions)		8.0	2.9 TB	21	$3.9 \cdot 10^9$	2,652	1.8 GB	4.8 GB	6.6 GB	437×
Kingsford	70.0	40.0 TB	41	$1.1 \cdot 10^9$	9,759	0.4 GB	8.0 GB	8.4 GB	4,742×	
GTEEx	81.2	65.0 TB	31	$1.1 \cdot 10^9$	11,095	0.4 GB	10.7 GB	11.1 GB	5,831×	
TCGA	7.2	5.5 TB	19	$35.2 \cdot 10^9$	4,220	20.5 GB	185.9 GB	206.4 GB	27×	
MetaSUB	155.8	86.0 TB	31	$296.9 \cdot 10^9$	242,619	112.2 GB	999.0 GB	1,111.3 GB	77×	
SRA-MetaGut	221.1	170.0 TB	31	$39.5 \cdot 10^9$	446,506	15.4 GB	50.1 GB	65.5 GB	2,595×	
SRA-Microbe	160.2	80.0 TB	31	$129.7 \cdot 10^9$	121,900	43.5 GB	64.7 GB	108.1 GB	740×	
SRA-Fungi	1,109.2	575.9 TB	31	$923.4 \cdot 10^9$	531,736	333.8 GB	1,510.3 GB	1,844.1 GB	312×	
SRA-Plants	725.4	345.7 TB	31	$343.9 \cdot 10^9$	436,502	127.6 GB	3,274.5 GB	3,402.1 GB	102×	
SRA-Human	146.6	61.3 TB	31	$50.2 \cdot 10^9$	57,938	21.1 GB	270.5 GB	291.6 GB	210×	
SRA-Metazoa (Mouse)	33.4	16.5 TB	31	$293.3 \cdot 10^9$	10,000	94.0 GB	98.7 GB	192.7 GB	86×	
SRA-Metazoa (10k)	1,856.8	925.3 TB	31	$2749.8 \cdot 10^9$	797,883	960.4 GB	7,898.4 GB	8,858.8 GB	104×	

Indexing petabase-scale inputs

Dataset	Tbp	Input (gz)	k	# k-mers	# labels	Graph	Annot.	Index	Ratio
UHGG (catalog)	0.01 0.71 0.56 1.70 1.70	3.3 GB	31	$9.7 \cdot 10^9$	4,644	2.6 GB	0.6 GB	3.2 GB	1.0×
UHGG (all)		206.0 GB	31	$33.0 \cdot 10^9$	286,997	9.6 GB	17.7 GB	27.3 GB	7.6×
Tara Oceans (genomes)		Can we index k-mer positions as well?				4,647	7.3 GB	3.8 GB	11.1 GB
Tara Oceans (assemblies)		106.8 GB	31	$119.4 \cdot 10^9$	518,205,057	34.4 GB	89.7 GB	124.1 GB	0.9×
RefSeq (85k taxID)		502.4 GB	31	$626.2 \cdot 10^9$	85,375	176.0 GB	87.5 GB	263.6 GB	1.9×
RefSeq (33M accessions)		502.5 GB	31	$626.2 \cdot 10^9$	32,881,348	176.0 GB	287.0 GB	463.1 GB	1.1×
Kingsford	8.0 What about expression? 81.2	2.9 TB	21	$3.9 \cdot 10^9$	2,652	1.8 GB	4.8 GB	6.6 GB	437×
GTEEx		41		$1.1 \cdot 10^9$	9,759	0.4 GB	8.0 GB	8.4 GB	4,742×
TCGA		65.0 TB	31	$1.1 \cdot 10^9$	11,095	0.4 GB	10.7 GB	11.1 GB	5,831×
MetaSUB	7.2	5.5 TB	19	$35.2 \cdot 10^9$	4,220	20.5 GB	185.9 GB	206.4 GB	27×
SRA-MetaGut	155.8	86.0 TB	31	$296.9 \cdot 10^9$	242,619	112.2 GB	999.0 GB	1,111.3 GB	77×
SRA-Microbe	221.1	170.0 TB	31	$39.5 \cdot 10^9$	446,506	15.4 GB	50.1 GB	65.5 GB	2,595×
SRA-Fungi	160.2	80.0 TB	31	$129.7 \cdot 10^9$	121,900	43.5 GB	64.7 GB	108.1 GB	740×
SRA-Plants	1,109.2	575.9 TB	31	$923.4 \cdot 10^9$	531,736	333.8 GB	1,510.3 GB	1,844.1 GB	312×
SRA-Human	725.4	345.7 TB	31	$343.9 \cdot 10^9$	436,502	127.6 GB	3,274.5 GB	3,402.1 GB	102×
SRA-Metazoa (Mouse)	146.6	61.3 TB	31	$50.2 \cdot 10^9$	57,938	21.1 GB	270.5 GB	291.6 GB	210×
SRA-Metazoa (10k)	33.4	16.5 TB	31	$293.3 \cdot 10^9$	10,000	94.0 GB	98.7 GB	192.7 GB	86×
SRA-Metazoa *	1,856.8	925.3 TB	31	$2749.8 \cdot 10^9$	797,883	960.4 GB	7,898.4 GB	8,858.8 GB	104×

Background

De Bruijn Graphs

... widely used in Bioinformatics since 1989

Background

De Bruijn Graphs

... widely used in Bioinformatics since 1989

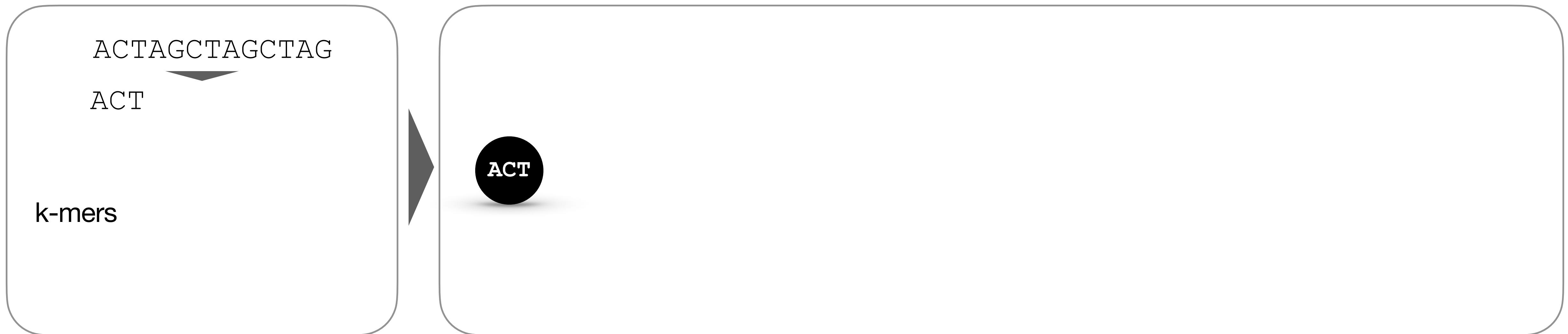
ACTAGCTAGCTAG

- ▶ Originally employed for ***de novo* assembly**

Background

De Bruijn Graphs

... widely used in Bioinformatics since 1989

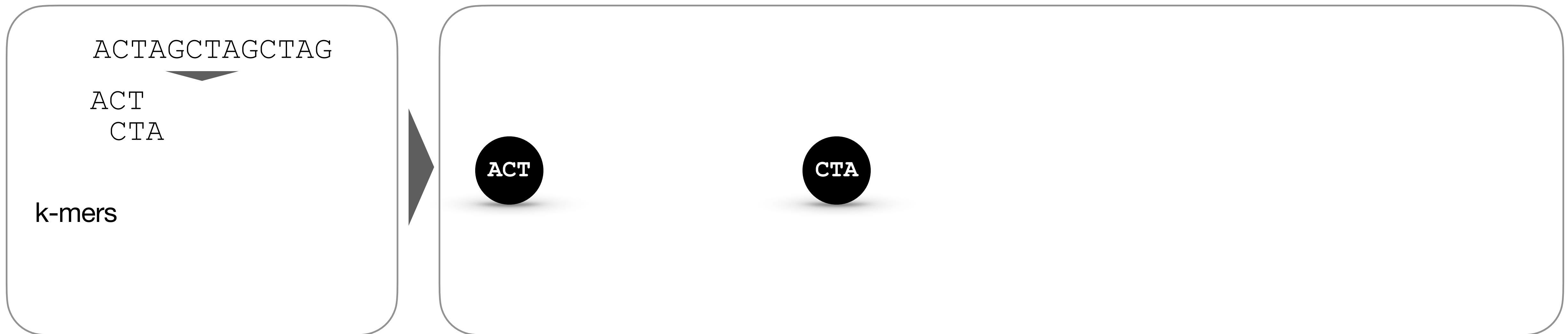


- ▶ Originally employed for ***de novo* assembly**

Background

De Bruijn Graphs

... widely used in Bioinformatics since 1989

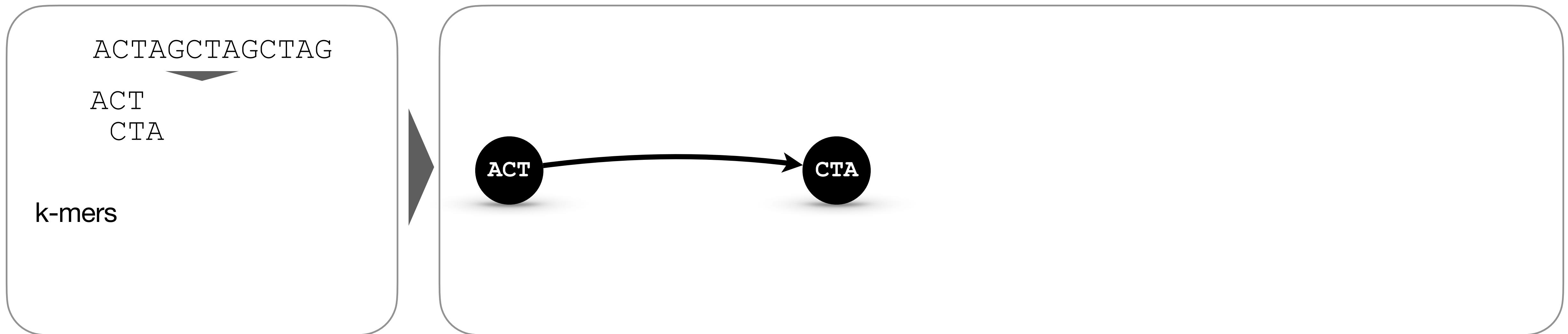


- ▶ Originally employed for ***de novo* assembly**

Background

De Bruijn Graphs

... widely used in Bioinformatics since 1989

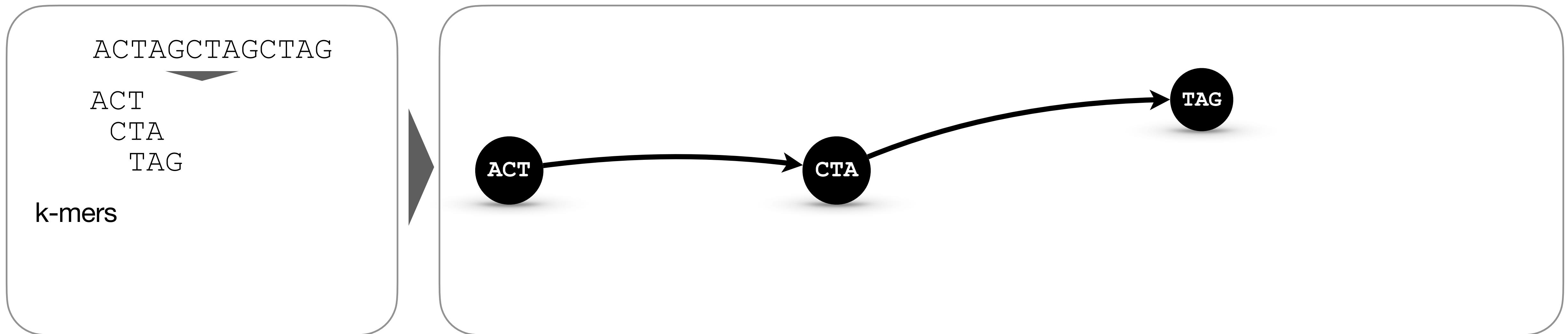


- ▶ Originally employed for ***de novo* assembly**

Background

De Bruijn Graphs

... widely used in Bioinformatics since 1989

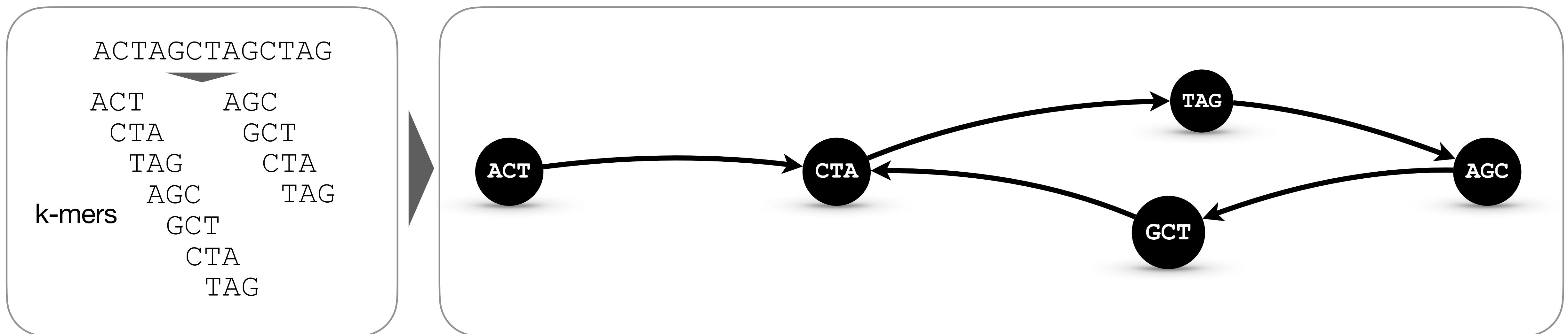


- ▶ Originally employed for ***de novo* assembly**

Background

De Bruijn Graphs

... widely used in Bioinformatics since 1989

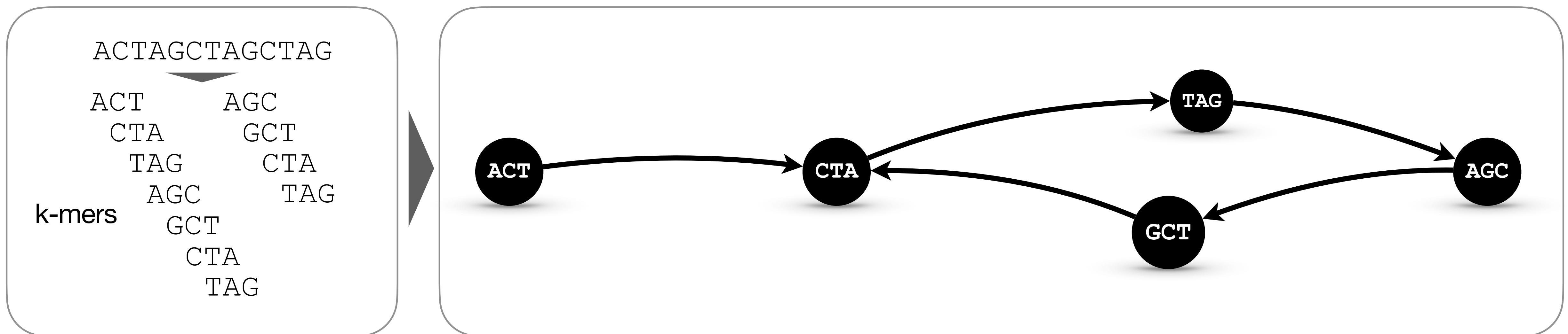


- ▶ Originally employed for ***de novo* assembly**

Background

De Bruijn Graphs

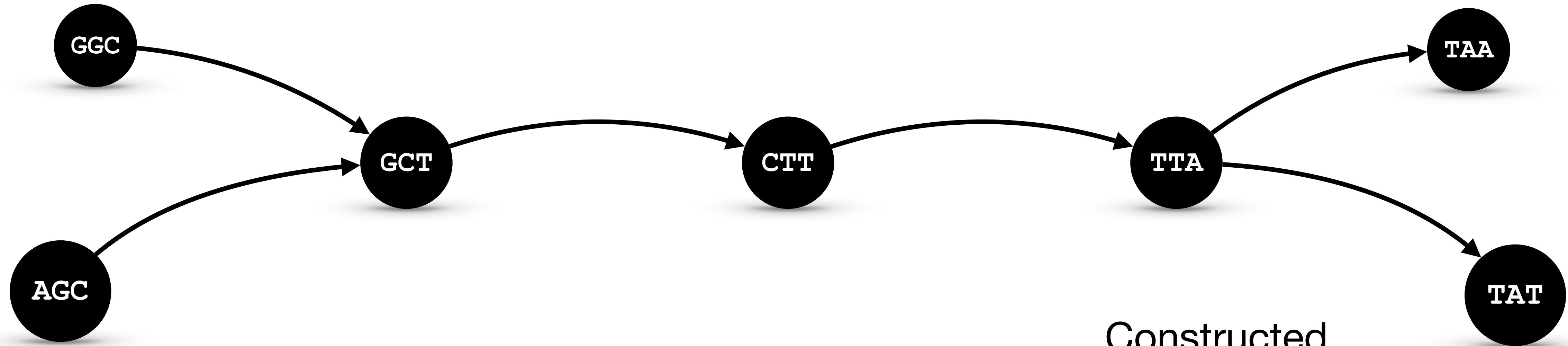
... widely used in Bioinformatics since 1989



- ▶ Originally employed for ***de novo* assembly**
- ▶ Now, also widely used for **indexing raw sequencing data**

Background

Annotated de Bruijn Graphs



Constructed
from sequences

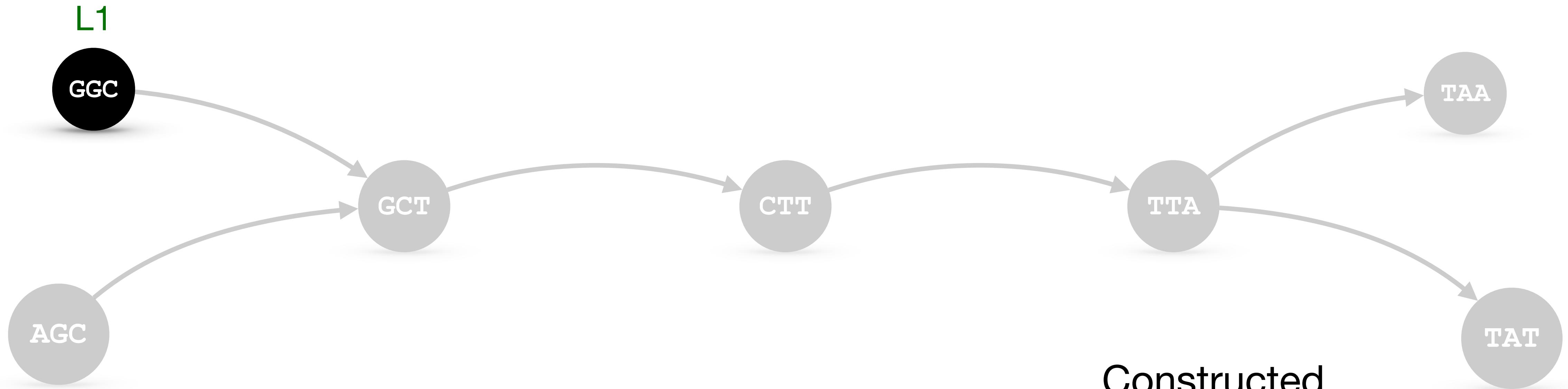
L1: GGCTTAT

L2: AGCTTAA

L3: TTAA

Background

Annotated de Bruijn Graphs



Constructed
from sequences

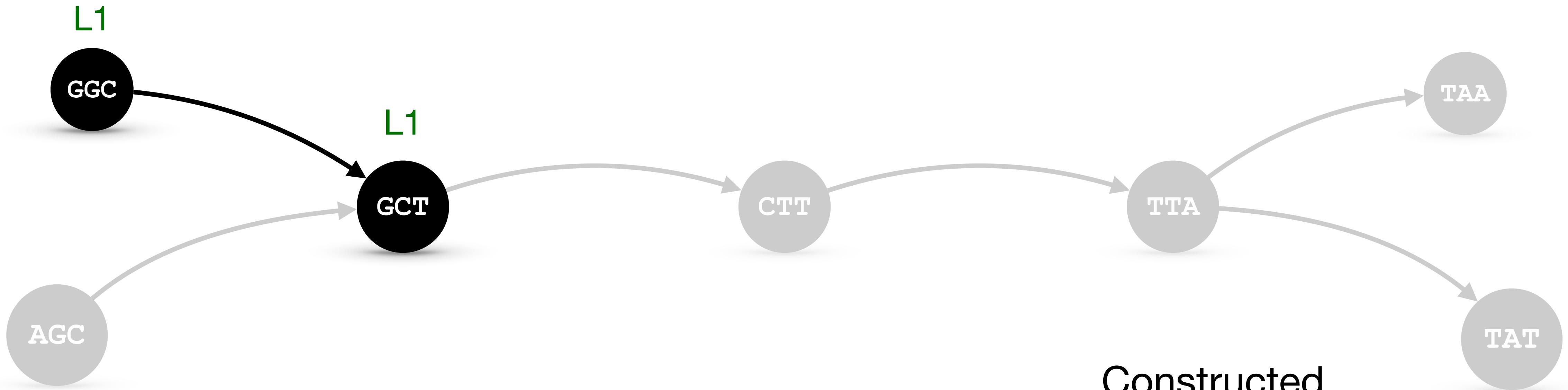
L1: GGC TTAT

L2: AGCTTAA

L3: TTAA

Background

Annotated de Bruijn Graphs



Constructed
from sequences

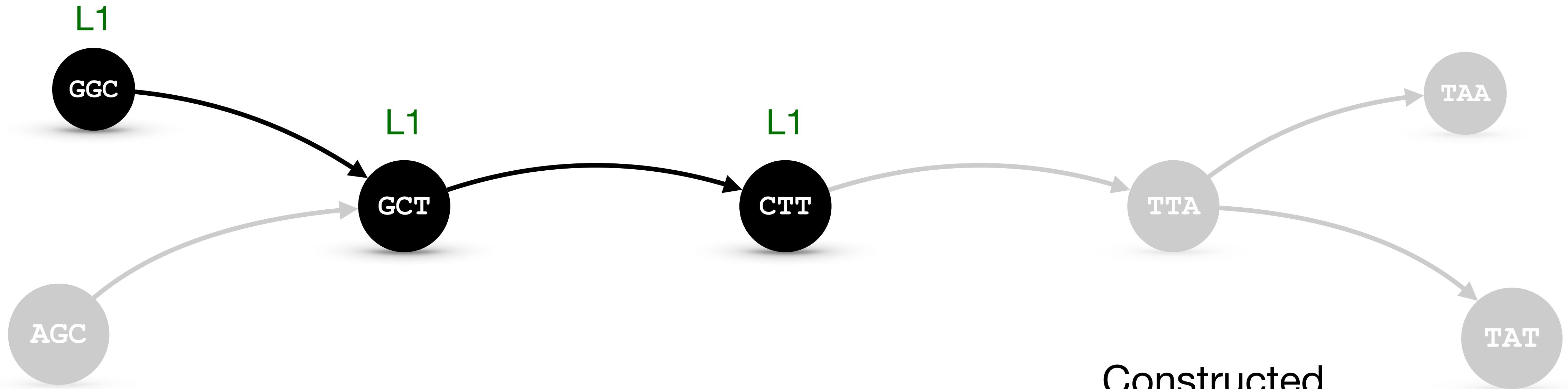
L1: GG**G**CTTAT

L2: AGCTTAA

L3: TTAA

Background

Annotated de Bruijn Graphs



Constructed
from sequences

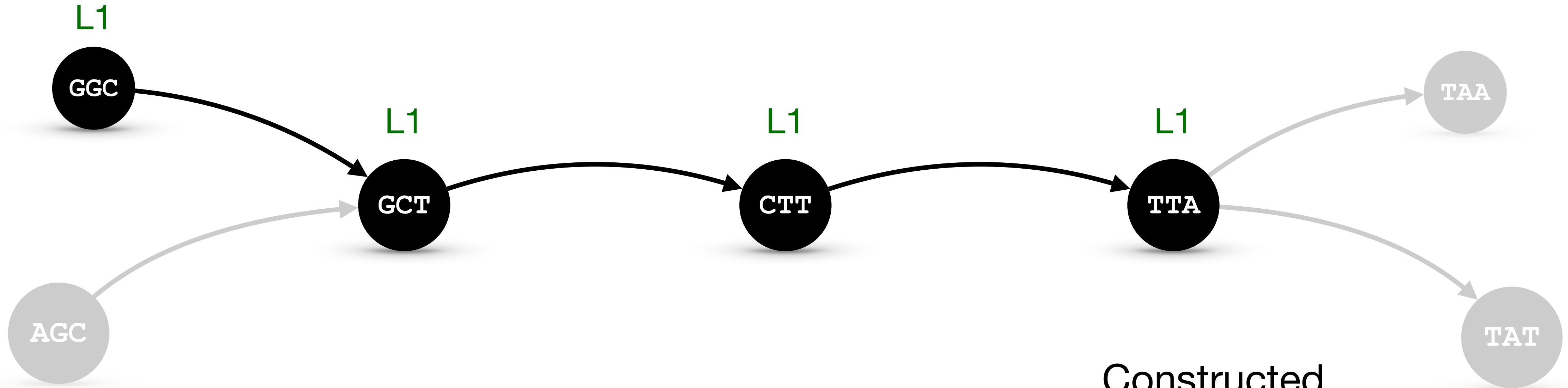
L1: GG**G**CTT**T**AT

L2: AG**G**CTTAA

L3: **T**TAA

Background

Annotated de Bruijn Graphs



Constructed
from sequences

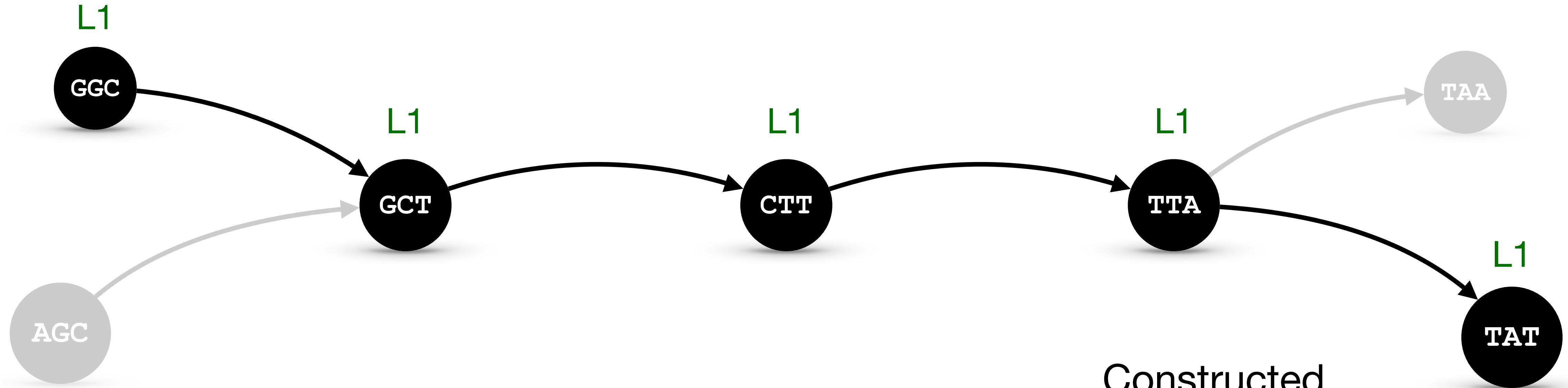
L1: GGCTTAT

L2: AGCTTAA

L3: TTAA

Background

Annotated de Bruijn Graphs



Constructed
from sequences

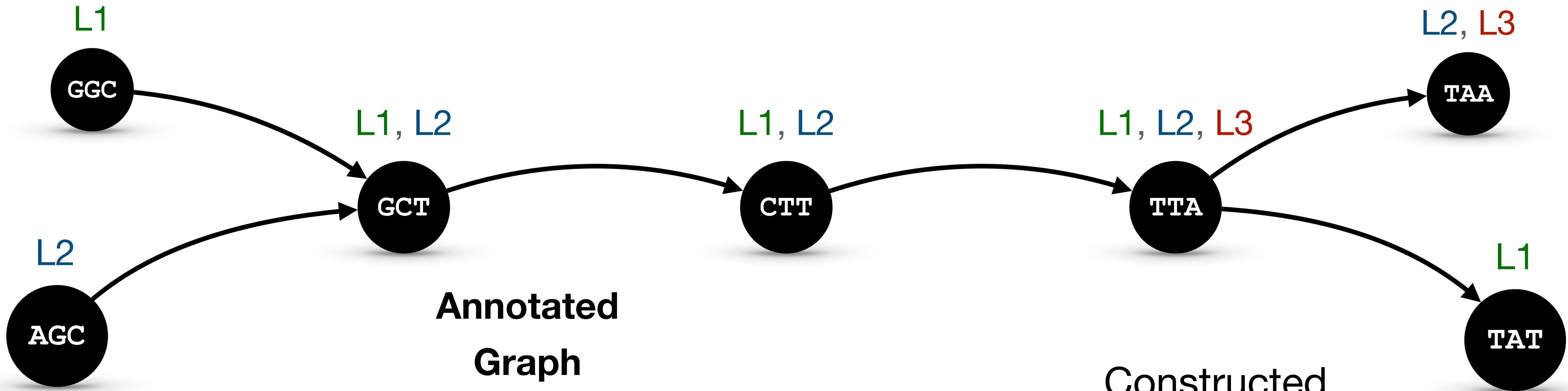
L1: GGCTTAT

L2: AGCTTAA

L3: TTAA

Background

Annotated de Bruijn Graphs



Constructed
from sequences

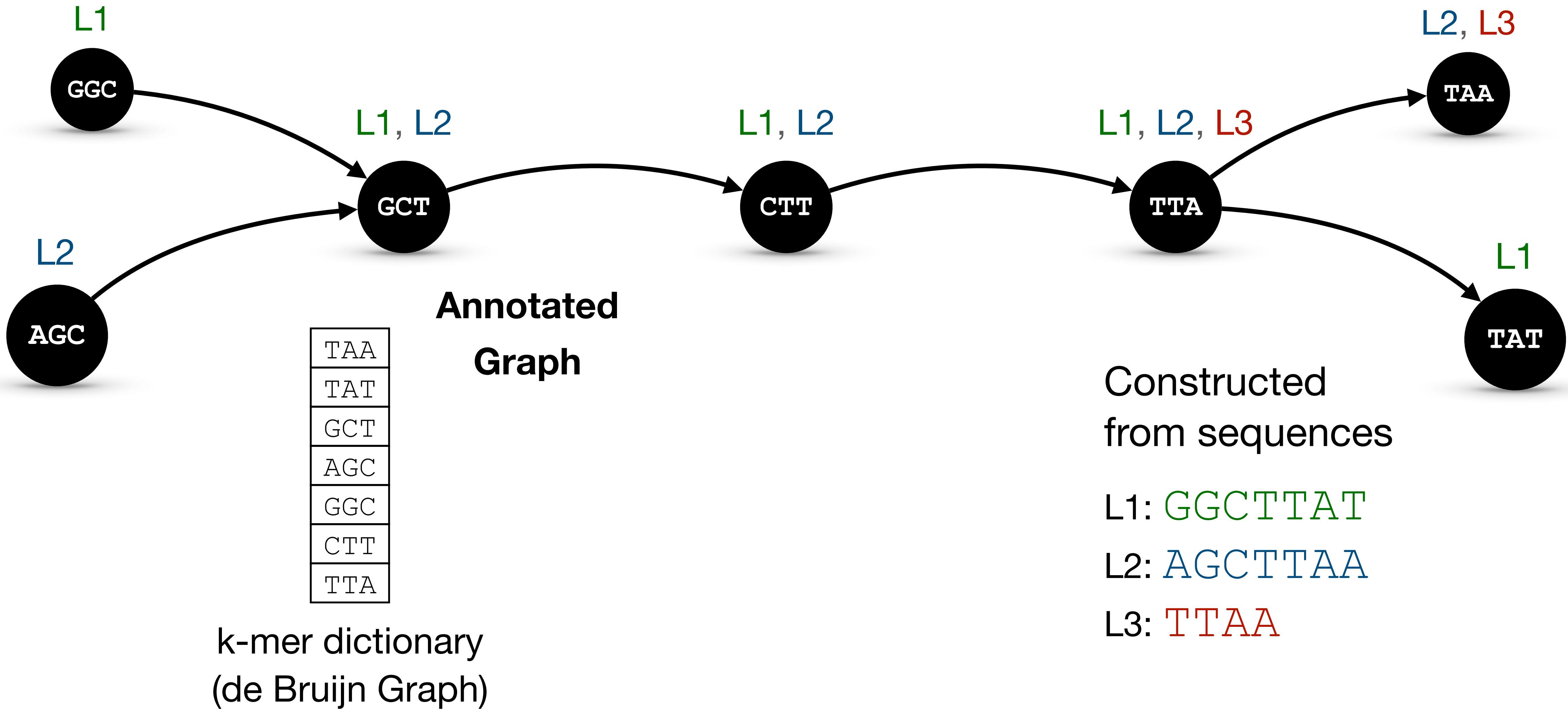
L1: GGCTTAT

L2: AGCTTAA

L3: TTAA

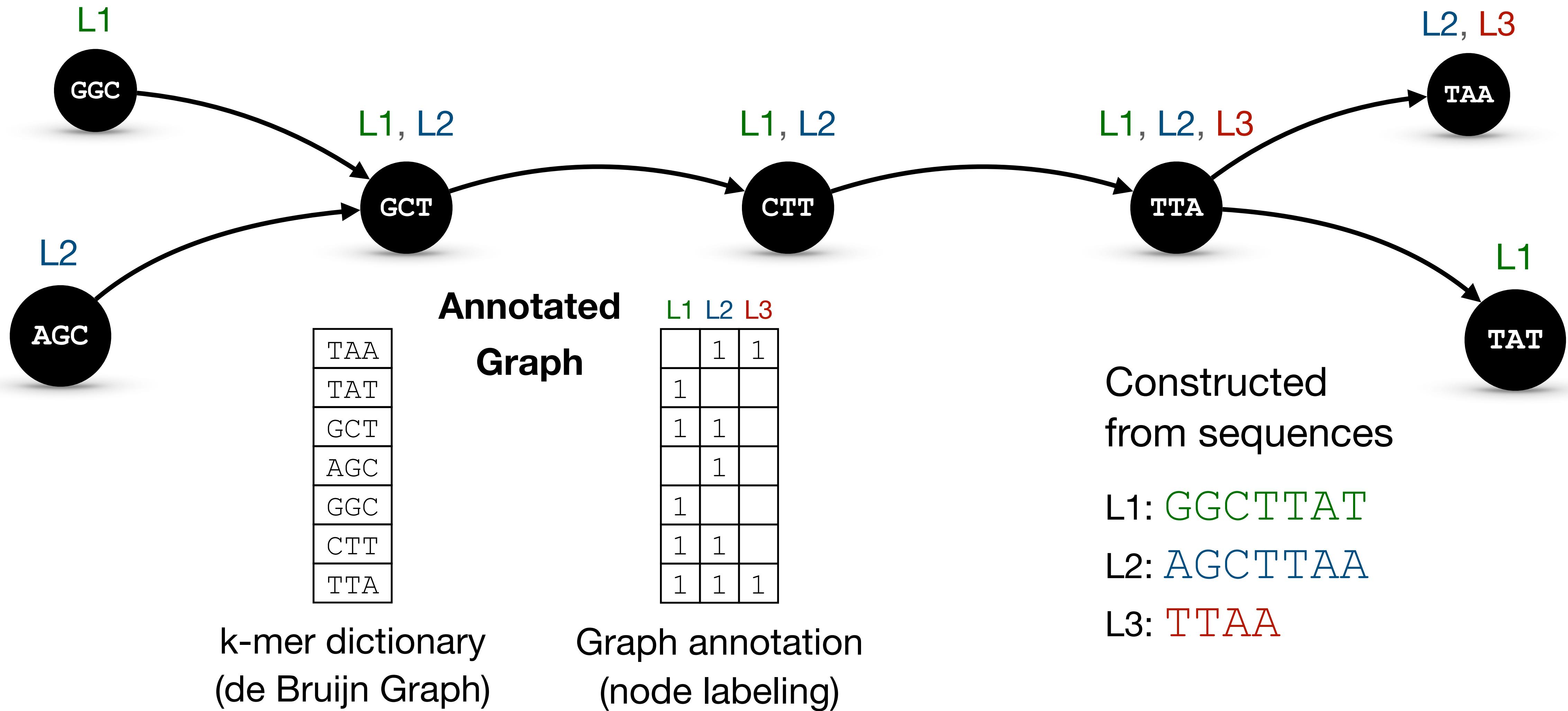
Background

Annotated de Bruijn Graphs



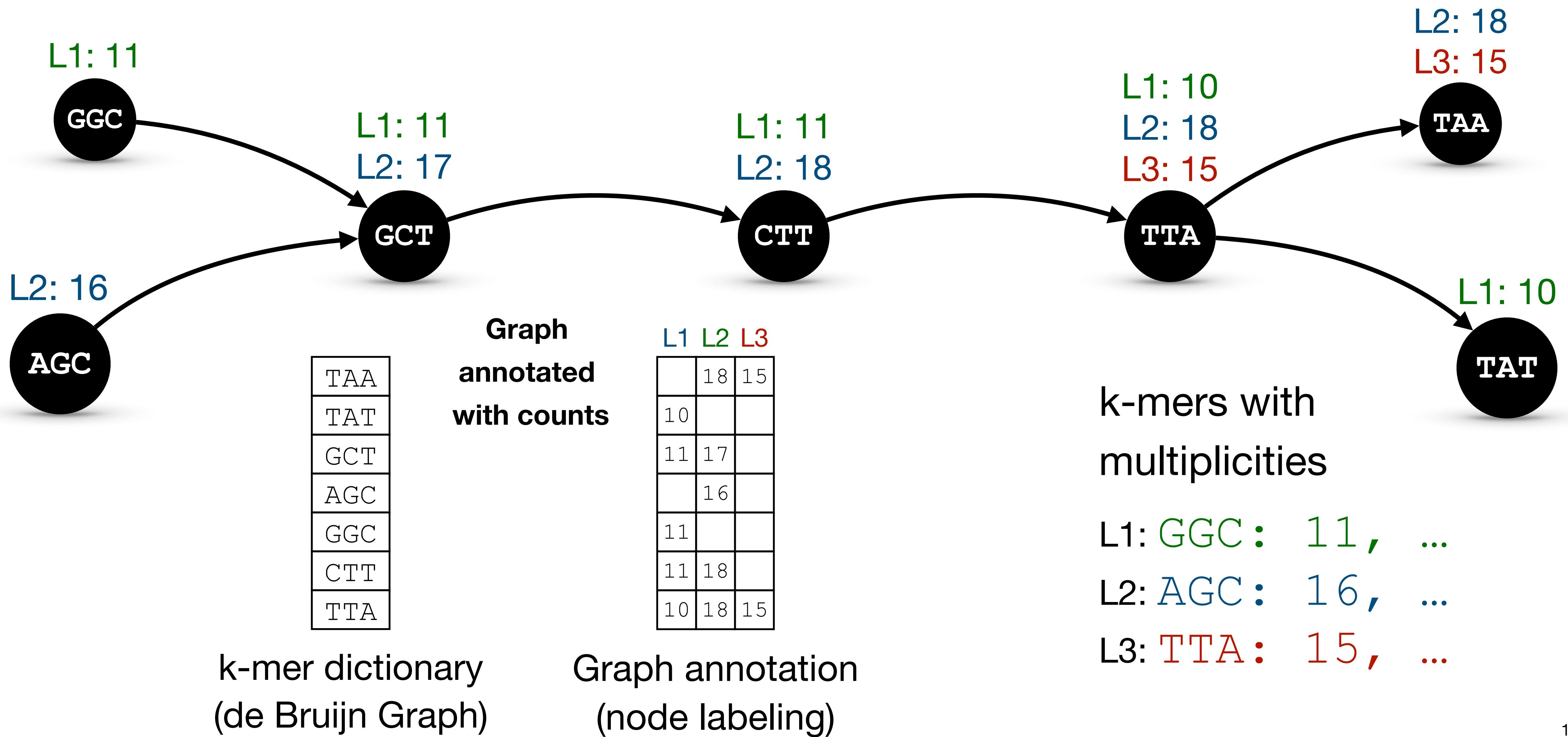
Background

Annotated de Bruijn Graphs



Motivation

1. Representing k-mer abundances



Motivation

2. Representing k-mer coordinates

ACTAGCTAGCTAG

Motivation

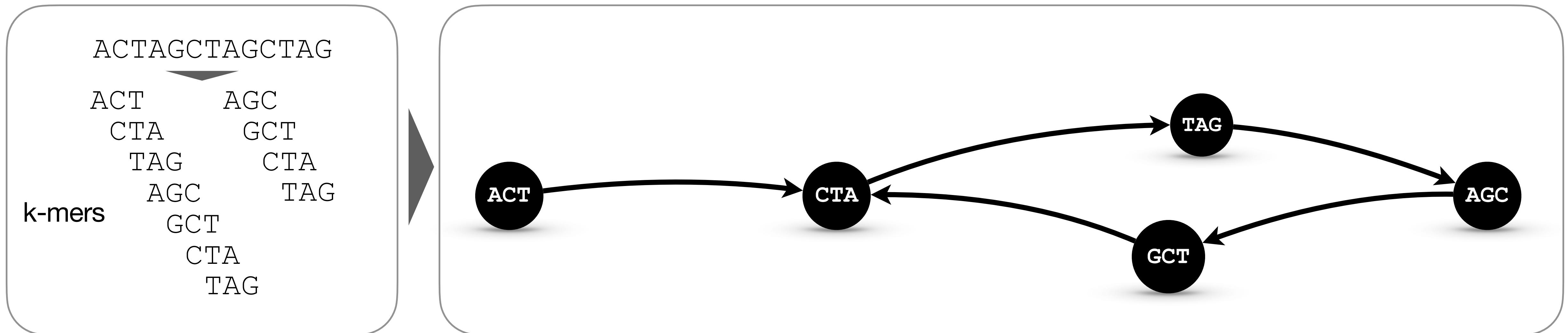
2. Representing k-mer coordinates

ACTAGCTAGCTAG
ACT AGC
CTA GCT
TAG CTA
AGC TAG
GCT
CTA
TAG

k-mers

Motivation

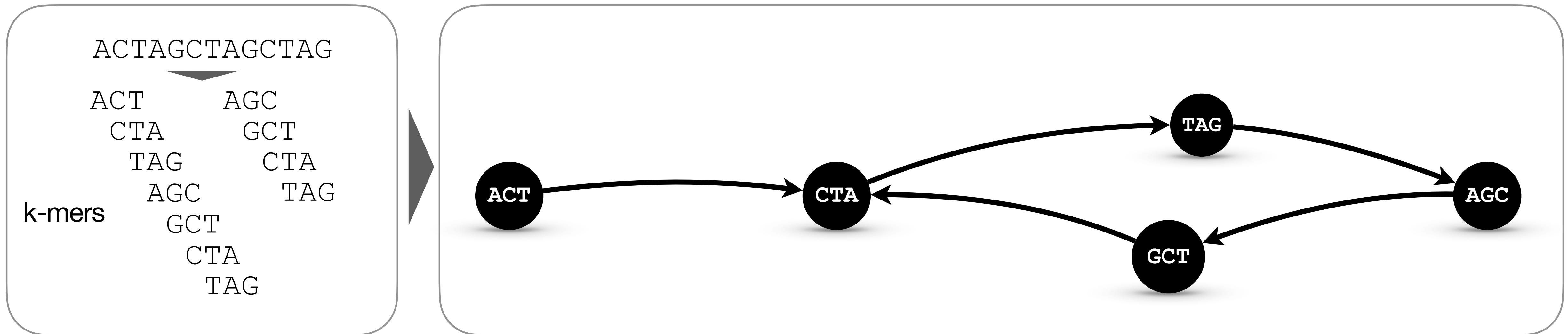
2. Representing k-mer coordinates



de Bruijn graph

Motivation

2. Representing k-mer coordinates

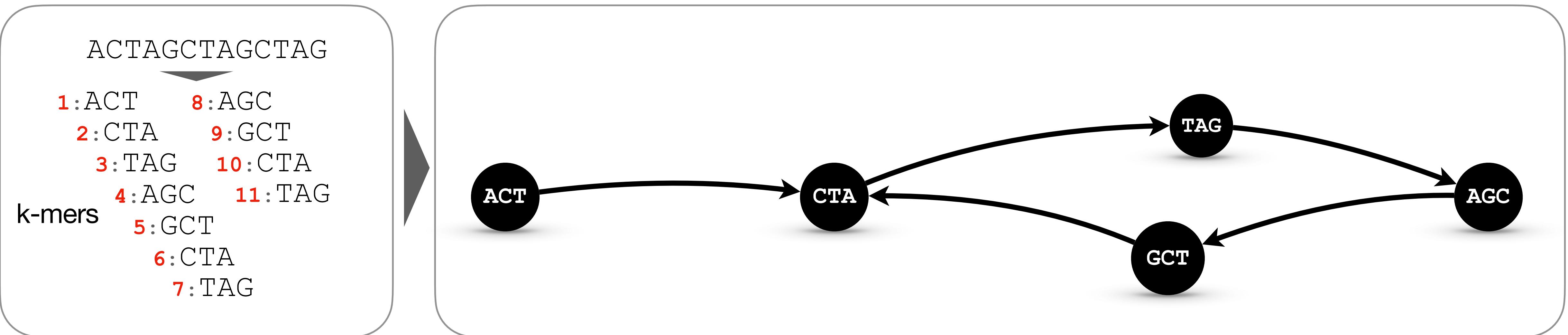


**Not invertible
representation** 😞

de Bruijn graph

Motivation

2. Representing k-mer coordinates

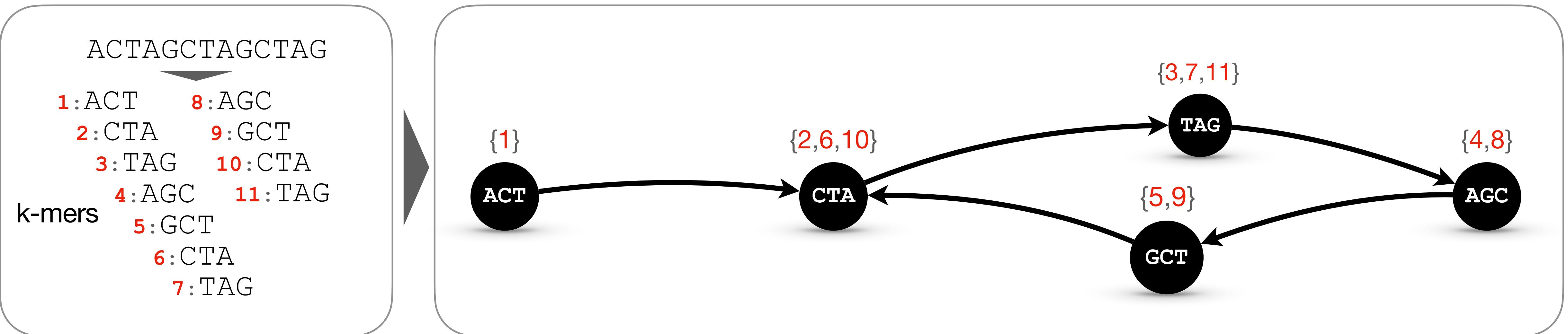


**Not invertible
representation** 😞

de Bruijn graph

Motivation

2. Representing k-mer coordinates

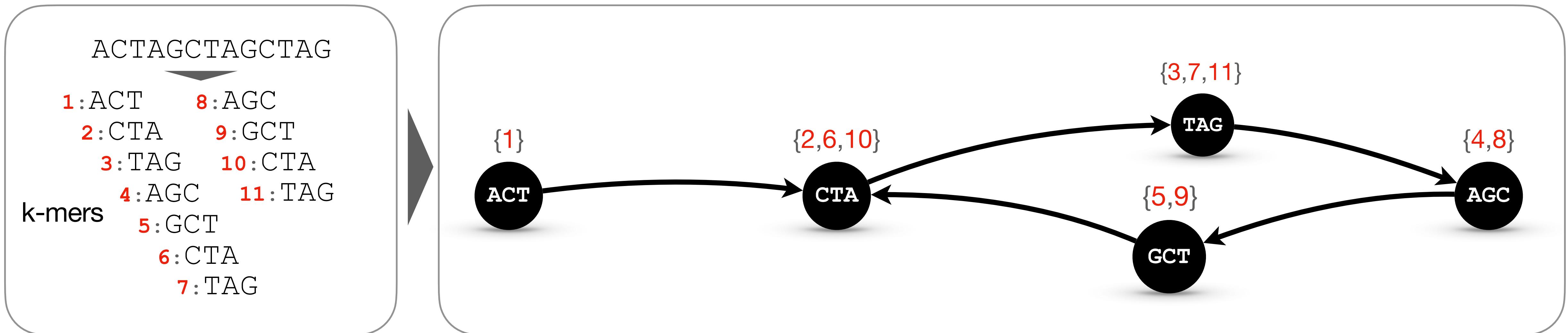


Invertible 😊

de Bruijn graph
with k-mer coordinates

Motivation

2. Representing k-mer coordinates



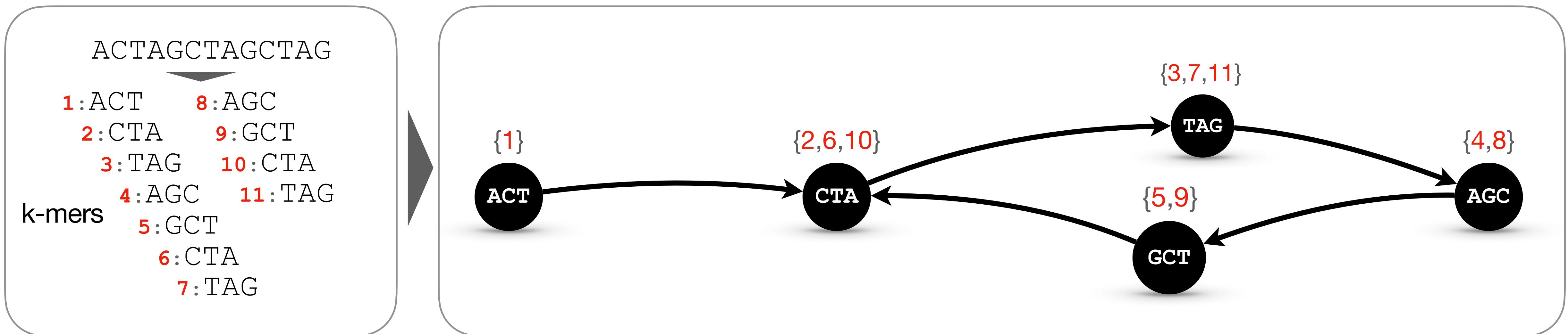
Encoding **sequence traces** allows

- reconstructing indexed sequences
(hence, **lossless sequence representation**)

de Bruijn graph
with k-mer coordinates

Motivation

2. Representing k-mer coordinates



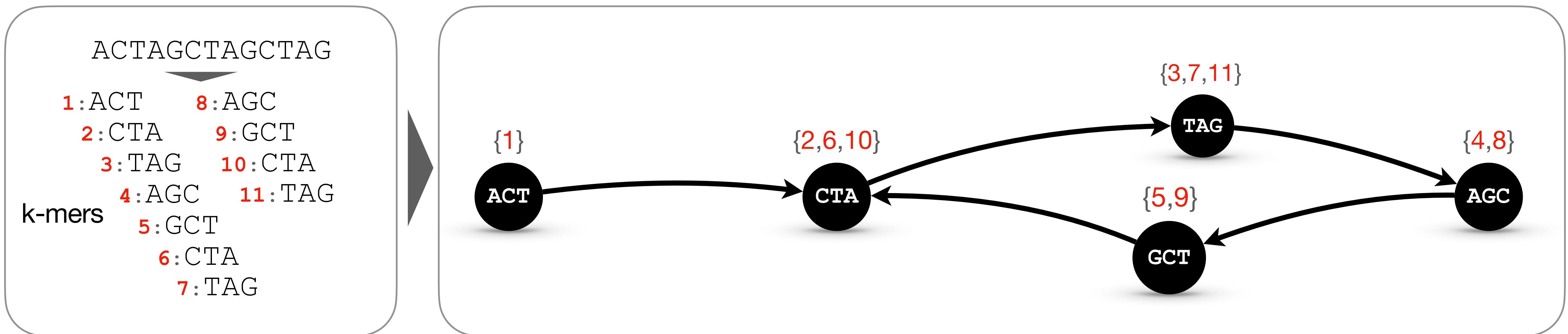
Encoding **sequence traces** allows

- reconstructing indexed sequences
(hence, **lossless sequence representation**)
- performing exact sequence alignment

de Bruijn graph
with k-mer coordinates

Motivation

2. Representing k-mer coordinates



Encoding **sequence traces** allows

- reconstructing indexed sequences
(hence, **lossless sequence representation**)
- performing exact sequence alignment
- retrieving all **traces** crossing a node

de Bruijn graph
with k-mer coordinates

Questions to address

Questions to address

1. How to **efficiently represent** a huge sparse **non-binary matrix**?

Questions to address

1. How to **efficiently represent** a huge sparse **non-binary matrix**?
2. Can we **employ existing repr. schemes** for binary matrices?

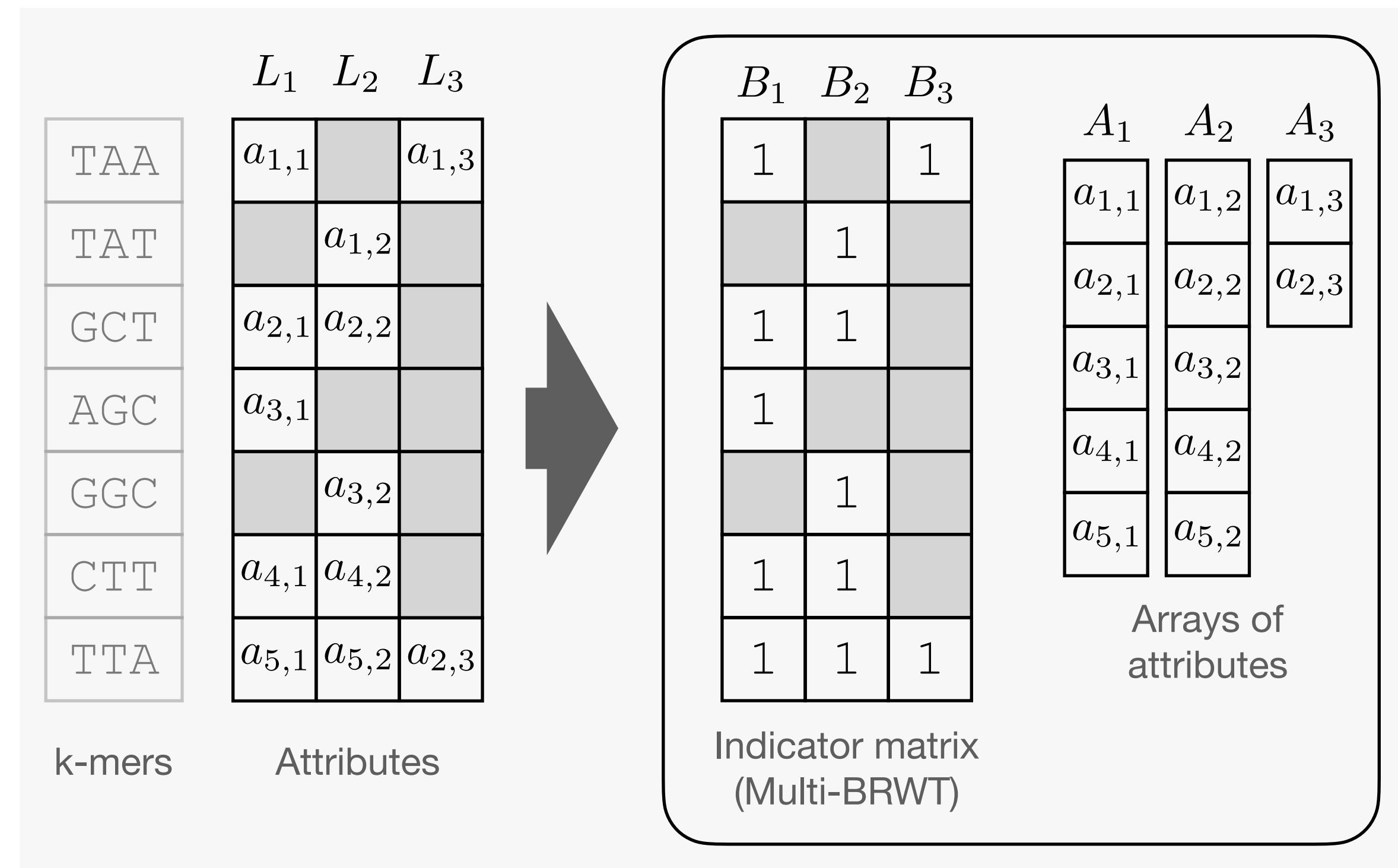
Questions to address

1. How to **efficiently represent** a huge sparse **non-binary matrix**?
2. Can we **employ existing repr. schemes** for binary matrices?
3. How to **exploit regularities in graph annotations**?

Method

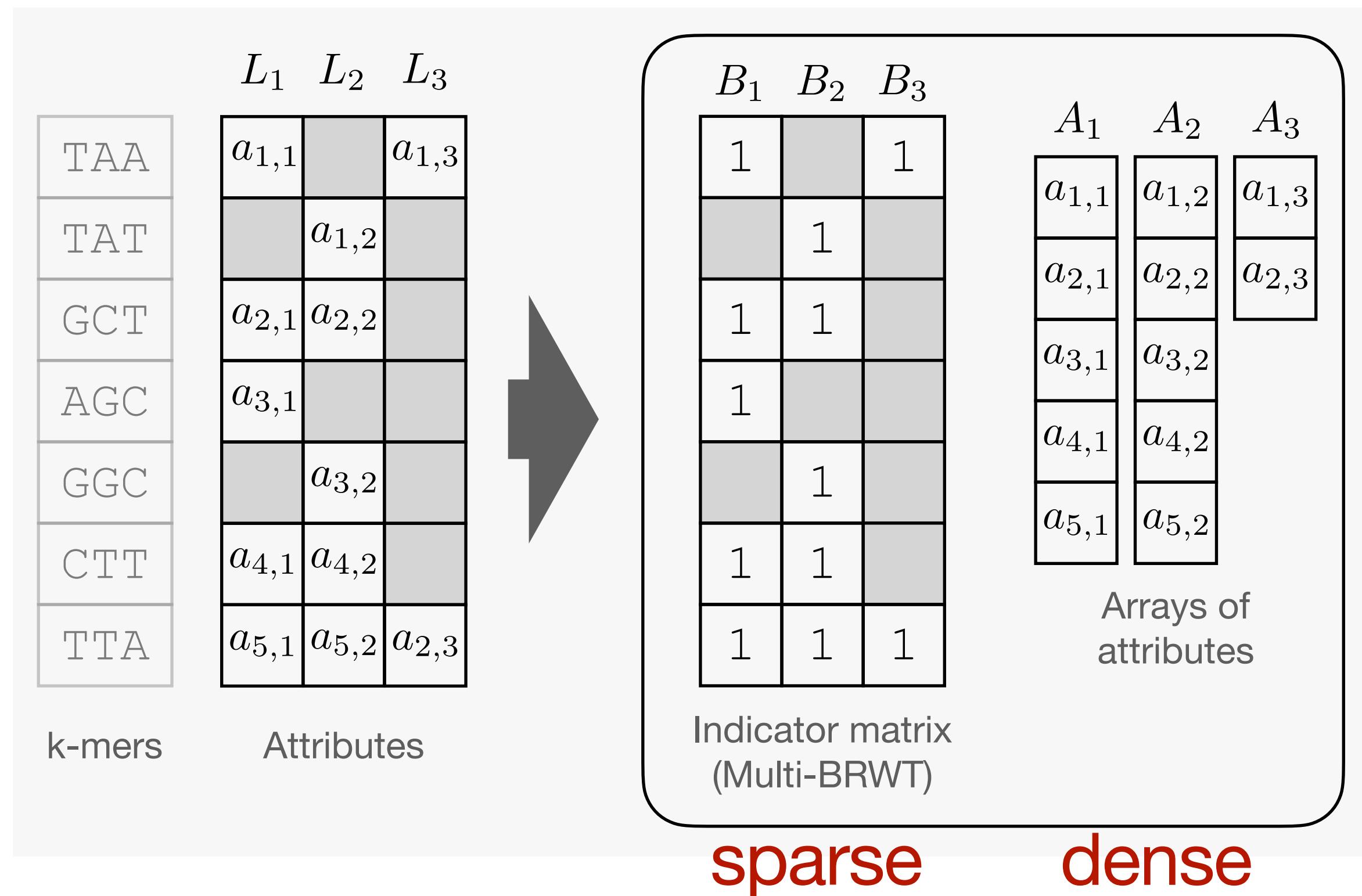
Method

General scheme for sparse matrices



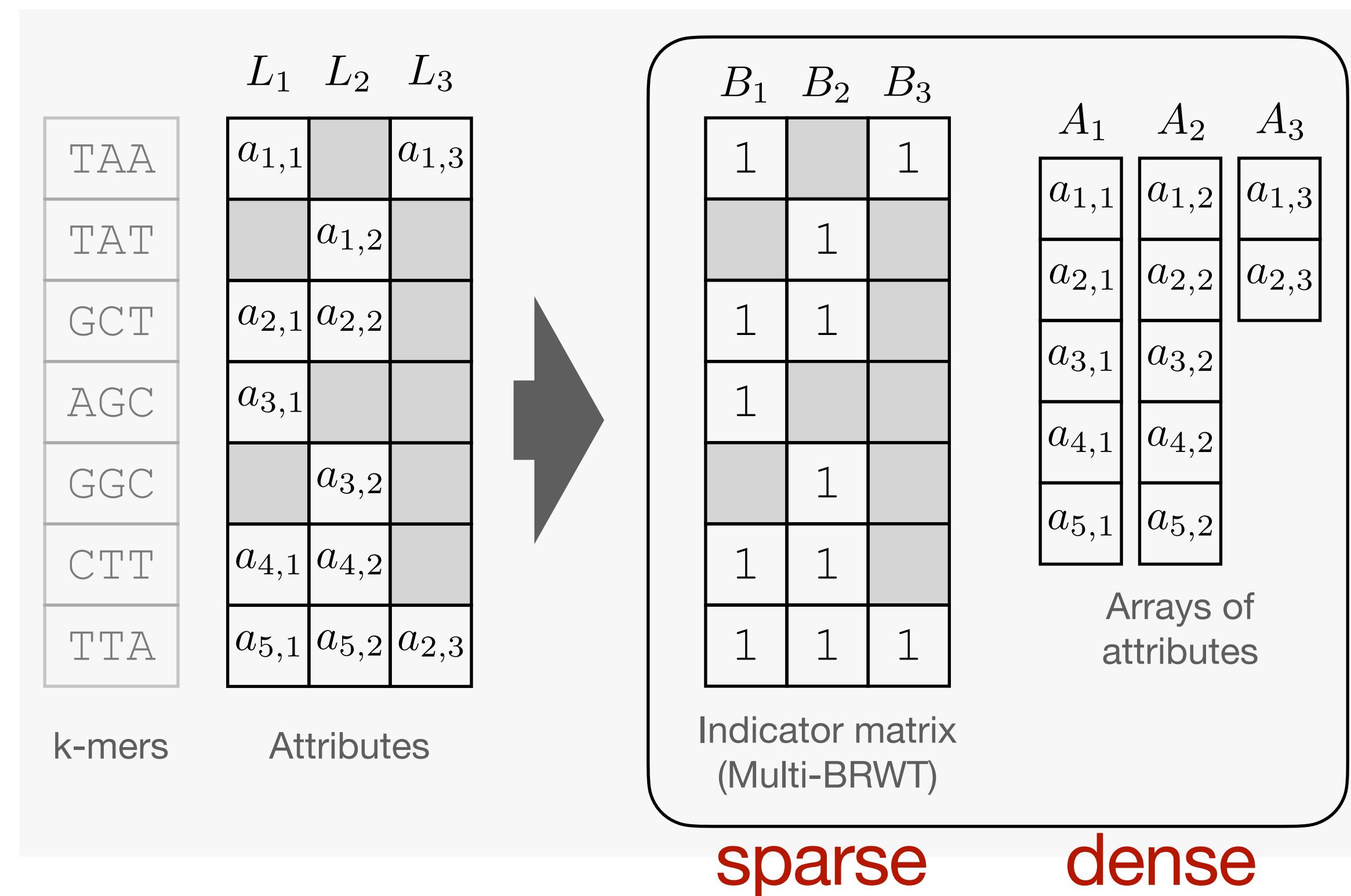
Method

General scheme for sparse matrices



Method

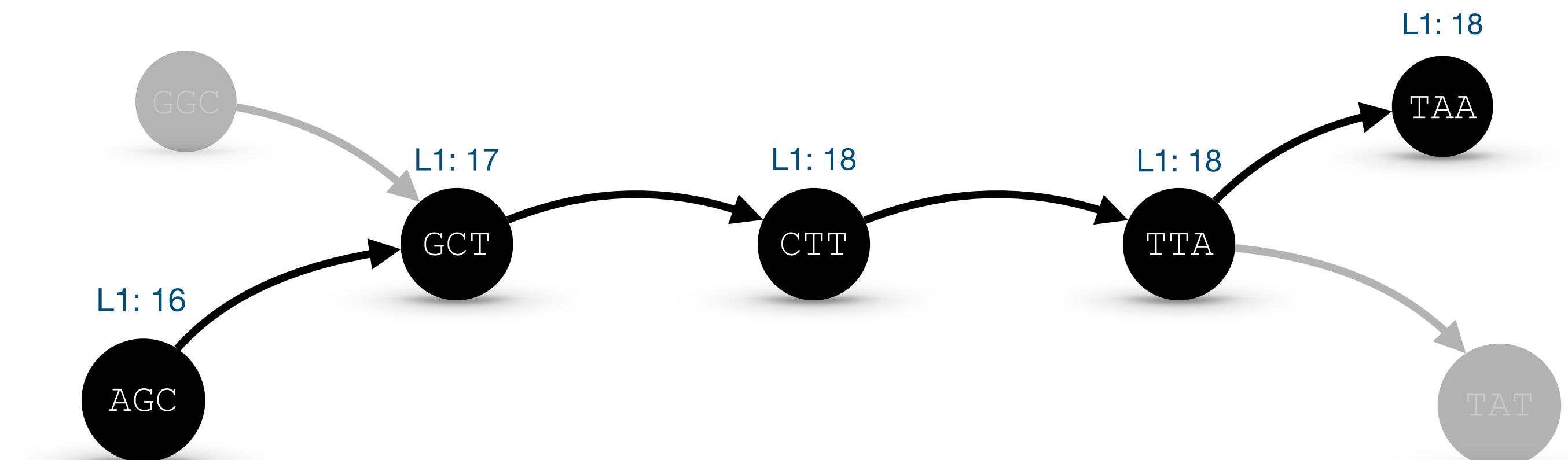
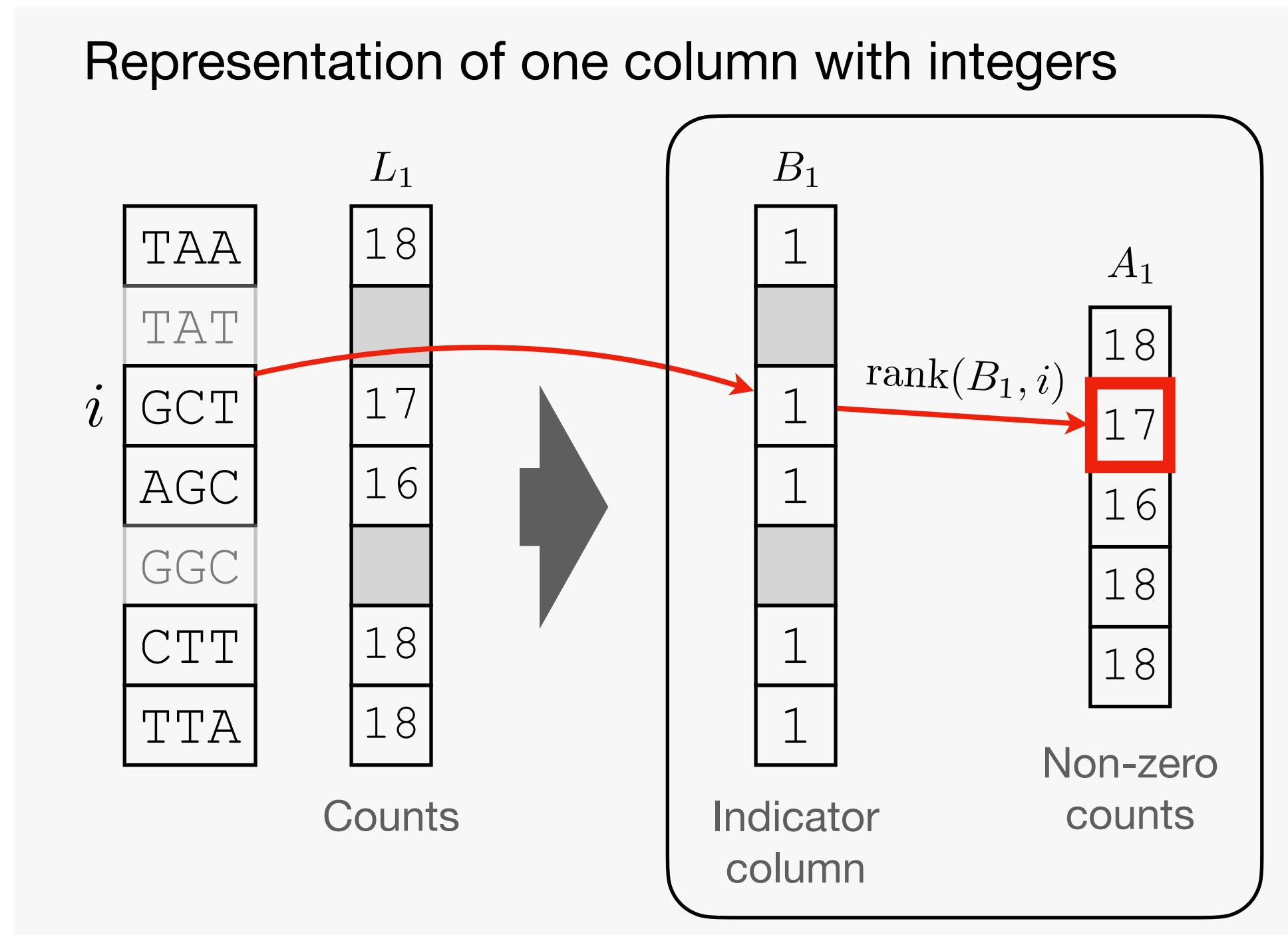
General scheme for sparse matrices



Theorem 1. *If both the indicator matrix and the arrays of attributes are represented succinctly, the proposed scheme also is a succinct representation of the matrix. That is, there is no other data structure that could represent any such matrix with asymptotically fewer bits.*

Method

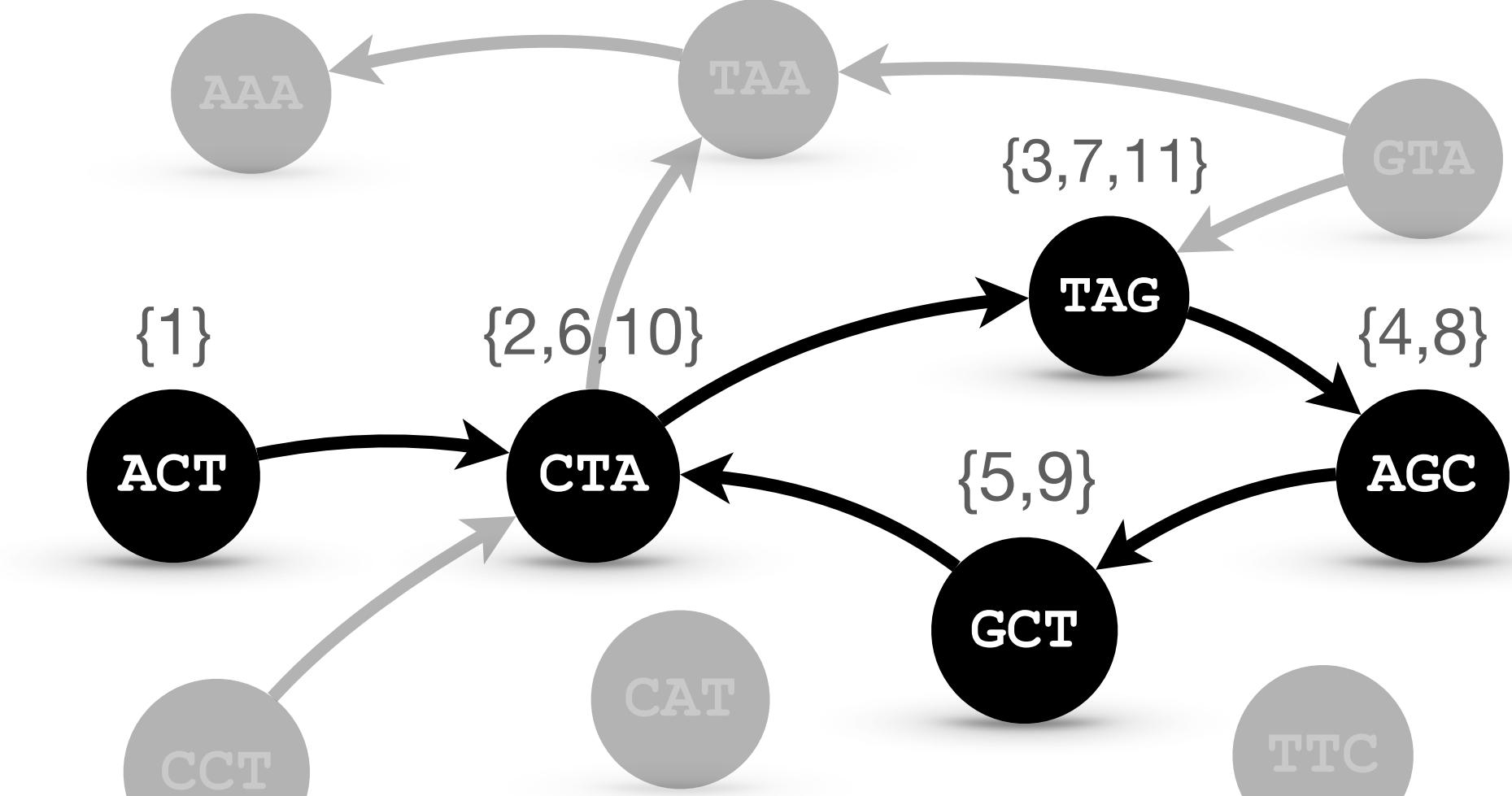
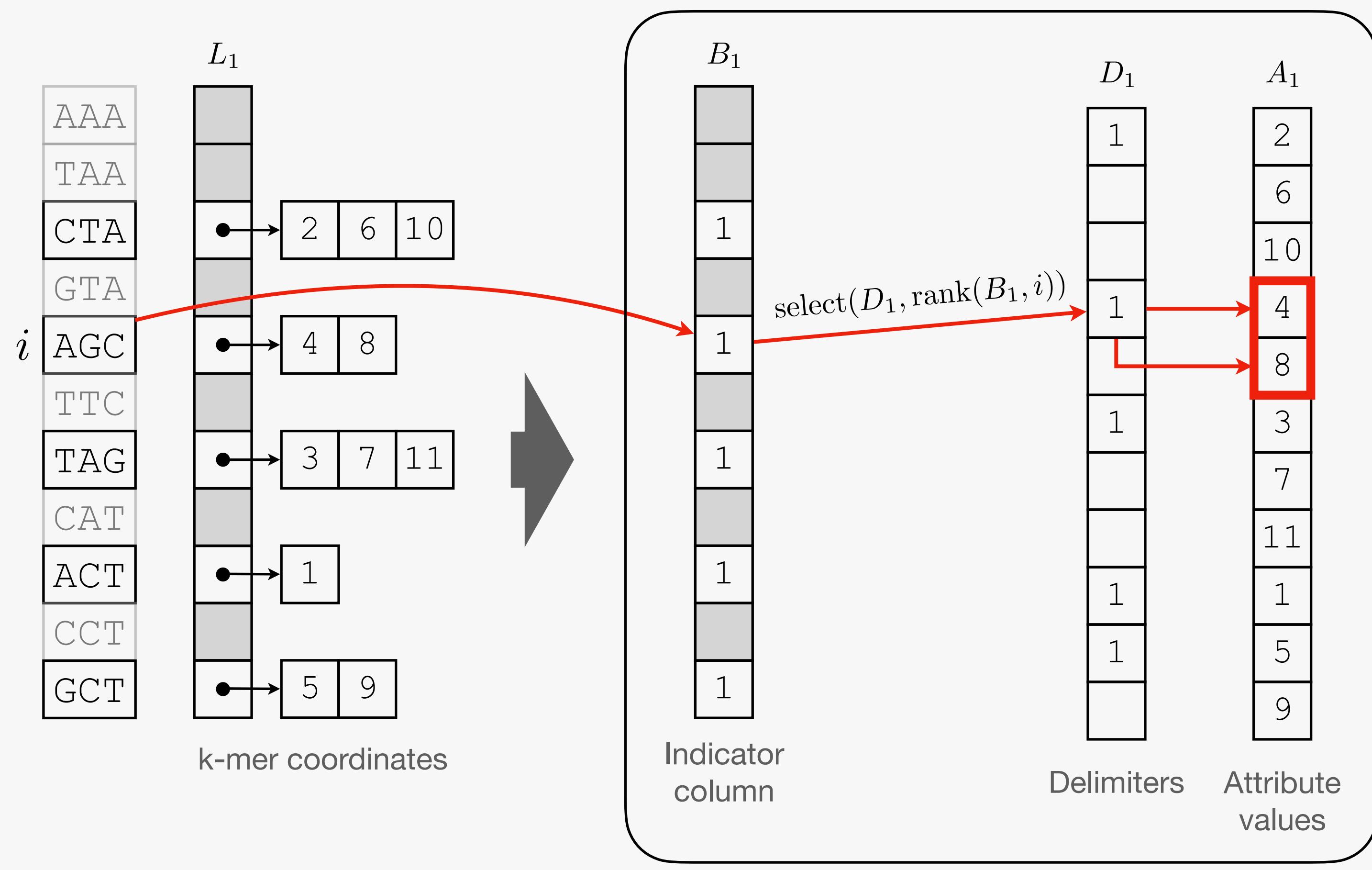
1. One column with integers (k-mer abundances)



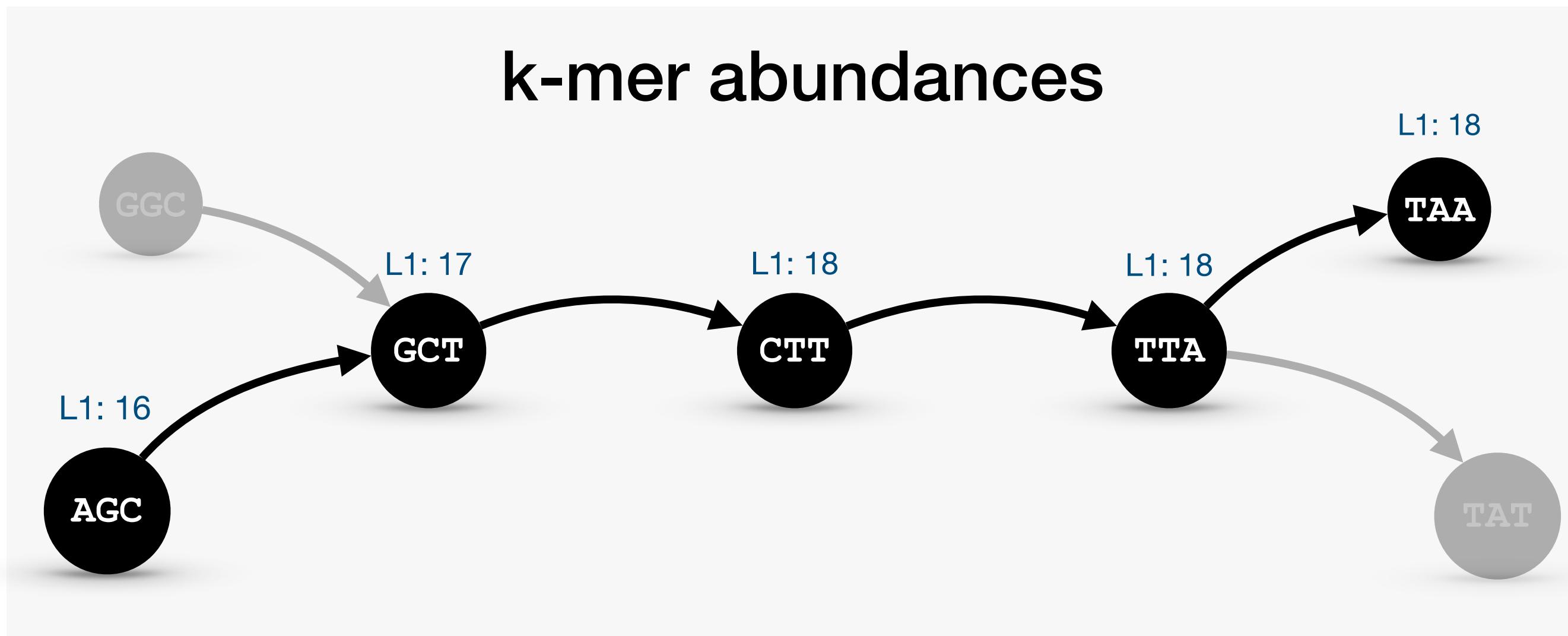
Method

2. One column with integer tuples (k-mer coordinates)

Representation of one column with sets, e.g., k-mer coordinates



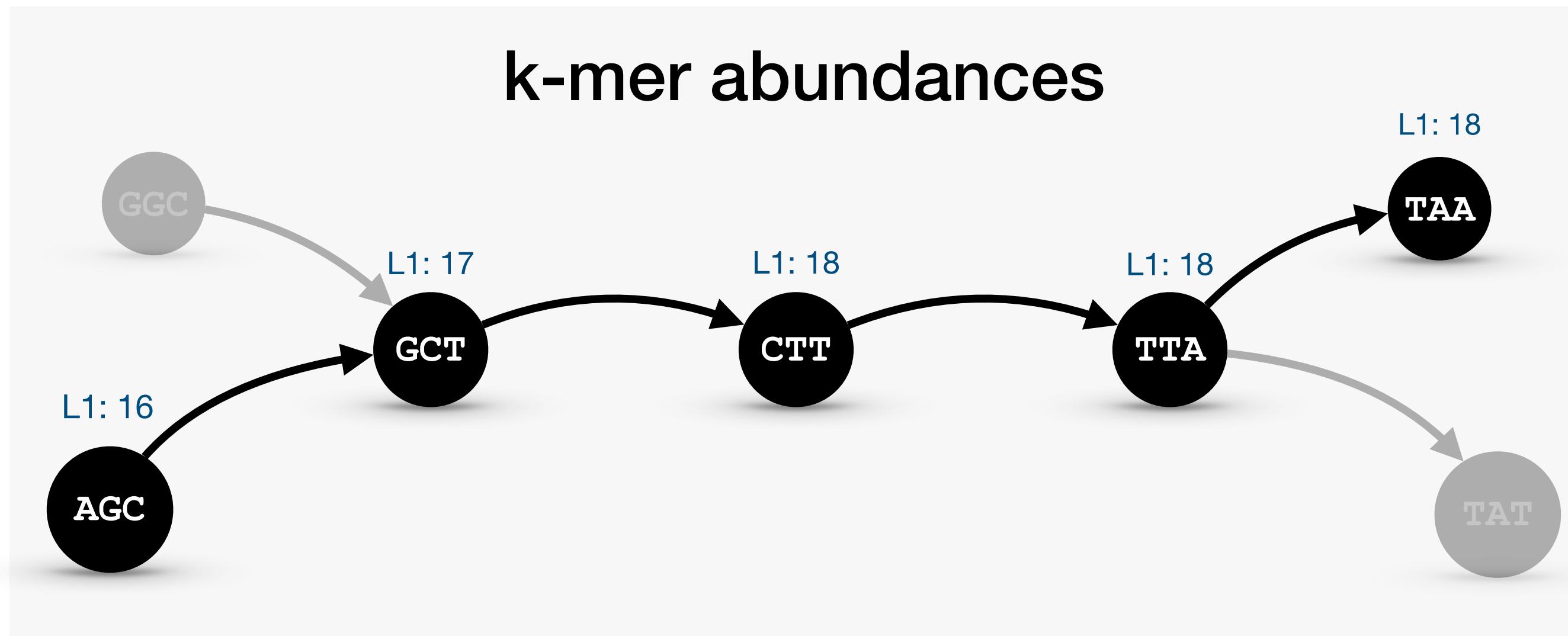
Exploiting regularities



Observe regularities:

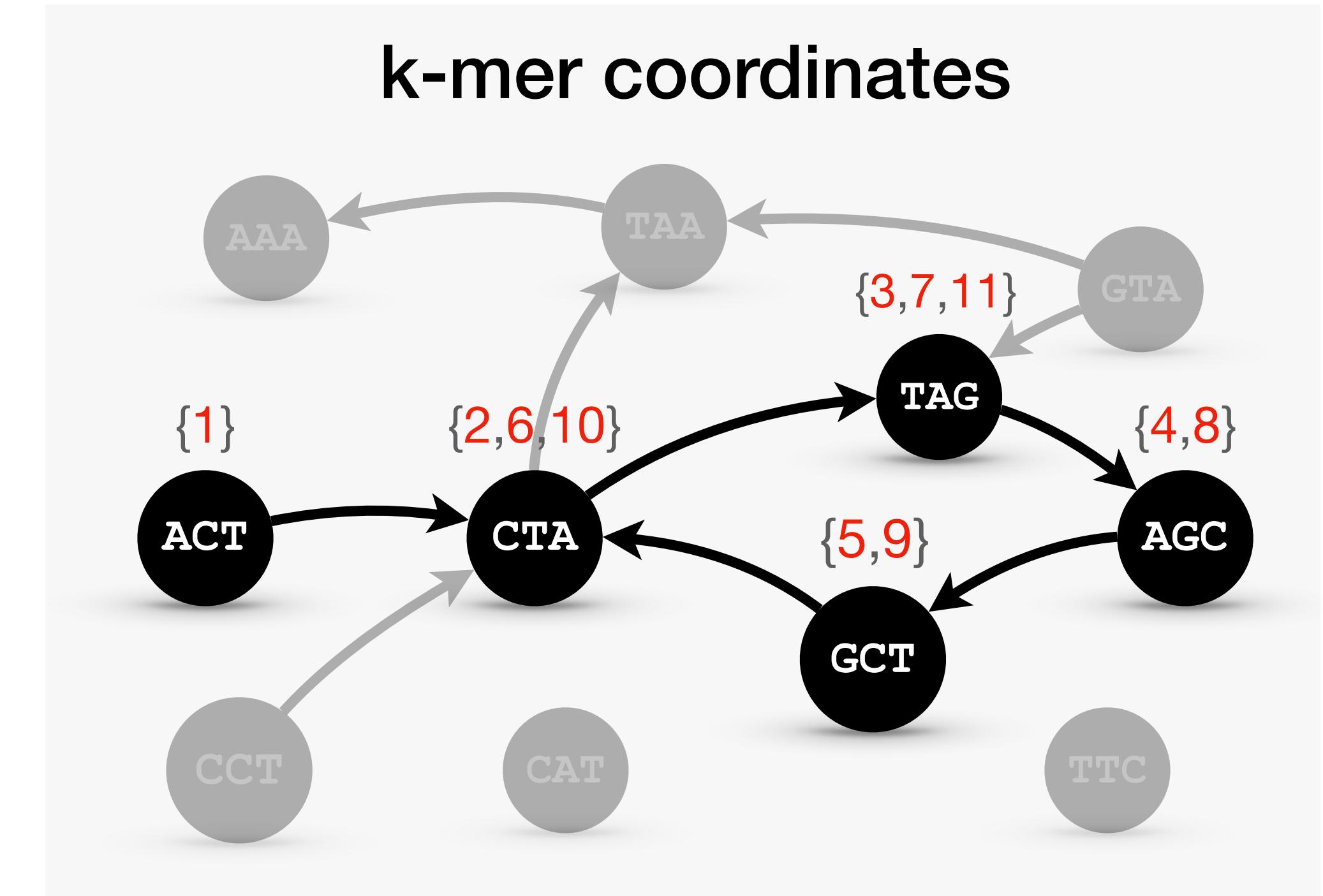
- Abundances of neighboring k-mers are often similar

Exploiting regularities

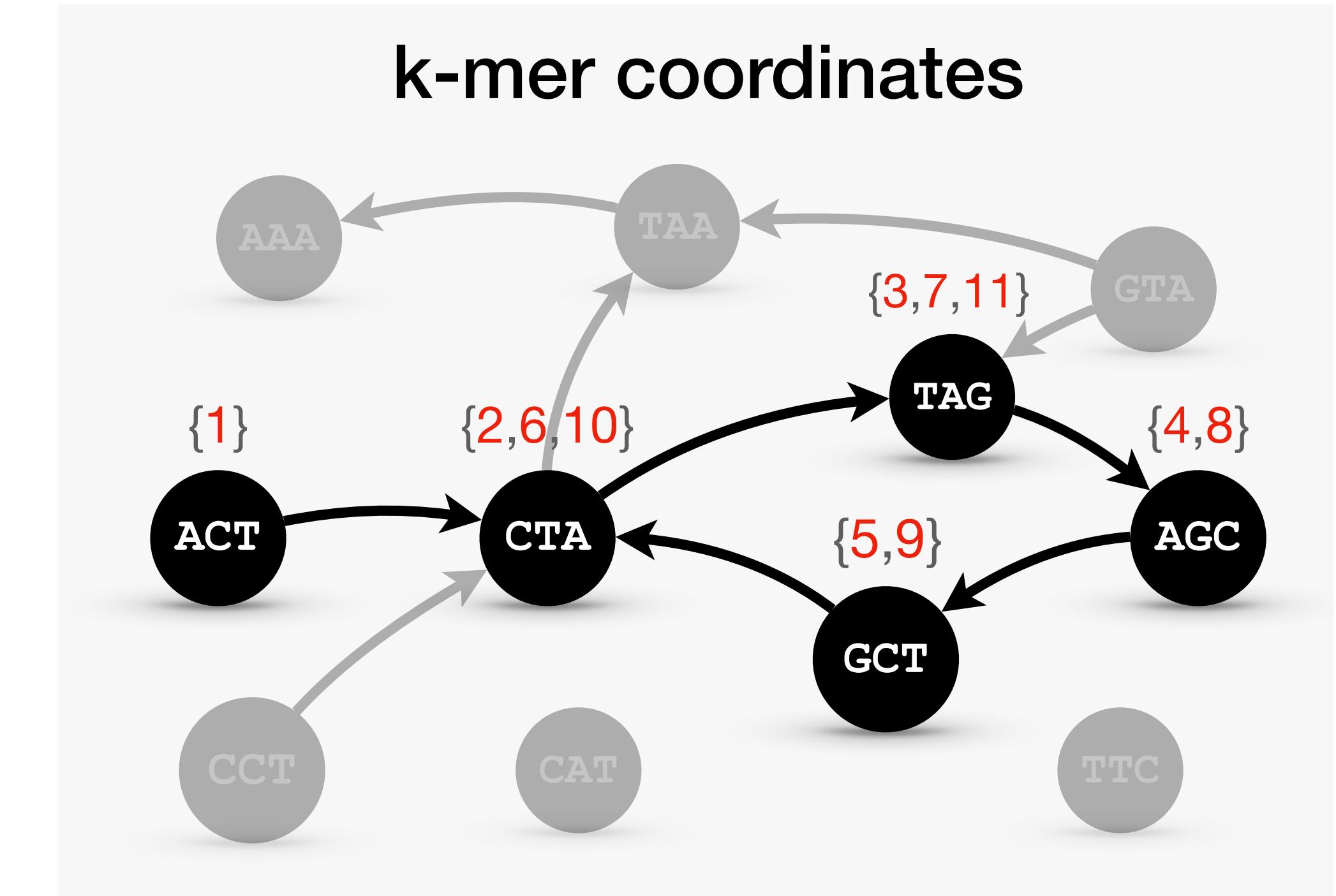
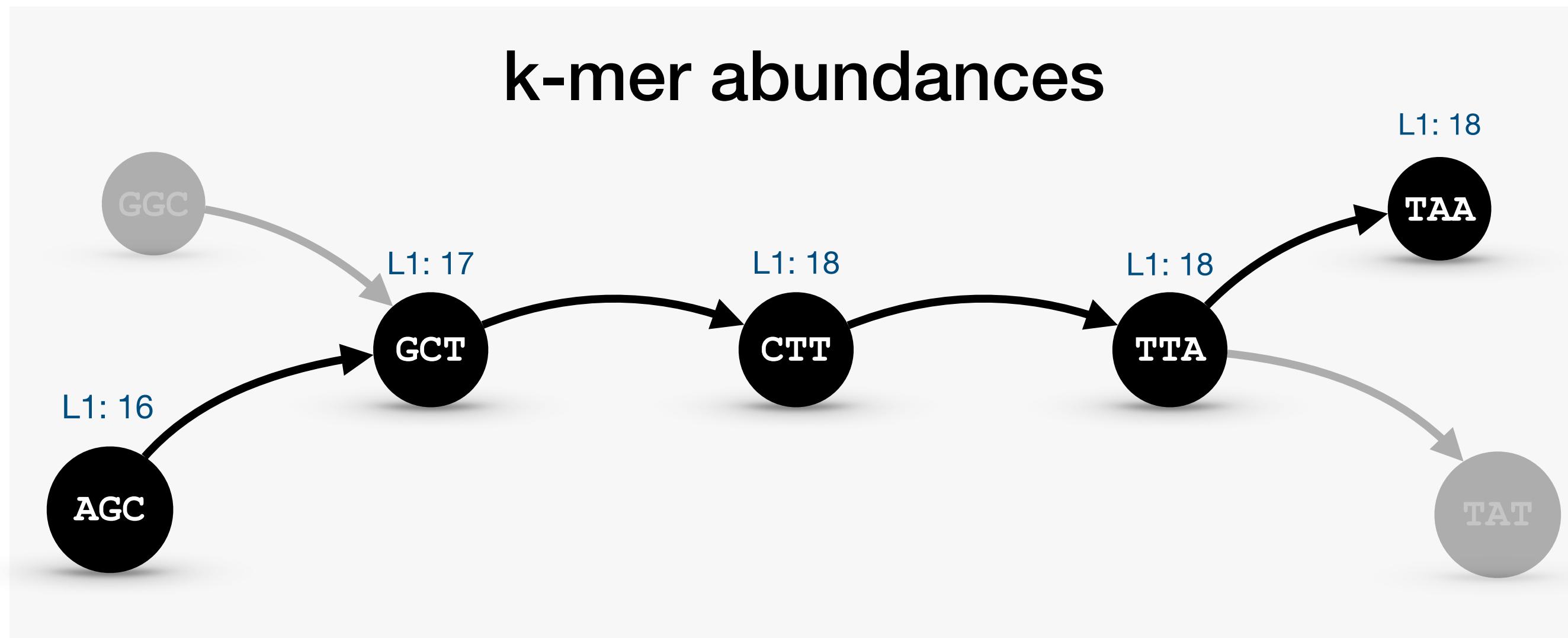


Observe regularities:

- Abundances of neighboring k-mers are often similar
- Coordinates for adjacent k-mers always differ by +1



Exploiting regularities



Observe regularities:

- Abundances of neighboring k-mers are often similar
- Coordinates for adjacent k-mers always differ by +1

Idea

Generalize the RowDiff scheme

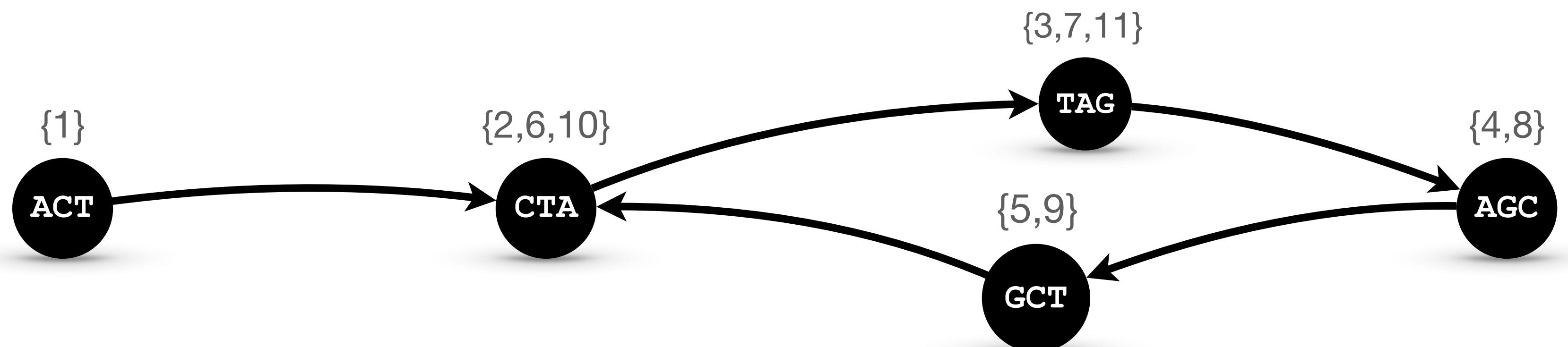


Delta coding for k-mer coordinates

A. Enumeration of k-mers

ACTAGCTAGCTAG
1:ACT 8:AGC
2:CTA 9:GCT
3:TAG 10:CTA
4:AGC 11:TAG
5:GCT
6:CTA
7:TAG

B. Delta-coding of coordinate annotations (sets)

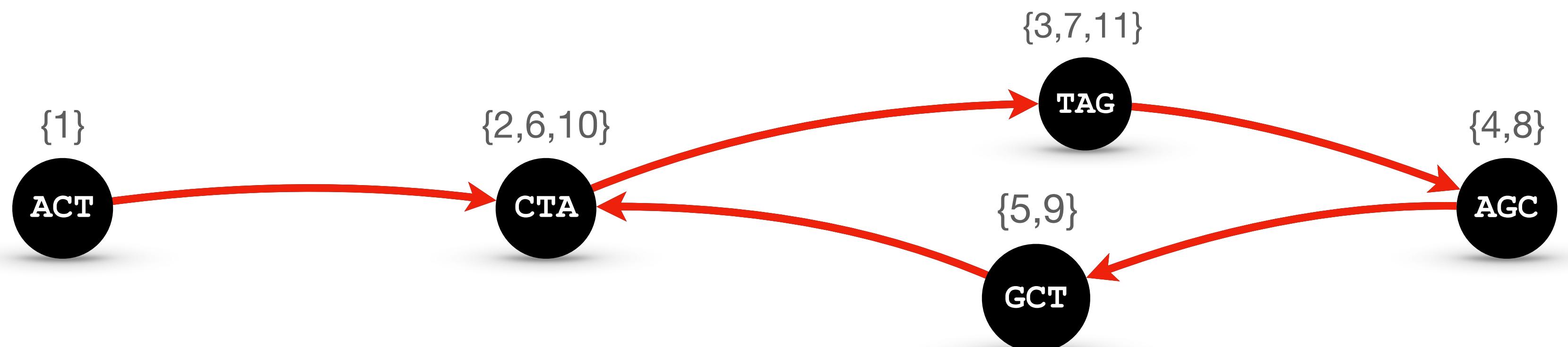


Delta coding for k-mer coordinates

A. Enumeration of k-mers

ACTAGCTAGCTAG
1:ACT 8:AGC
2:CTA 9:GCT
3:TAG 10:CTA
4:AGC 11:TAG
5:GCT
6:CTA
7:TAG

B. Delta-coding of coordinate annotations (sets)

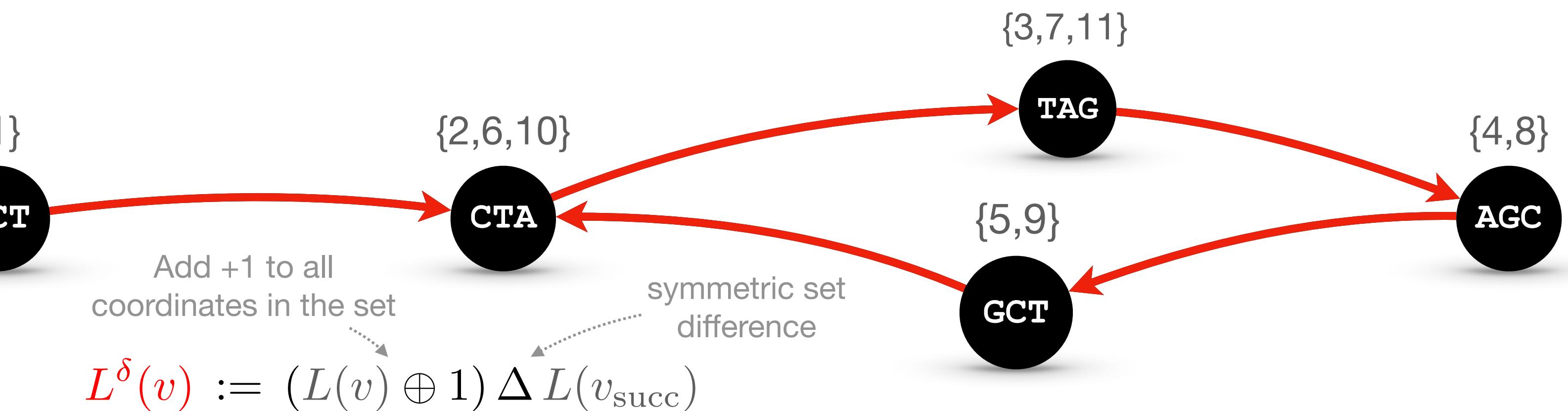


Delta coding for k-mer coordinates

A. Enumeration of k-mers

ACTAGCTAGCTAG
1:ACT 8:AGC
2:CTA 9:GCT
3:TAG 10:CTA
4:AGC 11:TAG
5:GCT
6:CTA
7:TAG

B. Delta-coding of coordinate annotations (sets)



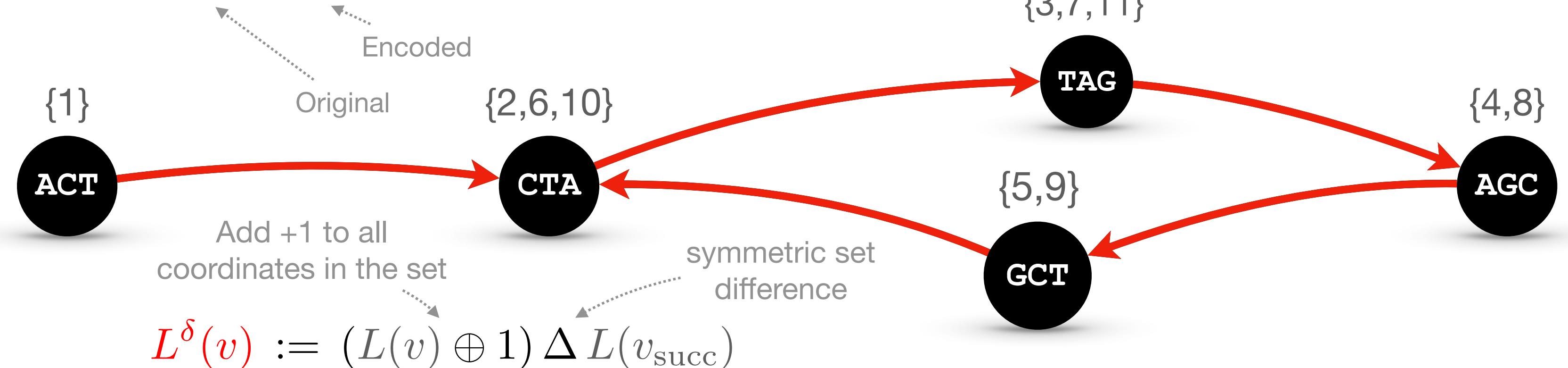
Delta coding for k-mer coordinates

A. Enumeration of k-mers

ACTAGCTAGCTAG
1:ACT 8:AGC
2:CTA 9:GCT
3:TAG 10:CTA
4:AGC 11:TAG
5:GCT
6:CTA
7:TAG

B. Delta-coding of coordinate annotations (sets)

$$L(v), L^\delta(v) \in 2^{\mathbb{N}} \quad \text{Annotations are sets}$$



Delta coding for k-mer coordinates

A. Enumeration of k-mers

ACTAGCTAGCTAG
1:ACT 8:AGC
2:CTA 9:GCT
3:TAG 10:CTA
4:AGC 11:TAG
5:GCT
6:CTA
7:TAG

B. Delta-coding of coordinate annotations (sets)

$$L(v), L^\delta(v) \in 2^{\mathbb{N}}$$

Original
Encoder

{1}

ACT

{2,6,10}

CTA

{3,7,11}

TAG

{5,9}

GCT

{4,8}

AGC

Add +1 to all
coordinates in the set

$$L^\delta(v) := (L(v) \oplus 1) \Delta L(v_{\text{succ}})$$

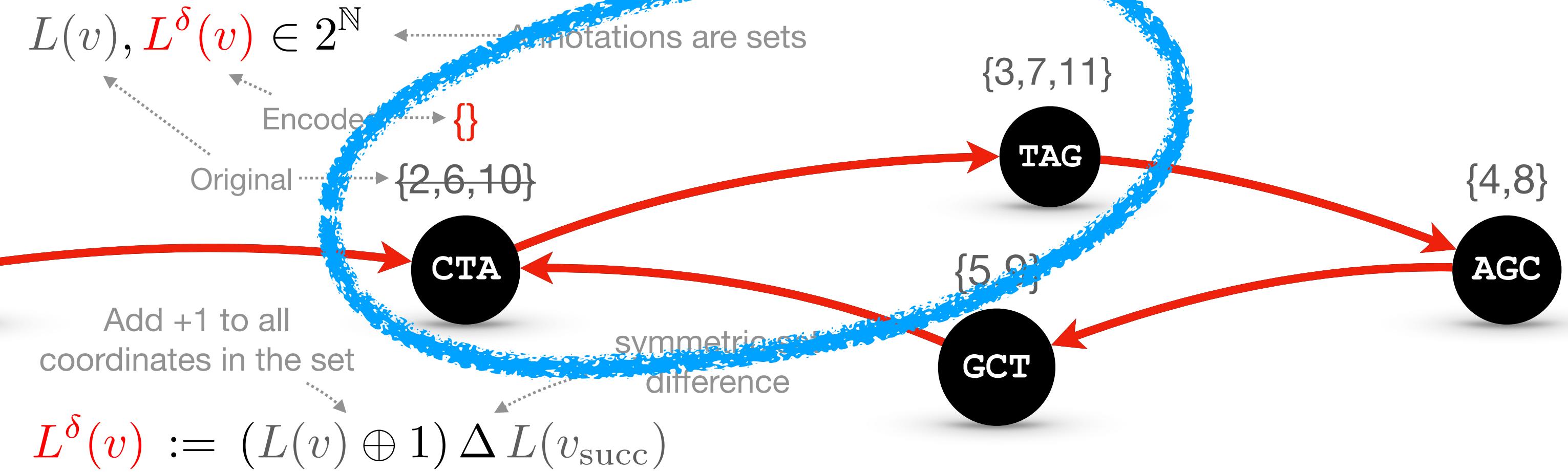
Annotations are sets

Delta coding for k-mer coordinates

A. Enumeration of k-mers

ACTAGCTAGCTAG
1:ACT 8:AGC
2:CTA 9:GCT
3:TAG 10:CTA
4:AGC 11:TAG
5:GCT
6:CTA
7:TAG

B. Delta-coding of coordinate annotations (sets)

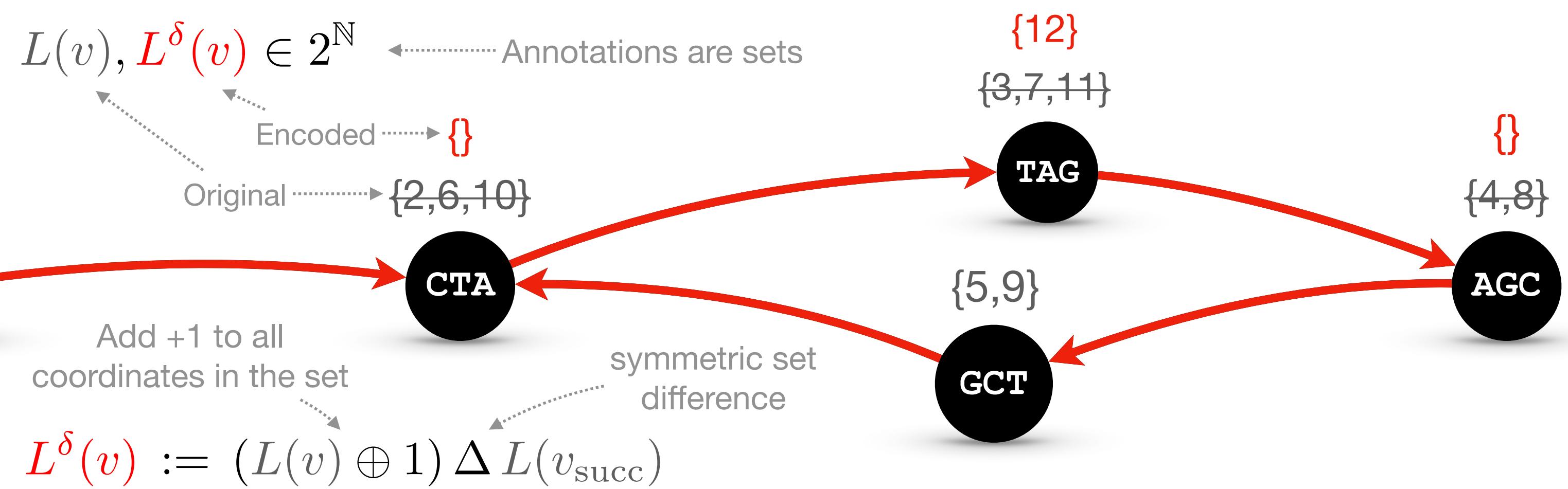


Delta coding for k-mer coordinates

A. Enumeration of k-mers

ACTAGCTAGCTAG
1:ACT 8:AGC
2:CTA 9:GCT
3:TAG 10:CTA
4:AGC 11:TAG
5:GCT
6:CTA
7:TAG

B. Delta-coding of coordinate annotations (sets)

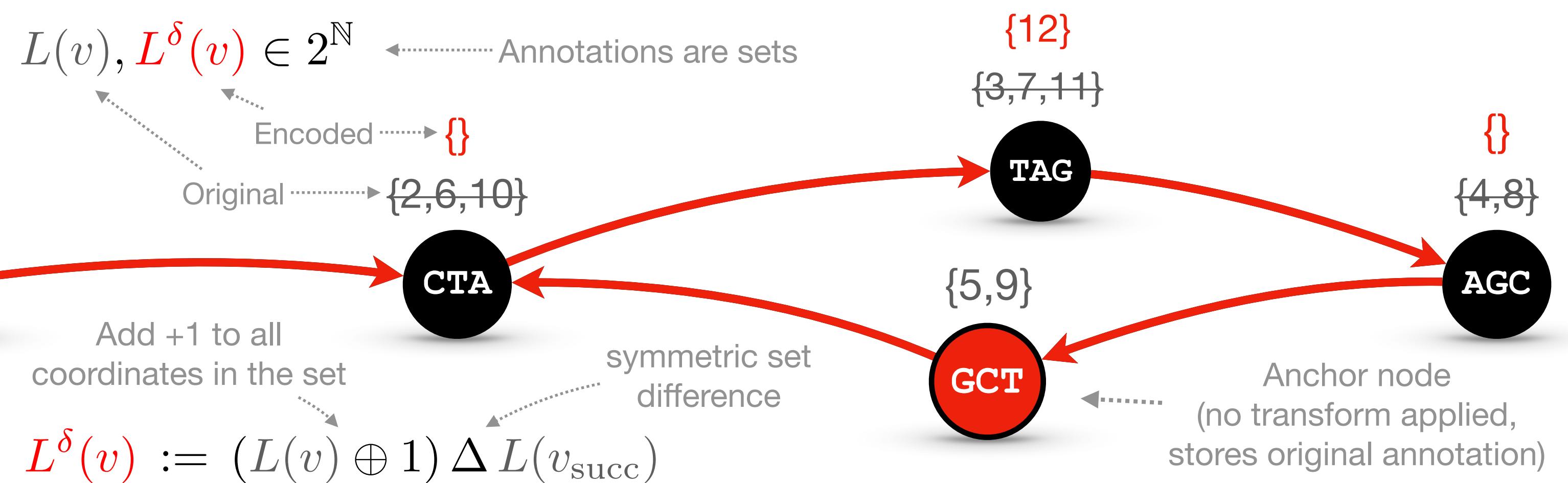


Delta coding for k-mer coordinates

A. Enumeration of k-mers

ACTAGCTAGCTAG
1:ACT 8:AGC
2:CTA 9:GCT
3:TAG 10:CTA
4:AGC 11:TAG
5:GCT
6:CTA
7:TAG

B. Delta-coding of coordinate annotations (sets)

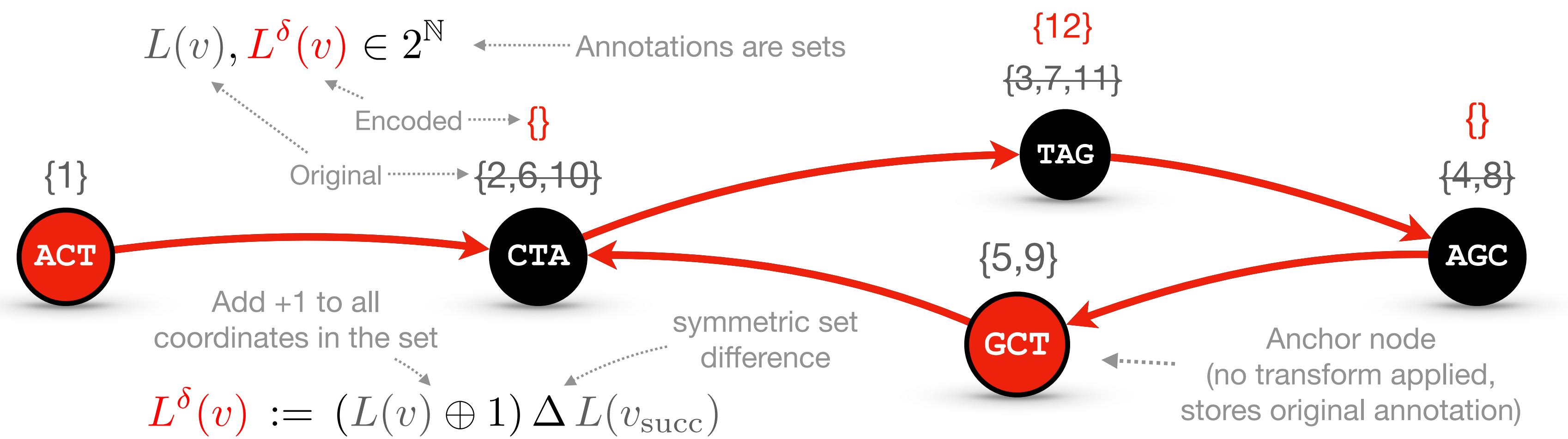


Delta coding for k-mer coordinates

A. Enumeration of k-mers

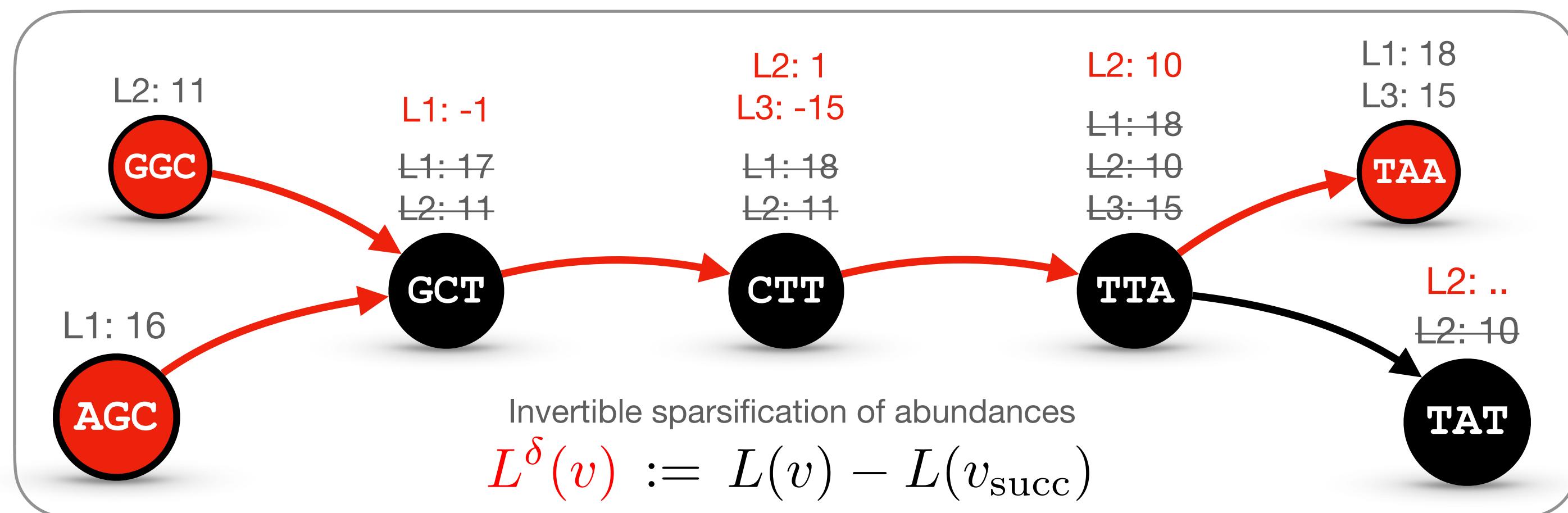
ACTAGCTAGCTAG
 1:ACT 8:AGC
 2:CTA 9:GCT
 3:TAG 10:CTA
 4:AGC 11:TAG
 5:GCT
 6:CTA
 7:TAG

B. Delta-coding of coordinate annotations (sets)

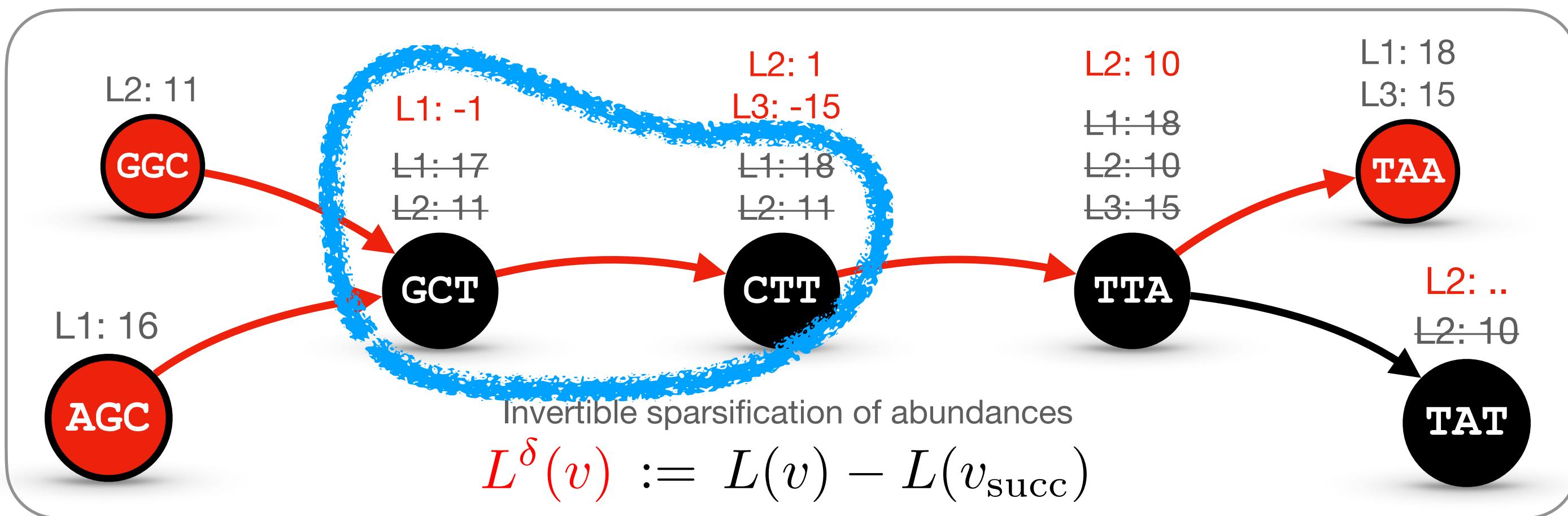


Decoding: $L(v) = (L^\delta(v) \Delta L(v_{succ})) \oplus 1$

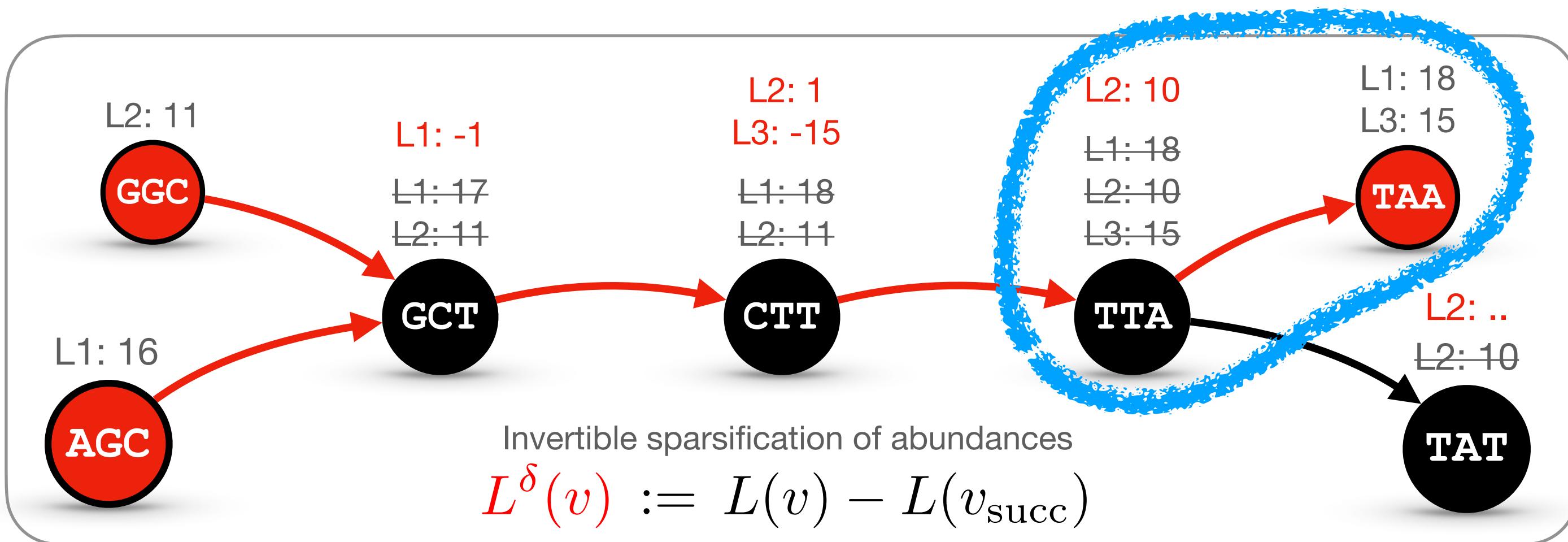
Delta coding for k-mer abundances



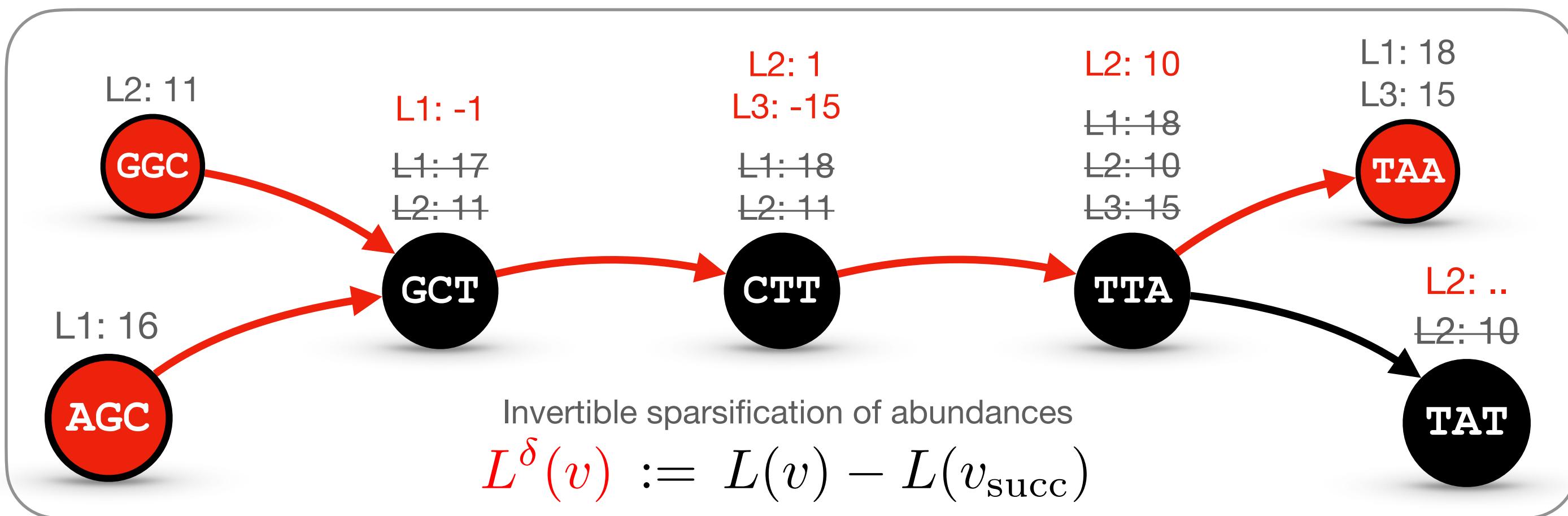
Delta coding for k-mer abundances



Delta coding for k-mer abundances

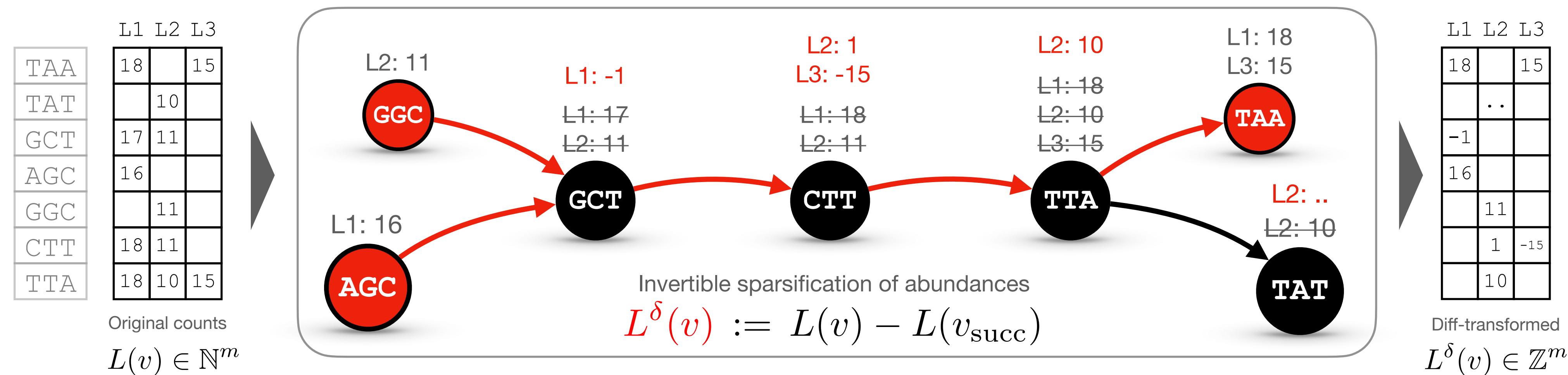


Delta coding for k-mer abundances



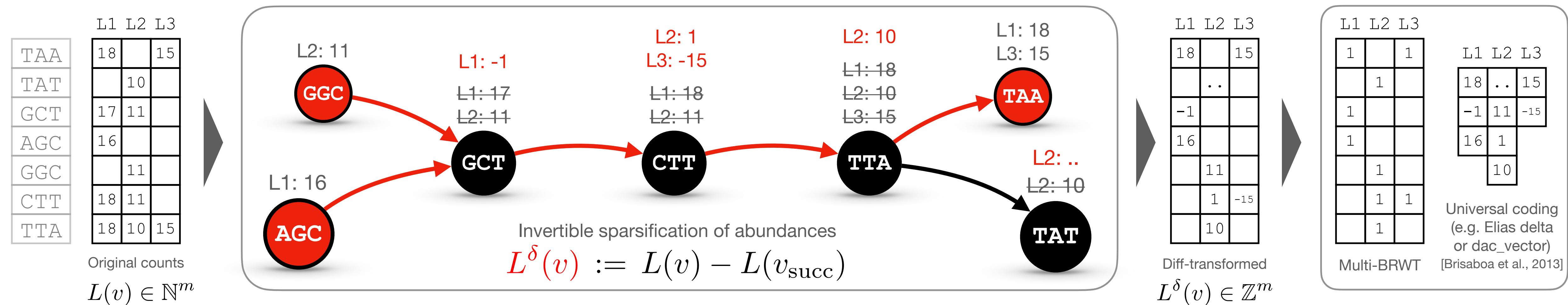
Decoding: $L(v) = L(v_{\text{succ}}) + L^\delta(v)$

Delta coding for k-mer abundances



Decoding: $L(v) = L(v_{\text{succ}}) + L^\delta(v)$

Delta coding for k-mer abundances

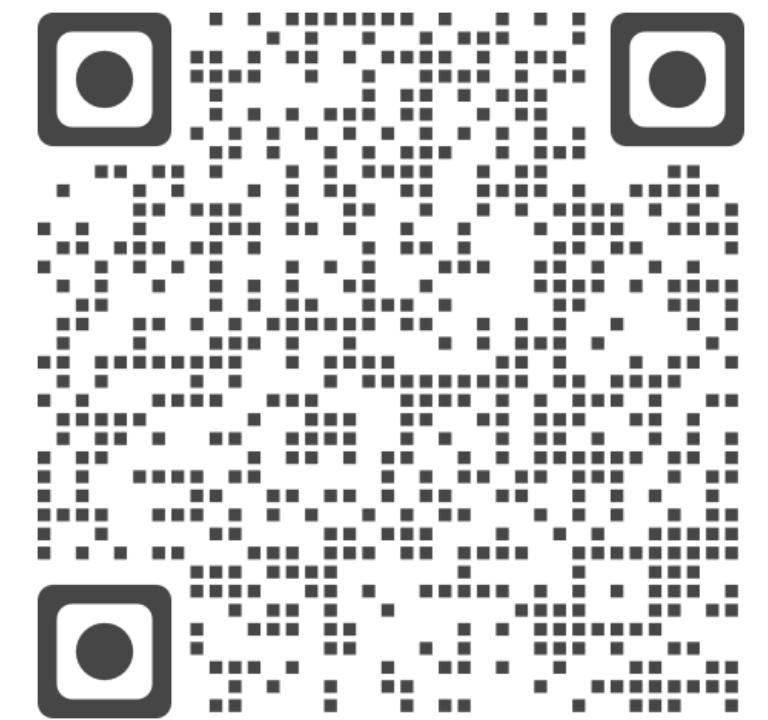


Decoding: $L(v) = L(v_{\text{succ}}) + L^\delta(v)$

Method

Counting DBG: Implementation

Repository: github.com/ratschlab/counting_dbg



Implemented as part of the MetaGraph framework

- succinct graph representations (based on the BOSS table Bowe et al., 2012)
- graph annotation representations (e.g., Multi-BRWT Karasikov et al., 2019)
- hybrid bit vector representations
- procedures for query and alignment

Base succinct data structures from sds1-lite (Succinct Data Structure Library)

- compressed and packed bitmaps
- compressed integer arrays (`sds1::dac_vector_dp<>`)

Experiments

1. Indexing k-mer abundances
2. Indexing k-mer coordinates
 - ▶ **Encoding traces** for lossless indexing of sequences
 - ▶ Trace Consistent Graph Alignment (**TCG-Aligner**)

Indexing k-mer abundances

Data: 2,586 Illumina RNA-Seq read sets*

On querying 100 random human transcripts (\approx 90 kbp in total)



(excluding index loading)

Method	Index size			Peak RAM during query			Query time		
	binary	smooth counts	raw counts	binary	smooth counts	raw counts	binary	smooth counts	raw counts
Mantis-MST	24.9 GB	-	-	25.1 GB	-	-	0.6 s	-	-
RowDiff	7.7 GB	-	-	8.0 GB	-	-	8.6 s	-	-
REINDEER	30.3 GB	59 GB	-	58.9 GB	91 GB	-	53.1 s	56.5 s	-
This work	6.6 GB	11 GB	21 GB	6.9 GB	11 GB	21 GB	6.4 s	17.6 s	21.2 s

* read sets from [Solomon and Kingsford, 2018]

Indexing k-mer abundances

Data: 2,586 Illumina RNA-Seq read sets*

On querying 100 random human transcripts (\approx 90 kbp in total)



(excluding index loading)

Method	Index size			Peak RAM during query			Query time		
	binary	smooth counts	raw counts	binary	smooth counts	raw counts	binary	smooth counts	raw counts
Mantis-MST	24.9 GB	-	-	25.1 GB	-	-	0.6 s	-	-
RowDiff	7.7 GB	-	-	8.0 GB	-	-	8.6 s	-	-
REINDEER	30.3 GB	59 GB	-	58.9 GB	91 GB	-	53.1 s	56.5 s	-
This work	6.6 GB	11 GB	21 GB	6.9 GB	11 GB	21 GB	6.4 s	17.6 s	21.2 s

- Generates **5x smaller** representations and uses **8x less RAM**

* read sets from [Solomon and Kingsford, 2018]

Indexing k-mer abundances

Data: 2,586 Illumina RNA-Seq read sets*

On querying 100 random human transcripts (≈ 90 kbp in total)



(excluding index loading)

Method	Index size			Peak RAM during query			Query time		
	binary	smooth counts	raw counts	binary	smooth counts	raw counts	binary	smooth counts	raw counts
Mantis-MST	24.9 GB	-	-	25.1 GB	-	-	0.6 s	-	-
RowDiff	7.7 GB	-	-	8.0 GB	-	-	8.6 s	-	-
REINDEER	30.3 GB	59 GB	-	58.9 GB	91 GB	-	53.1 s	56.5 s	-
This work	6.6 GB	11 GB	21 GB	6.9 GB	11 GB	21 GB	6.4 s	17.6 s	21.2 s

- Generates **5x smaller** representations and uses **8x less RAM**
- **3-5x faster** to query

* read sets from [Solomon and Kingsford, 2018]

Experiments

1. Indexing k-mer abundances
2. Indexing k-mer coordinates
 - ▶ **Encoding traces** for lossless indexing of sequences
 - ▶ Trace Consistent Graph Alignment (**TCG-Aligner**)

Trace Consistent Graph (TCG-) Aligner

Sequence alignment on top of Counting DBG with k-mer coordinates with the **seed-chain-extend** approach:

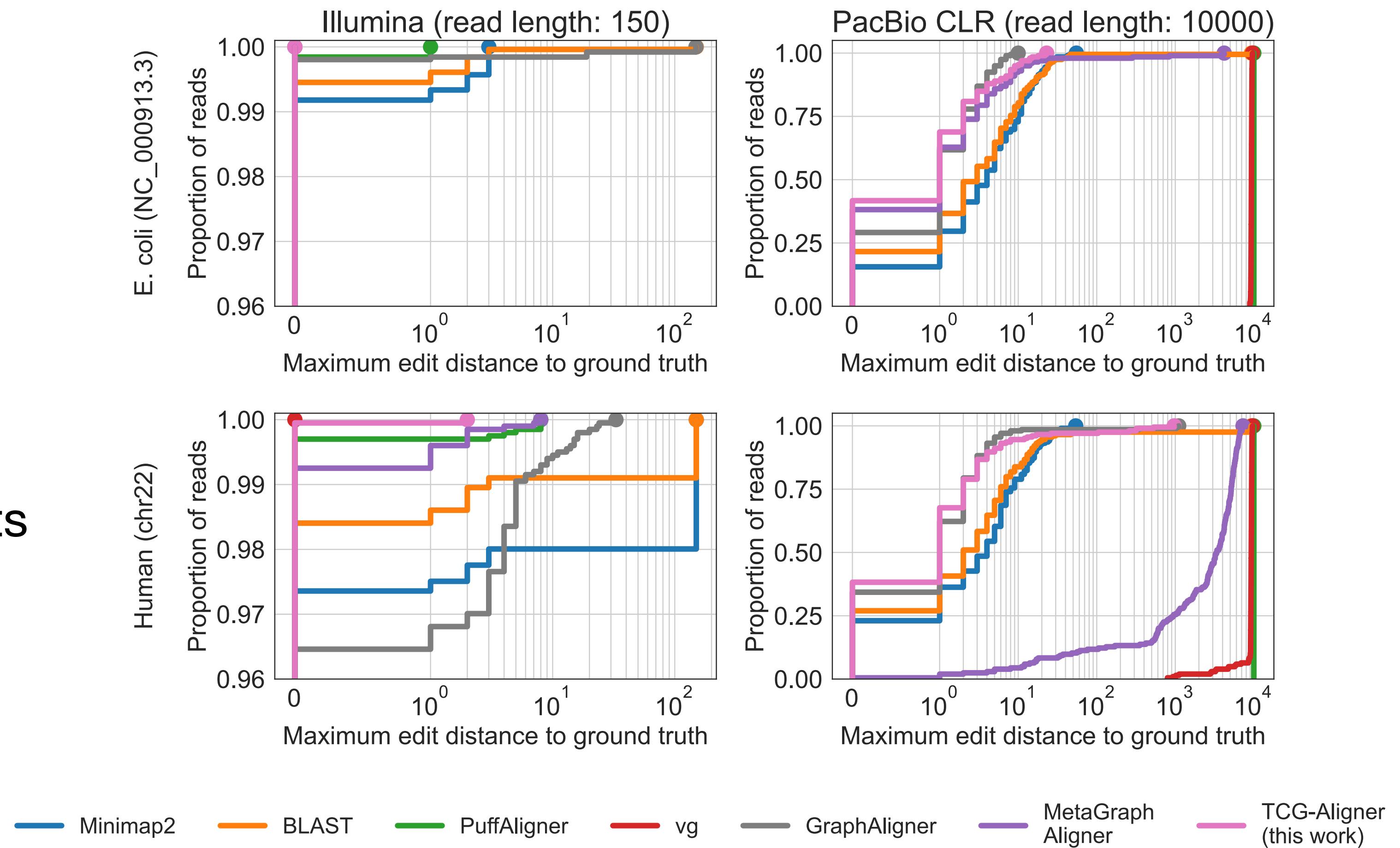
1. Find seeds of size k or less
2. Chaining (inspired by Minimap2 [Li, 2018])
 - DP table for pairs (seed, coordinate) sorted by coordinates
 - Dynamic Programming + Backtracking
3. Extension algorithm
 - generalization of Needleman-Wunsch algorithm
 - similar to the MetaGraph aligner [Karasikov, Mustafa et al., 2020]



Alignment accuracy

Evaluation approach

1. Index a reference genome
2. Simulate reads from the same reference
3. Align reads and measure the distance between the alignments and their generating references



Alignment accuracy for **Counting DBG** and **state-of-the-art aligners** on simulated Illumina- and PacBio-type reads (E. coli NC 000913.3 and human chr22). The edit distance is measured between the alignment (the returned path in the graph) and the ground truth sequence. In the top left subplot, the curves of vg- and TCG-Aligner are superimposed.

Lossless indexing of RefSeq

RefSeq (33M accessions, 1.7 Tbp, 483 GB)

	BLAST	MegaBLAST	This work
Index size	437 GB (2.05 bits/bp)	2,358 GB (11.07 bits/bp)	509 GB (2.39 bits/bp)
Align 1 read (*)	353 sec, 417 GB	12.5 sec, 0.090 GB	0.66 sec, 500 GB
Align 1,000 reads	1,857 sec, 428 GB	1,542 sec, 22.0 GB	575 sec, 513 GB

The alignment speed was measured on reads taken from a metagenomic sequencing sample SRR10002688_1.

(*) For aligning single reads, the experiment is independently performed for the 100 first reads and the average time and RAM usage are presented.

Lossless indexing of RefSeq

RefSeq (33M accessions, 1.7 Tbp, 483 GB)

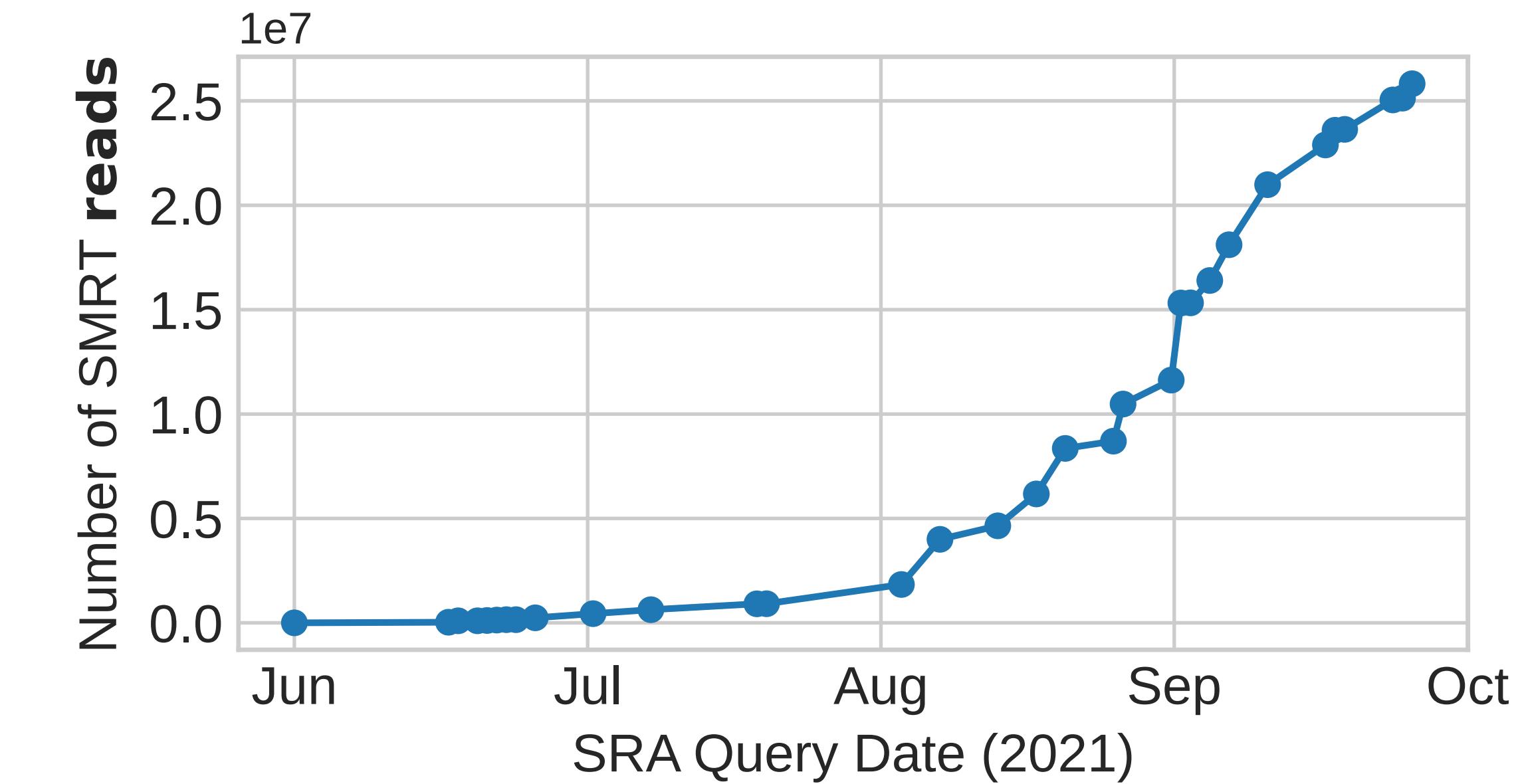
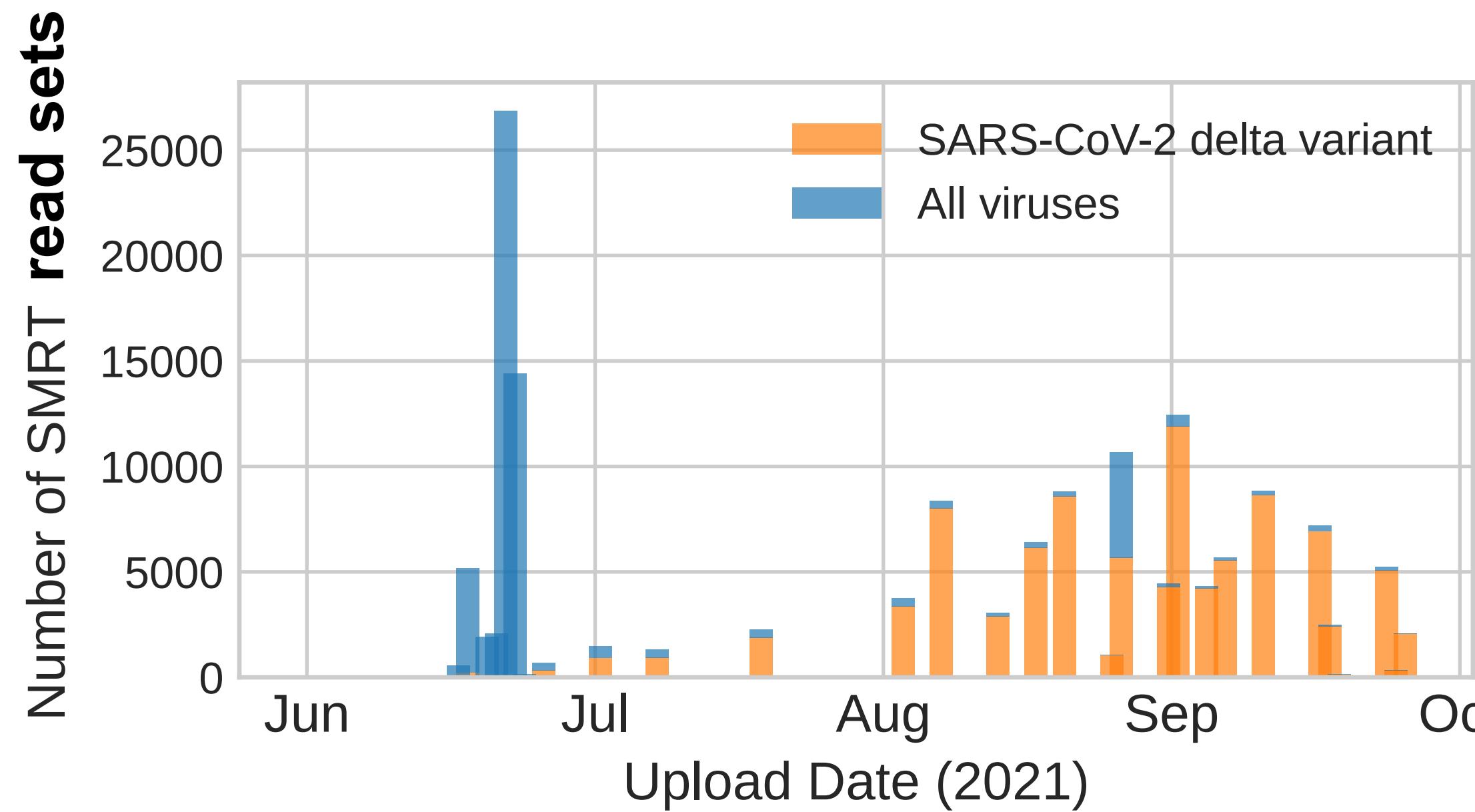
	BLAST	MegaBLAST	This work
Index size	437 GB (2.05 bits/bp)	2,358 GB (11.07 bits/bp)	509 GB (2.39 bits/bp)
Align 1 read (*)	353 sec, 417 GB	12.5 sec, 0.090 GB	0.66 sec, 500 GB
Align 1,000 reads	1,857 sec, 428 GB	1,542 sec, 22.0 GB	575 sec, 513 GB

The alignment speed was measured on reads taken from a metagenomic sequencing sample SRR10002688_1.

(*) For aligning single reads, the experiment is independently performed for the 100 first reads and the average time and RAM usage are presented.

Query Delta variant of SARS-CoV-2

1. Constructed a joint index of **all 152,884 viral PacBio SMRT read sets from SRA**
2. Assembled a list of 9 defining mutations of the SARS-CoV-2 21A (Delta) variant spike protein
3. Retrieve all the occurrences of these specific mutations within the reads (took under 4 min)



Coordinates allow retrieving not only the number of read sets (**left**) but also single reads (**right**)

Conclusion



Big Data
National Research Programme

Counting de Bruijn Graphs* generalize *Annotated de Bruijn graphs*

- ▶ Allow **efficiently encoding non-binary annotations**, e.g.:
 - k-mer abundances
 - k-mer coordinates
- ▶ **Scales to very large graphs**
 - constructs in linear time and constant memory
- ▶ Possible applications:
 - indexing and **querying k-mer abundances** (gene expression levels)
 - can be used as a **backend for aligners** (such as BLAST)
 - alignment with different error models (using k-mer abundances)

* Recently published in journal *Genome Research*

Team



Mikhail Karasikov



Marc Zimmermann



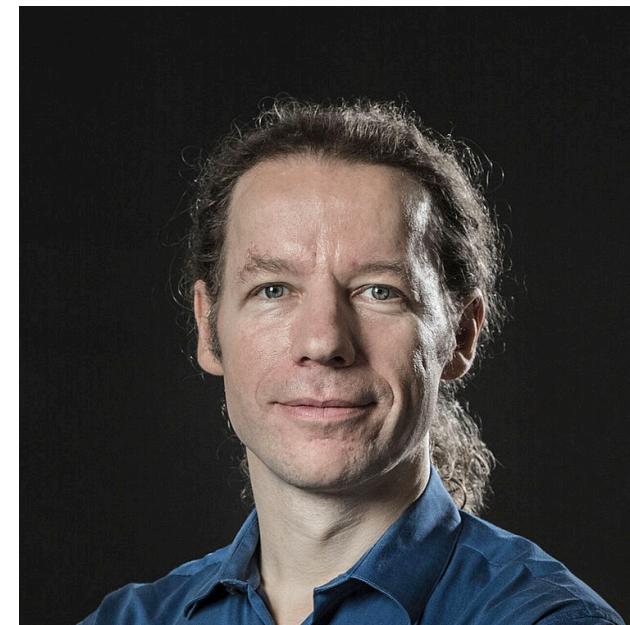
Andre Kahles



Harun Mustafa



Daniel Danciu



Gunnar Rätsch

NRP75 Team



Torsten Hoefler



Mario Stanke



Matthias Ebel



Giovanna Migliorelli

Further contributors

- Thomas Zhou
- Marek Kokot
- Chris Barber
- Radu Muntean
- Jan Studený
- Sara Javadzadeh-No
- Predrag Krnetic

Thanks for your attention!



Big Data
National Research Programme

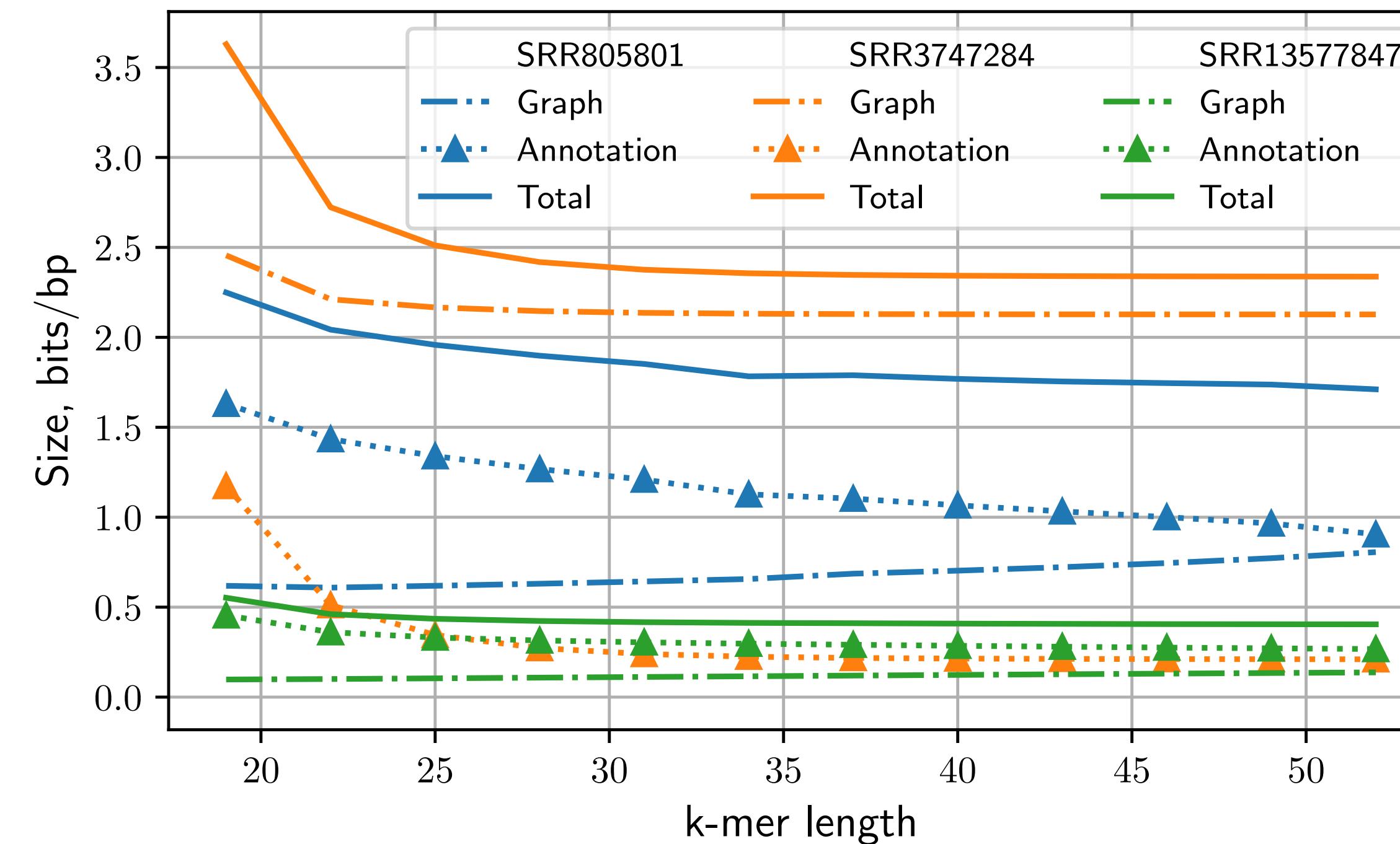


ETH zürich

Backup slides

Lossless indexing of k-mer coordinates

Dependence on k-mer length



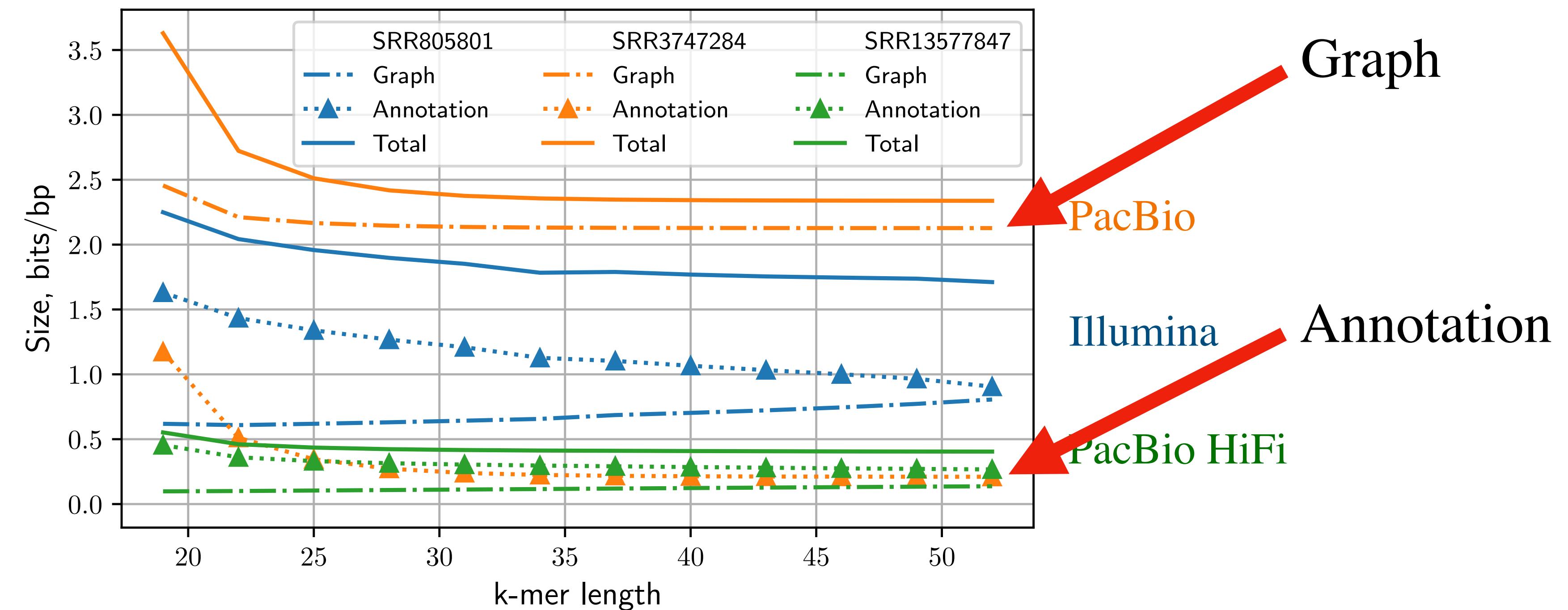
PacBio

Illumina

PacBio HiFi

Lossless indexing of k-mer coordinates

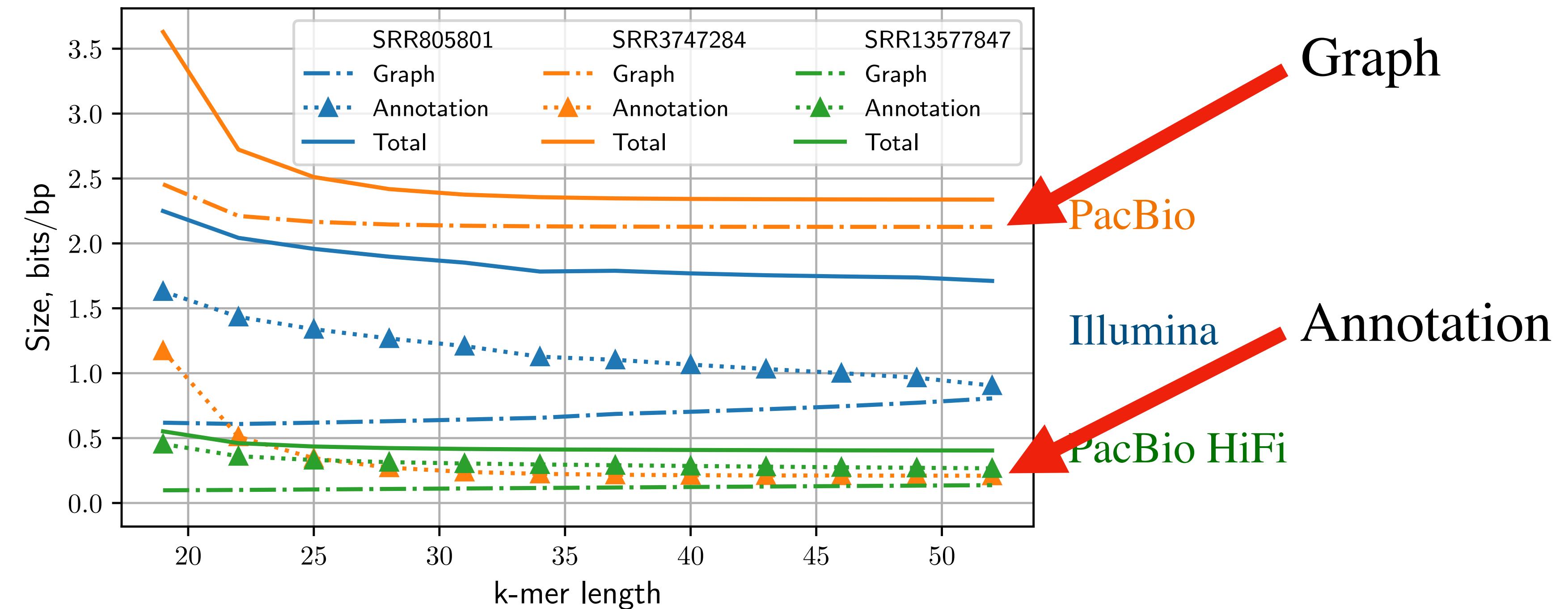
Dependence on k-mer length



1. For low-error reads (HiFi), annotation is tiny compared to graph

Lossless indexing of k-mer coordinates

Dependence on k-mer length



1. For low-error reads (HiFi), annotation is tiny compared to graph
2. Index size does not increase for larger values of k even for short Illumina reads

Construction time

Method	Construction time			Max. RAM		
	Binary	Smooth counts	Raw counts	Binary	Smooth counts	Raw counts
k-mer spectrum estimation with ntCard [†]	2.0 h	-	-	19 GB	-	-
k-mer counting with Squeakr [†]	35.4 h	-	-	333 GB	-	-
k-mer counting with Squeakr (single thread) ^{†*}	19.3 h	-	-	483 GB	-	-
Graph and annotation construction with Mantis*	9.6 h	-	-	42.7 GB	-	-
Annotation transform to MST	0.4 h	-	-	26.6 GB	-	-
Mantis-MST (all steps)	29.3 h	-	-	483 GB	-	-
k-mer counting with KMC [†] (with flag -m2)	3.2 h	-	-	41 GB	-	-
Assembling contigs from k-mers with MetaGraph [†]	0.7 h	-	-	51 GB	-	-
Graph construction and annotation with MetaGraph	1.7 h	-	-	52 GB	-	-
Annotation transform to RowDiff-MultiBRWT	1.3 h	-	-	58 GB	-	-
RowDiff (all steps)	6.9 h	-	-	58 GB	-	-
Assembling unitigs with BCALM [†] (with -max-memory 1000)	40.6 h	40.6 h	-	152 GB	152 GB	-
REINDEER (indexing)*	8.7 h	10.6 h	-	68.8 GB	59.4 GB	-
REINDEER (all steps)	49.3 h	51.2 h	-	152 GB	152 GB	-
k-mer counting with KMC [†] (with flag -m2)	3.2 h	3.2 h	3.2 h	41 GB	41 GB	41 GB
Assembling contigs from k-mers with MetaGraph [†]	0.7 h	0.9 h	0.8 h	51 GB	70 GB	70 GB
Graph construction and annotation with MetaGraph	1.6 h	2.2 h	2.3 h	52 GB	52 GB	52 GB
Annotation transformation (this work)	1.2 h	1.2 h	1.4 h	59 GB	88 GB	88 GB
This work (all steps)	6.7 h	7.5 h	7.7 h	59 GB	88 GB	88 GB

Alignment performance

Supplemental Table 2. Percentage of simulated reads mapping exactly to their respective ground-truth sequences (top) and total query time in seconds (bottom). For each reference, 2000 Illumina reads were simulated using ART and 200 PacBio CLR reads were simulated using pbsim, respectively. PuffAligner was unable to align the PacBio-type reads.

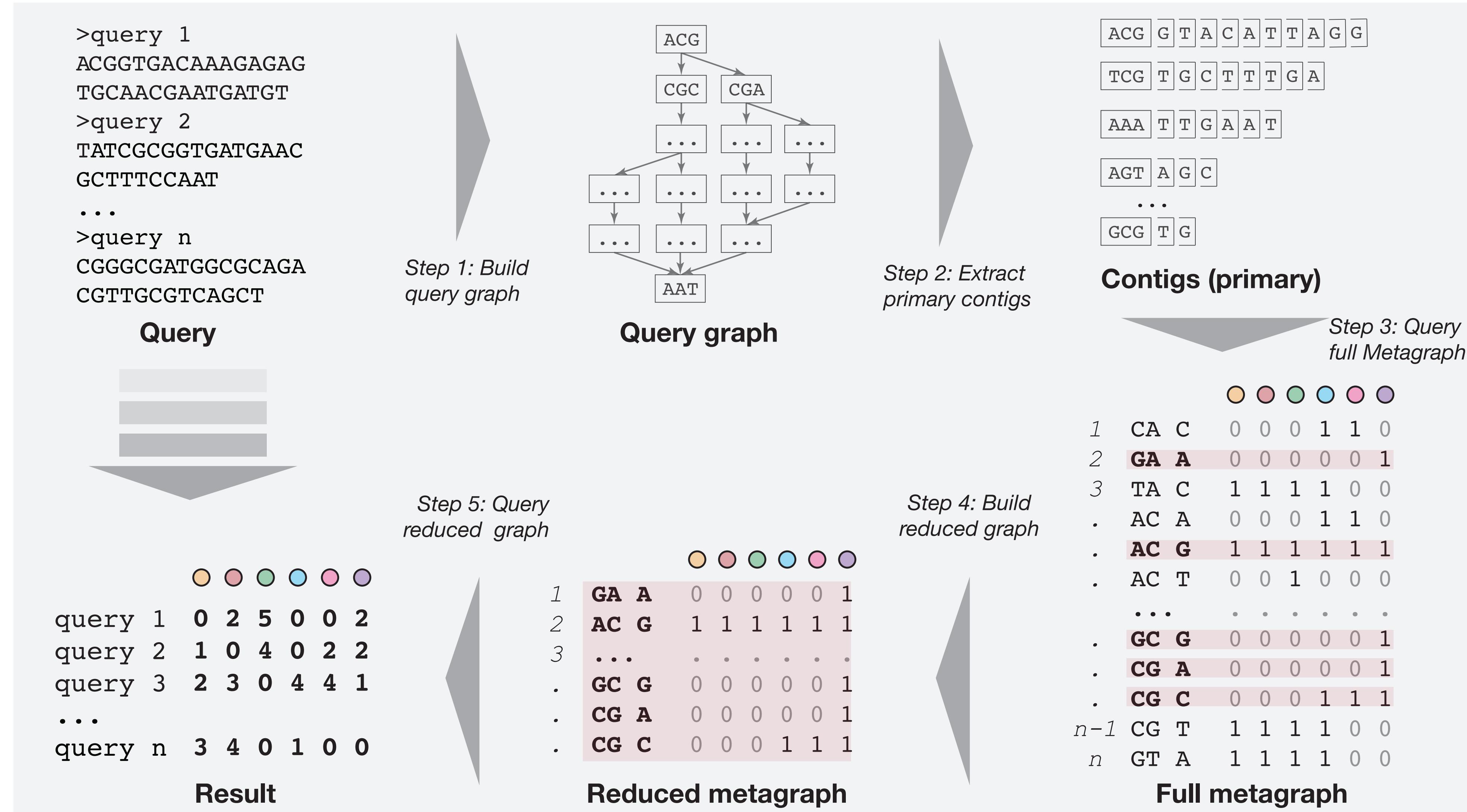
Reference	Minimap2	BLAST	PuffAligner	vg	GraphAligner	MetaGraph-Aligner	This work
Illumina: E. coli	99.18%	99.45%	99.84%	100.00%	99.8%		100.00%
Illumina: chr22	97.36%	98.4%	99.70%	100.00%	96.46%		99.25%
PacBio: E. coli	15.58%	21.61%	N/A	0.00%	29.15%		38.69%
PacBio: chr22	23.04%	26.96%	N/A	0.00%	34.31%		0.49%
Illumina: E. coli	0.13 s	0.39 s	0.03 s	1.16 s	3.22 s		1.13 s
Illumina: chr22	0.47 s	26.69 s	0.96 s	5.67 s	42.93 s		6.67 s
PacBio: E. coli	1.11 s	1.51 s	4.31 s	443.23 s	7.39 s		40.20 s
PacBio: chr22	2.43 s	34.80 s	4.91 s	1059.05 s	48.99 s		37.76 s

Data

Nine defining Delta gene variants of the SARS-CoV-2 spike protein

```
>OK091006.1:21536-25357:27-85
ACTAGTCTAGTCAGTGTGTTAATCTTAGAACCACTCAATTACCCCTGCATACA
>OK091006.1:21536-25357:444-502
CAACAAAAGTTGGATGGAAAGTGGAGTTATTCTAGTGCATAATTGCACTT
>OK091006.1:21536-25357:1326-1384
TTCTAACAGGTTGGTGGTAATTATAATTACCGGTATAGATTGTTAGGAAGTCTAATCTCA
>OK091006.1:21536-25357:1404-1462
TTCAACTGAAATCTATCAGGCCGGTAGCAAACCTTGTAAATGGTGTGAAGGTTTAATT
>OK091006.1:21536-25357:1812-1870
TTCTAACCAAGGTTGCTGTTCTTATCAGGGTGTAACTGCACAGAAGTCCCTGTTGCTA
>OK091006.1:21536-25357:2013-2071
CGCTAGTTATCAGACTCAGACTAATTCTCGTCGGCGGGCACGTAGTAGCTAGTCAT
>OK091006.1:21536-25357:2820-2878
CACAGCAAGTGCACTTGGAAAATTCAAAATGTGGTCAACCAAAATGCACAGCTTAA
```

Batch query



Background

Graph annotation representations

	L1	L2	L3
TAA	1		1
TAT		1	
GCT	1	1	
AGC	1		
GGC		1	
CTT	1	1	
TTA	1	1	1

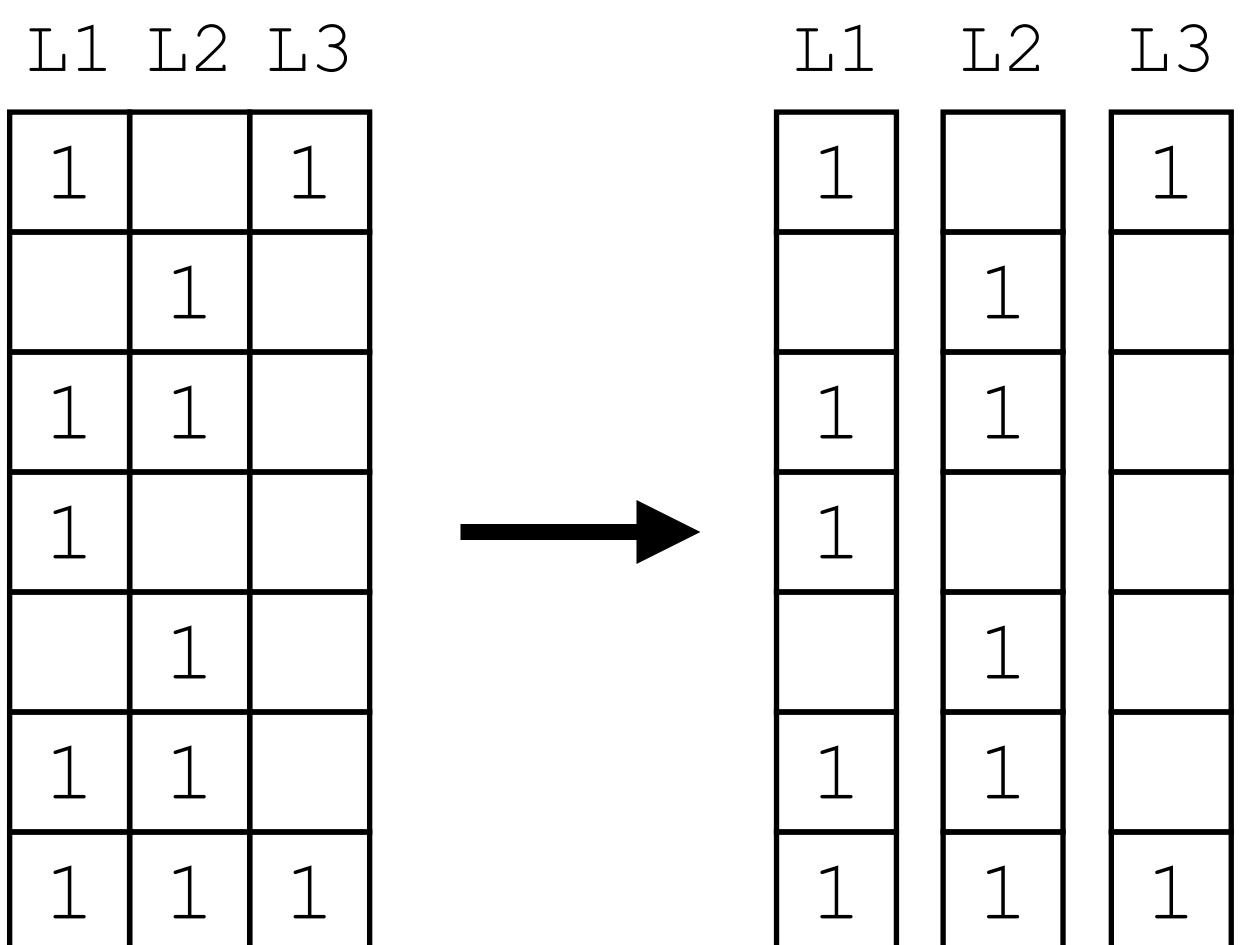
$\sim 10^{11}$

$\sim 10^6$

Background

Graph annotation representations

1. Column-major sparse representation



	L1	L2	L3
TAA	1		1
TAT		1	
GCT	1	1	
AGC	1		
GGC		1	
CTT	1	1	
TTA	1	1	1

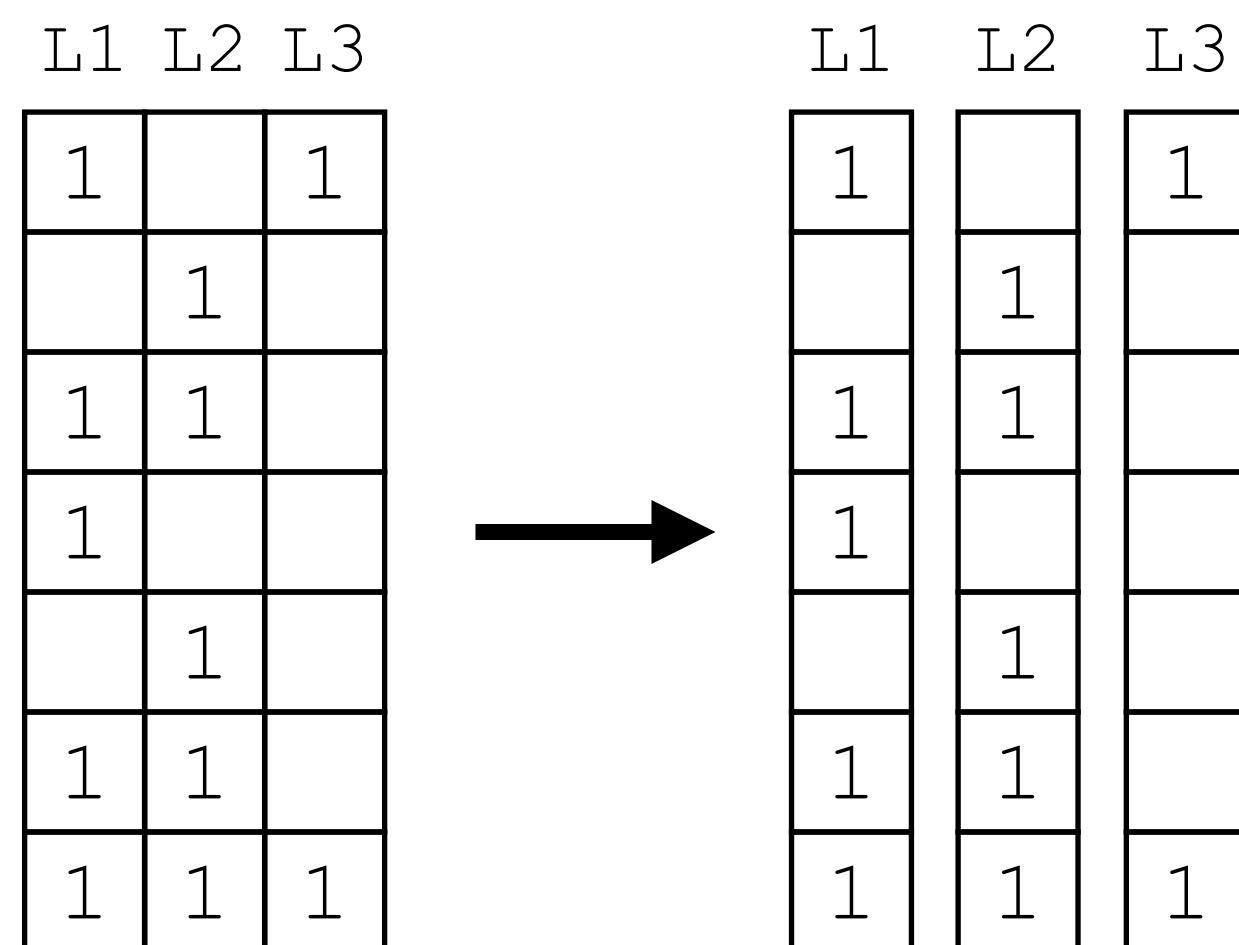
$\sim 10^{11}$

$\sim 10^6$

Background

Graph annotation representations

1. Column-major sparse representation



	L1	L2	L3
TAA	1		1
TAT		1	
GCT	1	1	
AGC	1		
GGC		1	
CTT	1	1	
TTA	1	1	1

$\sim 10^{11}$

$\sim 10^6$

Columns are stored independently as compressed bitmaps
(e.g. sd_vector [Okanohara et al., 2007])

Background

Graph annotation representations

1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]

1	1	1		
2			1	
3				1
4			1	1
5			1	
6	1			
7		1	1	

Background

Graph annotation representations

1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]

1	1	1		
2				
3			1	
4			1	1
5			1	
6	1			
7		1	1	

Background

Graph annotation representations

1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]

1	1	1					1
2							0
3						1	1
4				1		1	1
5					1		1
6	1						1
7		1	1				1

Background

Graph annotation representations

1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]

1	1	1					1
2							0
3						1	1
4				1		1	1
5					1		1
6	1						1
7		1	1				1

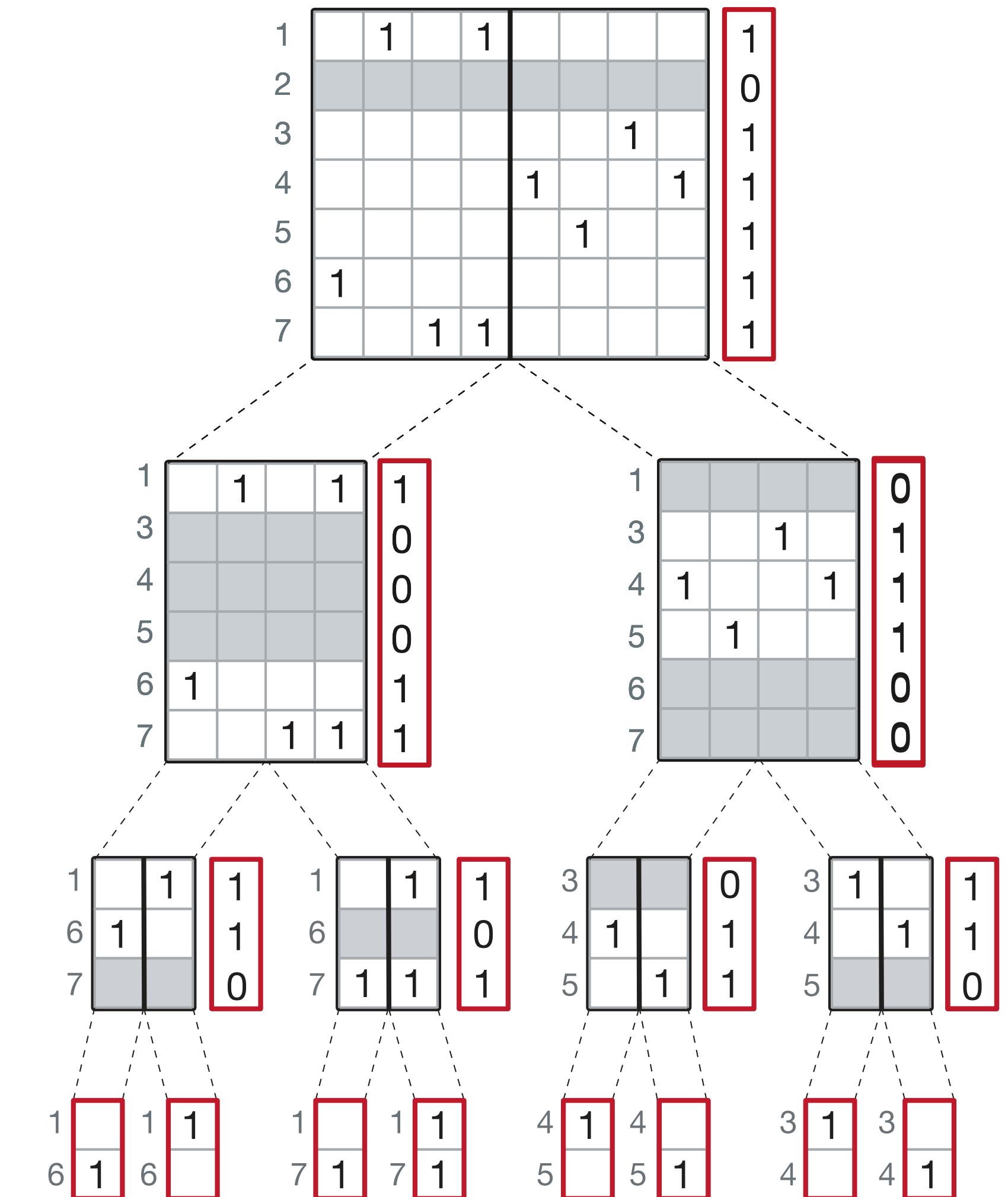
1	1	1				
3						
4						
5						
6	1					
7		1	1			

1						
3						
4						
5						
6	1					
7		1	1			

Background

Graph annotation representations

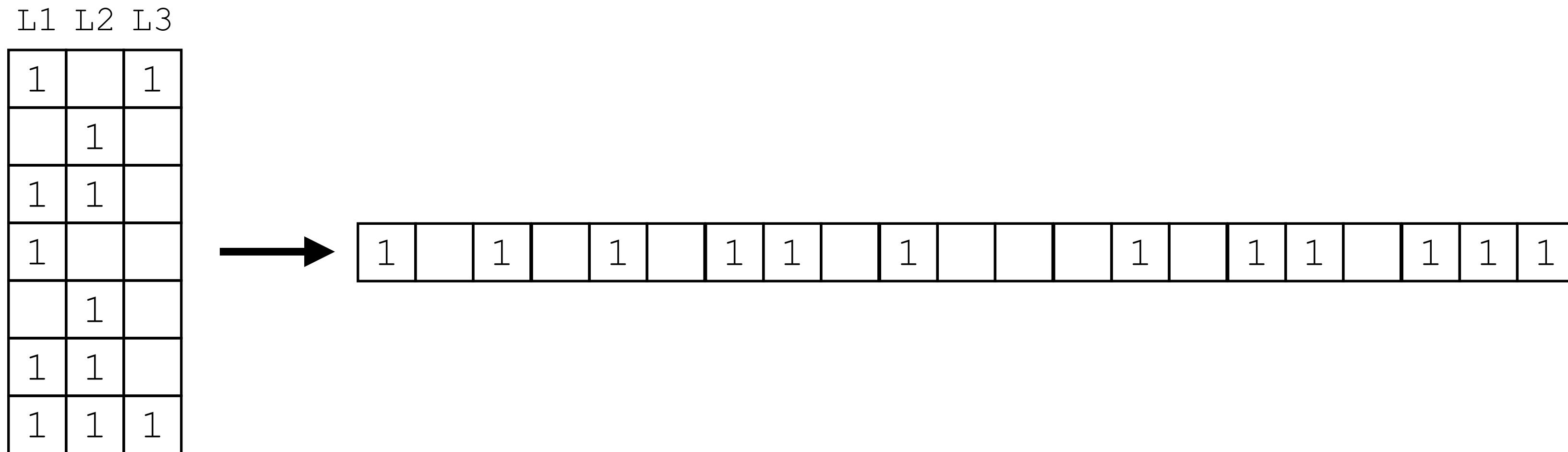
1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]



Background

Graph annotation representations

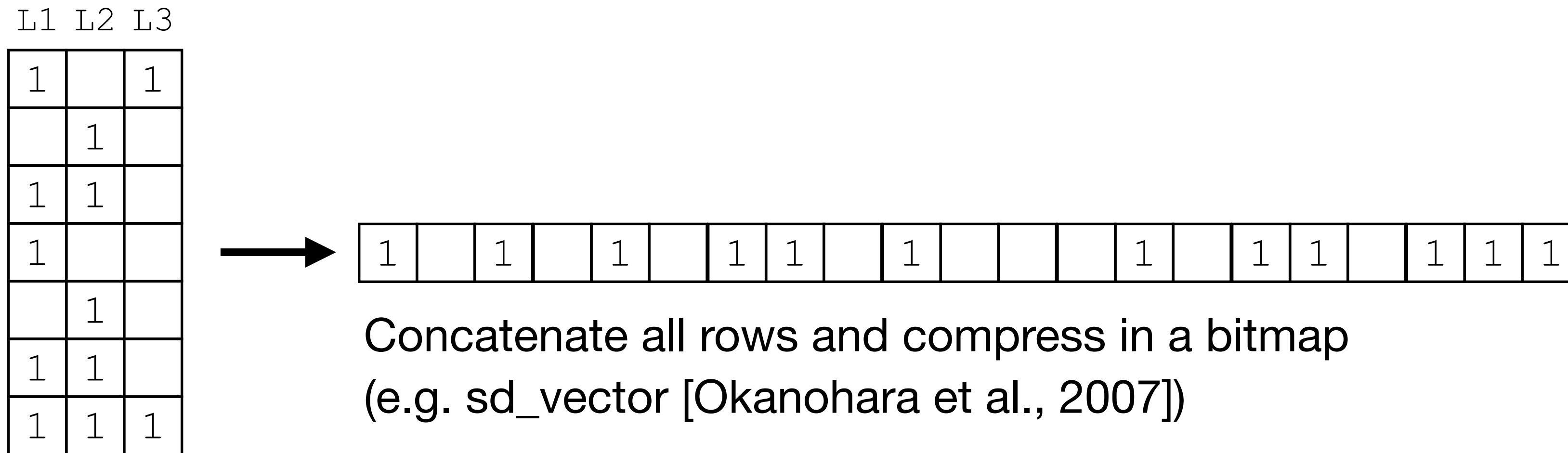
1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])



Background

Graph annotation representations

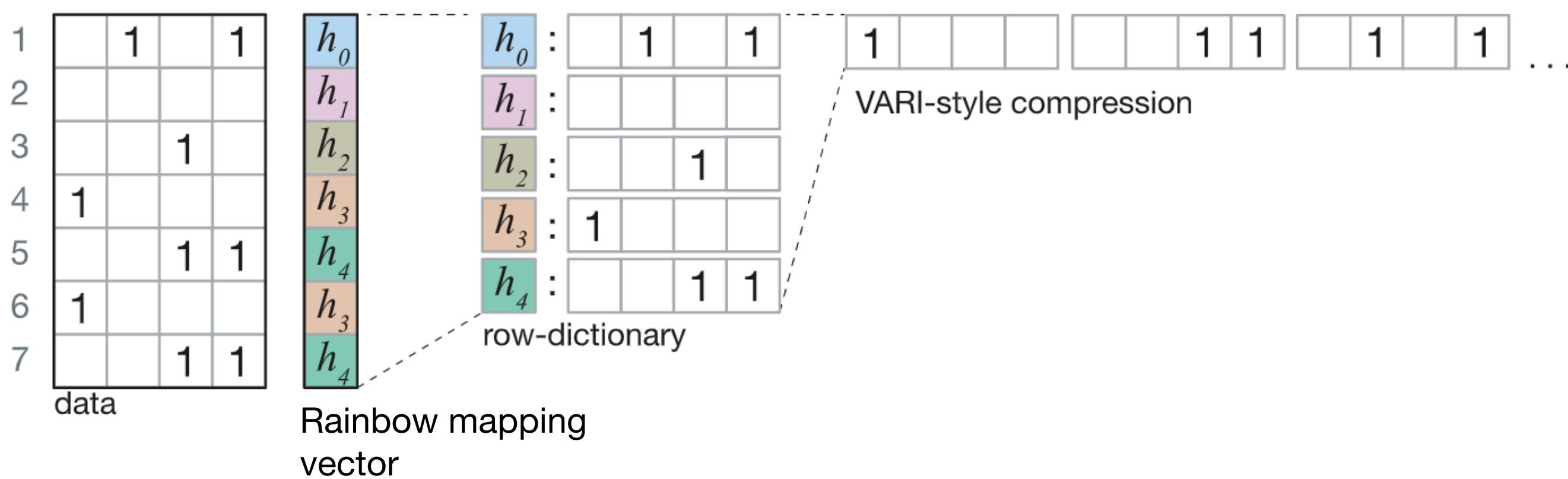
1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])



Background

Graph annotation representations

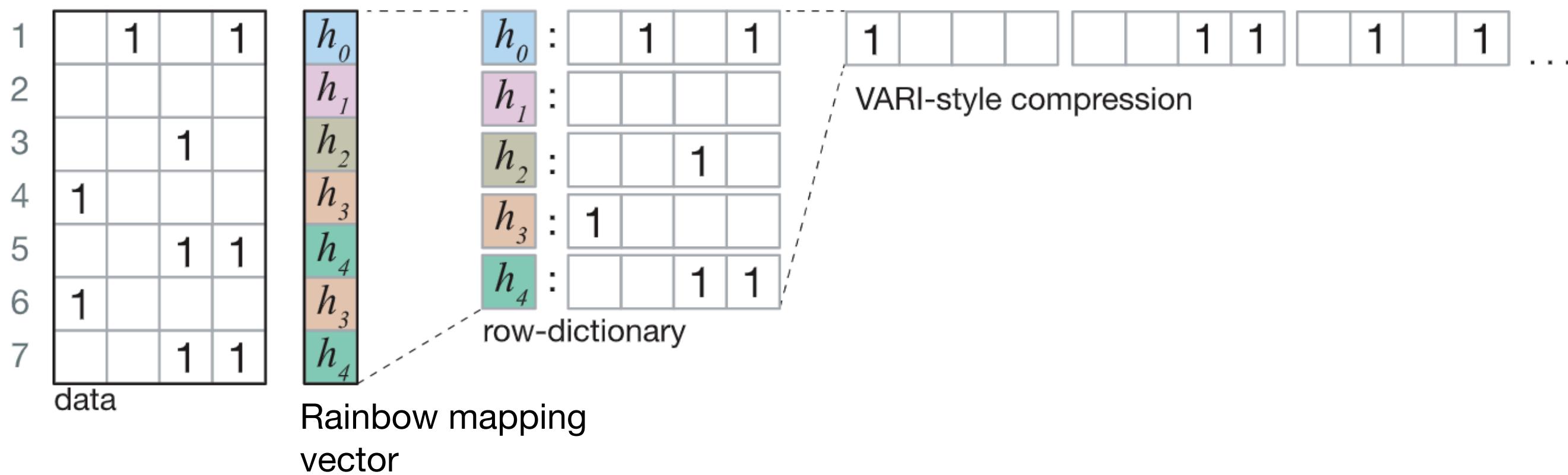
1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]



Background

Graph annotation representations

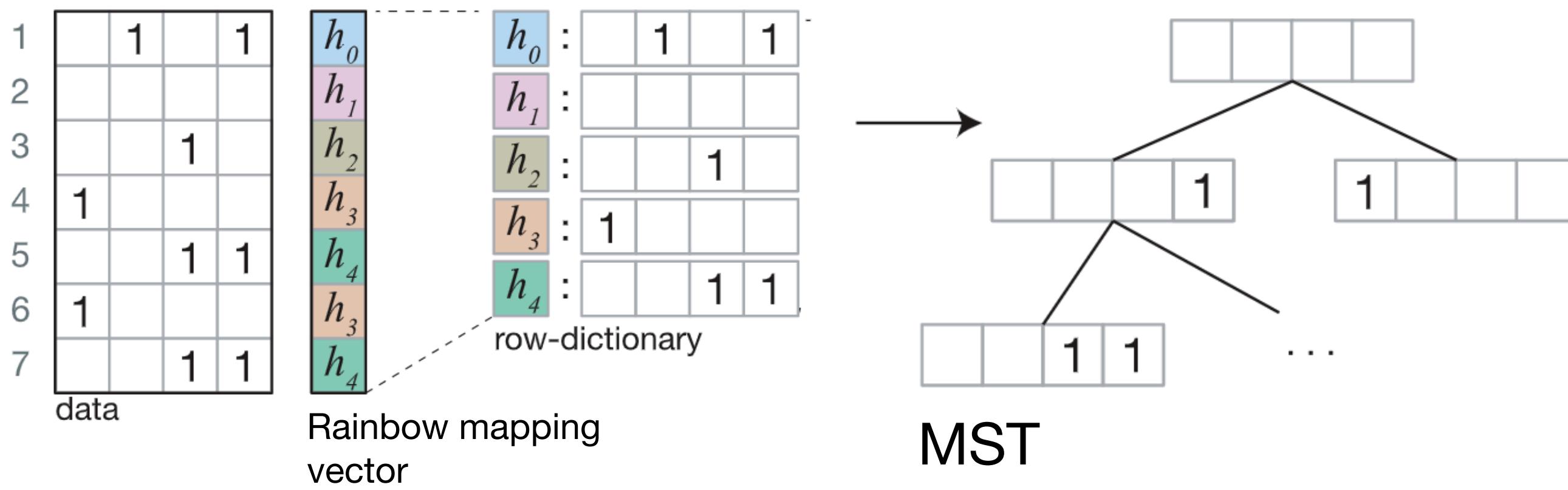
1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]



Background

Graph annotation representations

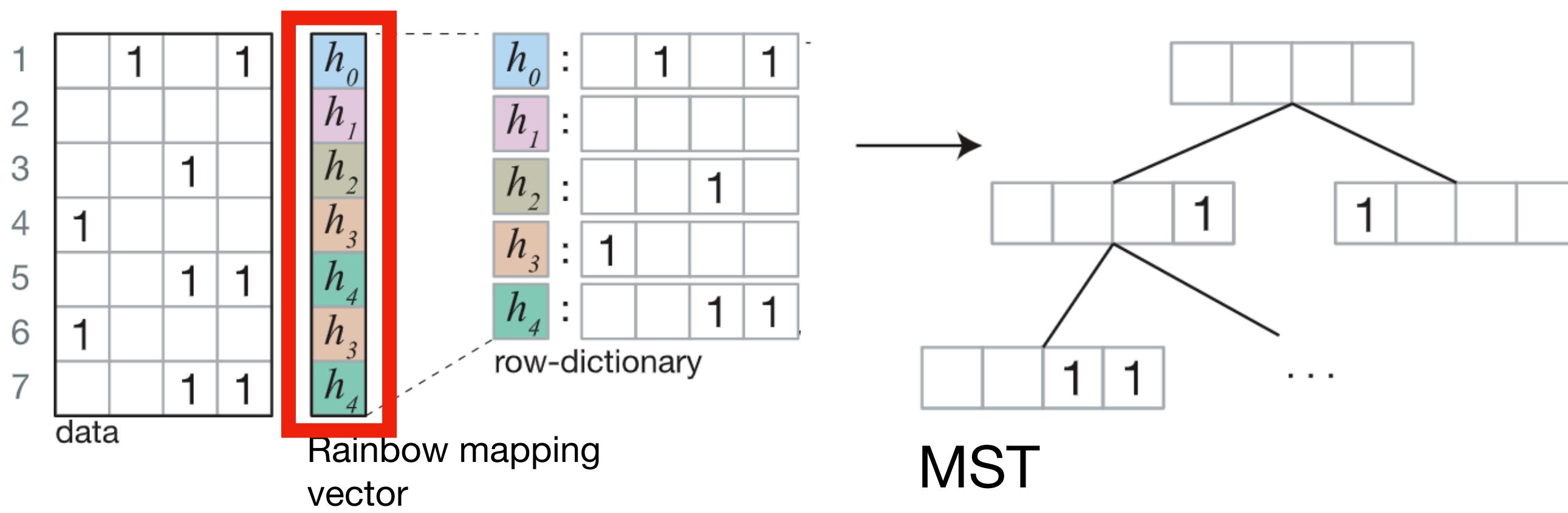
1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]



Background

Graph annotation representations

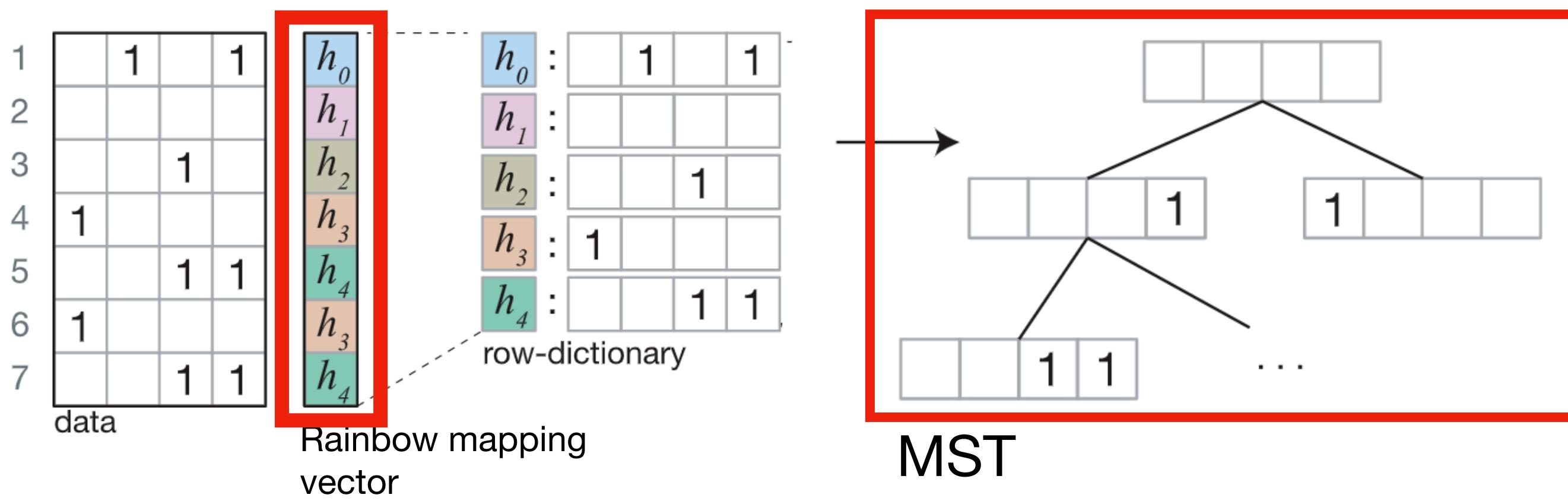
1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]



Background

Graph annotation representations

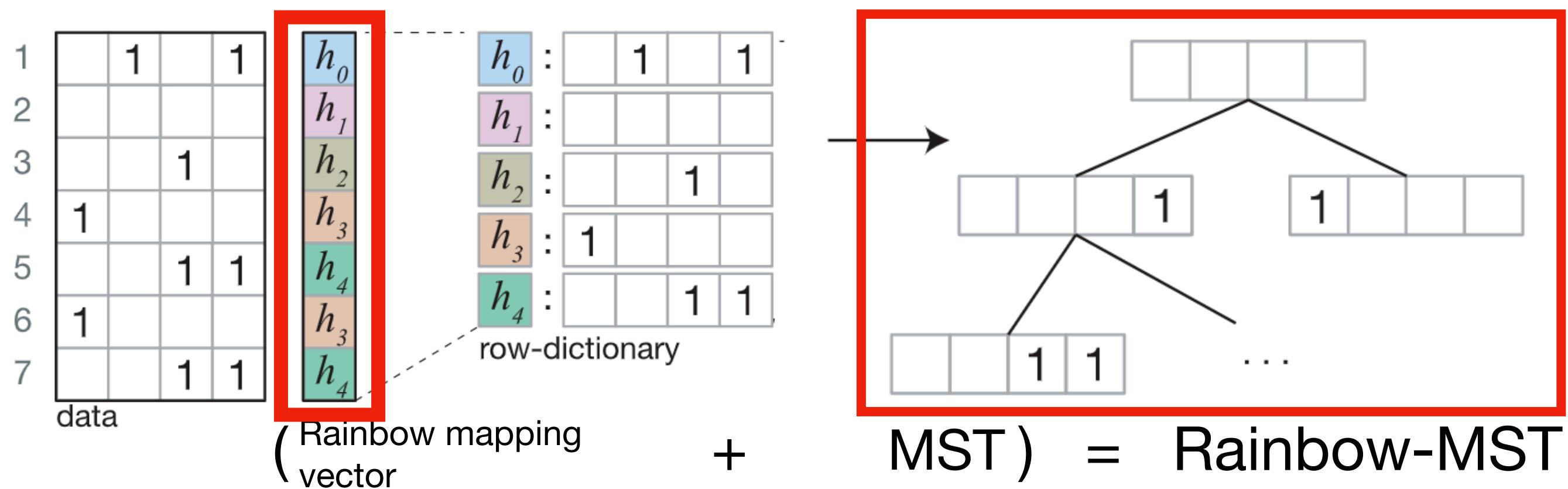
1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]



Background

Graph annotation representations

1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]



Background

Graph annotation representations

1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]

Background

Graph annotation representations

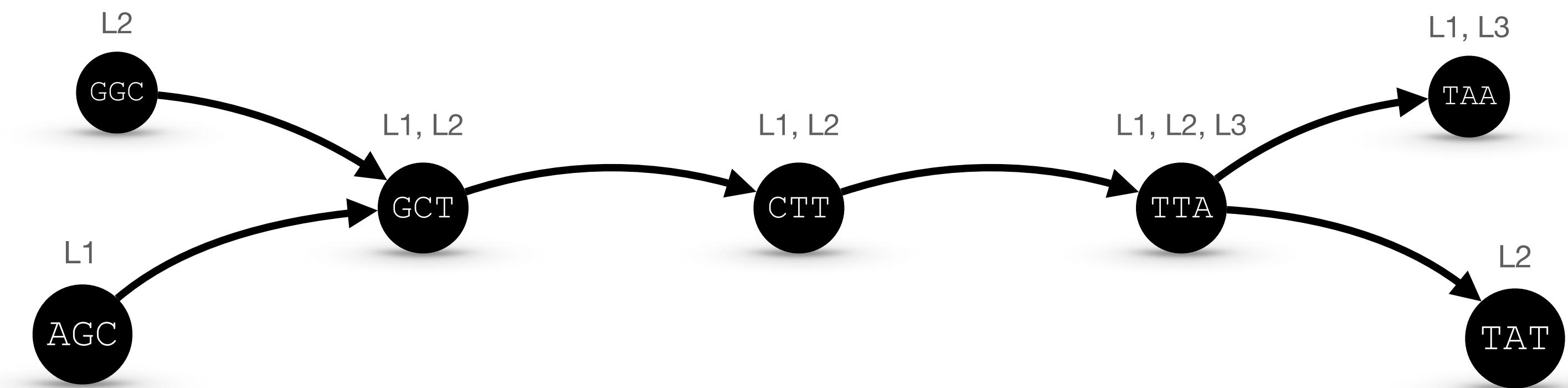
1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]
6. RowDiff [Danciu et al., 2021]

Background

Graph annotation representations

1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]
6. RowDiff [Danciu et al., 2021]

	L1	L2	L3
TAA	1		1
TAT		1	
GCT	1	1	
AGC	1		
GGC		1	
CTT	1	1	
TTA	1	1	1



Background

Graph annotation representations

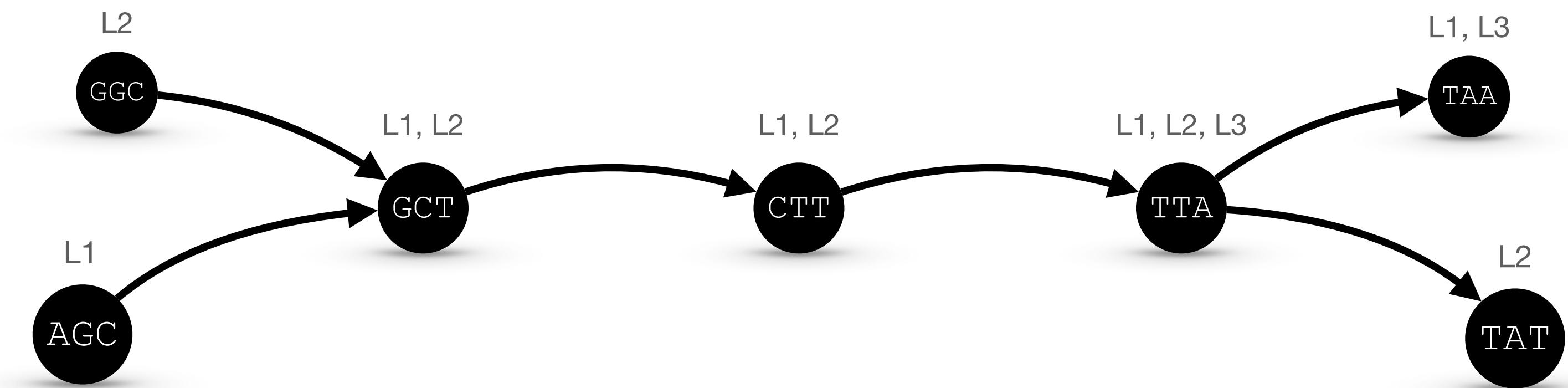
1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]
6. RowDiff [Danciu et al., 2021]

Store only **diffs**:

$$L^\delta(v) := L(v) \oplus L(v_{\text{succ}})$$

(\oplus is XOR)

	L1	L2	L3
TAA	1		1
TAT		1	
GCT	1	1	
AGC	1		
GGC		1	
CTT	1	1	
TTA	1	1	1



Background

Graph annotation representations

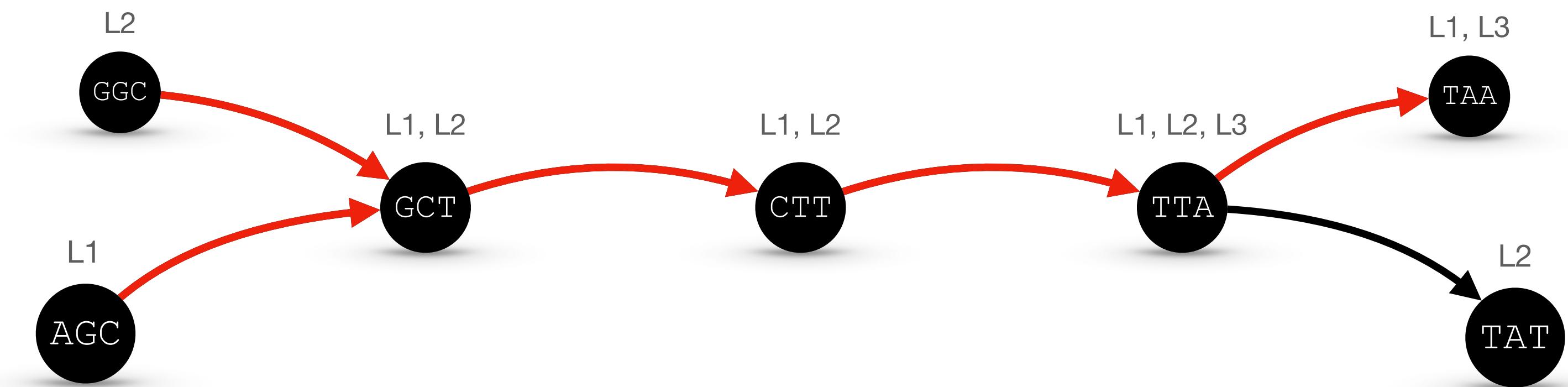
1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]
6. RowDiff [Danciu et al., 2021]

Store only **diffs**:

$$L^\delta(v) := L(v) \oplus L(v_{\text{succ}})$$

(\oplus is XOR)

	L1	L2	L3
TAA	1		1
TAT		1	
GCT	1	1	
AGC	1		
GGC		1	
CTT	1	1	
TTA	1	1	1



Background

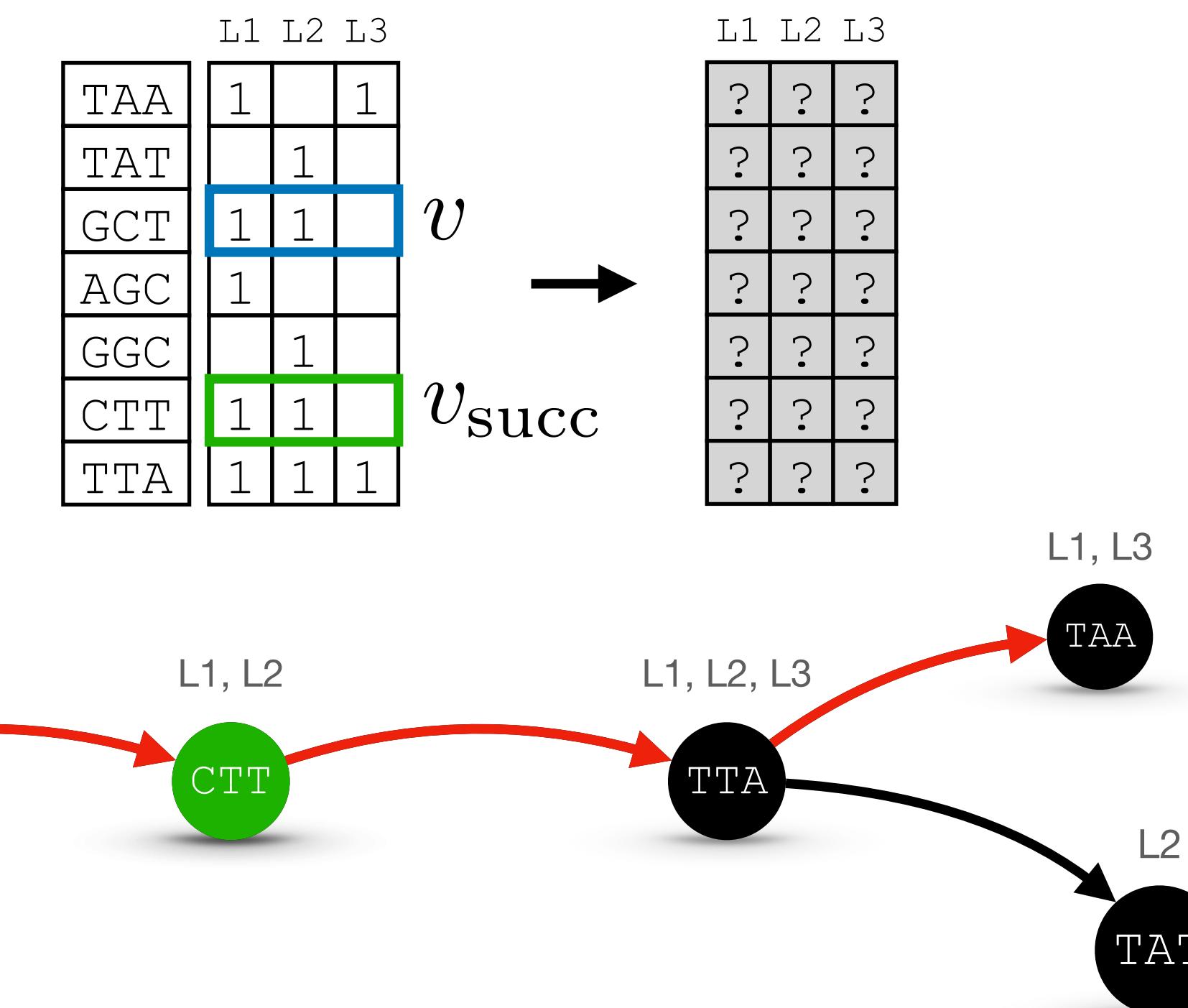
Graph annotation representations

1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]
6. RowDiff [Danciu et al., 2021]

Store only **diffs**:

$$L^\delta(v) := L(v) \oplus L(v_{\text{succ}})$$

(\oplus is XOR)



Background

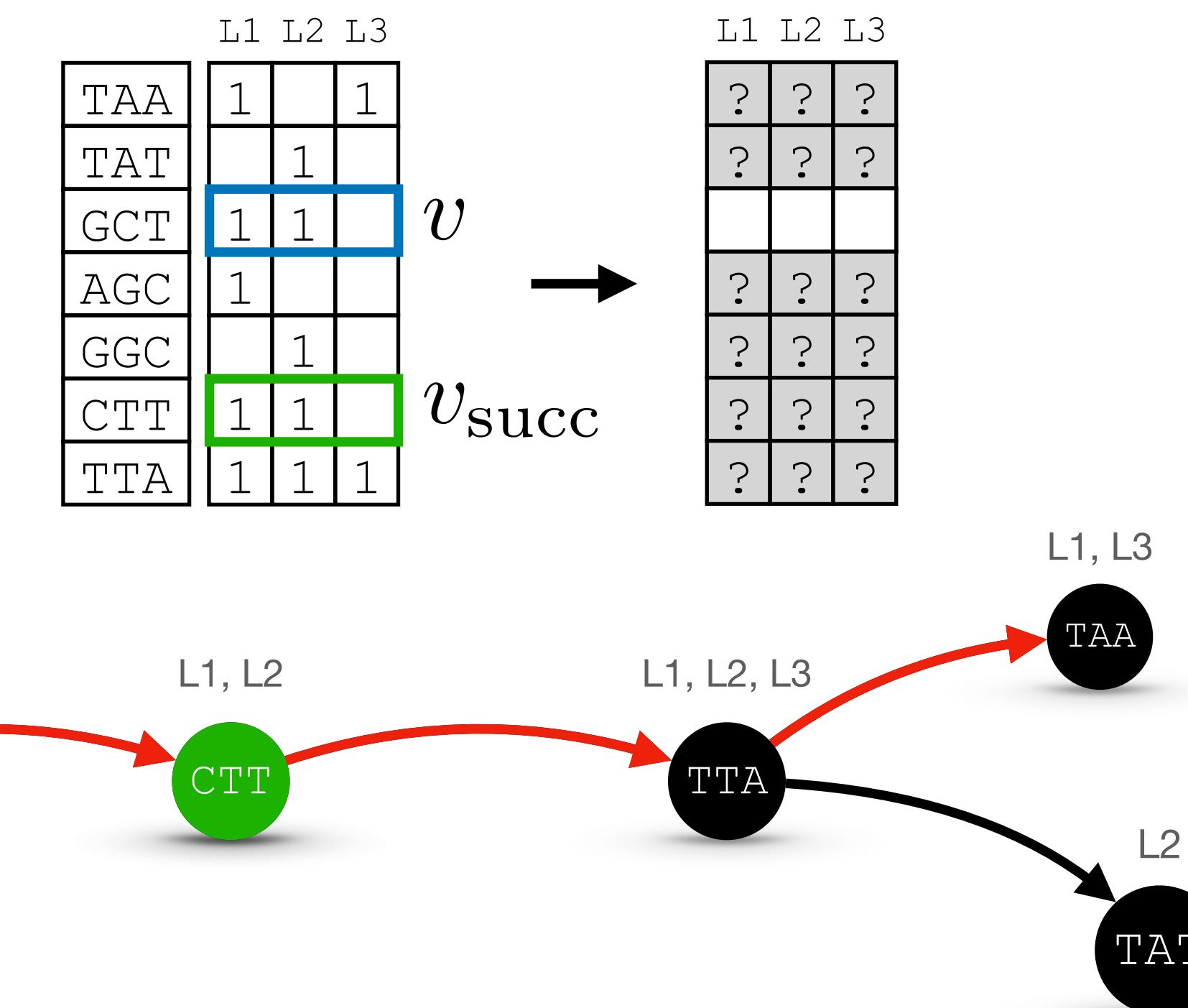
Graph annotation representations

1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]
6. RowDiff [Danciu et al., 2021]

Store only **diffs**:

$$L^\delta(v) := L(v) \oplus L(v_{\text{succ}})$$

(\oplus is XOR)



Background

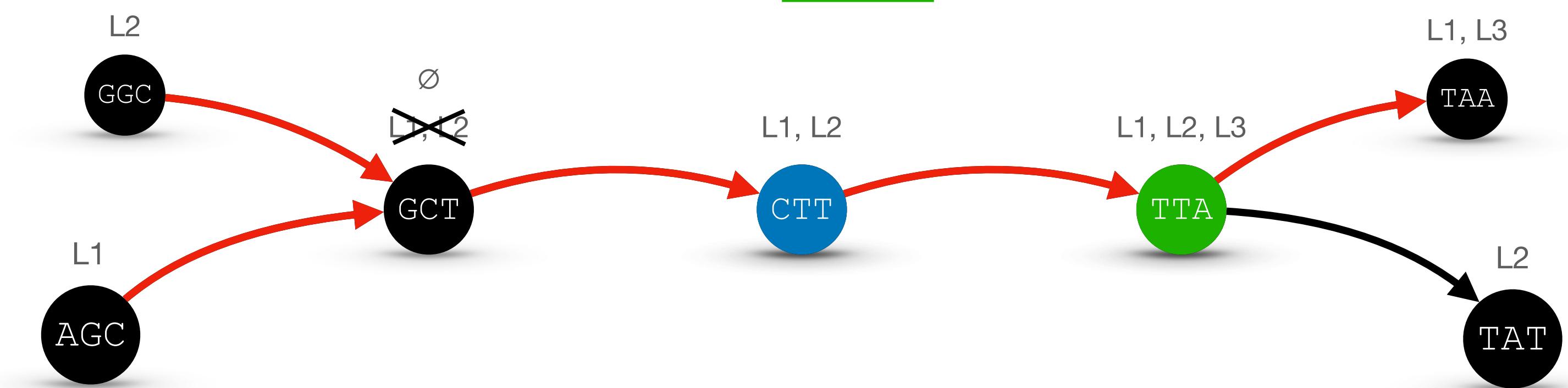
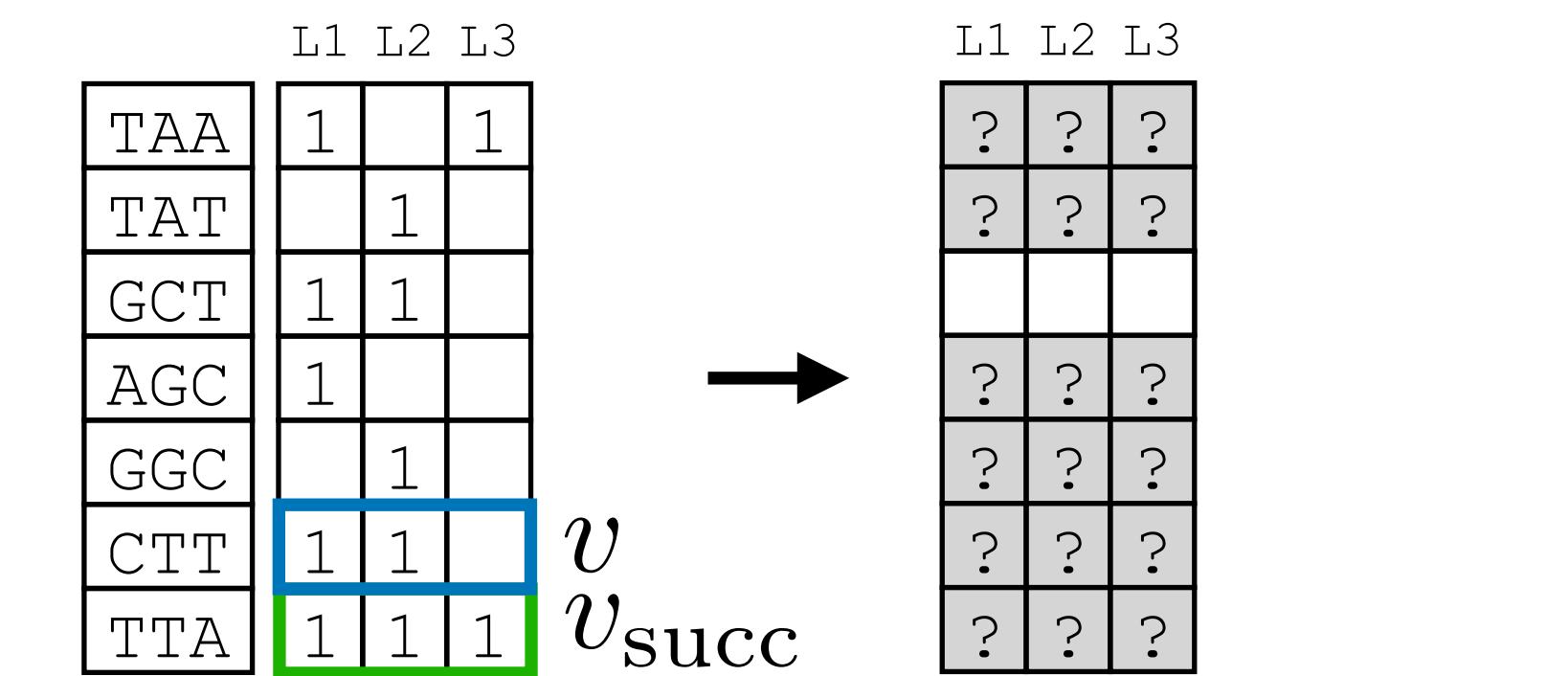
Graph annotation representations

1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]
6. RowDiff [Danciu et al., 2021]

Store only **diffs**:

$$L^\delta(v) := L(v) \oplus L(v_{\text{succ}})$$

(\oplus is XOR)



Background

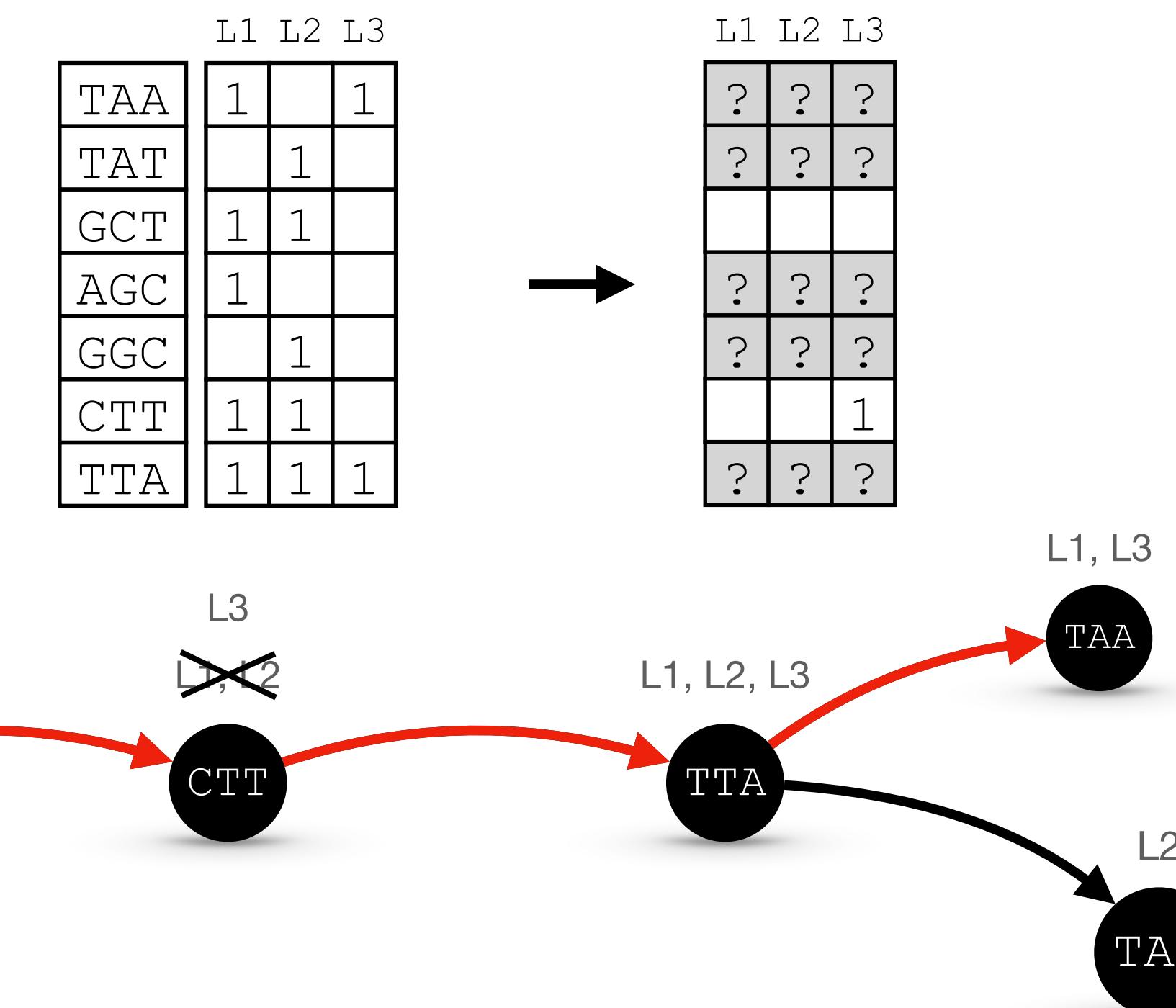
Graph annotation representations

1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]
6. RowDiff [Danciu et al., 2021]

Store only **diffs**:

$$L^\delta(v) := L(v) \oplus L(v_{\text{succ}})$$

(\oplus is XOR)



Background

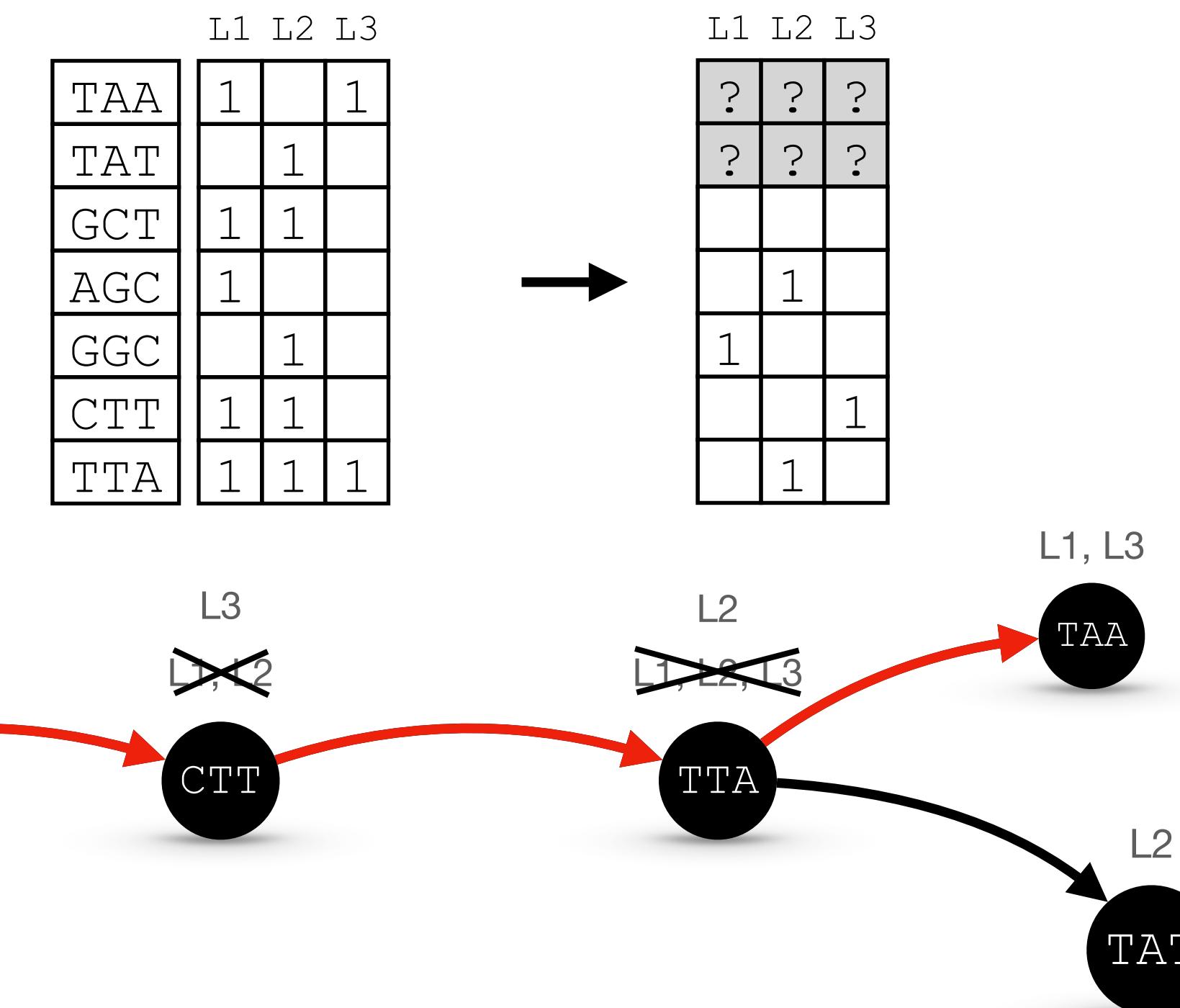
Graph annotation representations

1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]
6. RowDiff [Danciu et al., 2021]

Store only **diffs**:

$$L^\delta(v) := L(v) \oplus L(v_{\text{succ}})$$

(\oplus is XOR)



Background

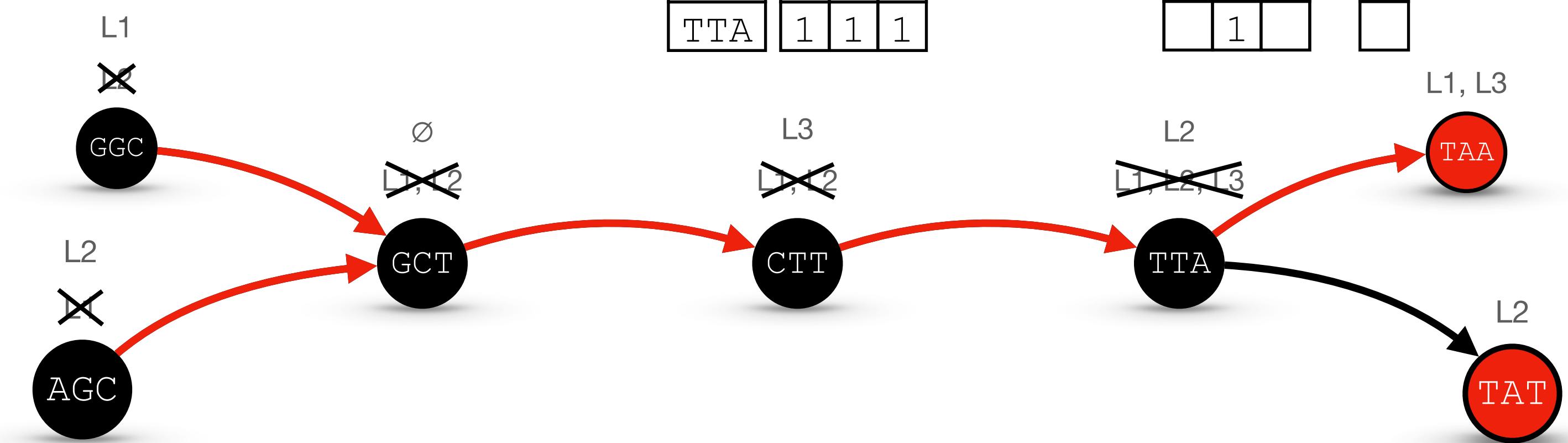
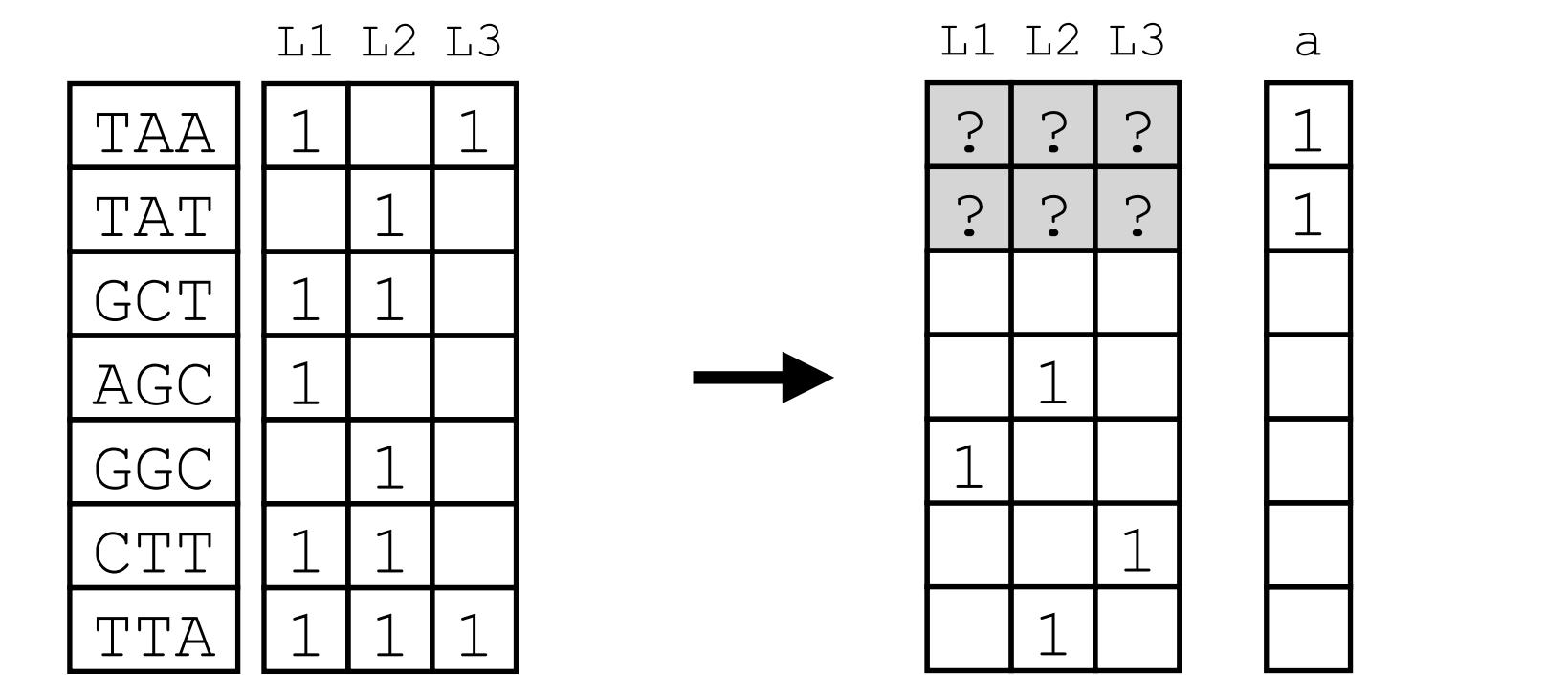
Graph annotation representations

1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]
6. RowDiff [Danciu et al., 2021]

Store only **diffs**:

$$L^\delta(v) := L(v) \oplus L(v_{\text{succ}})$$

(\oplus is XOR)



Background

Graph annotation representations

1. Column-major sparse representation
2. Multi-BRWT [Karasikov et al., 2019]
3. RowFlat (employed in VARI [Muggli et al., 2017])
4. Rainbowfish [Almodaresi et al., 2017]
5. Mantis-MST [Almodaresi et al., 2019]
6. RowDiff [Danciu et al., 2021]

Store only **diffs**:

$$L^\delta(v) := L(v) \oplus L(v_{\text{succ}})$$

(\oplus is XOR)

