Compressing and indexing pangenomes with meta-colored compacted de Bruijn graphs

Giulio Ermanno Pibiri

Ca' Foscari University of Venice

@giulio_pibiri

@jermp

ALPACA-PANGAIA Annual Workshop
Amsterdam, Netherlands, 22 November 2023

The colored k-mer indexing problem

- A **k-mer** is a sub-string of length k of some string R.

- We are given a collection $\mathscr{R} = \{R_1, \ldots, R_N\}$ of reference sequences.
  Each R_i is a (long) sequence over the DNA alphabet {A,C,G,T}.

- **Problem.** We want to build an *index* for $\mathscr{R}$ so that we can retrieve the set
  Color$(x) = \{i \,|\, x \in R_i\}$ efficiently for any k-mer x. Note that Color$(x) = \emptyset$ if $x \notin \mathscr{R}$.

The colored k-mer indexing problem

- A **k-mer** is a sub-string of length k of some string R.
- We are given a collection $\mathscr{R} = \{R_1, \ldots, R_N\}$ of reference sequences.
  Each R_i is a (long) sequence over the DNA alphabet {A,C,G,T}.
- **Problem.** We want to build an *index* for $\mathscr{R}$ so that we can retrieve the set
  $\text{Color}(x) = \{i \,|\, x \in R_i\}$ efficiently for any k-mer x. Note that $\text{Color}(x) = \emptyset$ if $x \notin \mathscr{R}$.
- A lot of hype in the indexing community for the case where $\mathscr{R}$ is a **pangenome**, i.e., a collection of related genomes.
- **Applications.** This problem is relevant for applications where sequences are first matched against known references (i.e., mapping/alignment algorithms): single-cell RNA-seq, metagenomics, etc.

Modular indexing layout

- **Goal.** What we want is the **map** $x \to \text{Color}(x) = \{i \mid x \in R_i\}$.

Modular indexing layout

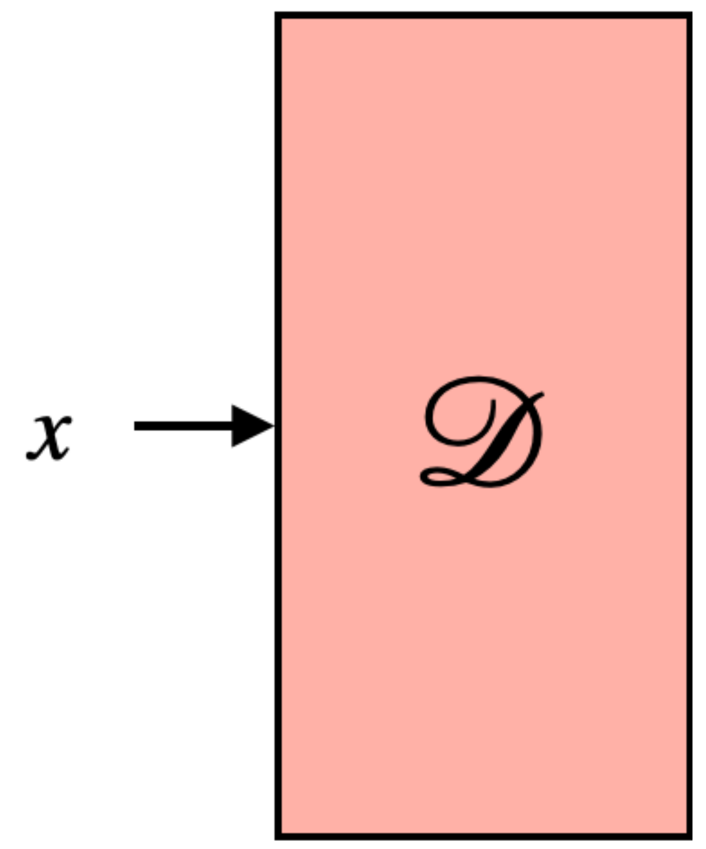
- **Goal.** What we want is the **map** $x \rightarrow \text{Color}(x) = \{i \,|\, x \in R_i\}$.
- **Two** data structures:

Modular indexing layout

- **Goal.** What we want is the **map** $x \rightarrow \text{Color}(x) = \{i \,|\, x \in R_i\}$.
- **Two** data structures:

1. All the distinct k-mers in $\mathscr{R} = \{R_1, \ldots, R_N\}$ are stored in the **dictionary** $\mathscr{D}$.

   Assume $\mathscr{D}$ stores n distinct k-mers and supports a Lookup(x) operation which, given a k-mer x, returns a unique integer $1 \leq h \leq n$ if $x \in \mathscr{D}$, and $\perp$ otherwise.



Modular indexing layout

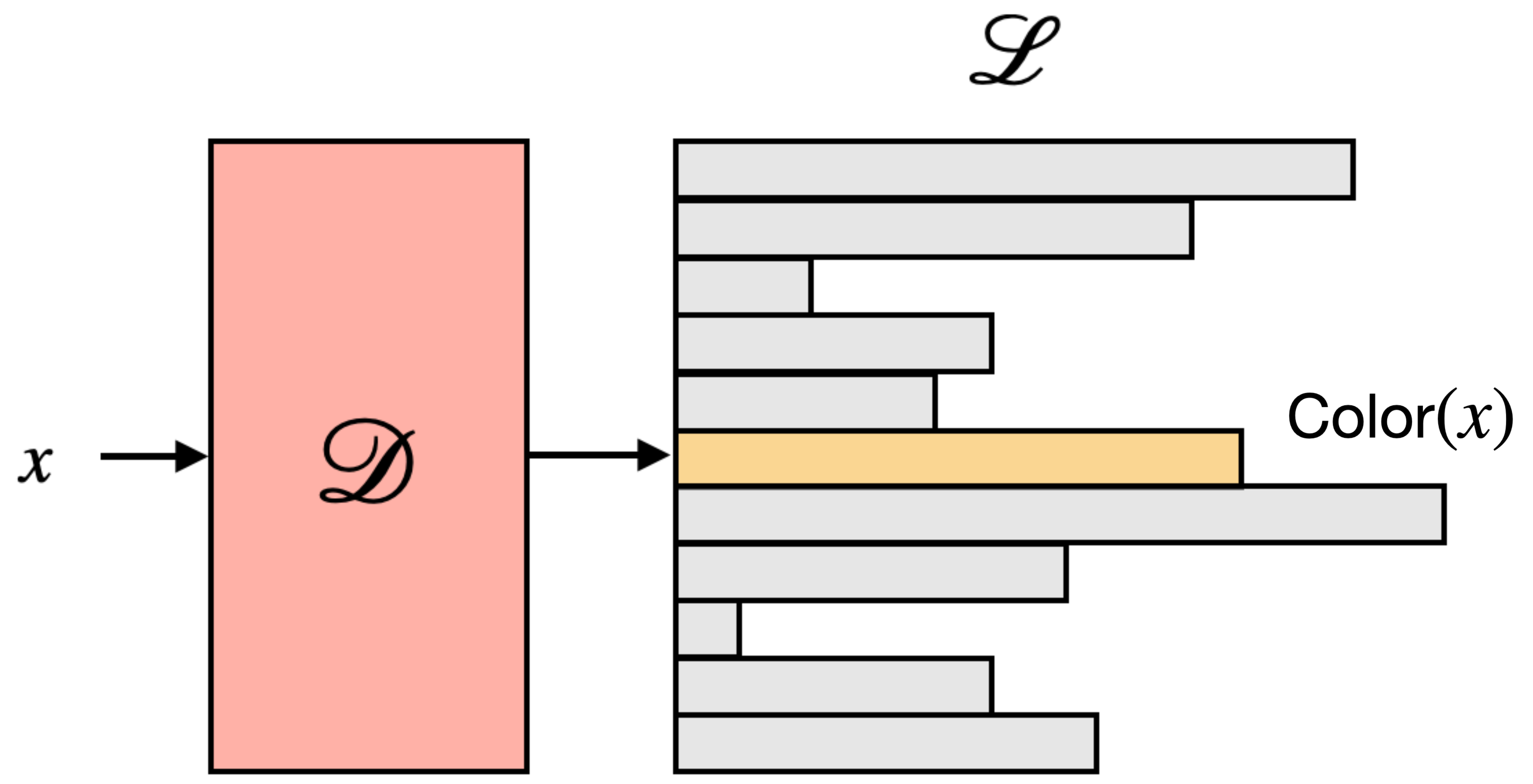
- **Goal.** What we want is the **map** $x \to \text{Color}(x) = \{i \,|\, x \in R_i\}$.

- **Two** data structures:

1. All the distinct k-mers in $\mathscr{R} = \{R_1, \ldots, R_N\}$ are stored in the **dictionary** $\mathscr{D}$.

   Assume $\mathscr{D}$ stores n distinct k-mers and supports a Lookup(x) operation which, given a k-mer x, returns a unique integer $1 \leq h \leq n$ if $x \in \mathscr{D}$, and $\perp$ otherwise.

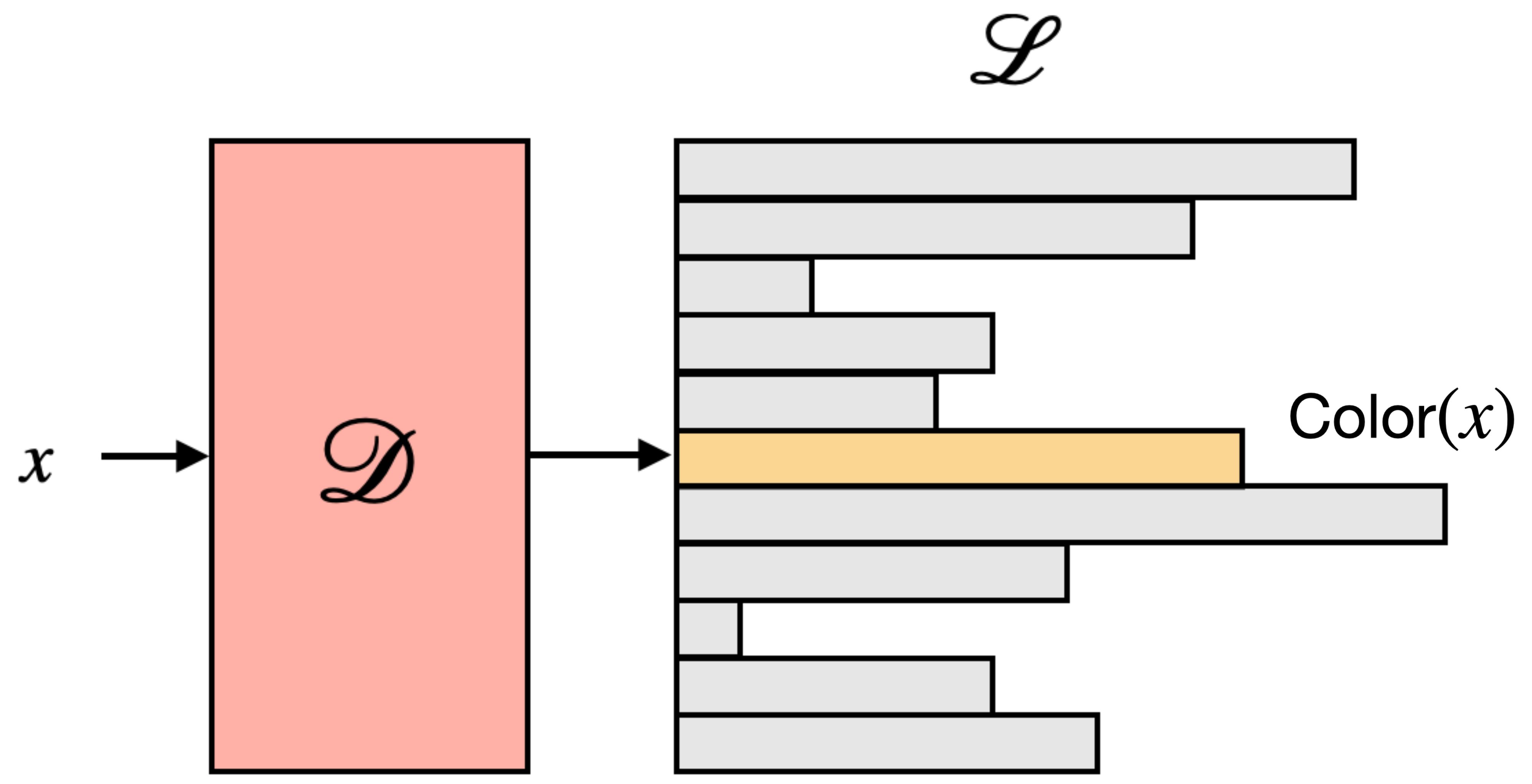
2. The sets $\{\text{Color}(x)\}_x$ are stored in order of Lookup(x) in the **inverted index** $\mathscr{L}$.



Modular indexing layout

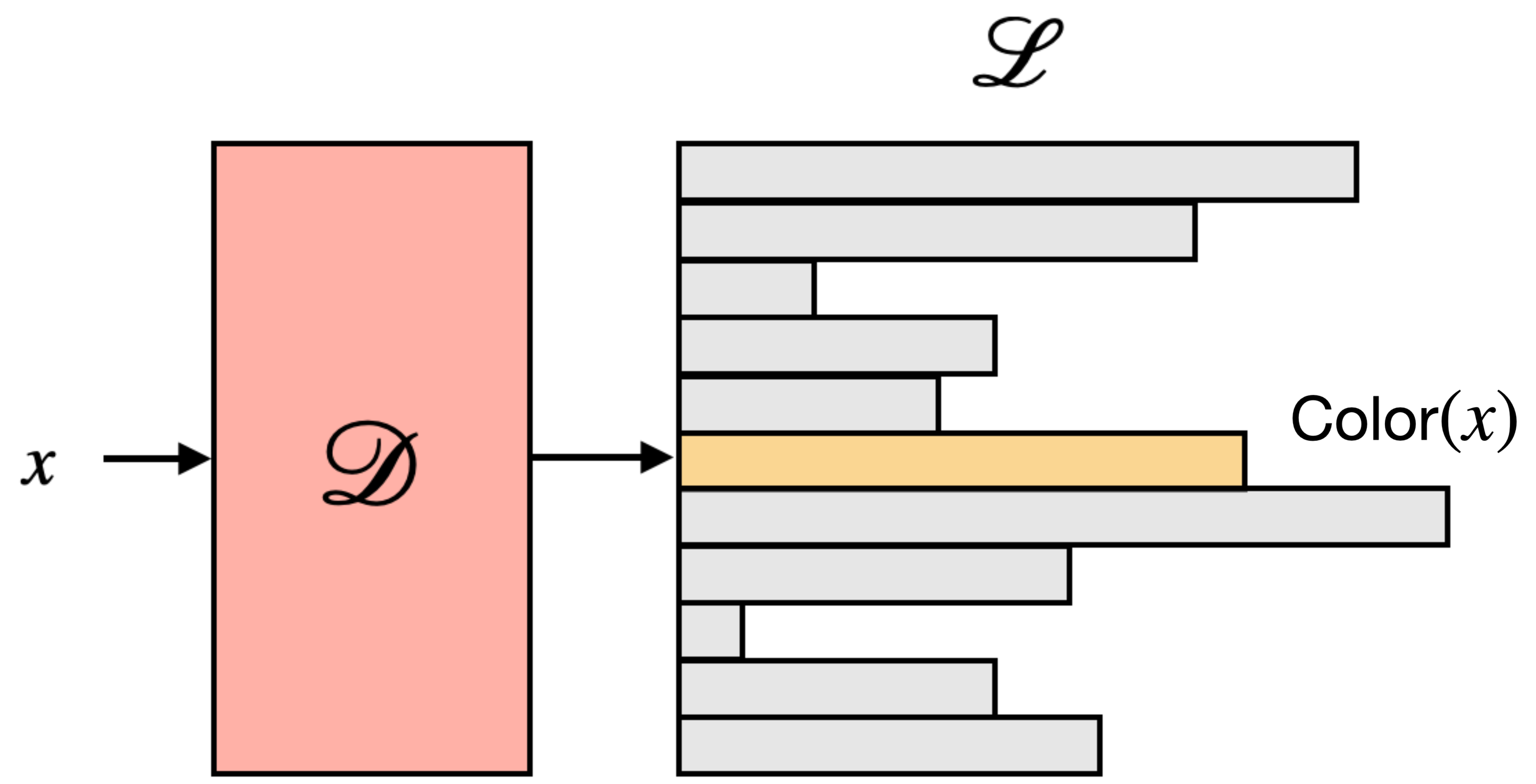
- Many k-mer based indexes (all of them?) are incarnations/adaptations of this **modular indexing layout**, $\mathscr{D} + \mathscr{L}$:

  - deBGA [Liu et al. 2016]
  - Kallisto [Bray et al. 2016]
  - BIGSI [Bradley et al. 2017]
  - Rainbowfish [Almodaresi et al. 2017]
  - Mantis [Pandey et al. 2018]
  - Pufferfish [Almodaresi et al. 2018]
  - SeqOthello [Yu et al. 2018]
  - COBS [Bingmann et al. 2019]
  - Reindeer [Marchet et al. 2020]
  - Raptor [Seiler et al. 2021]
  - Metagraph [Karasikov et al. 2022]
  - NIQKI [Agret et al. 2022]
  - Pufferfish2 [Fan et al. 2022]
  - Themisto [Alanko et. al 2023]
  - Fulgor-v1 [Fan et. al 2023]
  - Fulgor-v2 [P., Fan, Patro, 2023]



Modular indexing layout

- Our problem reduces to that of **representing two data structures**, $\mathscr{D}$ and $\mathscr{L}$.
- To do so at best, we **must understand/exploit** the **properties** of our problem.
- **Q.** What are these properties?



What is special about k-mers?

- Since we take *all* the distinct k-mers (i.e., consecutive) from our references, **they share (k-1)-symbol overlaps**.

```
ACGGTAGAACCGATTCAAATTCGACGTAGC...
ACGGTAGAACCGA
 CGGTAGAACCGAT
  GGTAGAACCGATT
   GTAGAACCGATTC
    TAGAACCGATTCA
     AGAACCGATTCAA
      GAACCGATTCAAA
       AACCGATTCAAAT
        ...
```

$\longleftarrow$ Example for $k = 13$.

What is special about k-mers?

- Since we take *all* the distinct k-mers (i.e., consecutive) from our references, **they share (k-1)-symbol overlaps**.

```
ACGGTAGAACCGATTCAAATTCGACGTAGC...
ACGGTAGAACCGA
 CGGTAGAACCGAT
  GGTAGAACCGATT
   GTAGAACCGATTC
    TAGAACCGATTCA
     AGAACCGATTCAA
      GAACCGATTCAAA
       AACCGATTCAAAT
        ...
```

$\longleftarrow$ Example for $k = 13$.

- **Two** important consequences:
  1. It is very likely that, given x in a query sequence Q and its answer returned from the index, $next(x)$ has a very similar answer (if not identical) $\rightarrow$ **Compression for satellite data.**
  2. **Faster query time**: given the answer to x, the answer to $next(x)$ can be computed faster than the answer for another arbitrary k-mer $y \neq next(x)$.

What is special about k-mers?

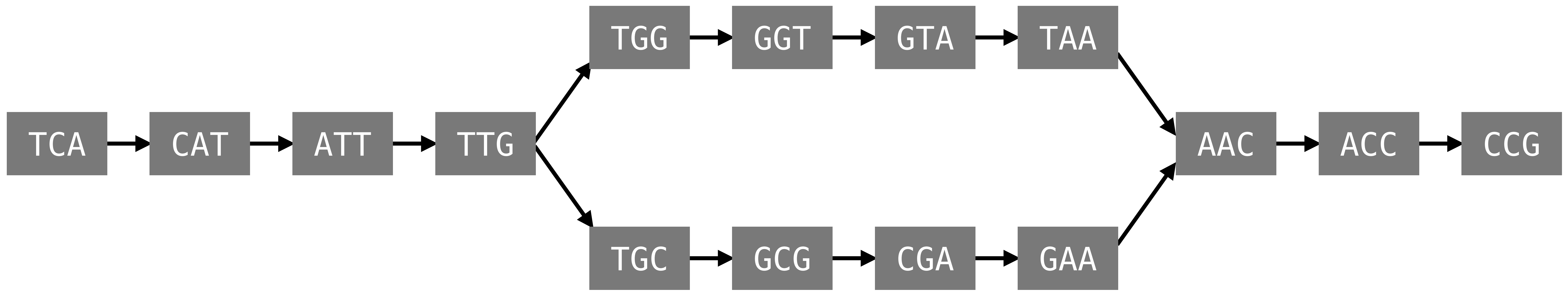
- Since we take *all* the distinct k-mers (i.e., consecutive) from our references, **they share (k-1)-symbol overlaps**.

```
ACGGTAGAACCGATTCAAATTCGACGTAGC...
ACGGTAGAACCGA
 CGGTAGAACCGAT
  GGTAGAACCGATT
   GTAGAACCGATTC          ←   Example for k = 13.
    TAGAACCGATTCA
     AGAACCGATTCAA
      GAACCGATTCAAA
       AACCGATTCAAAT
        ...
```

- As simple as this might sound, indexes usually **overlook** this property to achieve better space and query time. Only few of them actually exploit it.
- Examples:
  - findere [Robidou and Peterlongo, 2021/22]
  - SSHash [P., 2022]
  - LPHash [P., Shibuya, Limasset 2022]

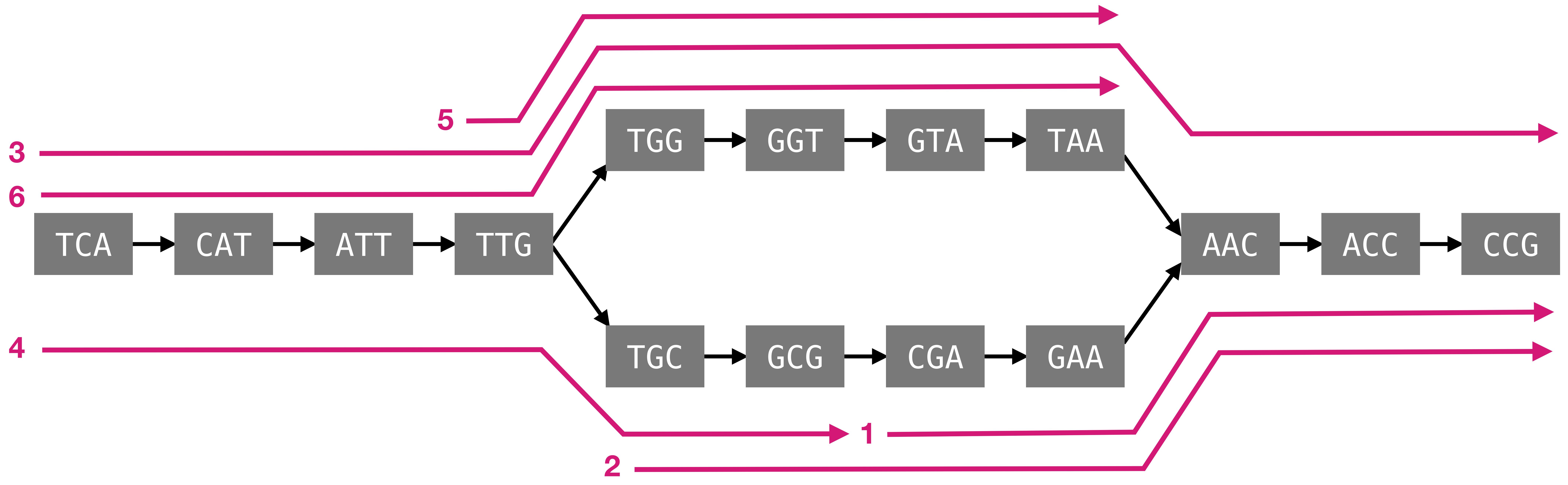
de Bruijn graphs

- The dictionary $\mathscr{D}$ is a set of k-mers with (k-1)-symbol overlaps.
- One-to-one correspondence between $\mathscr{D}$ and a *de Bruijn* graph (dBG).
- Example for **k = 3**.



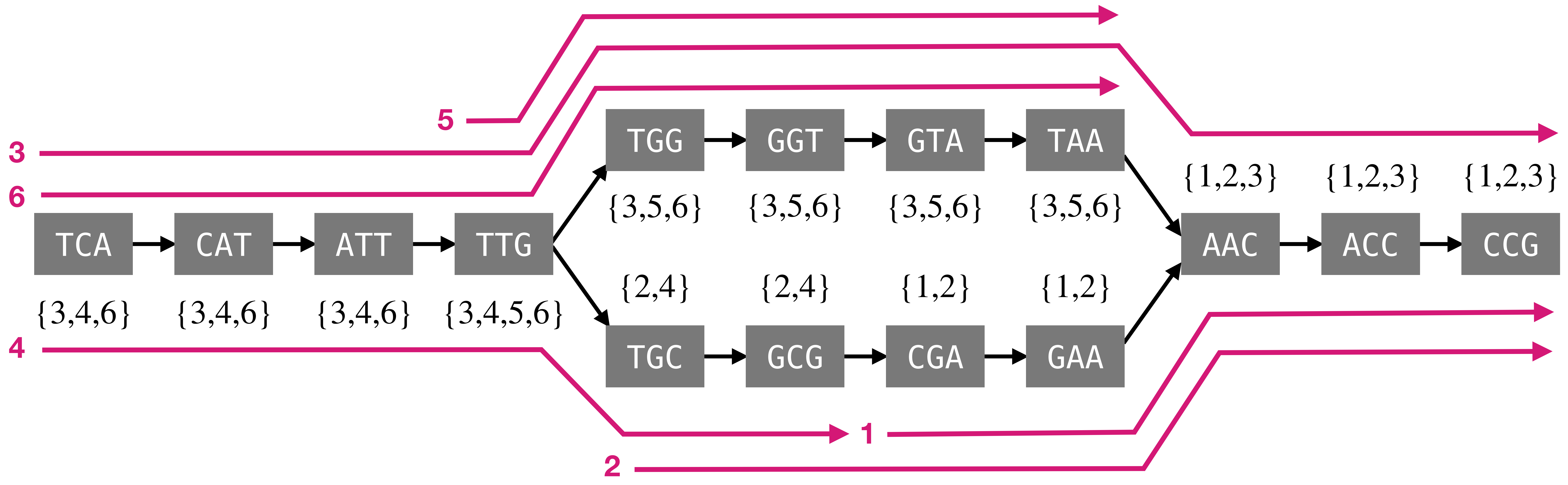
Colored de Bruijn graphs

- Example for k = 3 and **N = 6 references.** References in $\mathscr{R}$ are spelled by **paths** in the graph.



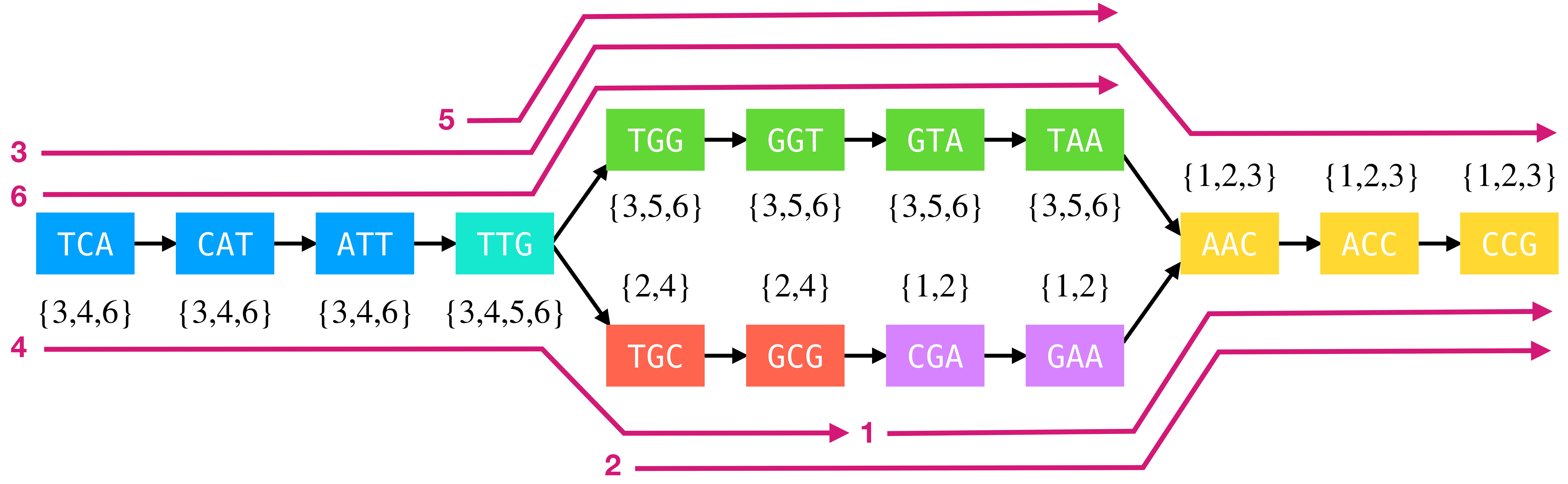
Colored de Bruijn graphs

- Example for k = 3 and **N = 6 references.** References in $\mathscr{R}$ are spelled by **paths** in the graph.



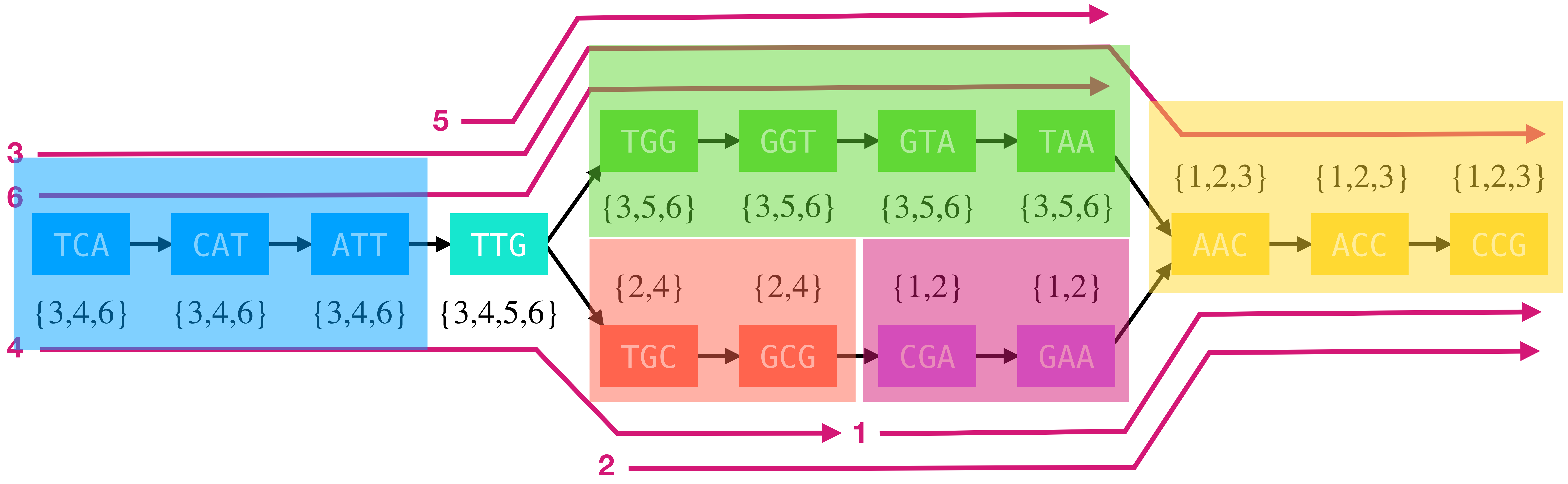
Colored de Bruijn graphs

- Example for k = 3 and **N = 6 references.** References in $\mathscr{R}$ are spelled by **paths** in the graph.



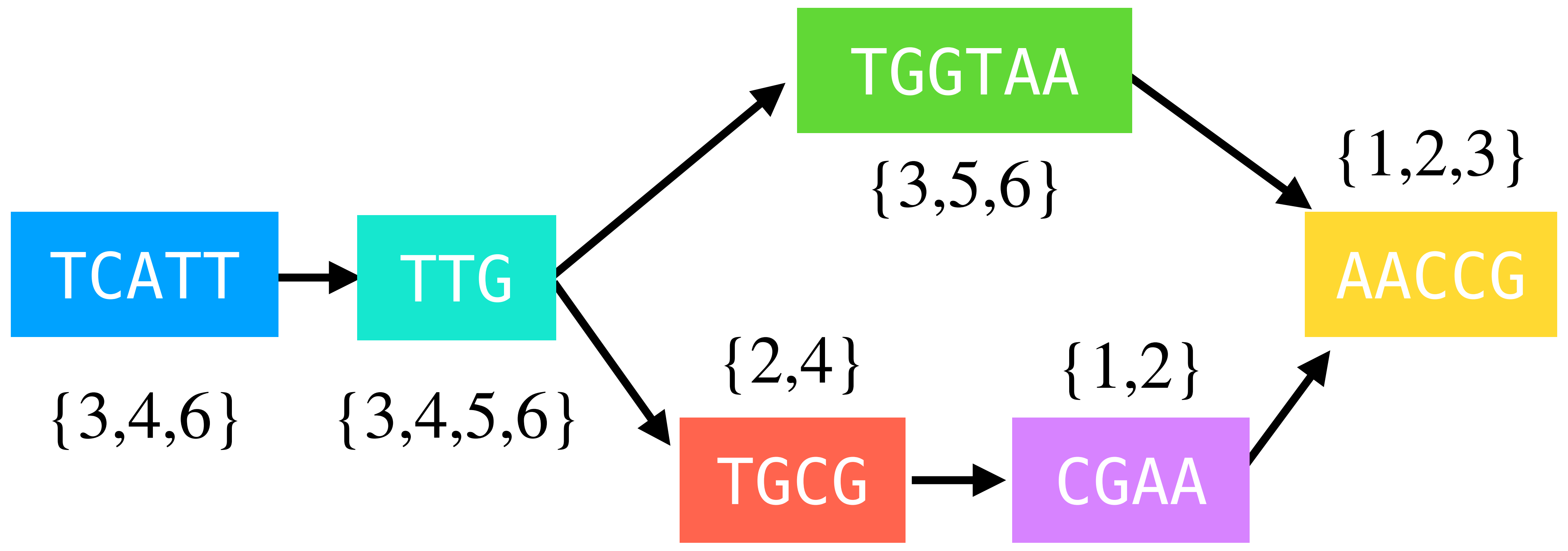
Colored de Bruijn graphs

- Example for k = 3 and **N = 6 references.** References in $\mathscr{R}$ are spelled by **paths** in the graph.



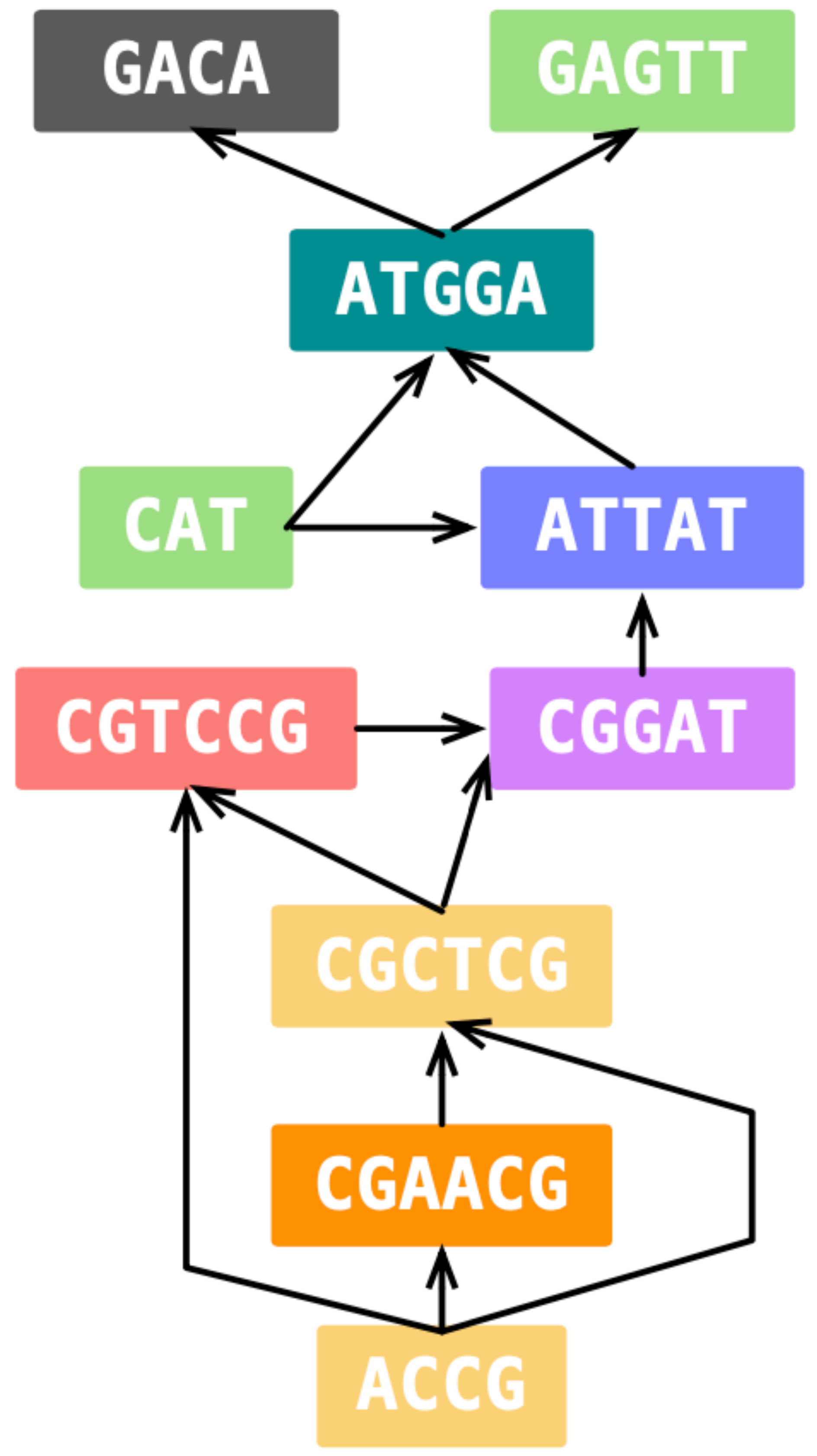
Colored compacted de Bruijn graphs

- Example for k = 3 and **N = 6 references.**
- Nodes having the **same color along non-branching paths** are collapsed into **unitigs**.



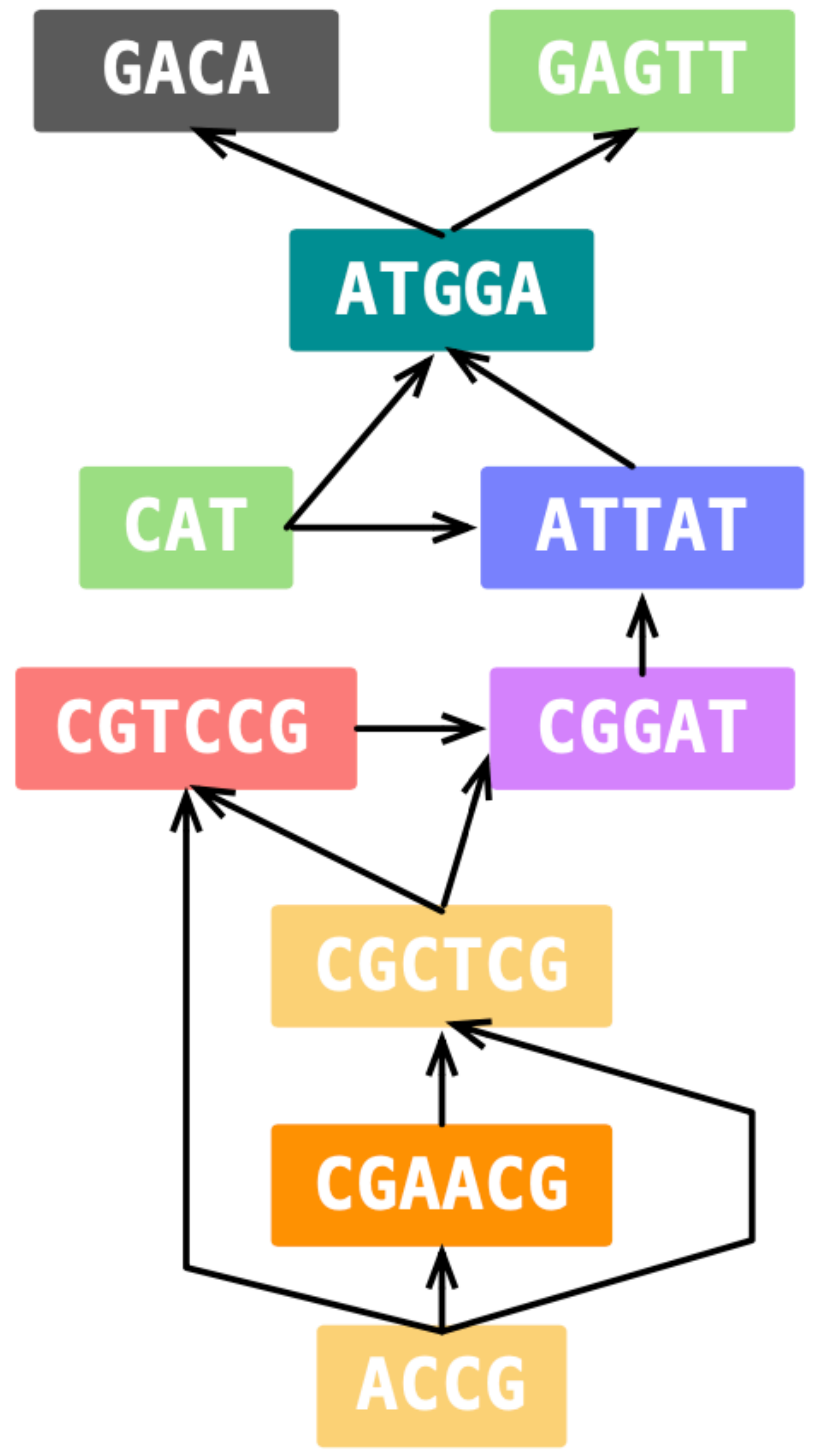
Colored compacted de Bruijn graphs

- Another, larger, example for **N = 16** references that we will use in the following.



Colored compacted de Bruijn graphs

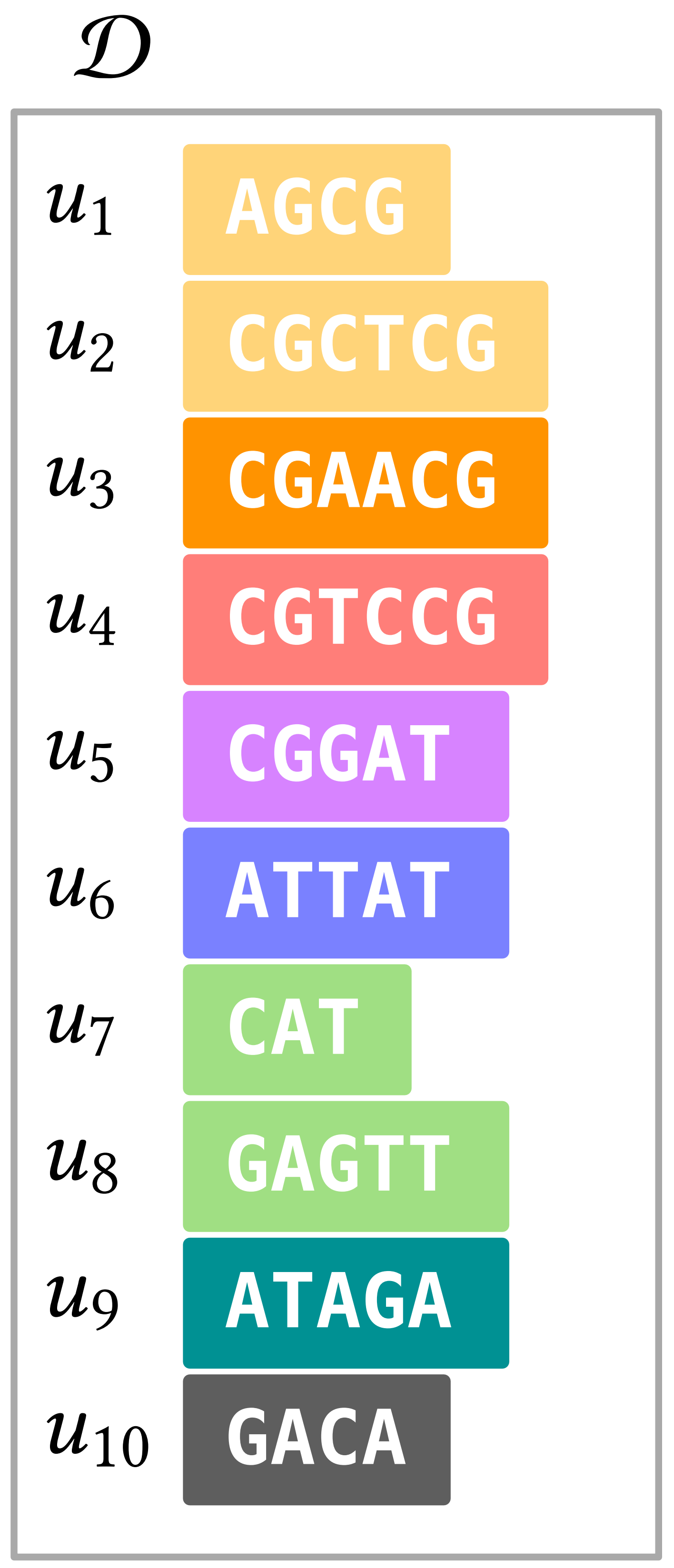
- Another, larger, example for **N = 16** references that we will use in the following.



- Let's now consider the properties of colored compacted dBGs and **how** we can exploit them for efficient indexing.

Properties of colored compacted dBGs

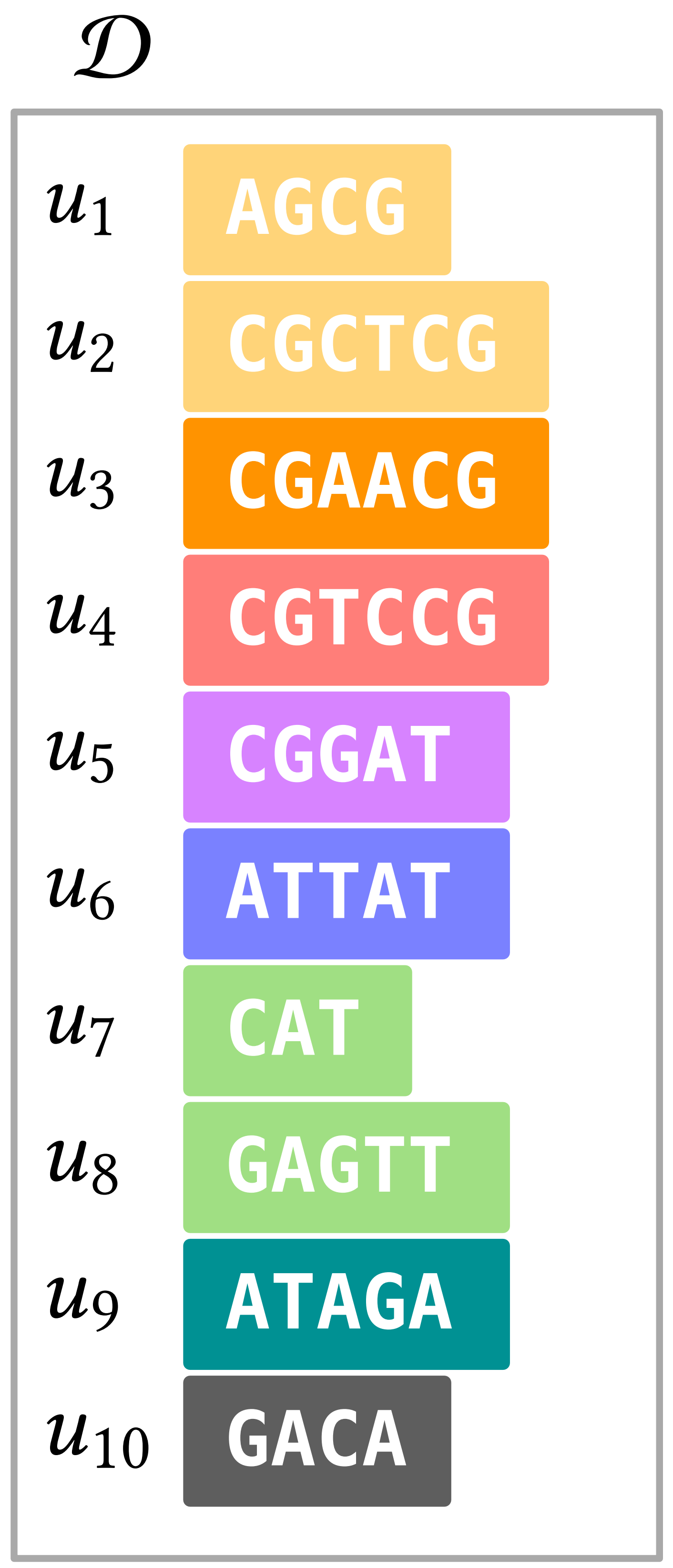
1. **Unitigs spell references in $\mathscr{R}$.** → We can represent the set of unitigs instead of the set of k-mers. Better space effectiveness and cache locality.



Properties of colored compacted dBGs

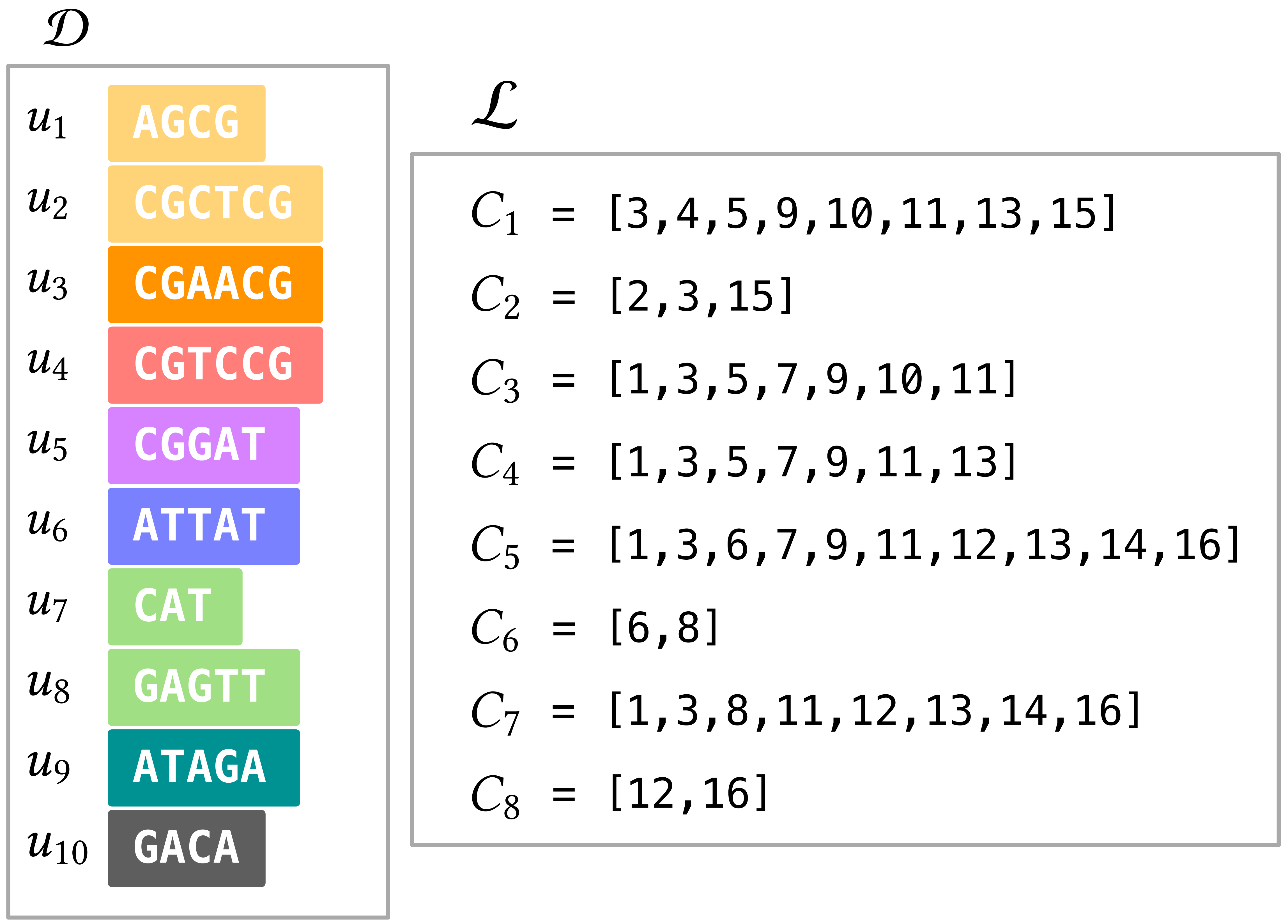
1. **Unitigs spell references in $\mathscr{R}$.** → We can represent the set of unitigs instead of the set of k-mers. Better space effectiveness and cache locality.

- Represent $\mathscr{D}$ with **SSHash** [P., 2022].
- SSHash stores a set of unitigs **in any wanted order** (*order-preserving*).



Properties of colored compacted dBGs

2. **Unitigs are monochromatic.** → We store a color set for each unitig, rather than for each k-mer because Color(x) = Color(y) if k-mers x and y are part of the same unitig.
   Thus, we need an efficient **map from k-mers to unitigs**, $x \rightarrow unitig(x)$.



$\mathcal{D}$

u_1	AGCG
u_2	CGCTCG
u_3	CGAACG
u_4	CGTCCG
u_5	CGGAT
u_6	ATTAT
u_7	CAT
u_8	GAGTT
u_9	ATAGA
u_{10}	GACA

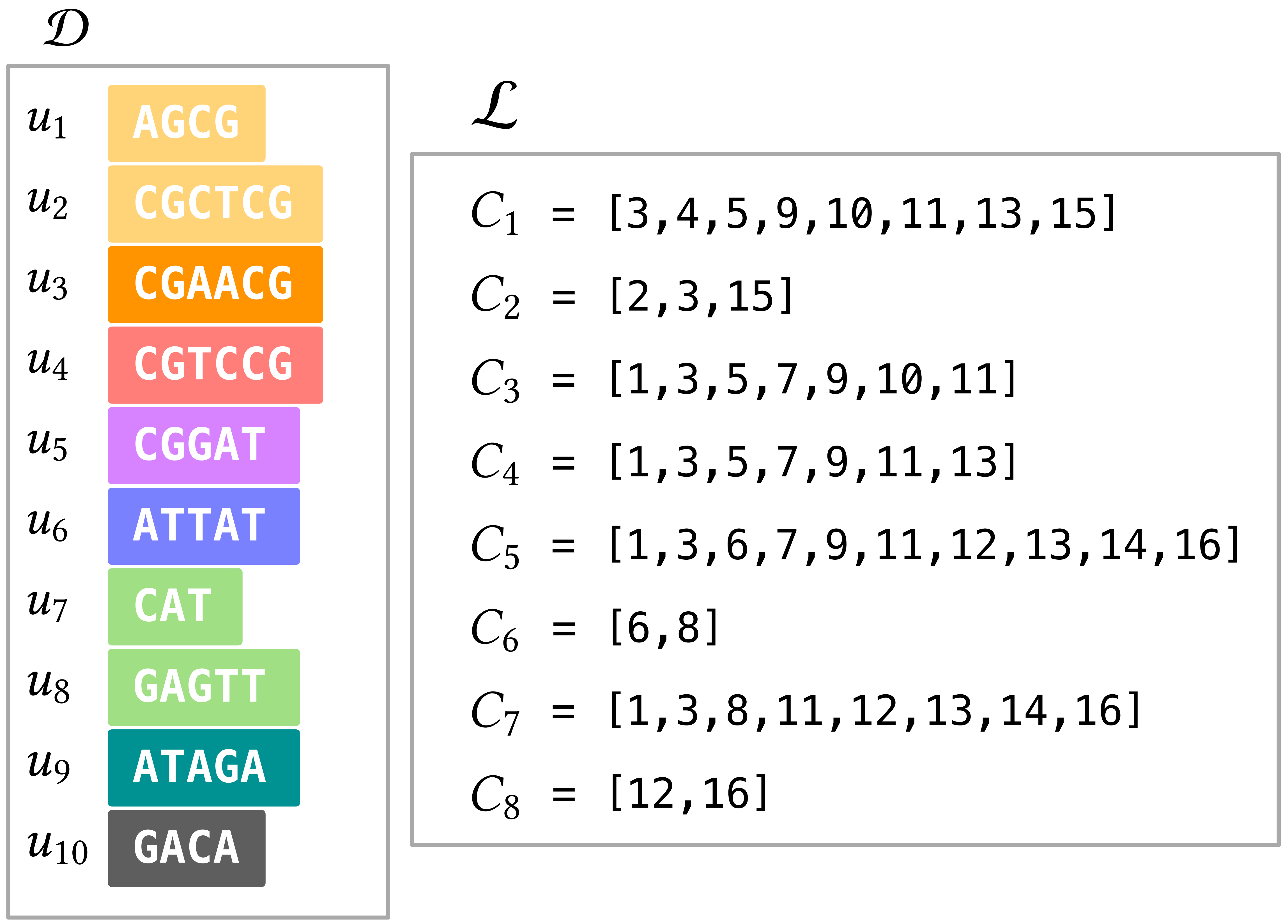
$\mathcal{L}$

```
C1 = [3,4,5,9,10,11,13,15]
C2 = [2,3,15]
C3 = [1,3,5,7,9,10,11]
C4 = [1,3,5,7,9,11,13]
C5 = [1,3,6,7,9,11,12,13,14,16]
C6 = [6,8]
C7 = [1,3,8,11,12,13,14,16]
C8 = [12,16]
```

Properties of colored compacted dBGs

2. **Unitigs are monochromatic.** → We store a color set for each unitig, rather than for each k-mer because Color(x) = Color(y) if k-mers x and y are part of the same unitig.
Thus, we need an efficient **map from k-mers to unitigs**, $x \rightarrow unitig(x)$.

- Represent $\mathscr{D}$ with **SSHash** [P., 2022].
- SSHash stores a set of unitigs in any wanted order, so it is easy to compute the **unitig identifier** $unitig(x)$ given the k-mer x.
- Now $\mathscr{L}$ stores Color(x) for each unitig in the order given by $unitig(x)$.



$\mathcal{D}$

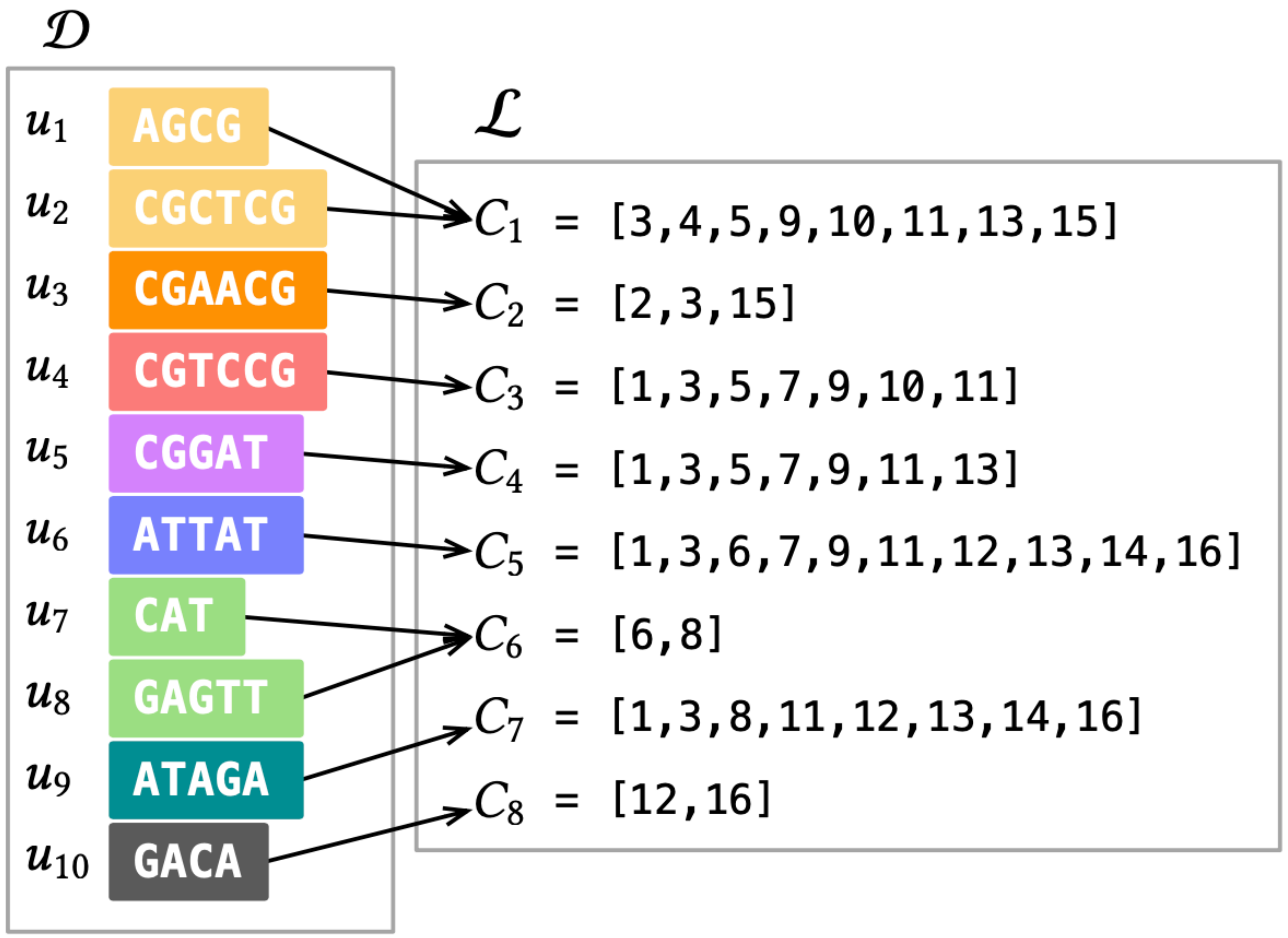
u_1	AGCG
u_2	CGCTCG
u_3	CGAACG
u_4	CGTCCG
u_5	CGGAT
u_6	ATTAT
u_7	CAT
u_8	GAGTT
u_9	ATAGA
u_{10}	GACA

$\mathcal{L}$

C_1 = [3,4,5,9,10,11,13,15]
C_2 = [2,3,15]
C_3 = [1,3,5,7,9,10,11]
C_4 = [1,3,5,7,9,11,13]
C_5 = [1,3,6,7,9,11,12,13,14,16]
C_6 = [6,8]
C_7 = [1,3,8,11,12,13,14,16]
C_8 = [12,16]

Properties of colored compacted dBGs

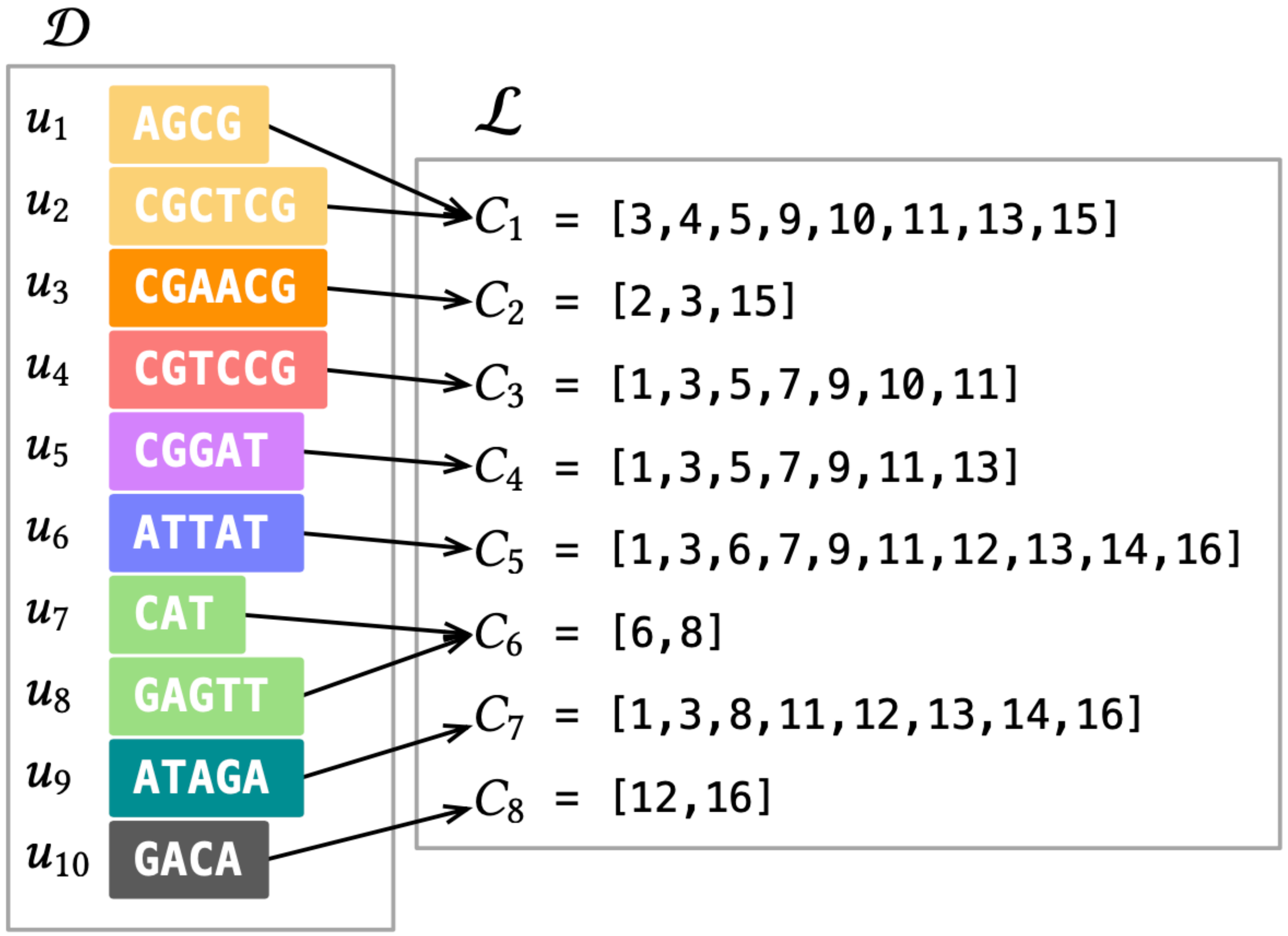
3. **Unitigs co-occur.** → Distinct unitigs often have the same color, i.e., they co-occur in the same subset of references. We have way less distinct colors than unitigs. We need an efficient **map from unitigs to colors** $unitig(x) \rightarrow \text{Color}(x)$.



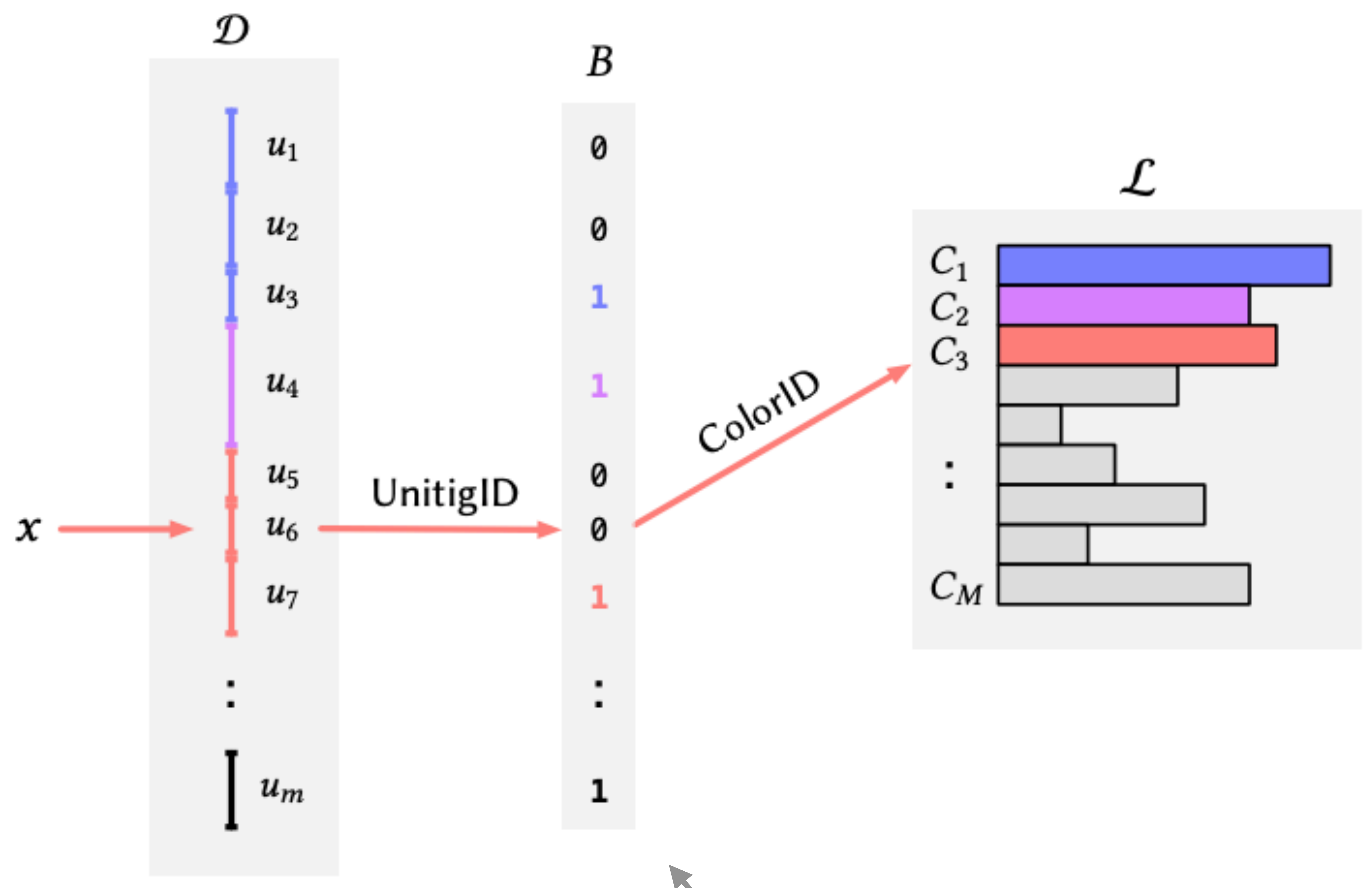
Properties of colored compacted dBGs

3. **Unitigs co-occur.** → Distinct unitigs often have the same color, i.e., they co-occur in the same subset of references. We have way less distinct colors than unitigs.
We need an efficient **map from unitigs to colors** $unitig(x) \rightarrow \text{Color}(x)$.

- Represent $\mathscr{D}$ with **SSHash** [P., 2022].
- SSHash stores a set of unitigs in any wanted order, so we can **permute** the unitigs in $\mathscr{D}$ so that **consecutive unitigs have the same color**.
- Then, mapping a unitig to its color is as simple as a Rank query over a bitmap.



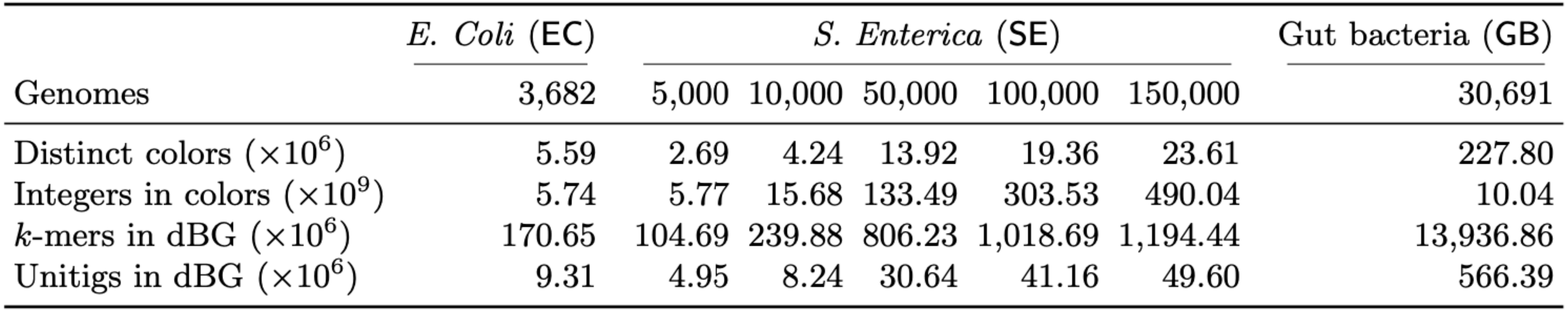
Mapping unitigs to colors in succinct space



Just $1 + o(1)$ bits per unitig.

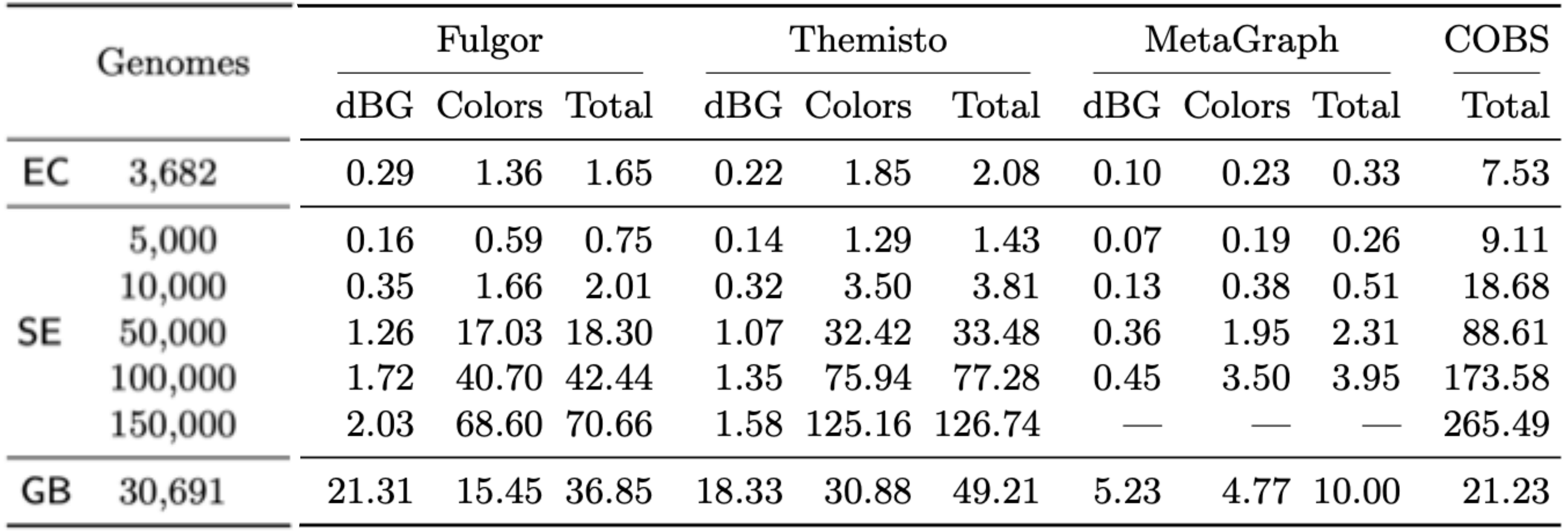
Results

- Results on some large pangenomes of different complexities.
- We implemented the method in a tool called **Fulgor** [Fan et al., 2023]: https://github.com/jermp/fulgor/releases/tag/v1.0.0



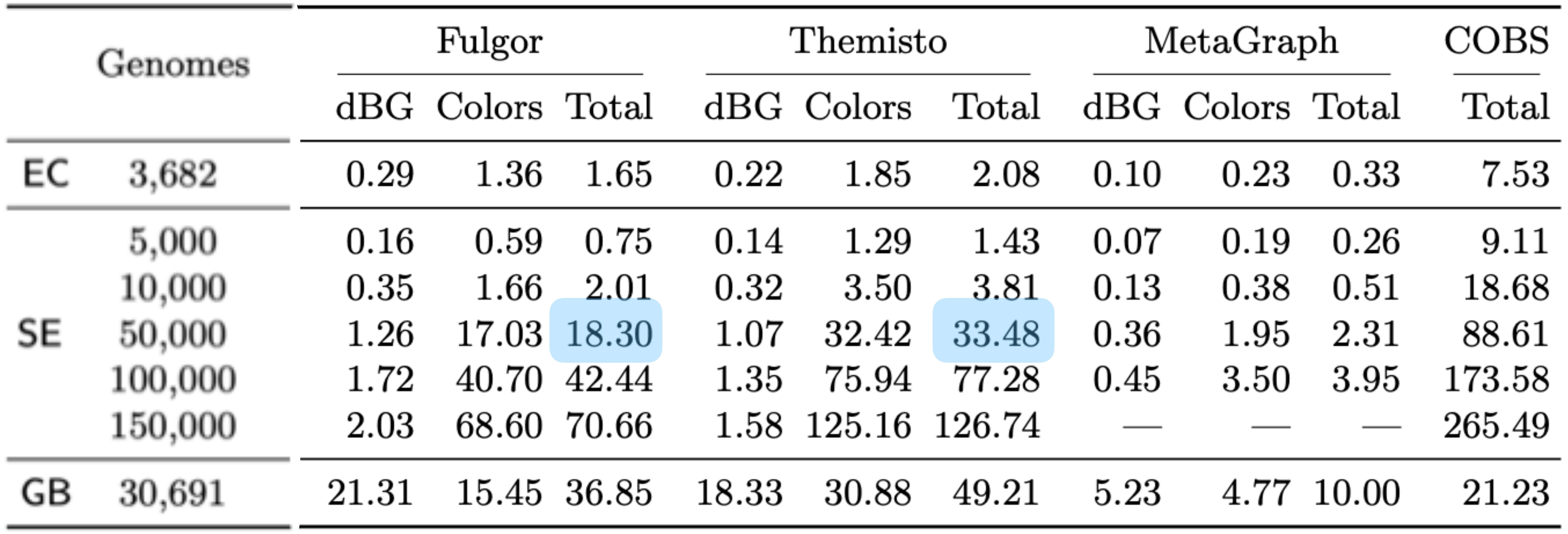
	E. Coli (EC)	*S. Enterica* (SE)					Gut bacteria (GB)
Genomes	3,682	5,000	10,000	50,000	100,000	150,000	30,691
Distinct colors ($\times 10^6$)	5.59	2.69	4.24	13.92	19.36	23.61	227.80
Integers in colors ($\times 10^9$)	5.74	5.77	15.68	133.49	303.53	490.04	10.04
k-mers in dBG ($\times 10^6$)	170.65	104.69	239.88	806.23	1,018.69	1,194.44	13,936.86
Unitigs in dBG ($\times 10^6$)	9.31	4.95	8.24	30.64	41.16	49.60	566.39

Index space in GB



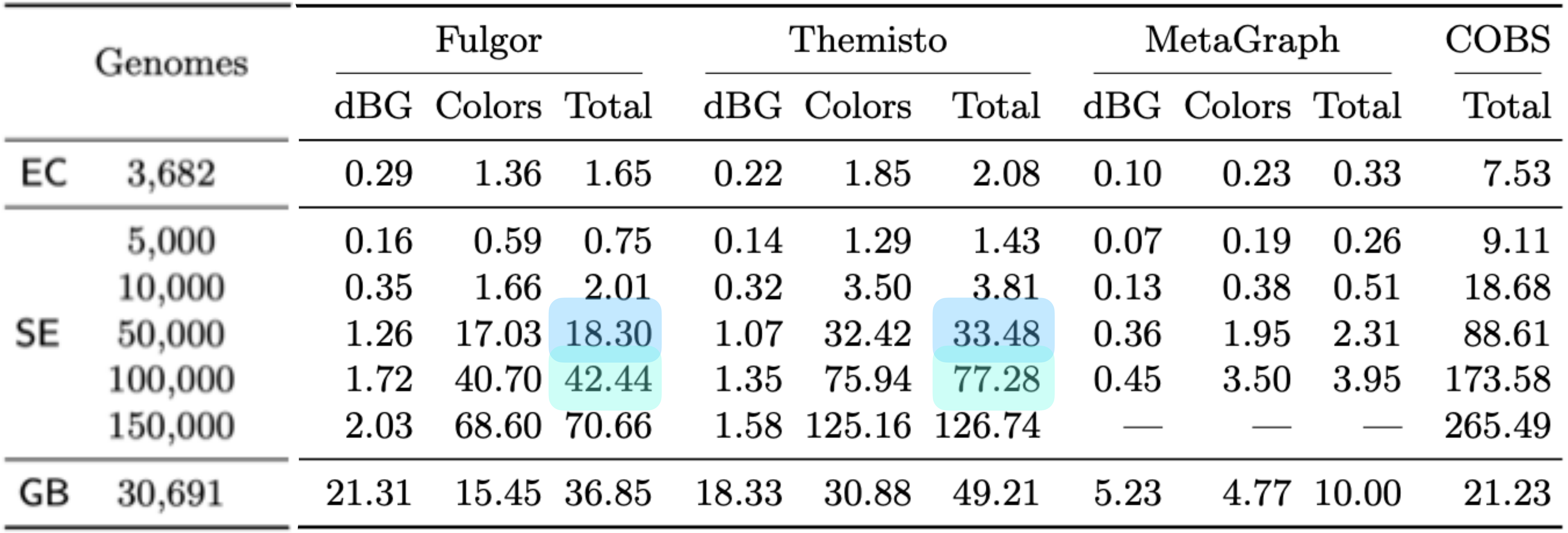
Genomes		Fulgor			Themisto			MetaGraph			COBS
		dBG	Colors	Total	dBG	Colors	Total	dBG	Colors	Total	Total
EC	3,682	0.29	1.36	1.65	0.22	1.85	2.08	0.10	0.23	0.33	7.53
SE	5,000	0.16	0.59	0.75	0.14	1.29	1.43	0.07	0.19	0.26	9.11
	10,000	0.35	1.66	2.01	0.32	3.50	3.81	0.13	0.38	0.51	18.68
	50,000	1.26	17.03	18.30	1.07	32.42	33.48	0.36	1.95	2.31	88.61
	100,000	1.72	40.70	42.44	1.35	75.94	77.28	0.45	3.50	3.95	173.58
	150,000	2.03	68.60	70.66	1.58	125.16	126.74	—	—	—	265.49
GB	30,691	21.31	15.45	36.85	18.33	30.88	49.21	5.23	4.77	10.00	21.23

Index space in GB



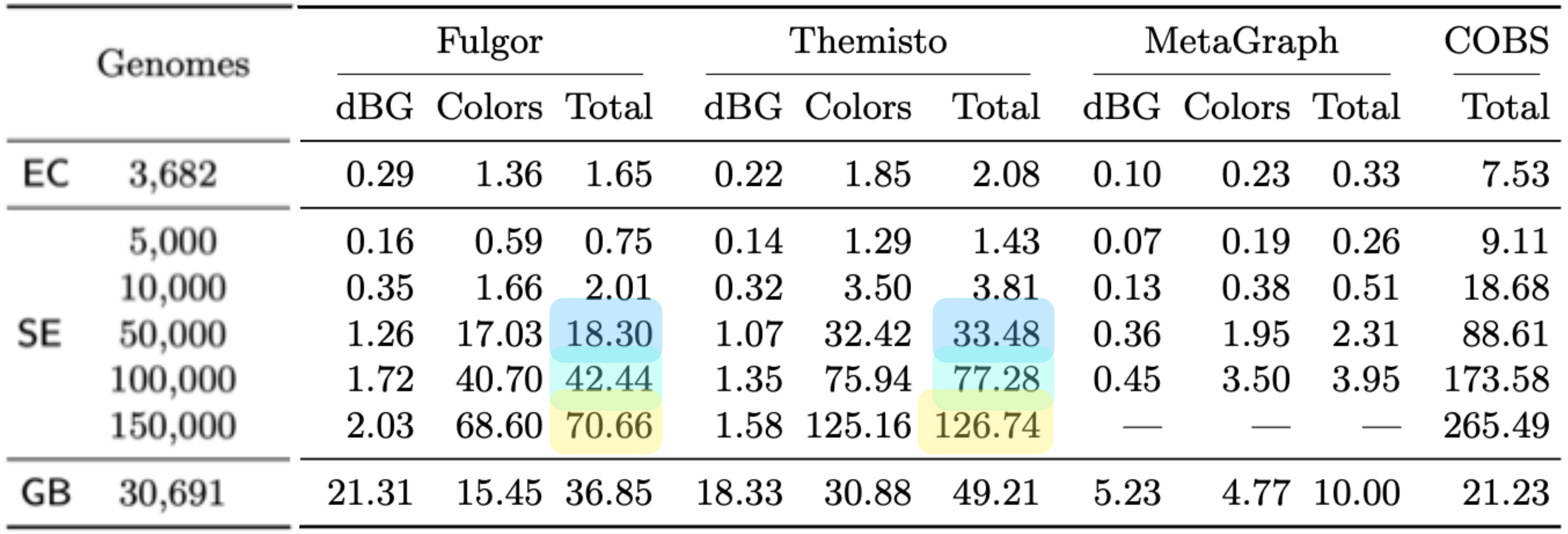
Genomes		Fulgor			Themisto			MetaGraph			COBS
		dBG	Colors	Total	dBG	Colors	Total	dBG	Colors	Total	Total
EC	3,682	0.29	1.36	1.65	0.22	1.85	2.08	0.10	0.23	0.33	7.53
SE	5,000	0.16	0.59	0.75	0.14	1.29	1.43	0.07	0.19	0.26	9.11
	10,000	0.35	1.66	2.01	0.32	3.50	3.81	0.13	0.38	0.51	18.68
	50,000	1.26	17.03	18.30	1.07	32.42	33.48	0.36	1.95	2.31	88.61
	100,000	1.72	40.70	42.44	1.35	75.94	77.28	0.45	3.50	3.95	173.58
	150,000	2.03	68.60	70.66	1.58	125.16	126.74	—	—	—	265.49
GB	30,691	21.31	15.45	36.85	18.33	30.88	49.21	5.23	4.77	10.00	21.23

Index space in GB



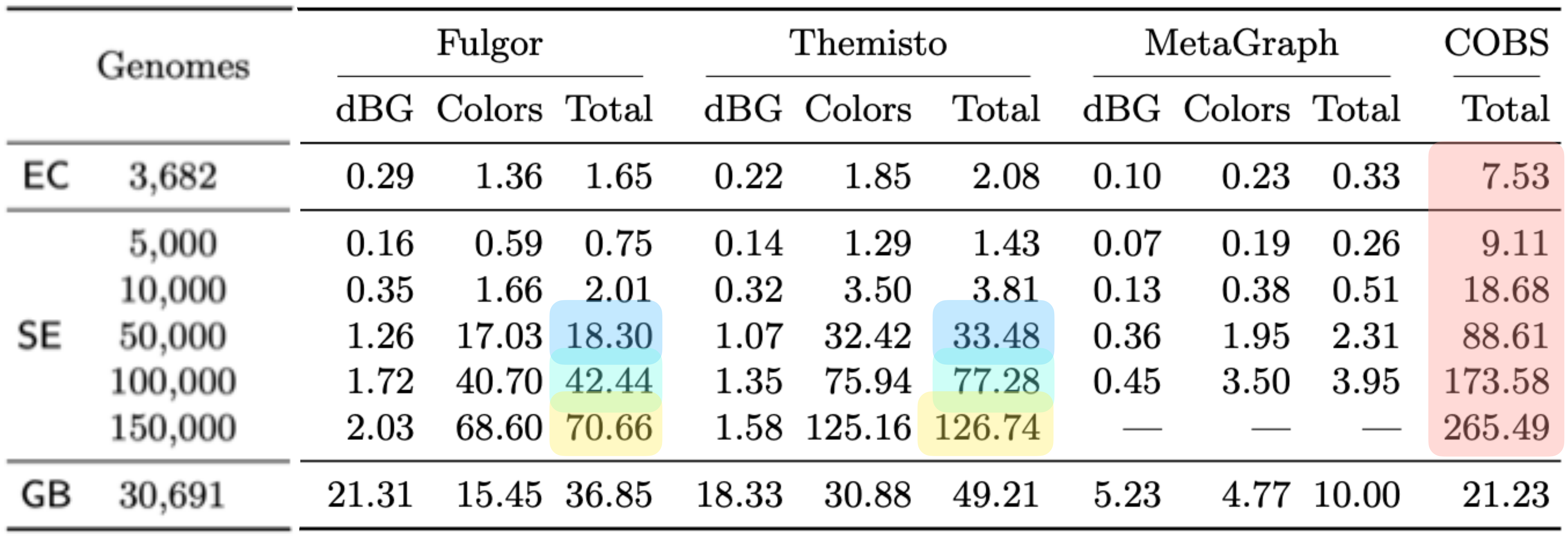
Genomes		Fulgor			Themisto			MetaGraph			COBS
		dBG	Colors	Total	dBG	Colors	Total	dBG	Colors	Total	Total
EC	3,682	0.29	1.36	1.65	0.22	1.85	2.08	0.10	0.23	0.33	7.53
SE	5,000	0.16	0.59	0.75	0.14	1.29	1.43	0.07	0.19	0.26	9.11
	10,000	0.35	1.66	2.01	0.32	3.50	3.81	0.13	0.38	0.51	18.68
	50,000	1.26	17.03	18.30	1.07	32.42	33.48	0.36	1.95	2.31	88.61
	100,000	1.72	40.70	42.44	1.35	75.94	77.28	0.45	3.50	3.95	173.58
	150,000	2.03	68.60	70.66	1.58	125.16	126.74	—	—	—	265.49
GB	30,691	21.31	15.45	36.85	18.33	30.88	49.21	5.23	4.77	10.00	21.23

Index space in GB



Genomes		Fulgor			Themisto			MetaGraph			COBS
		dBG	Colors	Total	dBG	Colors	Total	dBG	Colors	Total	Total
EC	3,682	0.29	1.36	1.65	0.22	1.85	2.08	0.10	0.23	0.33	7.53
SE	5,000	0.16	0.59	0.75	0.14	1.29	1.43	0.07	0.19	0.26	9.11
	10,000	0.35	1.66	2.01	0.32	3.50	3.81	0.13	0.38	0.51	18.68
	50,000	1.26	17.03	18.30	1.07	32.42	33.48	0.36	1.95	2.31	88.61
	100,000	1.72	40.70	42.44	1.35	75.94	77.28	0.45	3.50	3.95	173.58
	150,000	2.03	68.60	70.66	1.58	125.16	126.74	—	—	—	265.49
GB	30,691	21.31	15.45	36.85	18.33	30.88	49.21	5.23	4.77	10.00	21.23

Index space in GB

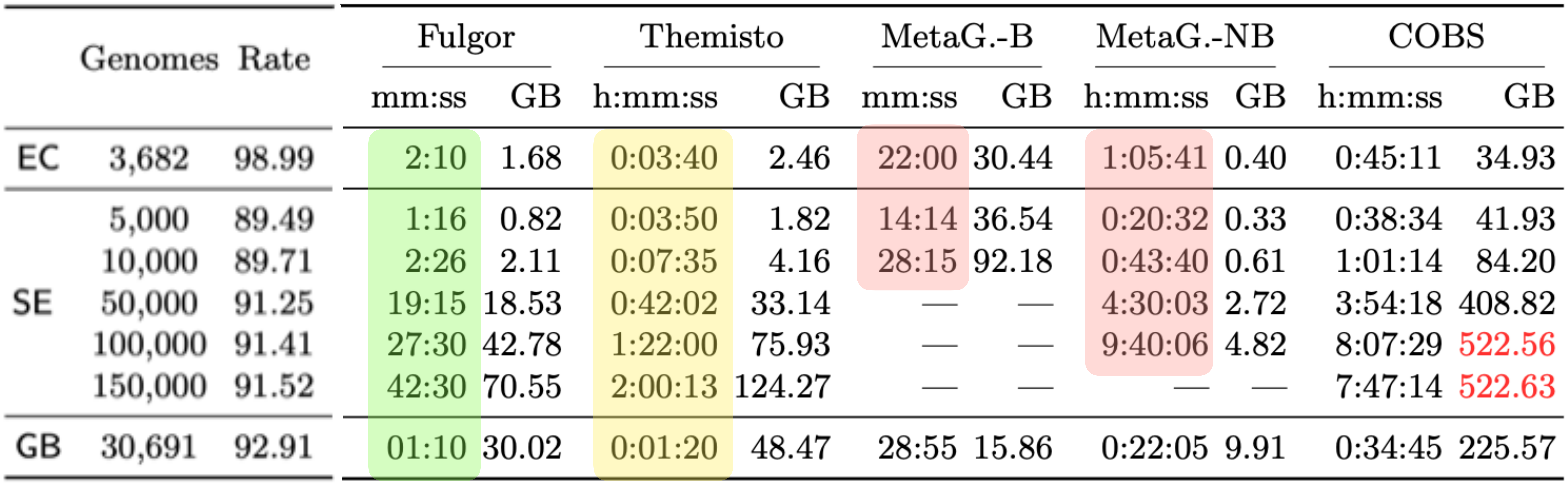


Genomes		Fulgor			Themisto			MetaGraph			COBS
		dBG	Colors	Total	dBG	Colors	Total	dBG	Colors	Total	Total
EC	3,682	0.29	1.36	1.65	0.22	1.85	2.08	0.10	0.23	0.33	7.53
SE	5,000	0.16	0.59	0.75	0.14	1.29	1.43	0.07	0.19	0.26	9.11
	10,000	0.35	1.66	2.01	0.32	3.50	3.81	0.13	0.38	0.51	18.68
	50,000	1.26	17.03	18.30	1.07	32.42	33.48	0.36	1.95	2.31	88.61
	100,000	1.72	40.70	42.44	1.35	75.94	77.28	0.45	3.50	3.95	173.58
	150,000	2.03	68.60	70.66	1.58	125.16	126.74	—	—	—	265.49
GB	30,691	21.31	15.45	36.85	18.33	30.88	49.21	5.23	4.77	10.00	21.23

Index space in GB

Genomes		Fulgor			Themisto			MetaGraph			COBS
		dBG	Colors	Total	dBG	Colors	Total	dBG	Colors	Total	Total
EC	3,682	0.29	1.36	1.65	0.22	1.85	2.08	0.10	0.23	0.33	7.53
SE	5,000	0.16	0.59	0.75	0.14	1.29	1.43	0.07	0.19	0.26	9.11
	10,000	0.35	1.66	2.01	0.32	3.50	3.81	0.13	0.38	0.51	18.68
	50,000	1.26	17.03	18.30	1.07	32.42	33.48	0.36	1.95	2.31	88.61
	100,000	1.72	40.70	42.44	1.35	75.94	77.28	0.45	3.50	3.95	173.58
	150,000	2.03	68.60	70.66	1.58	125.16	126.74	—	—	—	265.49
GB	30,691	21.31	15.45	36.85	18.33	30.88	49.21	5.23	4.77	10.00	21.23

Pseudoalignment efficiency



	Genomes	Rate	Fulgor		Themisto		MetaG.-B		MetaG.-NB		COBS	
			mm:ss	GB	h:mm:ss	GB	mm:ss	GB	h:mm:ss	GB	h:mm:ss	GB
EC	3,682	98.99	2:10	1.68	0:03:40	2.46	22:00	30.44	1:05:41	0.40	0:45:11	34.93
SE	5,000	89.49	1:16	0.82	0:03:50	1.82	14:14	36.54	0:20:32	0.33	0:38:34	41.93
	10,000	89.71	2:26	2.11	0:07:35	4.16	28:15	92.18	0:43:40	0.61	1:01:14	84.20
	50,000	91.25	19:15	18.53	0:42:02	33.14	—	—	4:30:03	2.72	3:54:18	408.82
	100,000	91.41	27:30	42.78	1:22:00	75.93	—	—	9:40:06	4.82	8:07:29	522.56
	150,000	91.52	42:30	70.55	2:00:13	124.27	—	—	—	—	7:47:14	522.63
GB	30,691	92.91	01:10	30.02	0:01:20	48.47	28:55	15.86	0:22:05	9.91	0:34:45	225.57

Yet another property!

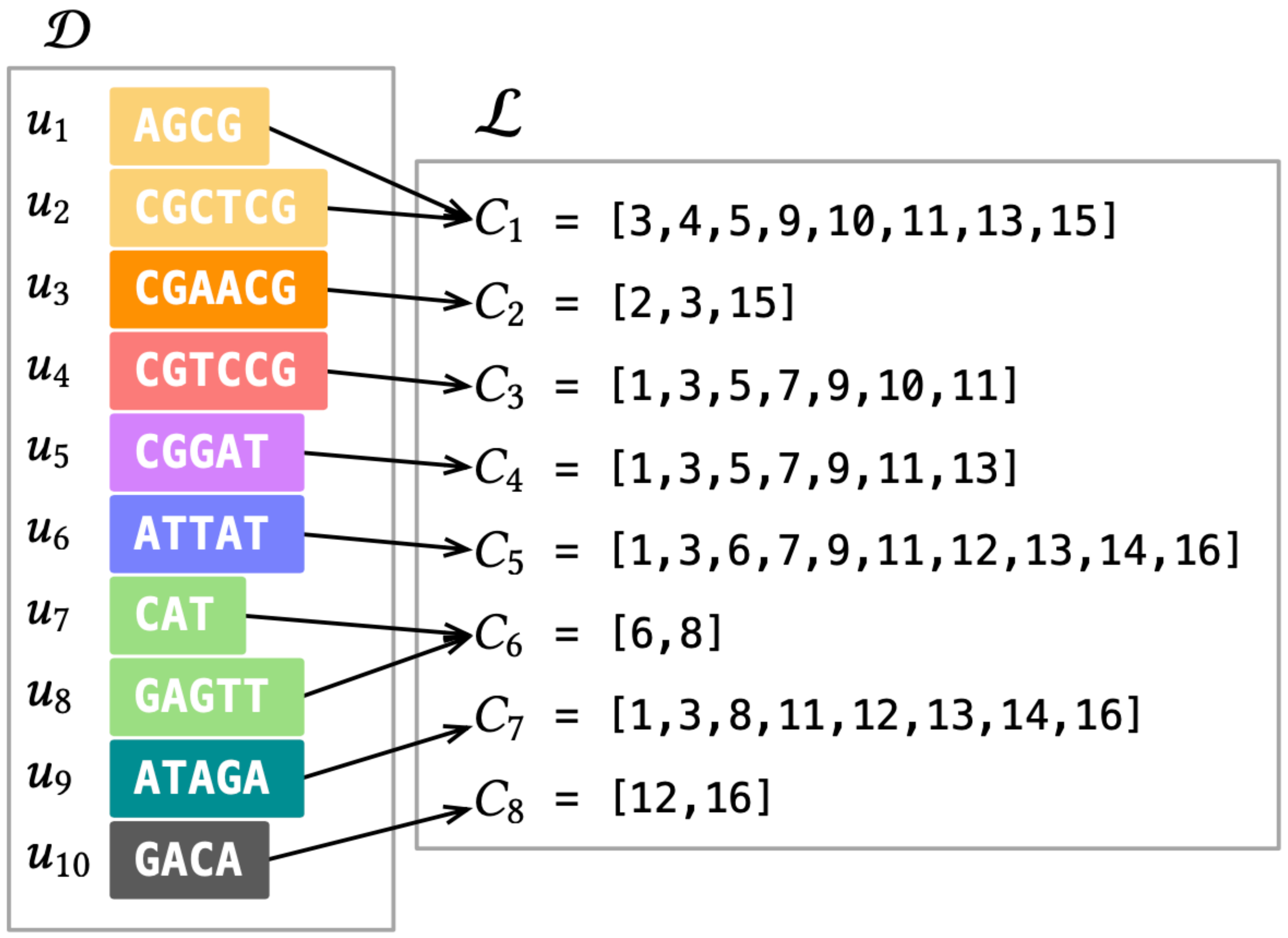
1. **Unitigs spell references in $\mathscr{R}$.**

2. **Unitigs are monochromatic.**

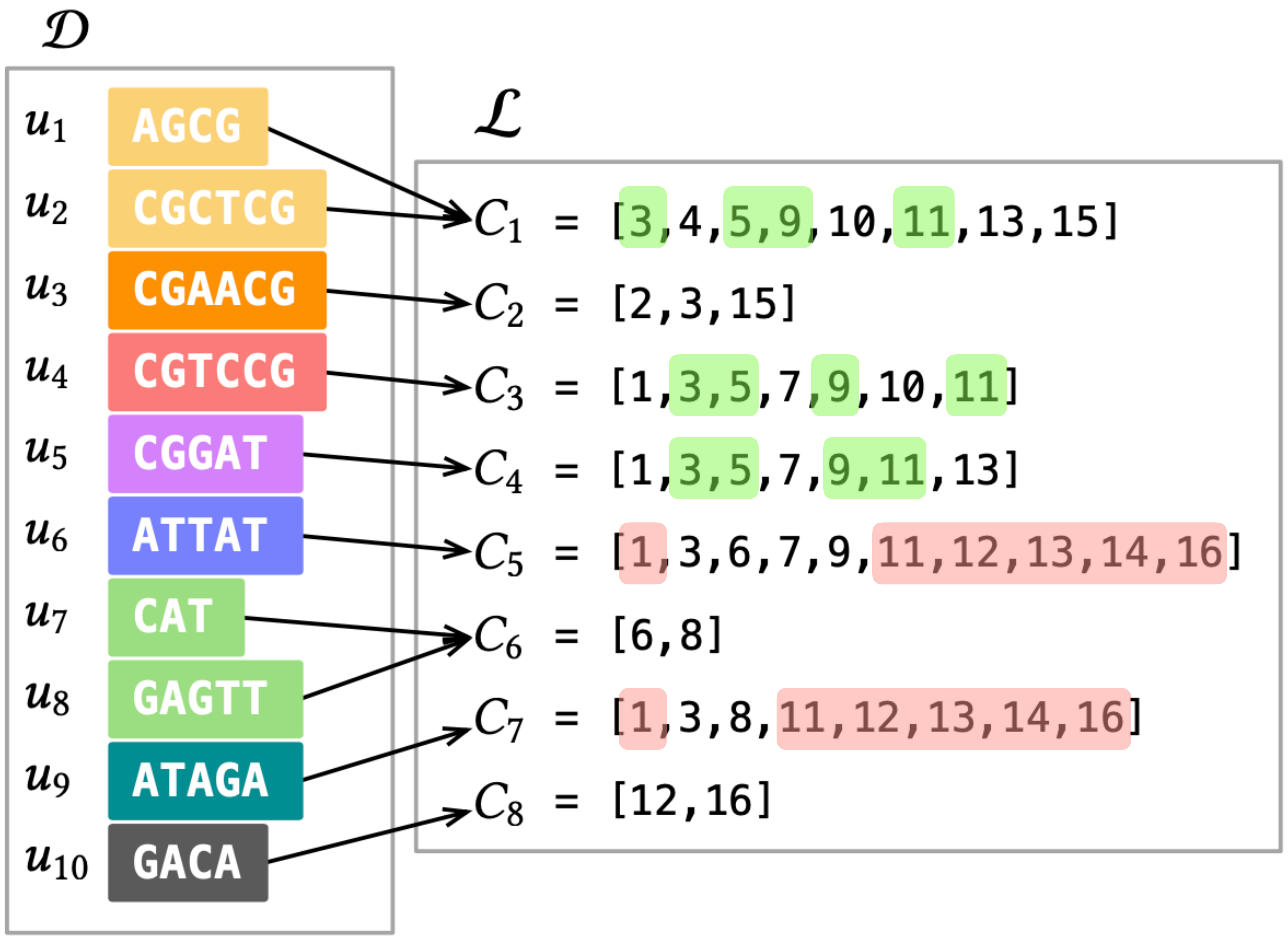
3. **Unitigs co-occur.**

4. **Colors are similar when indexing pangenomes.** → Opportunity to achieve **much better compression** if colors are not compressed *individually* (each set independently of the others) but *common patterns are factored out and compressed once*.

Colors are similar when indexing pangenomes

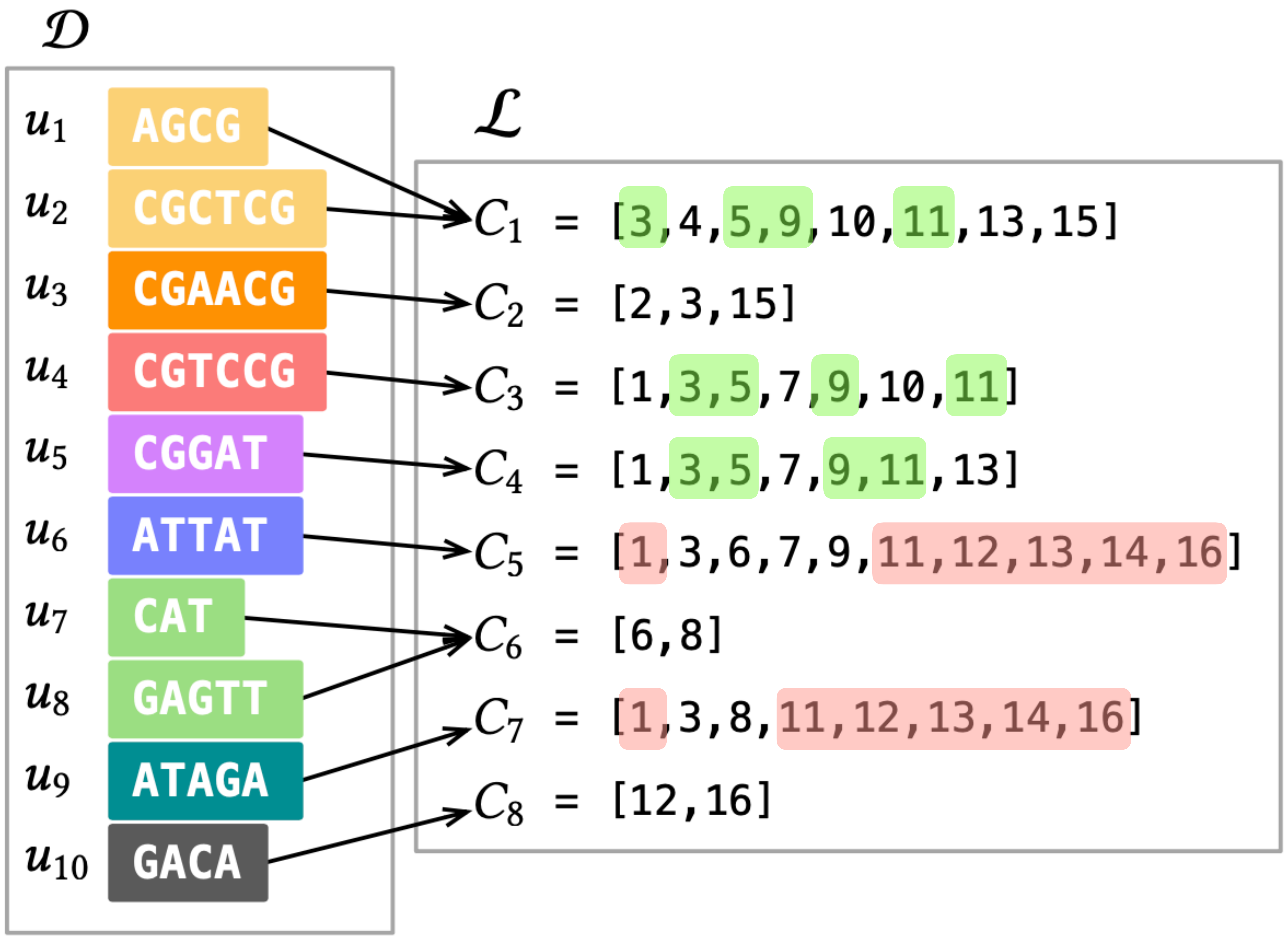


Colors are similar when indexing pangenomes



- The pattern $\{3,5,9,11\}$ is currently represented **three times**.
- The pattern $\{1,11,12,13,14,16\}$ is represented **twice**.

Colors are similar when indexing pangenomes



- The pattern $\{3,5,9,11\}$ is currently represented **three times**.
- The pattern $\{1,11,12,13,14,16\}$ is represented **twice**.
- **Q.** How to factor out this redundancy?

Introducing meta and partial colors

- Determine a **partition** of $[N] = \{1,\ldots,N\}$ so that references in the same partition are *similar*. Take the set of distinct **partial** colors for each partition → factor out the redundancy.
- **Intuition:** Similar references induce similar colors and thus *share patterns* → the number of distinct partial colors in a partition is small.
- Now we can render each original color as a sequence of references — or **meta** colors — to those **partial** colors.

Meta and partial colors — Example

- Example for **N = 16** references and **4** partitions.

$$\begin{array}{rl} & \{\ 1\ 12\ 13\ 14\ 16\ \}\{\ 3\ 5\ 9\ \}\{\ 7\ 11\ \}\{\ 2\ 4\ 6\ 8\ 10\ 15\ \} \\ \textit{new identifiers} \rightarrow & \ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\qquad 6\ 7\ 8\qquad 9\ 10\qquad 11\ 12\ 13\ 14\ 15\ 16 \end{array} \longleftarrow$$

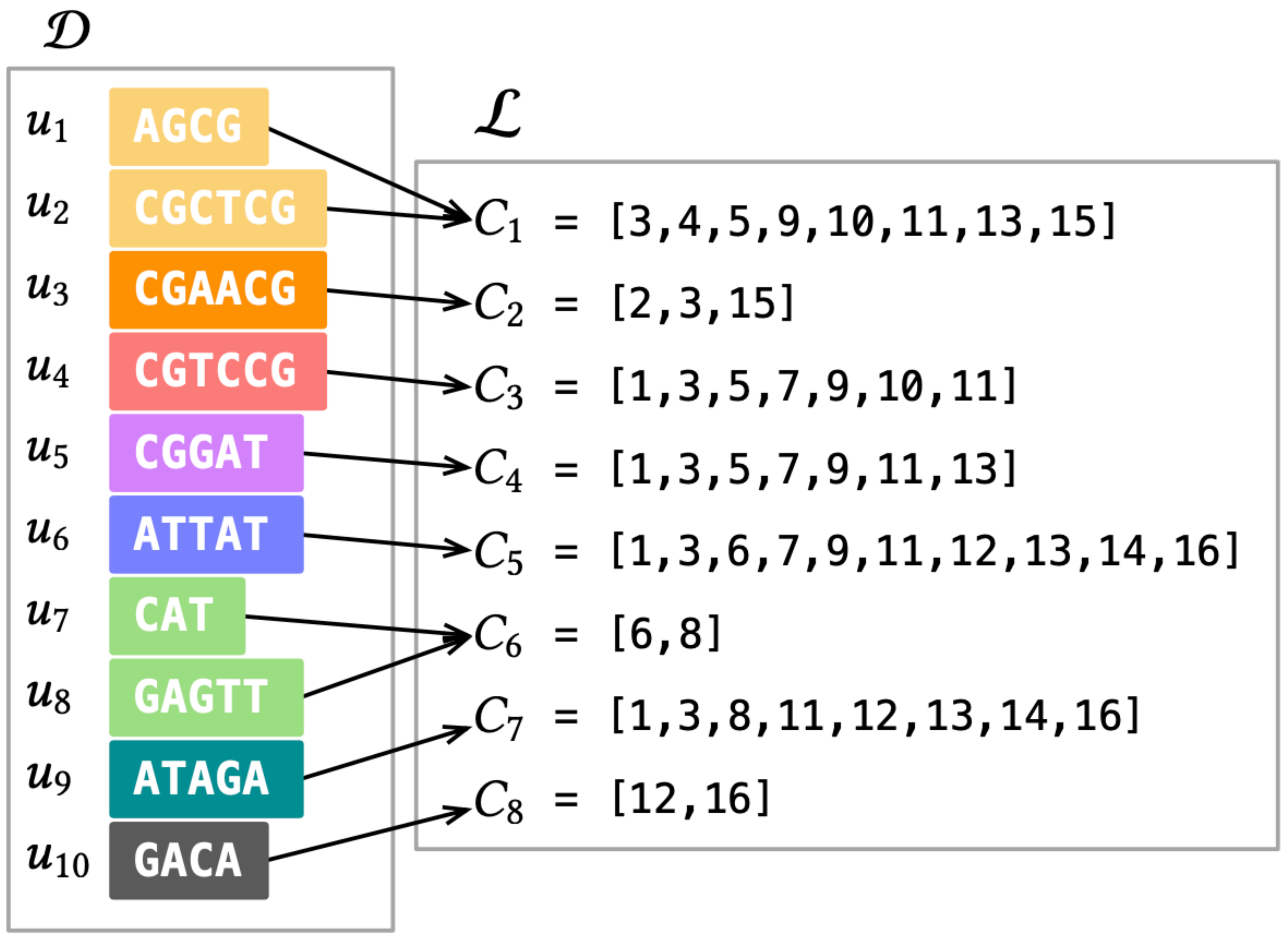
this defines a permutation π

Meta and partial colors — Example

- Example for **N = 16** references and **4** partitions.

$$\begin{array}{rccccccccccccccccccc} & \{ & 1 & 12 & 13 & 14 & 16 & \}\{ & 3 & 5 & 9 & \}\{ & 7 & 11 & \}\{ & 2 & 4 & 6 & 8 & 10 & 15 & \} \\ \textit{new identifiers} \rightarrow & & 1 & 2 & 3 & 4 & 5 & & 6 & 7 & 8 & & 9 & 10 & & 11 & 12 & 13 & 14 & 15 & 16 & \end{array}$$

$\longleftarrow$ this defines a permutation π



C_1 = [3,4,5,9,10,11,13,15]
C_2 = [2,3,15]
C_3 = [1,3,5,7,9,10,11]
C_4 = [1,3,5,7,9,11,13]
C_5 = [1,3,6,7,9,11,12,13,14,16]
C_6 = [6,8]
C_7 = [1,3,8,11,12,13,14,16]
C_8 = [12,16]

$\xrightarrow{\pi}$

$$C_1 = [3|6,7,8|10|12,15,16]$$
$$C_2 = [6|11,16]$$
$$C_3 = [1|6,7,8|9,10|15]$$
$$C_4 = [1,3|6,7,8|9,10]$$
$$C_5 = [1,2,3,4,5|6,8|9,10|13]$$
$$C_6 = [13,14]$$
$$C_7 = [1,2,3,4,5|6|10|14]$$
$$C_8 = [2,5]$$

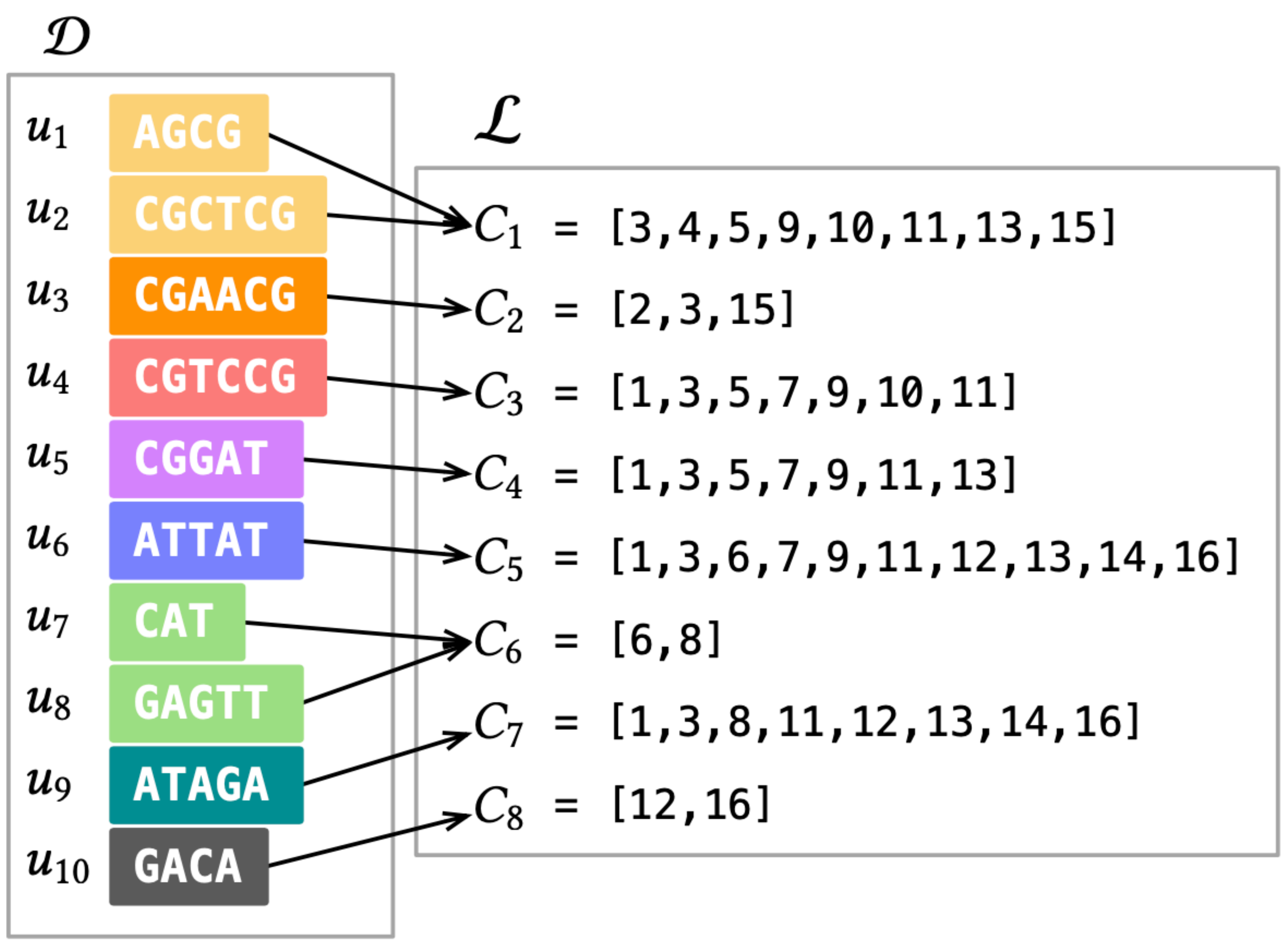
Meta and partial colors — Example

- Example for **N = 16** references and **4** partitions.

$$\begin{array}{r l}
 & \{\ 1\ 12\ 13\ 14\ 16\ \}\{\ 3\ 5\ 9\ \}\{\ 7\ 11\ \}\{\ 2\ \ 4\ \ 6\ \ 8\ \ 10\ 15\ \} \\
\textit{new identifiers} \rightarrow & 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ \ \boxed{6\ 7\ 8}\ \ \ 9\ 10\ \ \ 11\ 12\ 13\ 14\ 15\ 16
\end{array}$$

partition 2

this defines a permutation π



```
C1 = [3,4,5,9,10,11,13,15]
C2 = [2,3,15]
C3 = [1,3,5,7,9,10,11]
C4 = [1,3,5,7,9,11,13]
C5 = [1,3,6,7,9,11,12,13,14,16]
C6 = [6,8]
C7 = [1,3,8,11,12,13,14,16]
C8 = [12,16]
```

$\xrightarrow{\pi}$

$$C_1 = [3|6,7,8|10|12,15,16]$$
$$C_2 = [6|11,16]$$
$$C_3 = [1|6,7,8|9,10|15]$$
$$C_4 = [1,3|6,7,8|9,10]$$
$$C_5 = [1,2,3,4,5|6,8|9,10|13]$$
$$C_6 = [13,14]$$
$$C_7 = [1,2,3,4,5|6|10|14]$$
$$C_8 = [2,5]$$

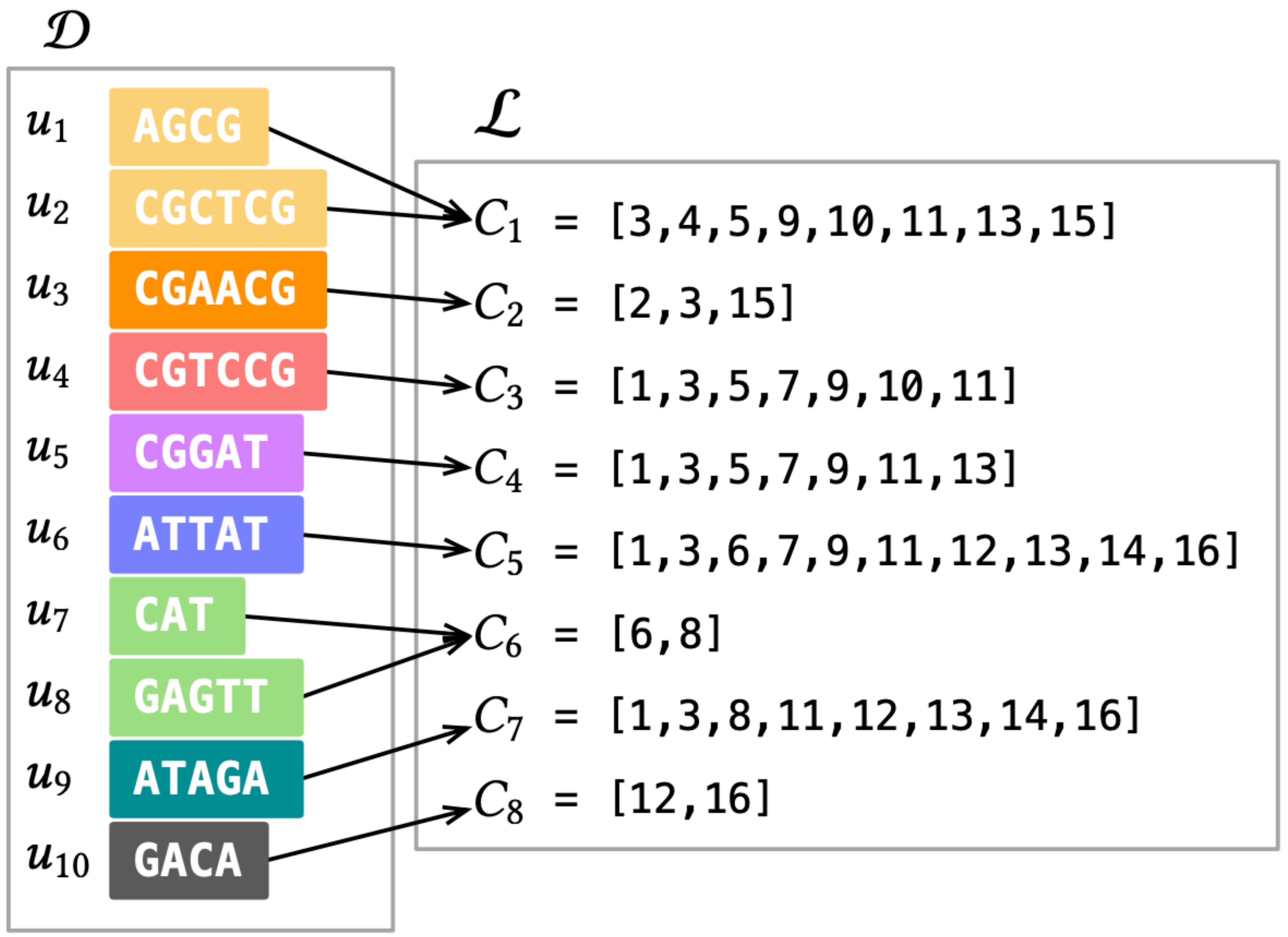
Meta and partial colors — Example

- Example for **N = 16** references and **4** partitions.

$$
\begin{array}{rcccc}
 & \{\ 1\ 12\ 13\ 14\ 16\ \} & \{\ 3\ 5\ 9\ \} & \{\ 7\ 11\ \} & \{\ 2\ \ 4\ \ 6\ \ 8\ \ 10\ 15\ \} \\
\textit{new identifiers} \rightarrow & 1\ \ 2\ \ 3\ \ 4\ \ 5 & \boxed{6\ 7\ 8} & 9\ 10 & 11\ 12\ 13\ 14\ 15\ 16
\end{array}
$$

partition 2

this defines a permutation π



C_1 = [3,4,5,9,10,11,13,15]
C_2 = [2,3,15]
C_3 = [1,3,5,7,9,10,11]
C_4 = [1,3,5,7,9,11,13]
C_5 = [1,3,6,7,9,11,12,13,14,16]
C_6 = [6,8]
C_7 = [1,3,8,11,12,13,14,16]
C_8 = [12,16]

$\xrightarrow{\pi}$

$$C_1 = [3|6,7,8|10|12,15,16]$$
$$C_2 = [6|11,16]$$
$$C_3 = [1|6,7,8|9,10|15]$$
$$C_4 = [1,3|6,7,8|9,10]$$
$$C_5 = [1,2,3,4,5|6,8|9,10|13]$$
$$C_6 = [13,14]$$
$$C_7 = [1,2,3,4,5|6|10|14]$$
$$C_8 = [2,5]$$

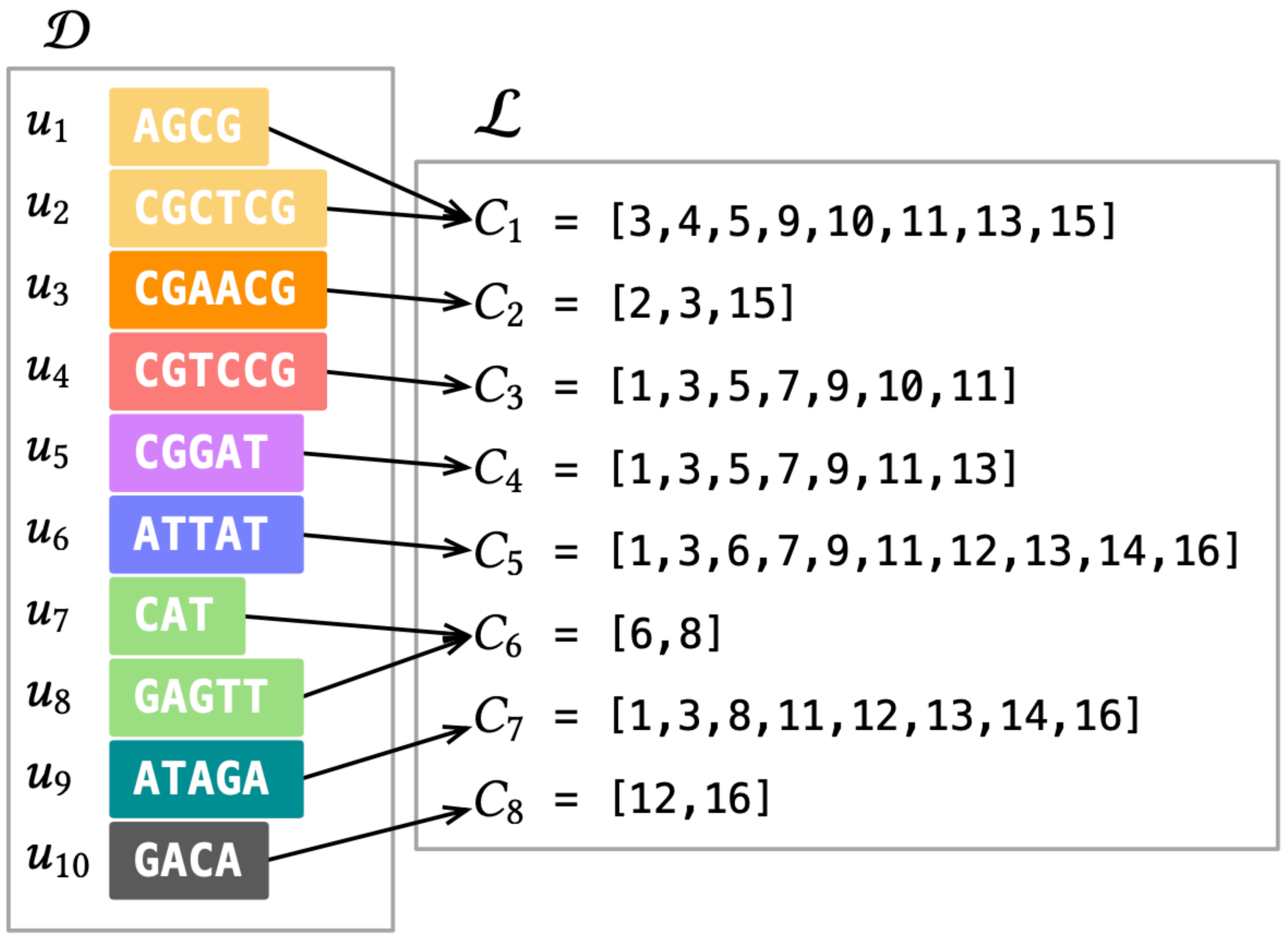
Meta and partial colors — Example

- Example for **N = 16** references and **4** partitions.

$$\{\ 1\ 12\ 13\ 14\ 16\ \}\{\ 3\ 5\ 9\ \}\{\ 7\ 11\ \}\{\ 2\ \ 4\ \ 6\ \ 8\ \ 10\ 15\ \}$$

$$\textit{new identifiers} \rightarrow 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ \ \underbrace{\boxed{6\ 7\ 8}}_{\text{partition 2}}\ \ \ 9\ 10\ \ \ 11\ 12\ 13\ 14\ 15\ 16$$

← this defines a permutation π



C_1 = [3,4,5,9,10,11,13,15]
C_2 = [2,3,15]
C_3 = [1,3,5,7,9,10,11]
C_4 = [1,3,5,7,9,11,13]
C_5 = [1,3,6,7,9,11,12,13,14,16]
C_6 = [6,8]
C_7 = [1,3,8,11,12,13,14,16]
C_8 = [12,16]

$\xrightarrow{\pi}$

$$C_1 = [3|6,7,8|10|12,15,16]$$
$$C_2 = [6|11,16]$$
$$C_3 = [1|6,7,8|9,10|15]$$
$$C_4 = [1,3|6,7,8|9,10]$$
$$C_5 = [1,2,3,4,5|6,8|9,10|13]$$
$$C_6 = [13,14]$$
$$C_7 = [1,2,3,4,5|6|10|14]$$
$$C_8 = [2,5]$$

[6,7,8]
[6]
[6,8]

Meta and partial colors — Example

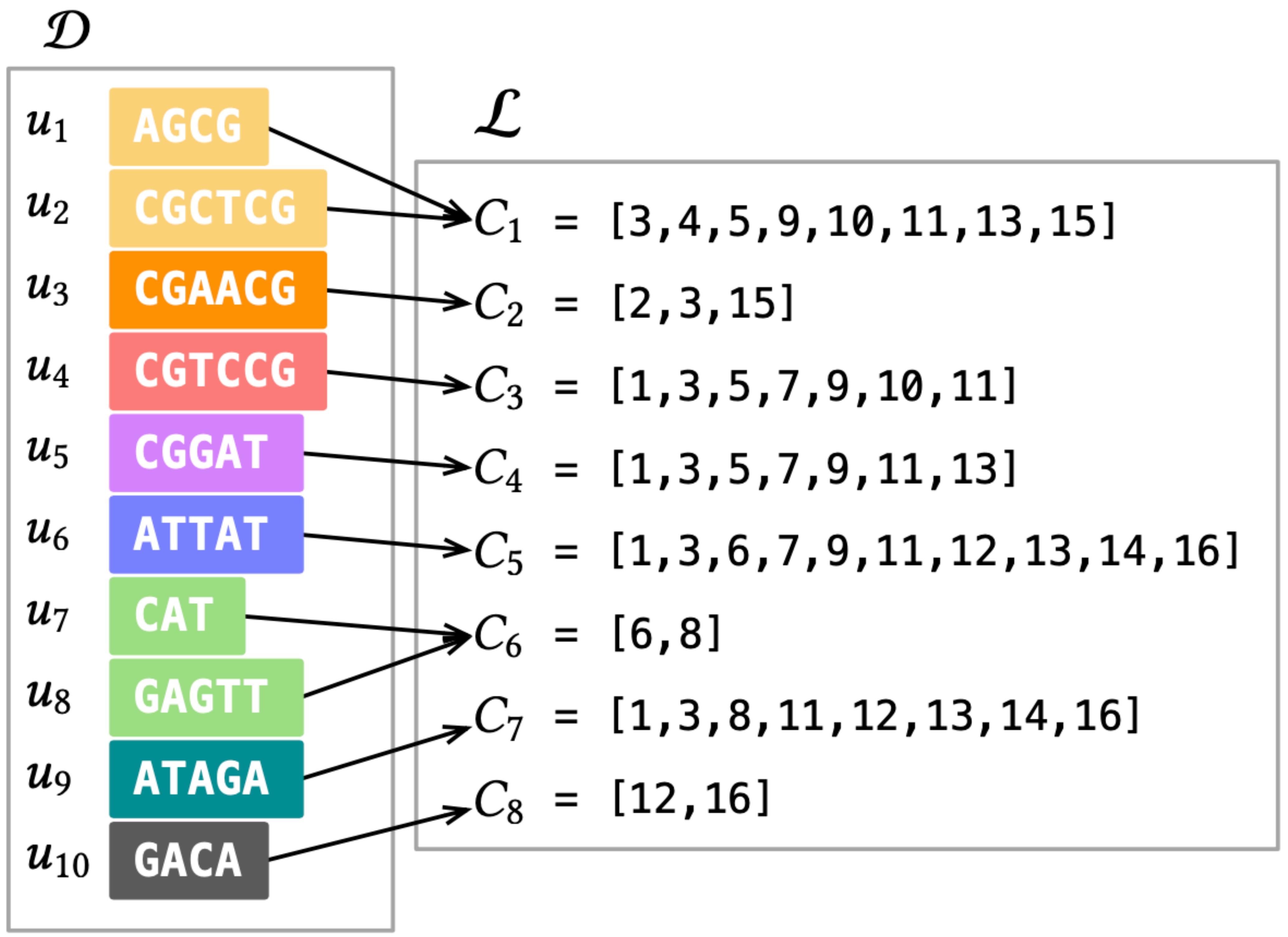
- Example for **N = 16** references and **4** partitions.

$$\{\ 1\ 12\ 13\ 14\ 16\ \}\{\ 3\ 5\ 9\ \}\{\ 7\ 11\ \}\{\ 2\ \ 4\ \ 6\ \ 8\ \ 10\ 15\ \}$$

$$\textit{new identifiers} \rightarrow 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ \boxed{6\ 7\ 8}\ \ 9\ 10\ \ 11\ 12\ 13\ 14\ 15\ 16$$

partition 2

$\leftarrow$ this defines a permutation π



C_1 = [3,4,5,9,10,11,13,15]
C_2 = [2,3,15]
C_3 = [1,3,5,7,9,10,11]
C_4 = [1,3,5,7,9,11,13]
C_5 = [1,3,6,7,9,11,12,13,14,16]
C_6 = [6,8]
C_7 = [1,3,8,11,12,13,14,16]
C_8 = [12,16]

$\xrightarrow{\pi}$

$$C_1 = [3|6,7,8|10|12,15,16]$$
$$C_2 = [6|11,16]$$
$$C_3 = [1|6,7,8|9,10|15]$$
$$C_4 = [1,3|6,7,8|9,10]$$
$$C_5 = [1,2,3,4,5|6,8|9,10|13]$$
$$C_6 = [13,14]$$
$$C_7 = [1,2,3,4,5|6|10|14]$$
$$C_8 = [2,5]$$

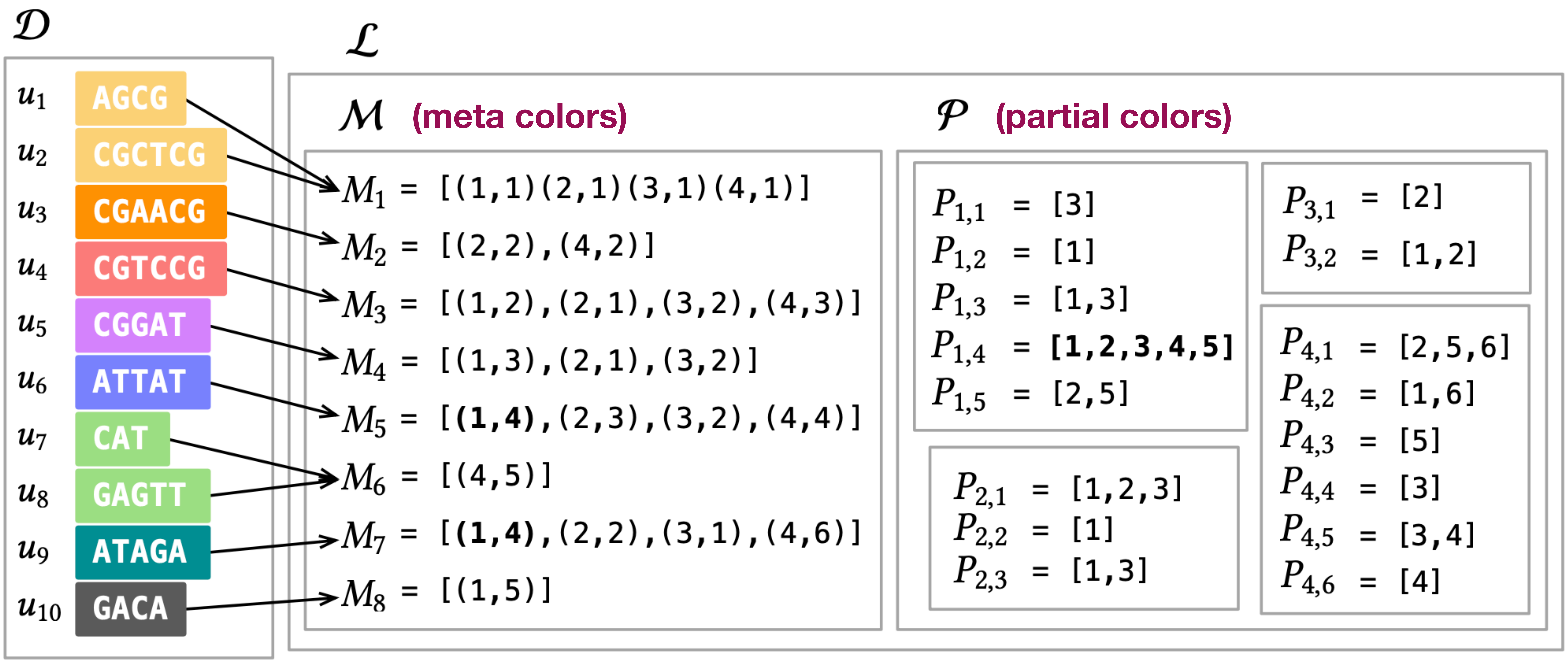
[6,7,8]
[6]
[6,8]

$\downarrow$ -5

[1,2,3]
[1]
[1,3]

Meta and partial colors — Example

- Example for **N = 16** references and **4** partitions.



M_1 = [(1,1)(2,1)(3,1)(4,1)]
M_2 = [(2,2),(4,2)]
M_3 = [(1,2),(2,1),(3,2),(4,3)]
M_4 = [(1,3),(2,1),(3,2)]
M_5 = [**(1,4)**,(2,3),(3,2),(4,4)]
M_6 = [(4,5)]
M_7 = [**(1,4)**,(2,2),(3,1),(4,6)]
M_8 = [(1,5)]

$P_{1,1}$ = [3]
$P_{1,2}$ = [1]
$P_{1,3}$ = [1,3]
$P_{1,4}$ = **[1,2,3,4,5]**
$P_{1,5}$ = [2,5]

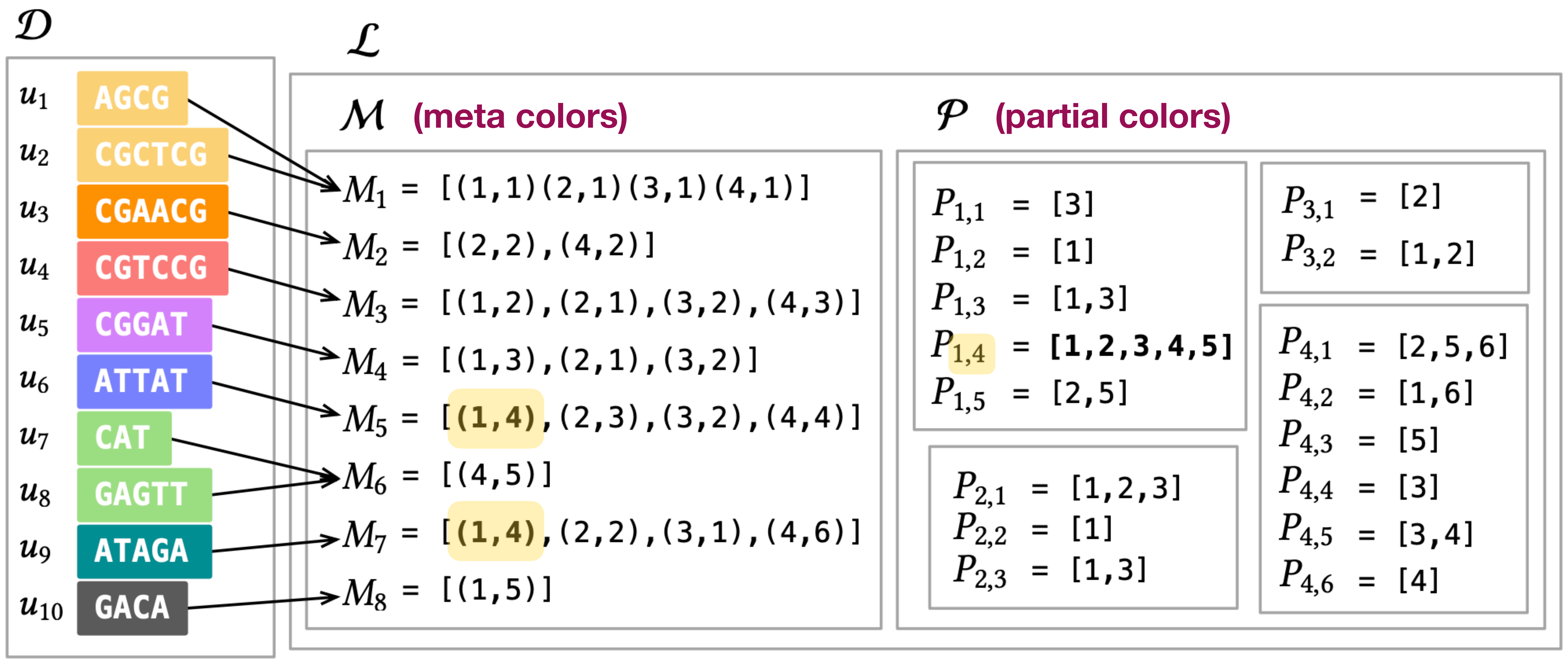
$P_{2,1}$ = [1,2,3]
$P_{2,2}$ = [1]
$P_{2,3}$ = [1,3]

$P_{3,1}$ = [2]
$P_{3,2}$ = [1,2]

$P_{4,1}$ = [2,5,6]
$P_{4,2}$ = [1,6]
$P_{4,3}$ = [5]
$P_{4,4}$ = [3]
$P_{4,5}$ = [3,4]
$P_{4,6}$ = [4]

Meta and partial colors — Example

- Example for **N = 16** references and **4** partitions.



M_1 = [(1,1)(2,1)(3,1)(4,1)]
M_2 = [(2,2),(4,2)]
M_3 = [(1,2),(2,1),(3,2),(4,3)]
M_4 = [(1,3),(2,1),(3,2)]
M_5 = [(1,4),(2,3),(3,2),(4,4)]
M_6 = [(4,5)]
M_7 = [(1,4),(2,2),(3,1),(4,6)]
M_8 = [(1,5)]

$P_{1,1}$ = [3]
$P_{1,2}$ = [1]
$P_{1,3}$ = [1,3]
$P_{1,4}$ = **[1,2,3,4,5]**
$P_{1,5}$ = [2,5]

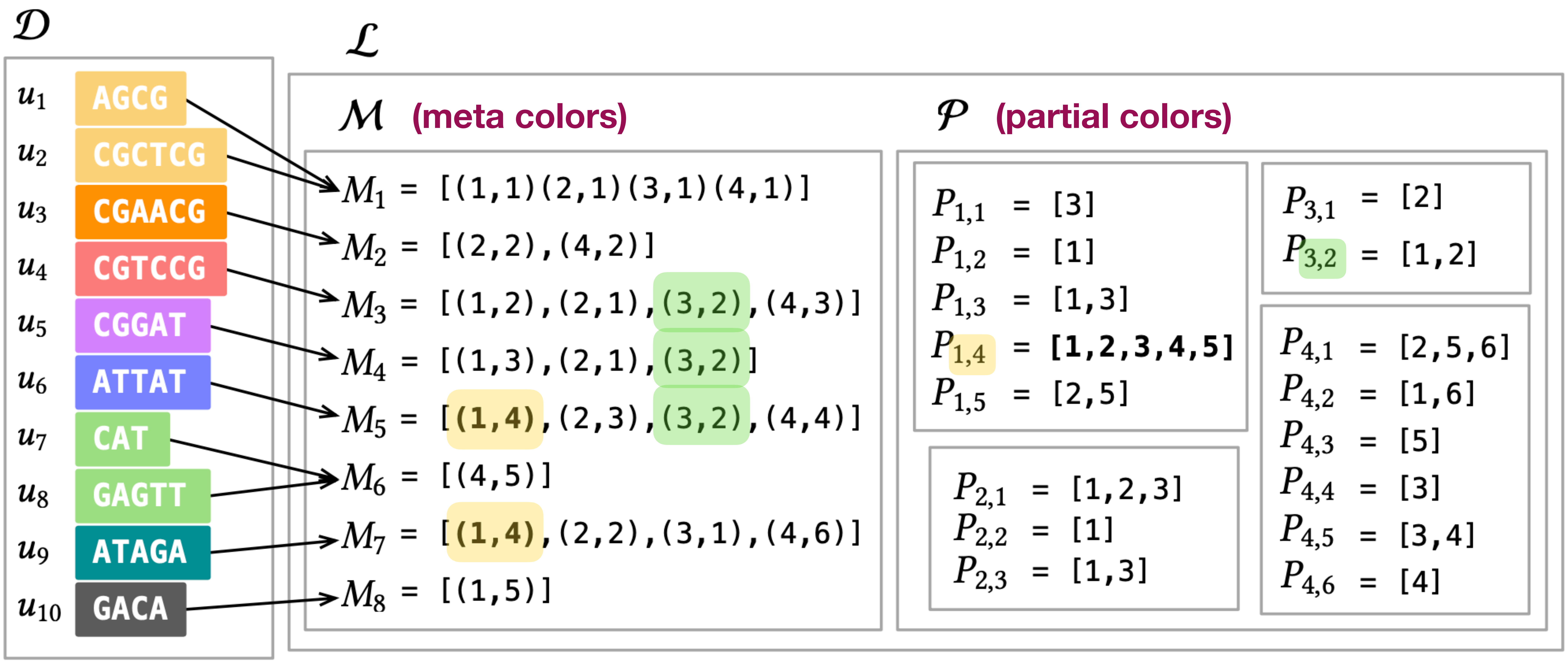
$P_{2,1}$ = [1,2,3]
$P_{2,2}$ = [1]
$P_{2,3}$ = [1,3]

$P_{3,1}$ = [2]
$P_{3,2}$ = [1,2]

$P_{4,1}$ = [2,5,6]
$P_{4,2}$ = [1,6]
$P_{4,3}$ = [5]
$P_{4,4}$ = [3]
$P_{4,5}$ = [3,4]
$P_{4,6}$ = [4]

Meta and partial colors — Example

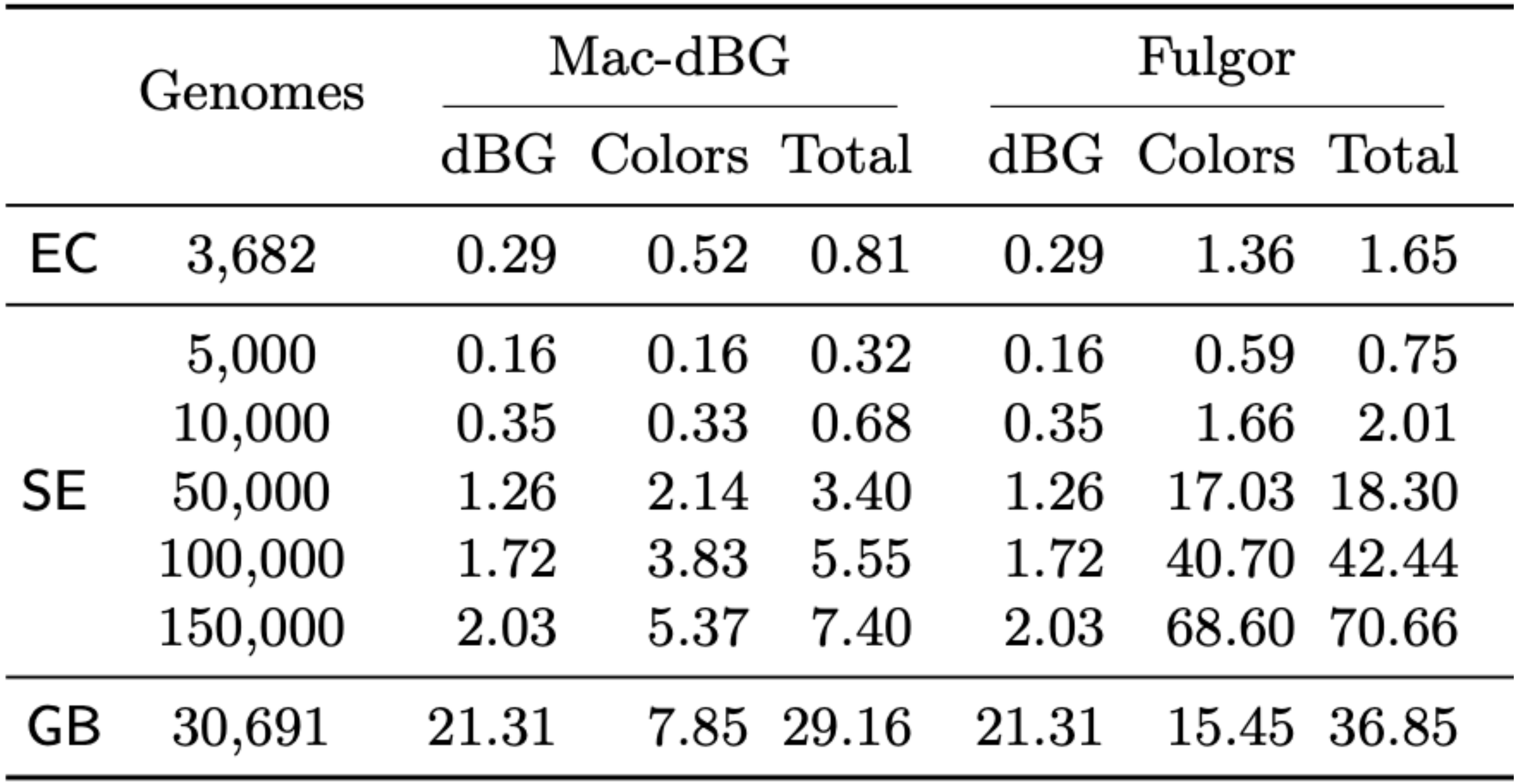
- Example for **N = 16** references and **4** partitions.



Results

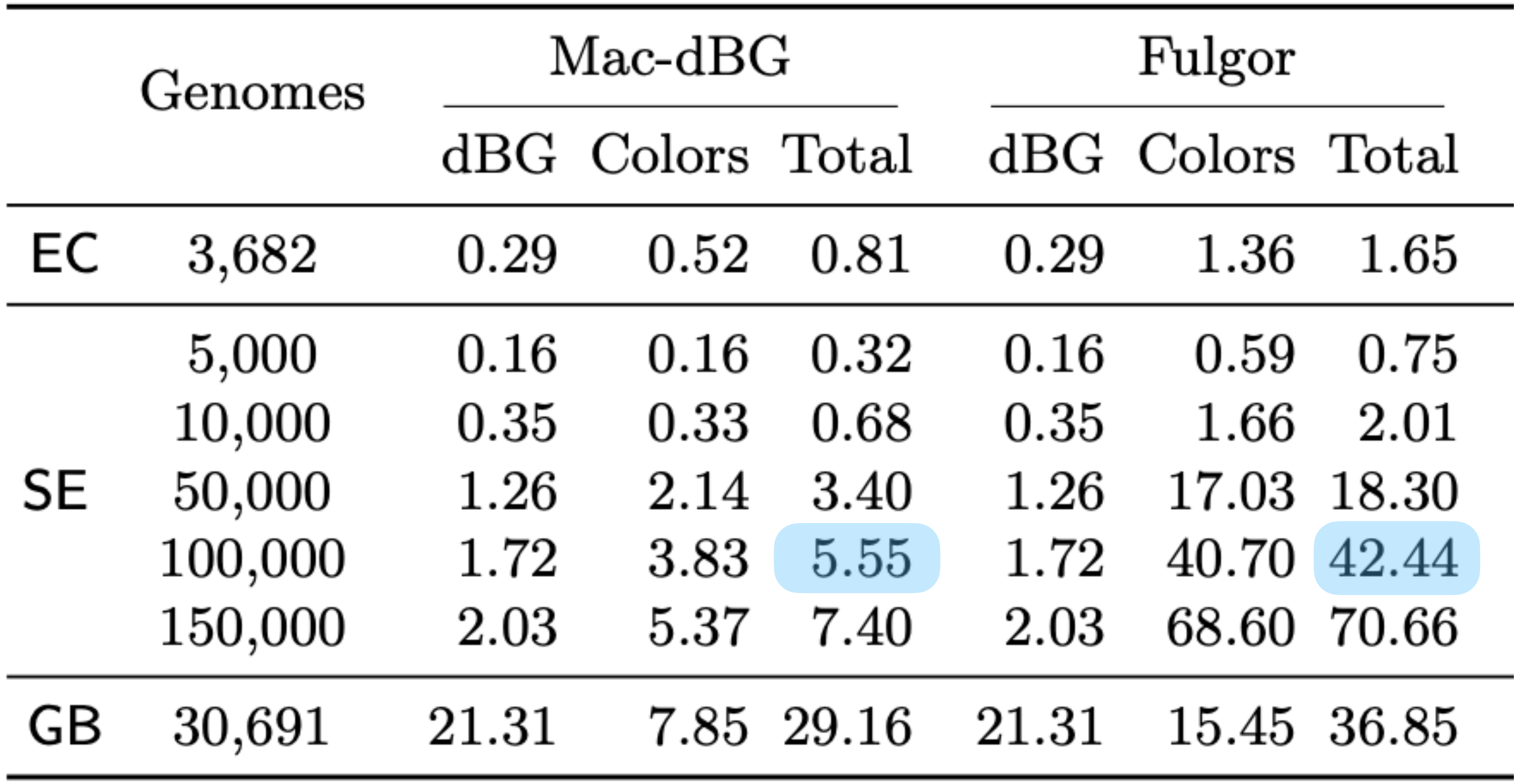
- We applied the meta/partial color optimisation to Fulgor.
- We call it the *meta-colored* compacted dBG (**Mac-dBG**, or **Fulgor-v2**).
- https://github.com/jermp/fulgor/releases/tag/v2.0.0

Space in GB



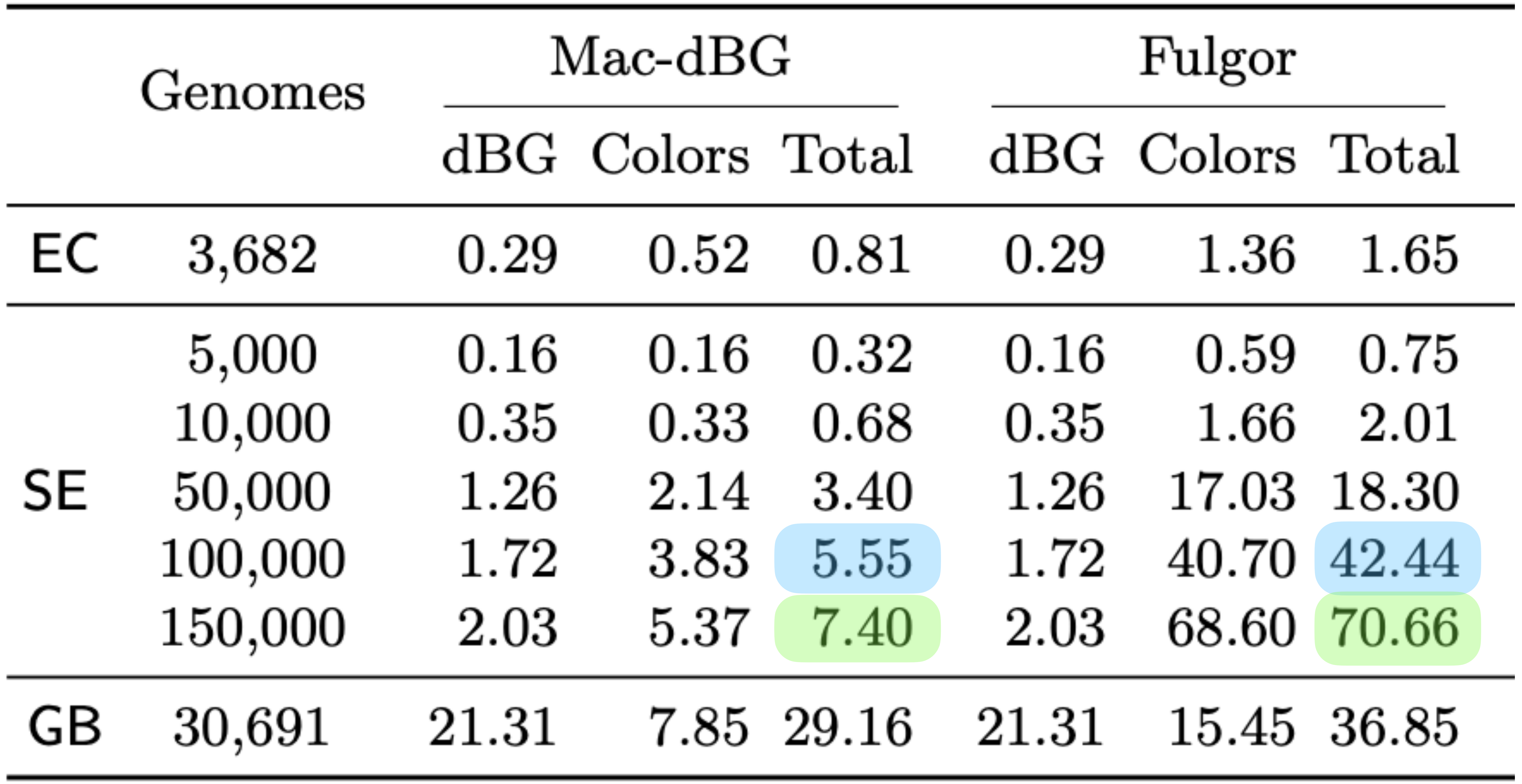
	Genomes	Mac-dBG			Fulgor		
		dBG	Colors	Total	dBG	Colors	Total
EC	3,682	0.29	0.52	0.81	0.29	1.36	1.65
SE	5,000	0.16	0.16	0.32	0.16	0.59	0.75
	10,000	0.35	0.33	0.68	0.35	1.66	2.01
	50,000	1.26	2.14	3.40	1.26	17.03	18.30
	100,000	1.72	3.83	5.55	1.72	40.70	42.44
	150,000	2.03	5.37	7.40	2.03	68.60	70.66
GB	30,691	21.31	7.85	29.16	21.31	15.45	36.85

Space in GB



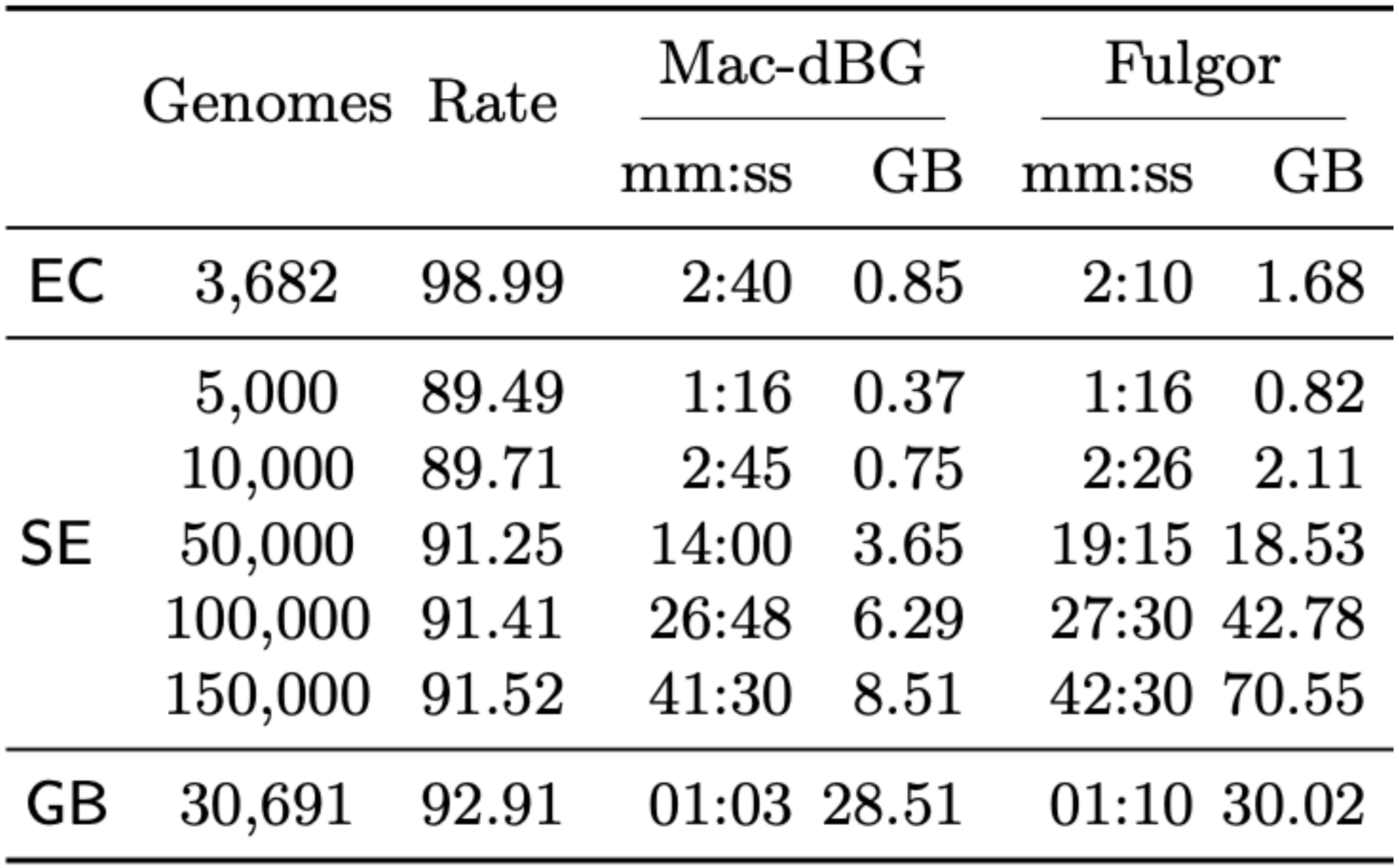
	Genomes	Mac-dBG			Fulgor		
		dBG	Colors	Total	dBG	Colors	Total
EC	3,682	0.29	0.52	0.81	0.29	1.36	1.65
SE	5,000	0.16	0.16	0.32	0.16	0.59	0.75
	10,000	0.35	0.33	0.68	0.35	1.66	2.01
	50,000	1.26	2.14	3.40	1.26	17.03	18.30
	100,000	1.72	3.83	5.55	1.72	40.70	42.44
	150,000	2.03	5.37	7.40	2.03	68.60	70.66
GB	30,691	21.31	7.85	29.16	21.31	15.45	36.85

Space in GB



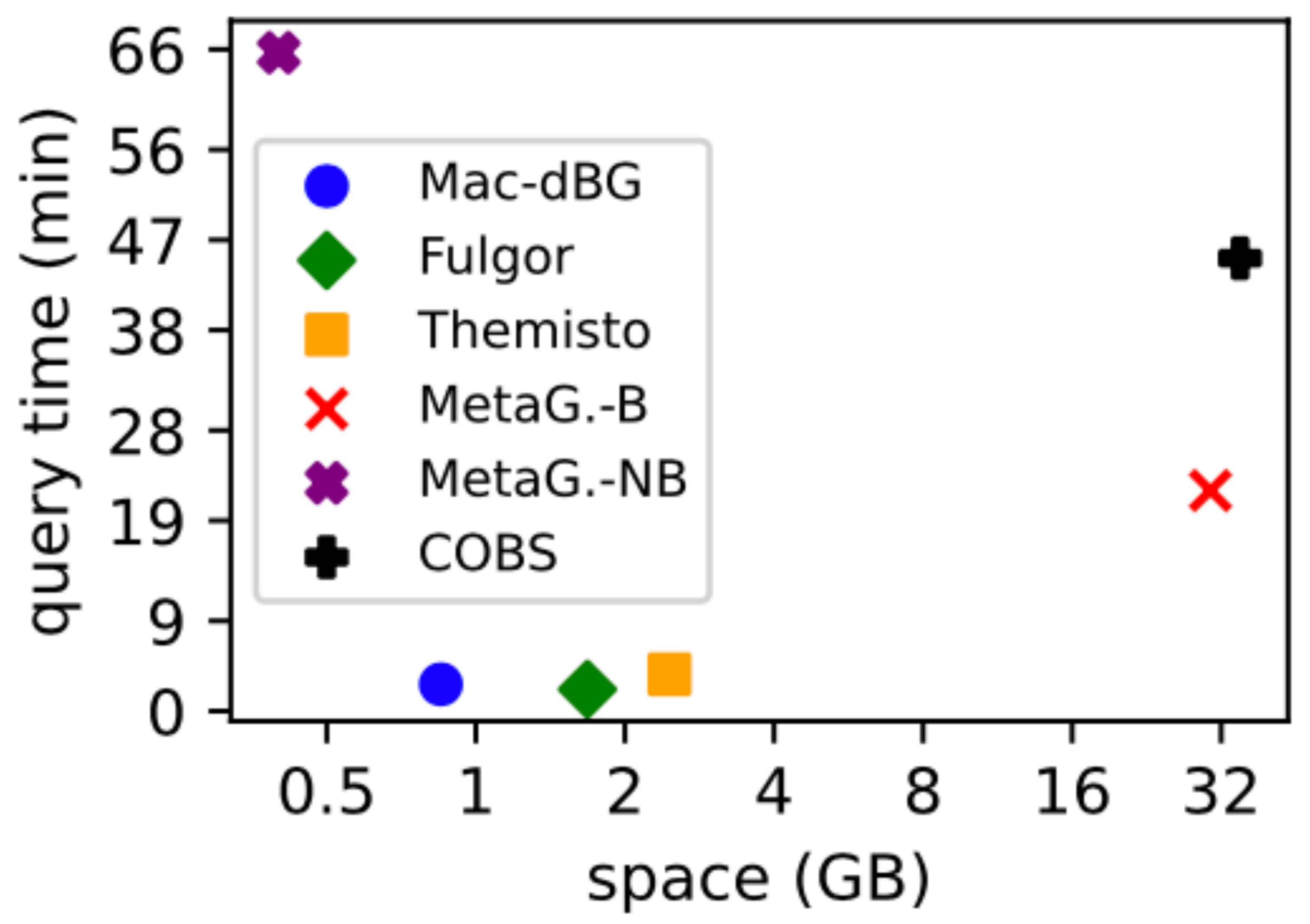
	Genomes	Mac-dBG			Fulgor		
		dBG	Colors	Total	dBG	Colors	Total
EC	3,682	0.29	0.52	0.81	0.29	1.36	1.65
SE	5,000	0.16	0.16	0.32	0.16	0.59	0.75
	10,000	0.35	0.33	0.68	0.35	1.66	2.01
	50,000	1.26	2.14	3.40	1.26	17.03	18.30
	100,000	1.72	3.83	5.55	1.72	40.70	42.44
	150,000	2.03	5.37	7.40	2.03	68.60	70.66
GB	30,691	21.31	7.85	29.16	21.31	15.45	36.85

Pseudoalignment efficiency

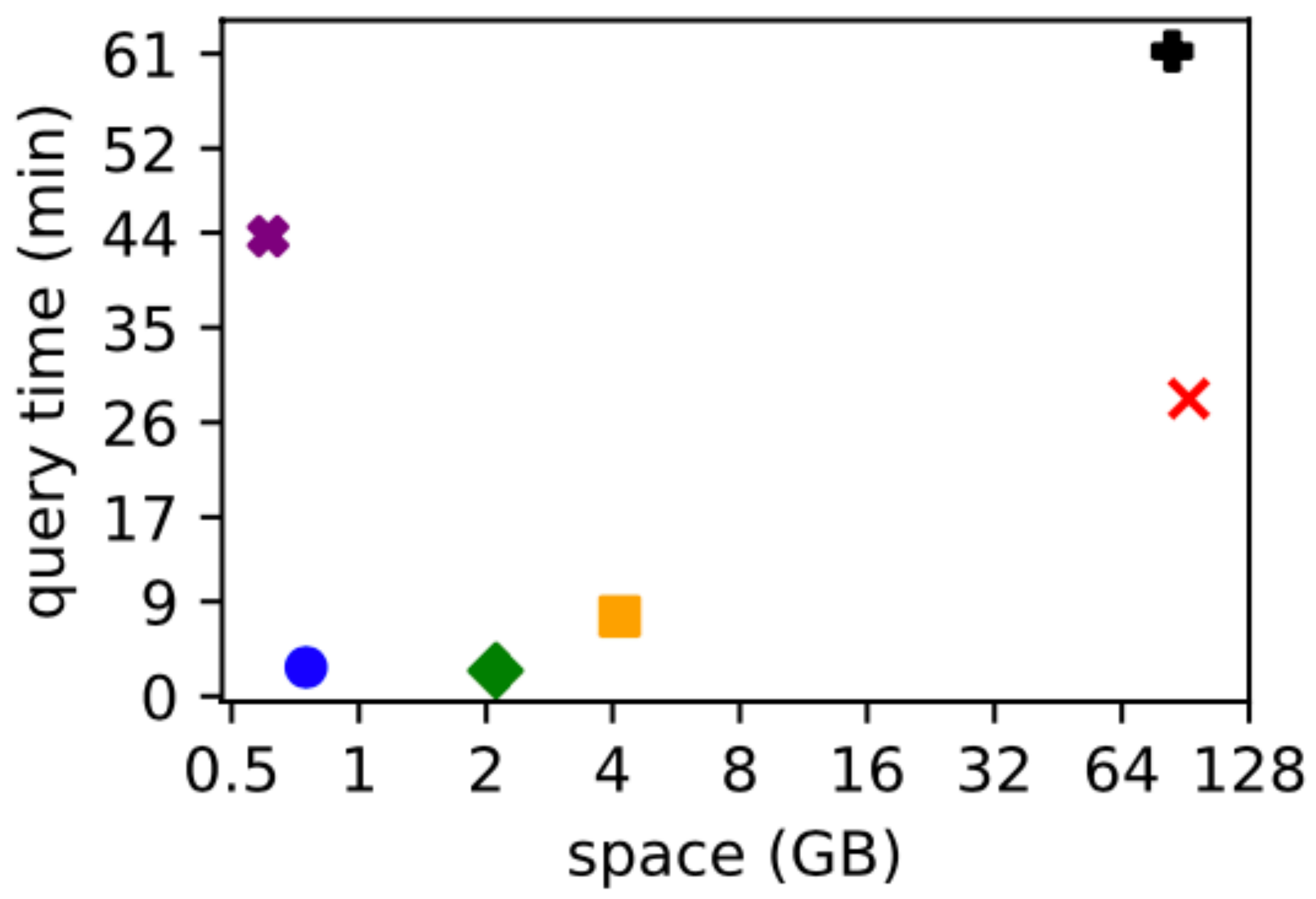


	Genomes	Rate	Mac-dBG		Fulgor	
			mm:ss	GB	mm:ss	GB
EC	3,682	98.99	2:40	0.85	2:10	1.68
	5,000	89.49	1:16	0.37	1:16	0.82
	10,000	89.71	2:45	0.75	2:26	2.11
SE	50,000	91.25	14:00	3.65	19:15	18.53
	100,000	91.41	26:48	6.29	27:30	42.78
	150,000	91.52	41:30	8.51	42:30	70.55
GB	30,691	92.91	01:03	28.51	01:10	30.02

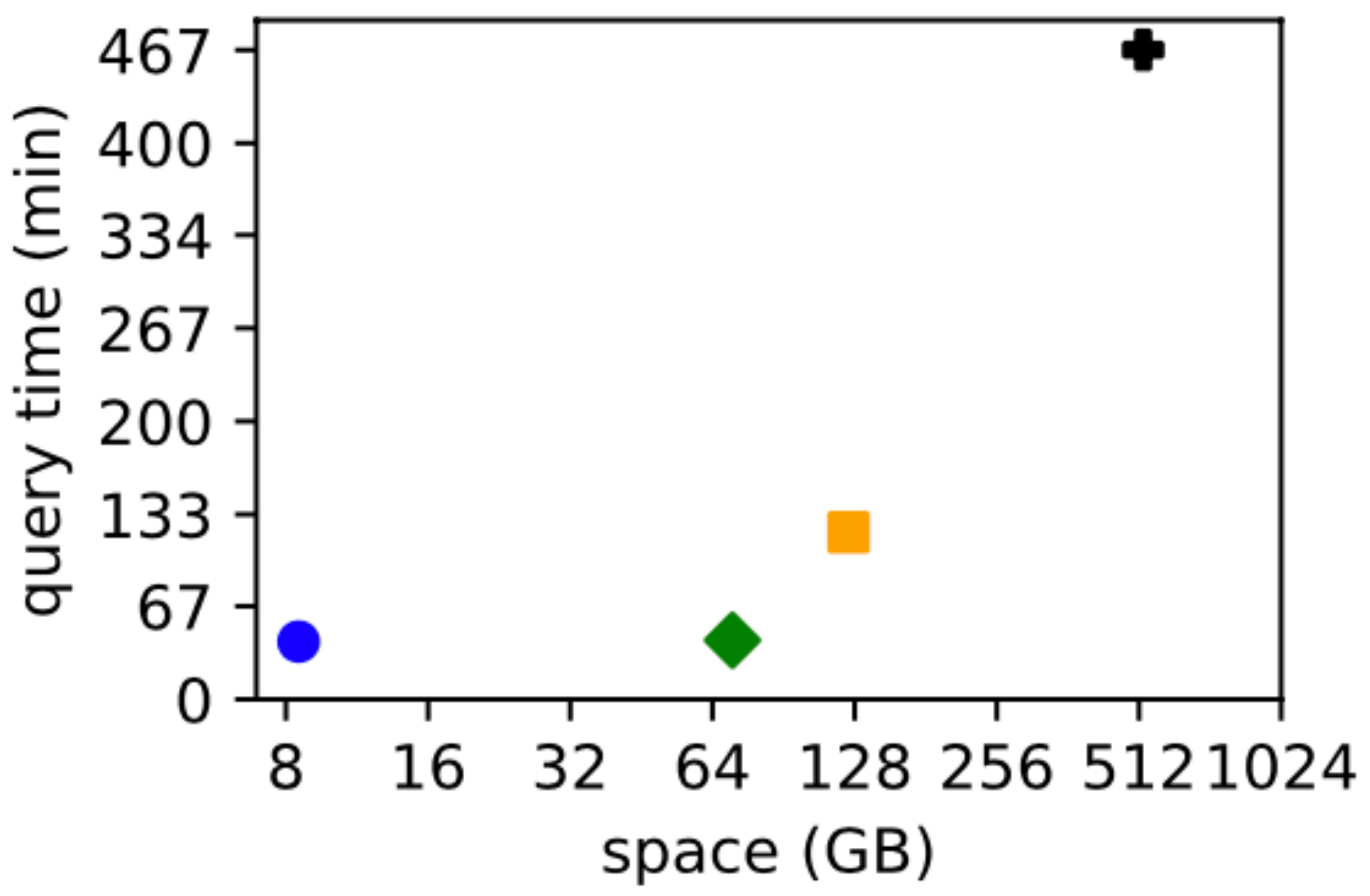
Overall space/time trade-off



(a) EC



(b) SE 10,000



(c) SE 150,000

Conclusions

- SSHash to obtain an **efficient map from k-mers to unitigs**.
- Permute unitigs in color order to enable a **space-efficient mapping from unitigs to colors**.
- Factorize the redundancy in large color matrixes via **meta/partial colors**.
- Result: the meta-colored dBG embodies a superior space/time trade-off compared to the state of the art. Space improvement can be dramatic but query efficiency not harmed.
- Many **open problems and future directions**, check our pre-print out: https://doi.org/10.1101/2023.07.21.550101.

Conclusions

- SSHash to obtain an **efficient map from k-mers to unitigs**.
- Permute unitigs in color order to enable a **space-efficient mapping from unitigs to colors**.
- Factorize the redundancy in large color matrixes via **meta/partial colors**.
- Result: the meta-colored dBG embodies a superior space/time trade-off compared to the state of the art. Space improvement can be dramatic but query efficiency not harmed.
- Many **open problems and future directions**, check our pre-print out: https://doi.org/10.1101/2023.07.21.550101.

Thank you for the attention!