DNA-Based Storage of RDF Graph Data: A Futuristic Approach to Data Analytics

Mr. Asad Usmani
Supervisor: Prof. Dr. Lena Wiese

Department of Computer Science
Goethe University Frankfurt am Main

20th June 2025

Problem Statement 1: Why Partial Data Retrieval?

- Even though a movie, an image, or a book could be retrieved as a whole from DNA storage as needed, this is not the same for **complex graph data**.
- Indeed, compressed data storage is helpful in both synthesis and sequencing operations to handle data cost-efficiently. However, even if only **partial information** is required, it is unnecessary to sequence and decode the complete archived data about a single complex graph, which is expensive and impractical.
- If **future query demands** are considered, any of the existing proposed DNA storage models are not appropriate.
- We have, therefore, designed a **DNA-based storage model for RDF graph data**, which enables us to retrieve partial information by sequencing a subset of DNA strands rather than all. Consequently, this **reduces significant sequencing costs and time**.

ID-based Triple Storage (IBTS) Method

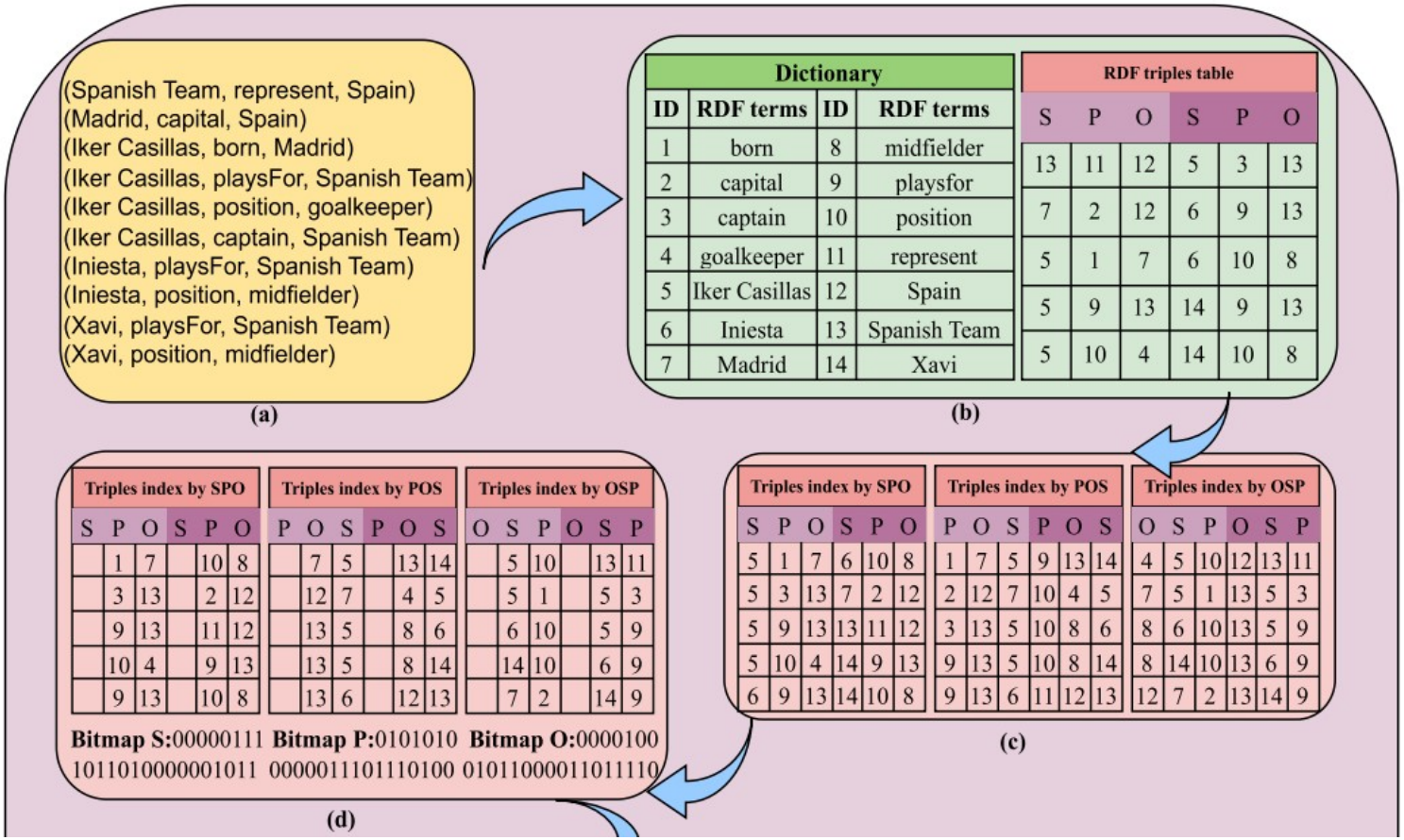


Figure: **(a)** reveals the data modelling of RDF graph data in terms of **SPOs**. We store the RDF strings in a **dictionary** table using an ID-based database model, as shown in **(b)**. **(c,d)** show the **index tables** and **bitmaps** obtained for efficient data retrieval.

DNA Strands Mapping to RDF Graph Data

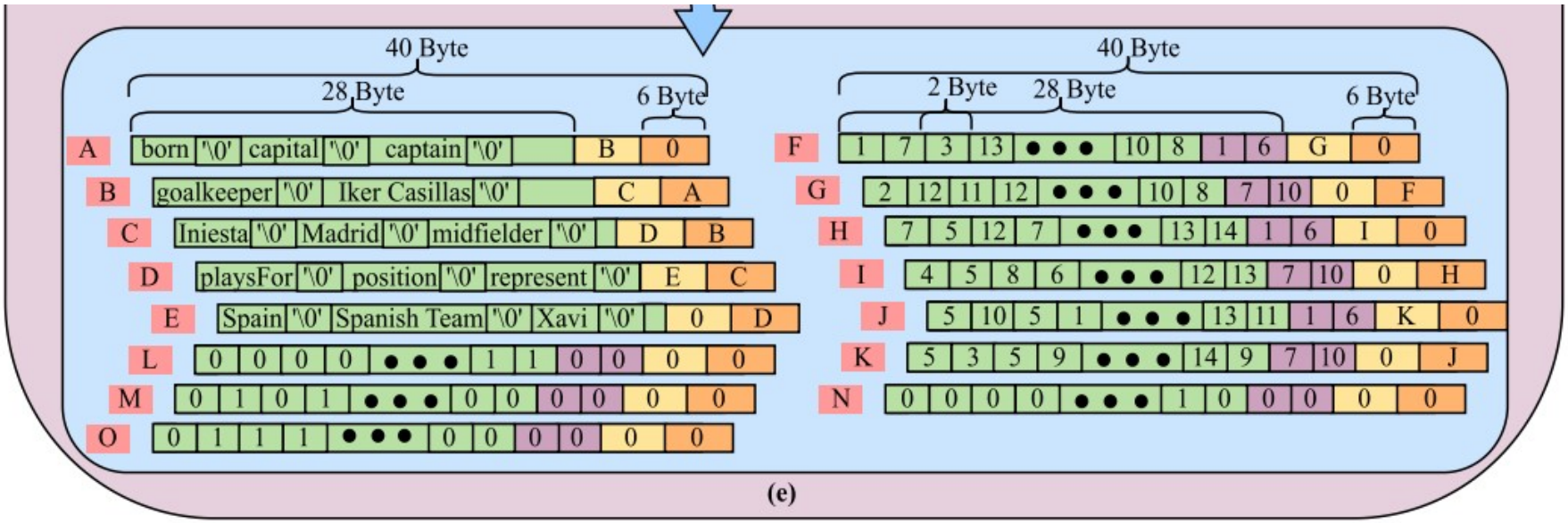


Figure: **(e)** depicts the organization of that RDF graph data modelling in the DNA pool.

SPARQL Query Processing Algorithm

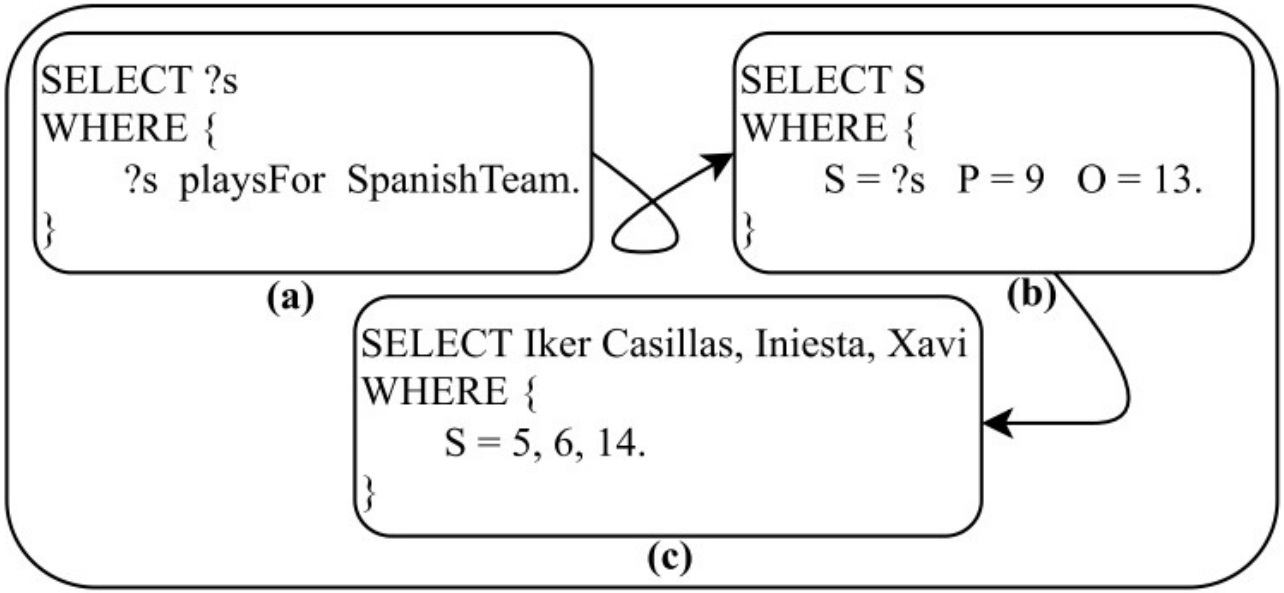


Figure: The query requests for all the subjects associated with the object: **Spanish Team** in the triple pattern (?S, P, O) through the property: **playsFor**. **(a)** The query framework algorithm identifies the dictionary IDs for both of these RDF string items. **(b)** Based on the numerical values of both the predicate and object, we find the IDs of all related subjects. **(c)** Through reverse mapping from ID-to-String, we obtain three RDF strings: **Iker Casillas**, **Iniesta** and **Xavi**, using their IDs: 5, 6 and 14.

Quantitative Experimental Results

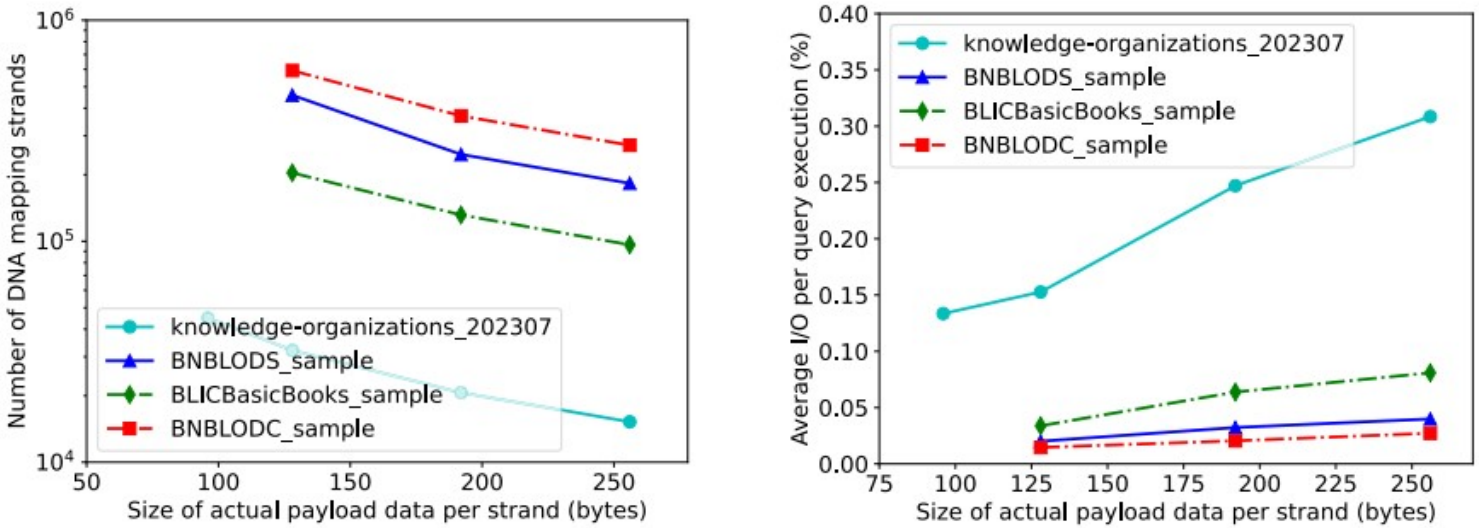


Figure: Plots the line charts on query processing over four different RDF graphs based on minimum data retrieval by DNA strand size. This figure shows **(a)** the total number of strand maps across different strand sizes for each of the four graphs and **(b)** the corresponding **average number of strands retrieved** after six query processing.

Statistical Experimental Results

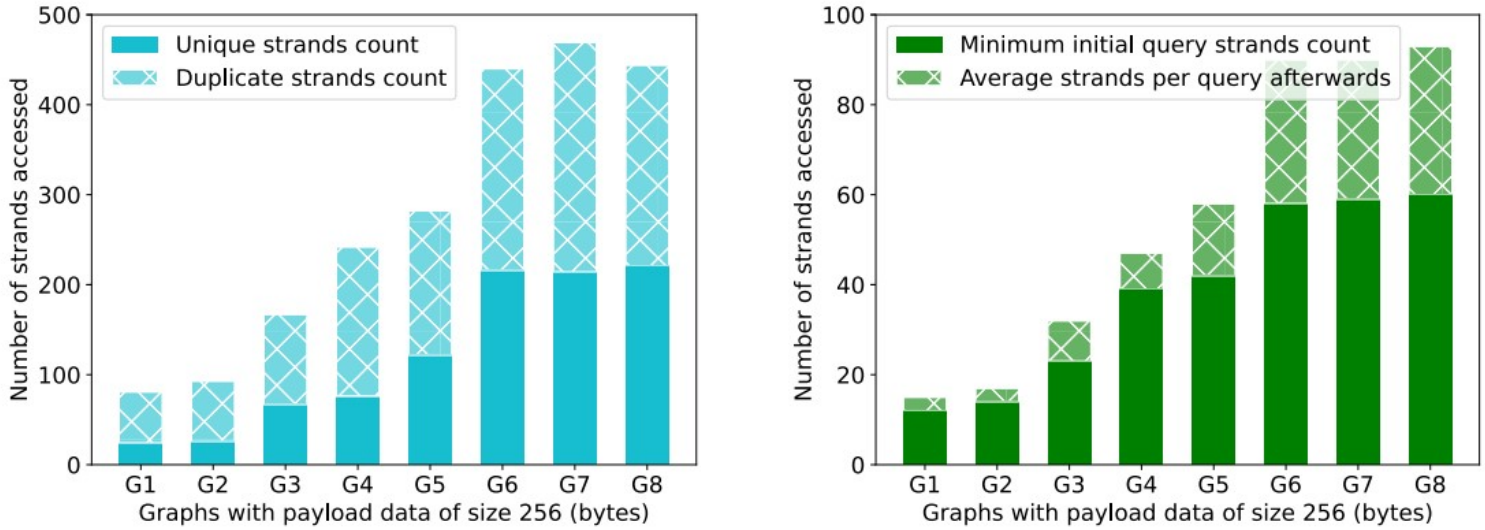
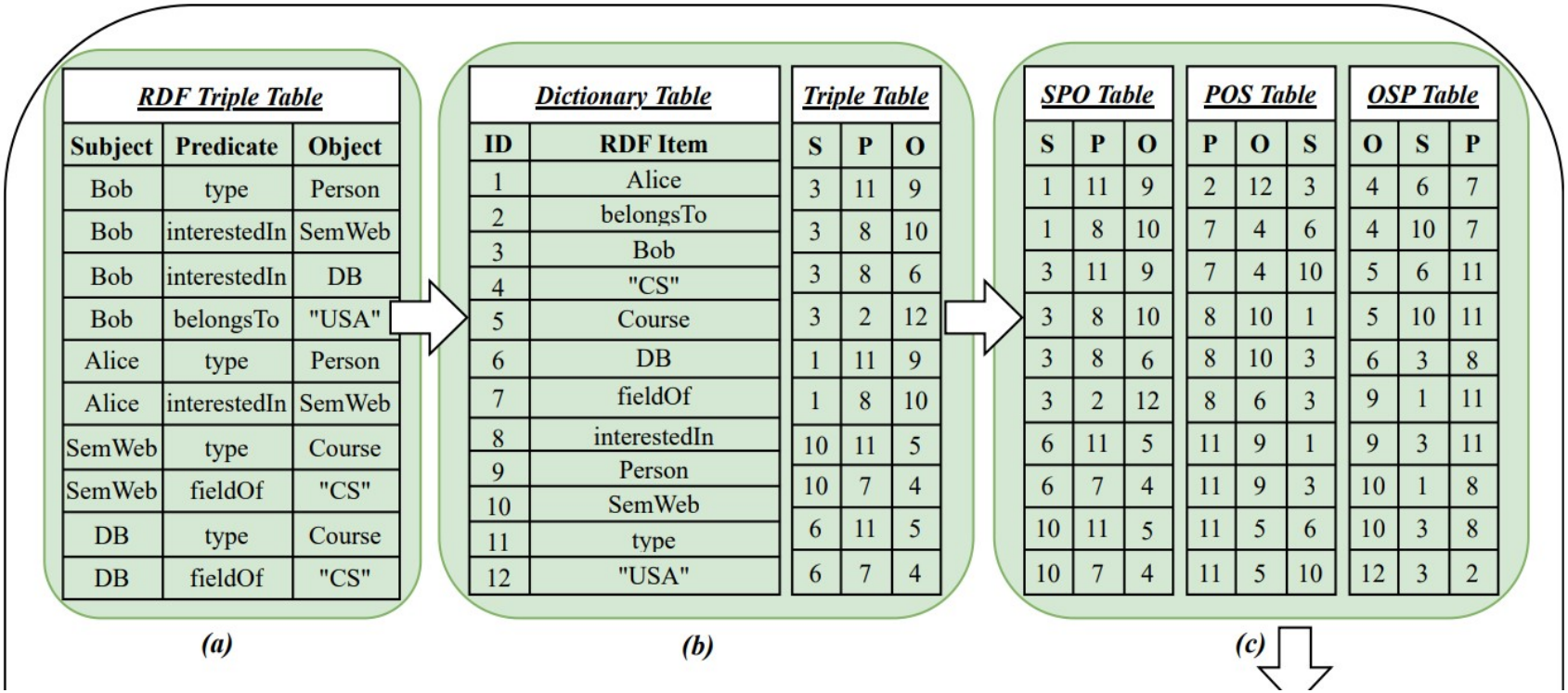


Figure: (Left) With six queries, various strands are accessed multiple times (referred to as **duplicate strands**), revealing the number of unique and duplicate strands for each graph data with a certain payload size. Similarly, with six queries, an average strand count accessed after an initial query for each graph can be visualized in the opposite figure (Right).

Problem Statement 2: Why Fast Partial Data Retrieval?

- When it comes to **scalability**, the prior method consumes numerous sequencing runs and is not the best option for large graph datasets.
- A fewer number of primers means more hamming distance, reducing the possibility of **unwanted data** being retrieved and consequently reducing the amount of data analysis needed. The prior approach does not restrict us to limited sequencing of undesired data and is flexible in choosing a **primers set** available.
- Variations in DNA strand length cause the other factors to vary. Using NGS is best for mapping all strand lengths and is suitable for achieving results quickly with our new block-based method.

Block-based Triple Storage (BBTS) Method



RDF Triple Table

Subject	Predicate	Object
Bob	type	Person
Bob	interestedIn	SemWeb
Bob	interestedIn	DB
Bob	belongsTo	"USA"
Alice	type	Person
Alice	interestedIn	SemWeb
SemWeb	type	Course
SemWeb	fieldOf	"CS"
DB	type	Course
DB	fieldOf	"CS"

(a)

Dictionary Table

ID	RDF Item
1	Alice
2	belongsTo
3	Bob
4	"CS"
5	Course
6	DB
7	fieldOf
8	interestedIn
9	Person
10	SemWeb
11	type
12	"USA"

Triple Table

S	P	O
3	11	9
3	8	10
3	8	6
3	2	12
1	11	9
1	8	10
10	11	5
10	7	4
6	11	5
6	7	4

(b)

SPO Table

S	P	O
1	11	9
1	8	10
3	11	9
3	8	10
3	8	6
3	2	12
6	11	5
6	7	4
10	11	5
10	7	4

POS Table

P	O	S
2	12	3
7	4	6
7	4	10
8	10	1
8	10	3
8	6	3
11	9	1
11	9	3
11	5	6
11	5	10

OSP Table

O	S	P
4	6	7
4	10	7
5	6	11
5	10	11
6	3	8
9	1	11
9	3	11
10	1	8
10	3	8
12	3	2

(c)

Figure: **(a)** reveals the data modelling of RDF graph data in terms of SPOs. We store the RDF strings in a dictionary by using an ID-based database model, as shown in **(b)**. **(c)** shows the three index tables obtained for efficient data retrieval.

Block-based Triple Storage (BBTS) Method

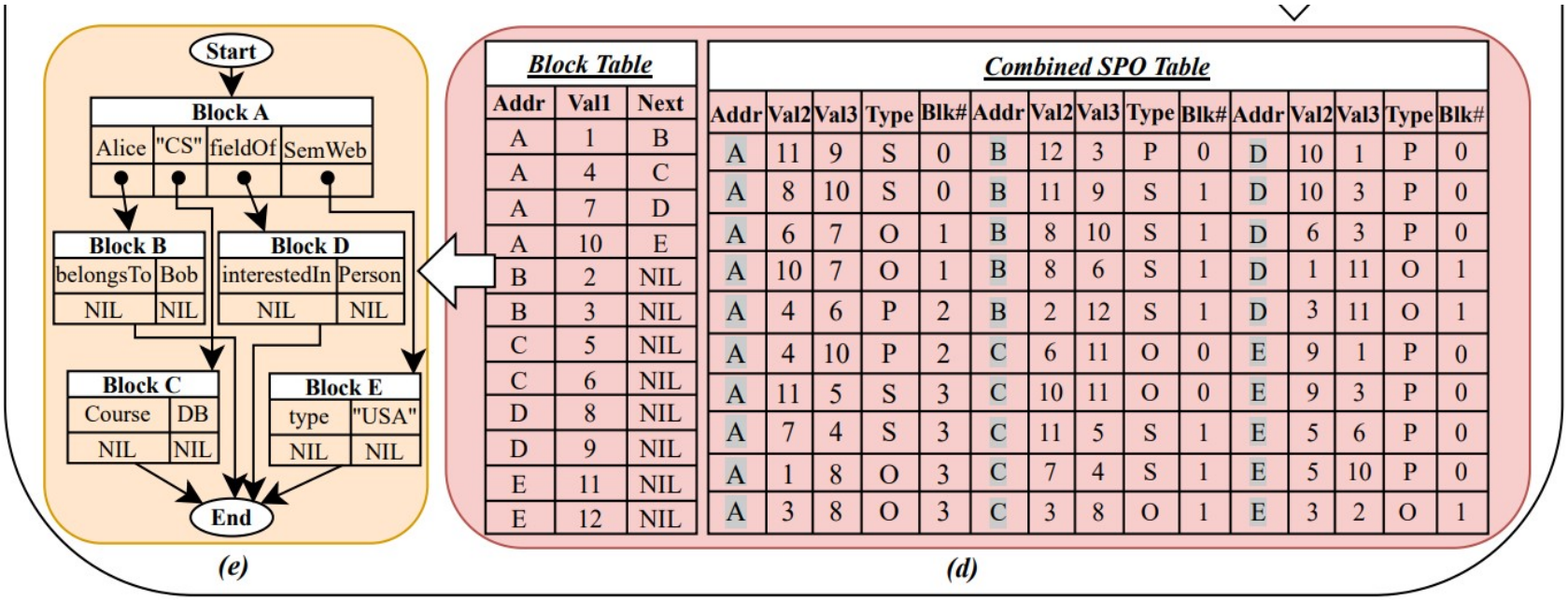


Figure: **(d)** reveals both the **block table** and **combined spo table** based on a **block size of 4**. Each block table entry contains three fields: a primer address to amplify the corresponding block contents, an RDF string item and another address to reach the next block if needed. Finally, **(e)** depicts the organization of that RDF graph data modelling as a concatenation of various blocks.

Mapping Strands to RDF Graph Data

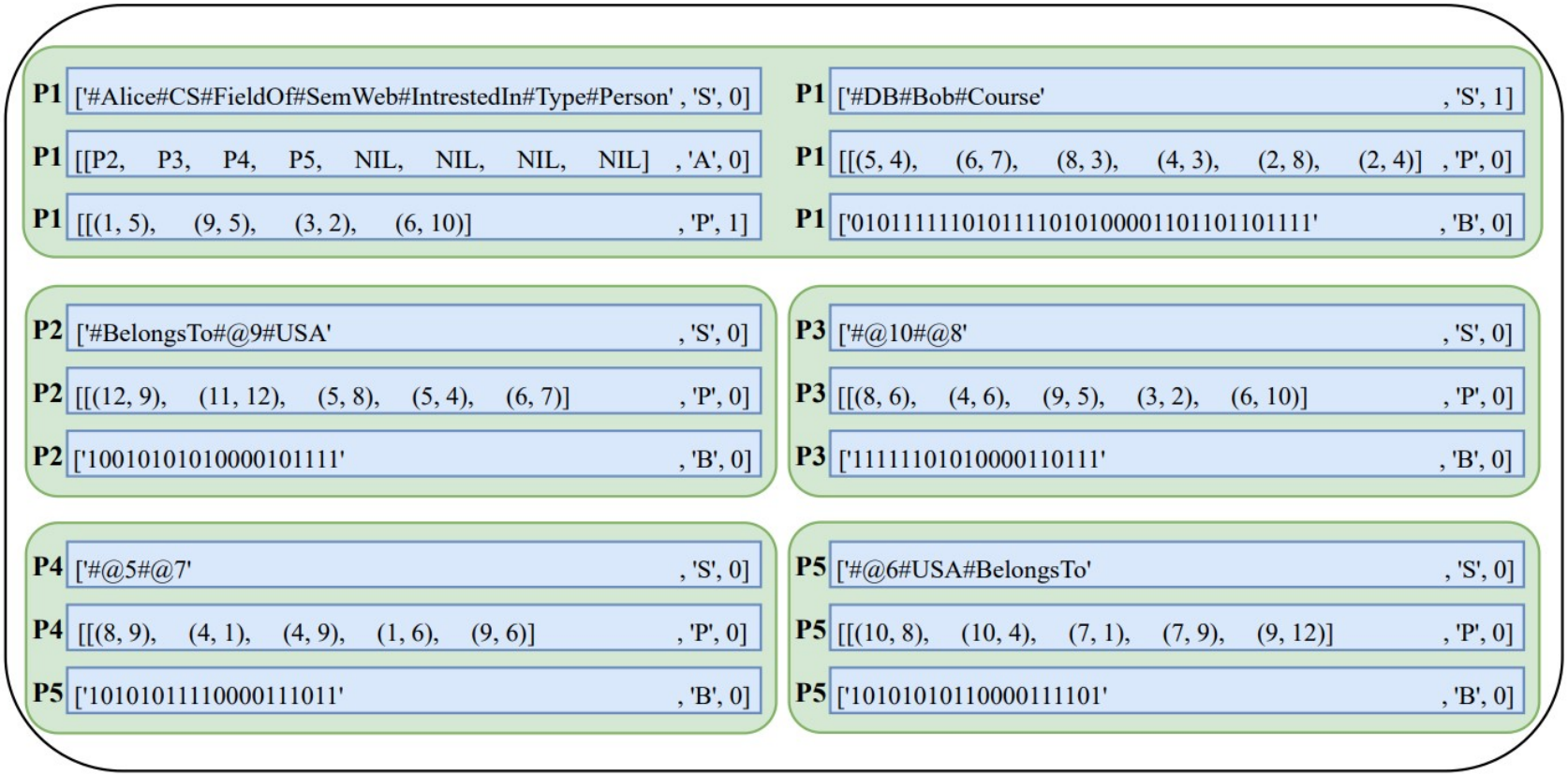


Figure: This figure depicts the mapping structure of **four basic payload types: S, A, P and B of a DNA strand** to map RDF graph data components. Moreover, 18 strands with a payload of 196nt each for the five blocks (P1 — P5) are used to map the exemplary dataset.

Quantitative Experimental Results

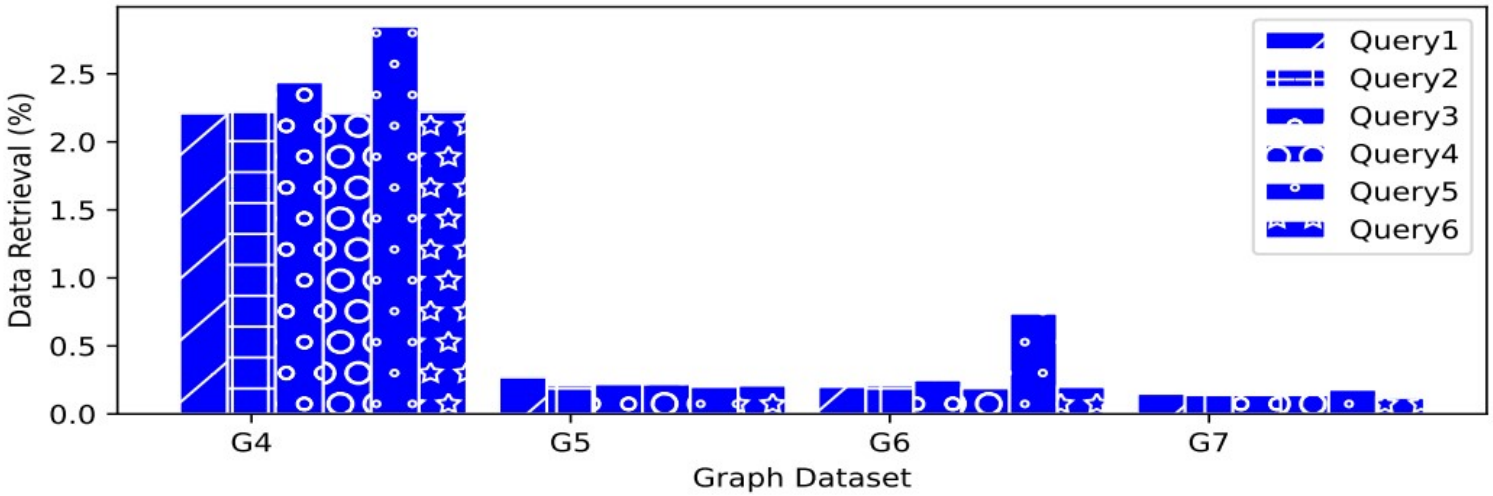


Figure: The figure shows the **data retrieval percentage** of four large graphs with a block size of 768 for six different random queries each.

Quantitative Comparison Results

	BASELINE METHOD ($\mathcal{M}_{base}$)							BLOCK-BASED METHOD ($\mathcal{M}_{block}$)						
Graph	$\mathcal{M}_b$	$\mathcal{S}_m$	$\mathcal{S}_a$	$\mathcal{C}_s$	$\mathcal{B}_s$	$\mathcal{S}_r$	$\mathcal{C}_t$	$\mathcal{M}_b$	$\mathcal{S}_m$	$\mathcal{S}_a$	$\mathcal{C}_s$	$\mathcal{B}_s$	$\mathcal{S}_r$	$\mathcal{C}_t$
G1	0.38	1,179	83	7.04	1	27	+27	0.50	10,959	460	4.20	128	2	+2
G2	0.54	1,659	94	5.67	1	32	+32	0.53	11,580	396	3.42	128	2	+2
G3	0.41	1,283	125	9.74	1	33	+33	0.36	7,841	234	2.98	128	2	+2
G4	0.64	1,982	153	7.72	1	31	+31	0.60	13,002	950	7.31	128	2	+2
G5	4.92	15,221	108	0.71	1	60	+60	7.31	159,603	3,653	2.29	768	2	+2
G6	31.15	96,352	141	0.15	1	60	+60	58.46	1,277,103	4,463	0.35	768	2	+2
G7	59.12	182,869	110	0.06	1	57	+57	96.01	2,097,352	5,964	0.28	768	2	+2
G8	87.89	271,845	141	0.05	1	67	+67	137.85	3,011,327	6,660	0.22	768	2	+2
G9	122.28	378,237	127	0.03	1	69	+69	134.15	2,930,579	28,248	0.96	4096	2	+2

Figure: Shows a quantitative analysis of RDF graph datasets using a few key characteristics. There, the notations $\mathcal{M}_b$, $\mathcal{S}_m$, $\mathcal{S}_a$, $\mathcal{C}_s$, $\mathcal{B}_s$, $\mathcal{S}_r$ and $\mathcal{C}_t$ express the memory mapping of payload data in megabytes, number of strands mapped, number of strands accessed, **sequencing cost** by percentage, block size, **sequencing runs** and sequencing time for partial data retrieval, respectively.

Conclusion

- We presented two strategies: **ID-based binary search approach** and **Block-based random access method**, which offer an efficient DNA-based query processing system to retrieve partial information. Specifically, the average **partial data retrieval per query as output is found to be less than 1%** for RDF graphs with more than 10MB, thus reducing both sequencing costs and time.

Thank you!

Questions?

usmani@mathematik.uni-frankfurt.de