TVM: A Tile–based Video Management Framework
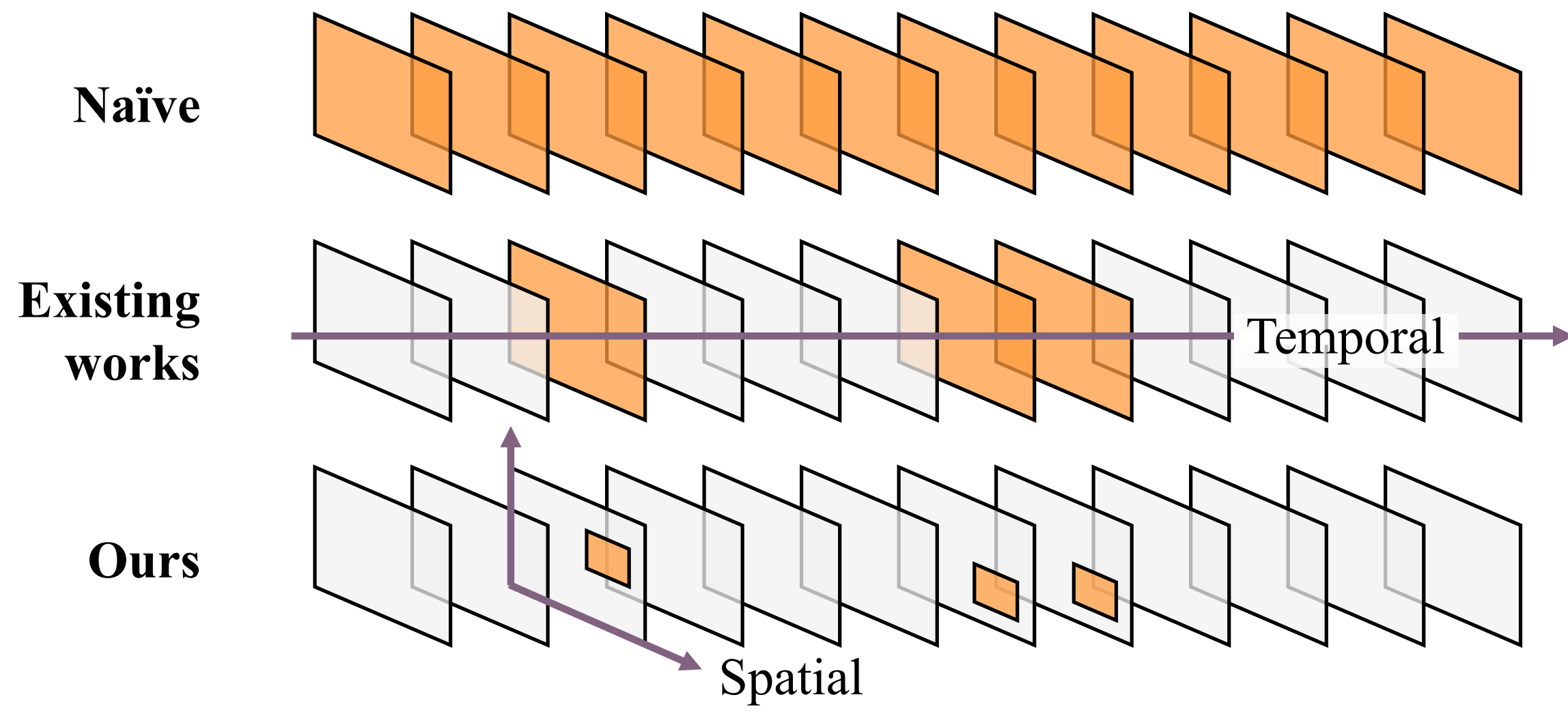
Tianxiong Zhong[1], Zhiwei Zhang[1], Guo Lu[2], Ye Yuan[1], Yu-Ping Wang[1], and Guoren Wang[1]

[1]Beijing Institute of Technology [2]Shanghai JiaoTong University

Motivation

Query: How many cars are there in the video?   ■ Decode+DNN

Naïve

Existing works — Temporal

Ours — Spatial

By reducing data access from the spatial scale to accelerate video analysis queries.

Challenge

5×11 tiles, 6 contain objects   5×5 tiles, 3 contain objects   3×3 tiles, 1 contain objects

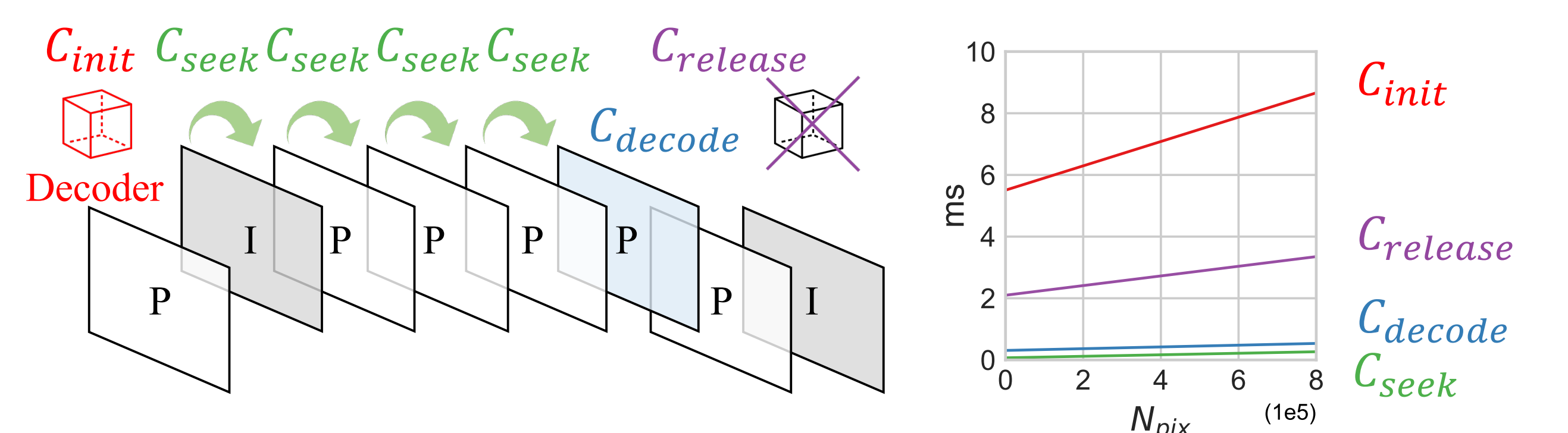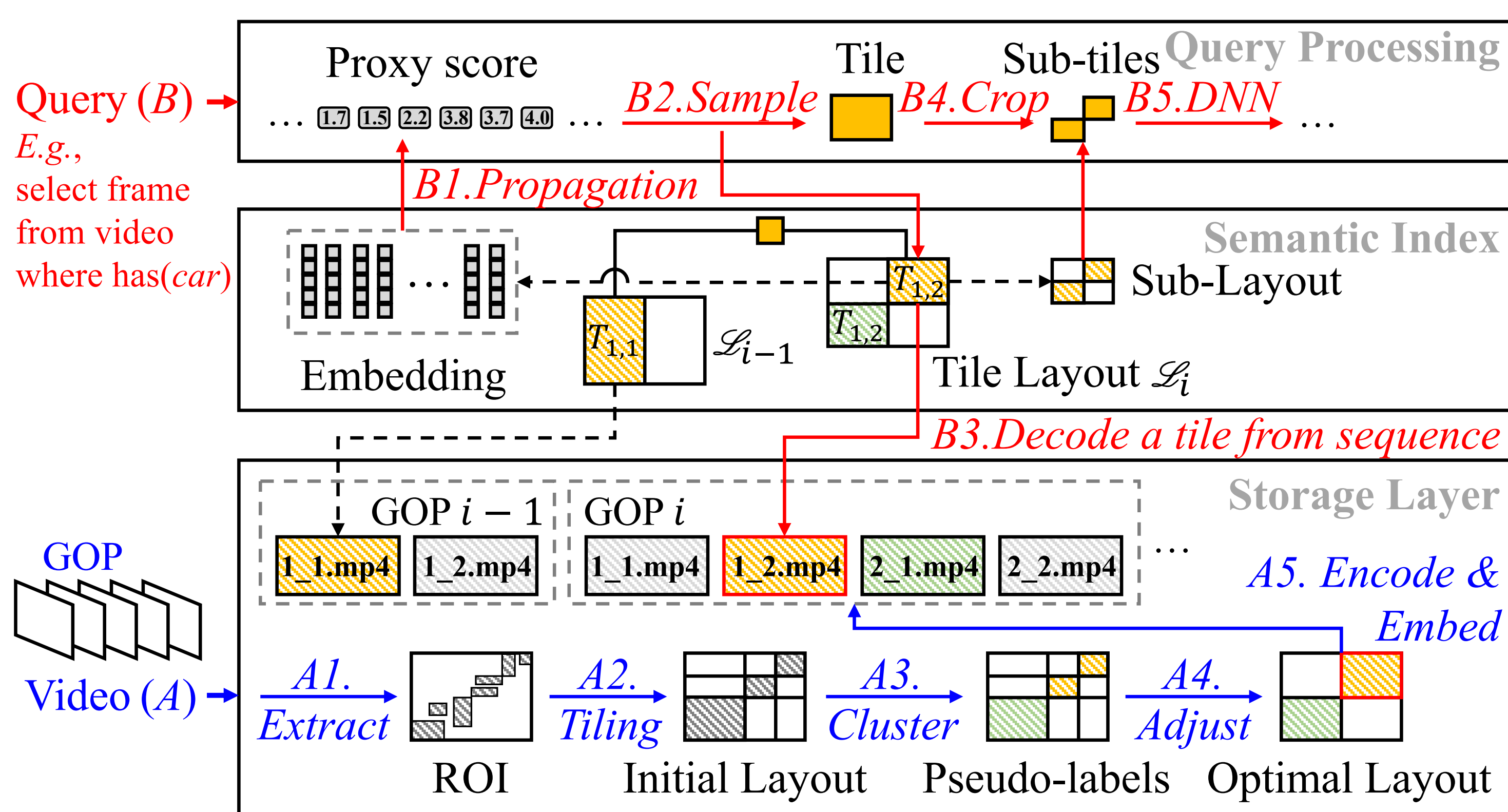Fine-grained ◀———— **Optimal** ————▶ Course-grained
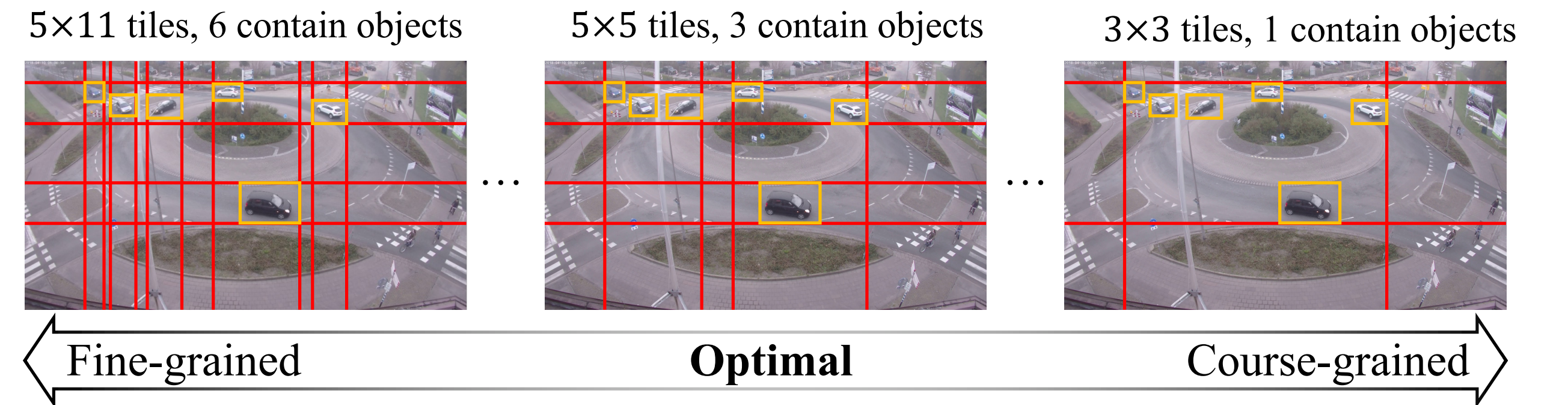
✓ Small tile size   ✓ **Suitable tile size**   ✗ Huge tile size
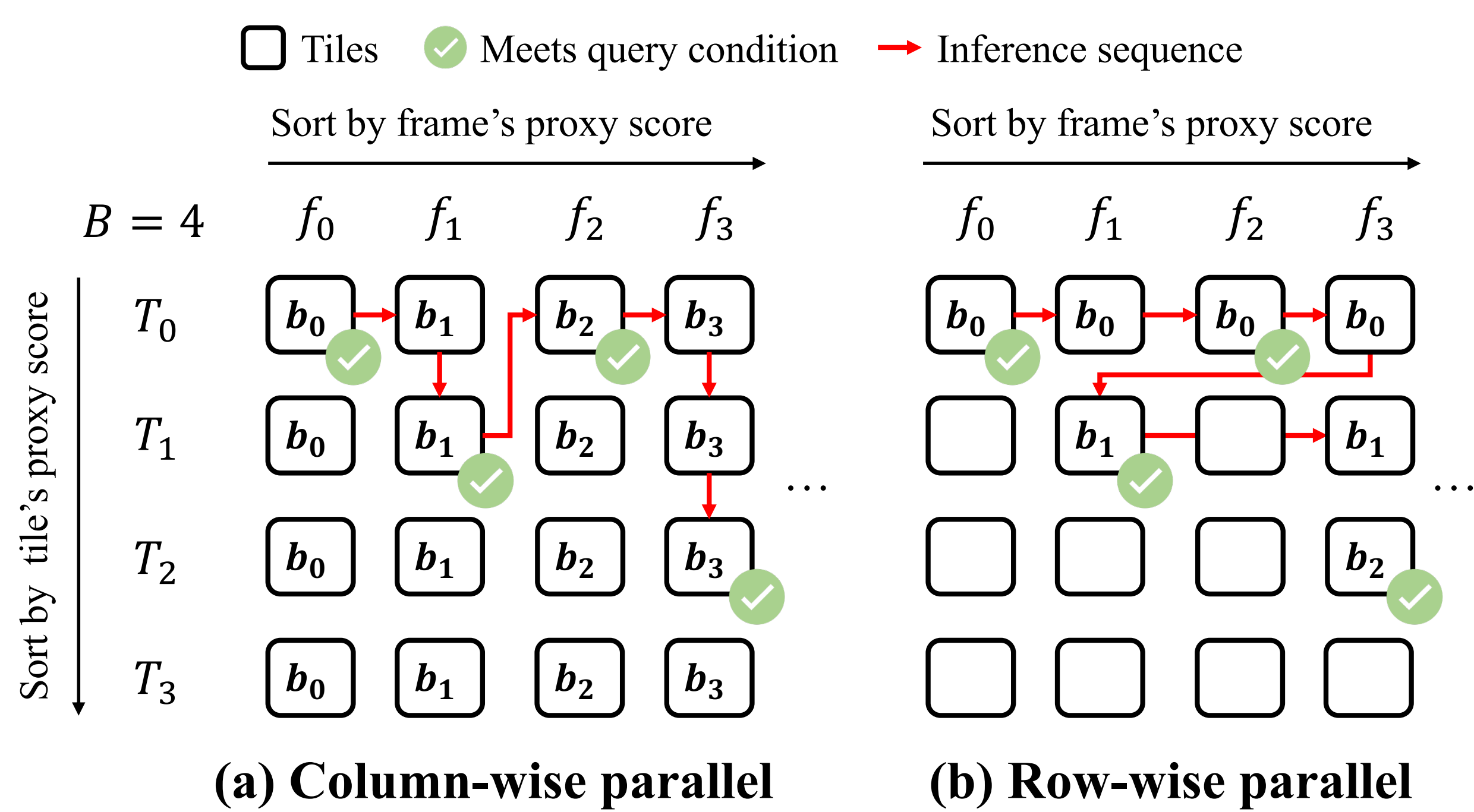✗ Decode many tiles   ✓ **Decode less tiles**   ✓ Decode only one tile

1. Design of an optimal tile layout that minimizes the overall decoding and processing costs.
2. Query workflows for accessing tiles are different. Need query-specific acceleration algorithms.

Optimal Index Structure

Query Processing

Proxy score … 1.7 1.5 2.2 3.8 3.7 4.0 …   Tile   Sub-tiles

Query (B)   *B2.Sample* $\rightarrow$ *B4.Crop* $\rightarrow$ *B5.DNN* …

E.g., select frame from video where has(car)

B1.Propagation

Semantic Index

Embedding $\mathscr{L}_{i-1}$ $T_{1,1}$ $T_{1,2}$ Sub-Layout

Tile Layout $\mathscr{L}_i$

B3.Decode a tile from sequence

Storage Layer

GOP $i-1$ GOP i
1_1.mp4 1_2.mp4   1_1.mp4 1_2.mp4 2_1.mp4 2_2.mp4 …

A5. Encode & Embed

GOP

Video (A)   *A1. Extract* → ROI *A2. Tiling* → Initial Layout *A3. Cluster* → Pseudo-labels *A4. Adjust* → Optimal Layout

C_{init} C_{seek} C_{seek} C_{seek} C_{seek} $C_{release}$
Decoder C_{decode}
P I P P P P P I

(a) Tile random access

(b) Random access costs

(plot axes: ms vs N_{pix} (1e5); curves C_{init}, $C_{release}$, C_{decode}, C_{seek})
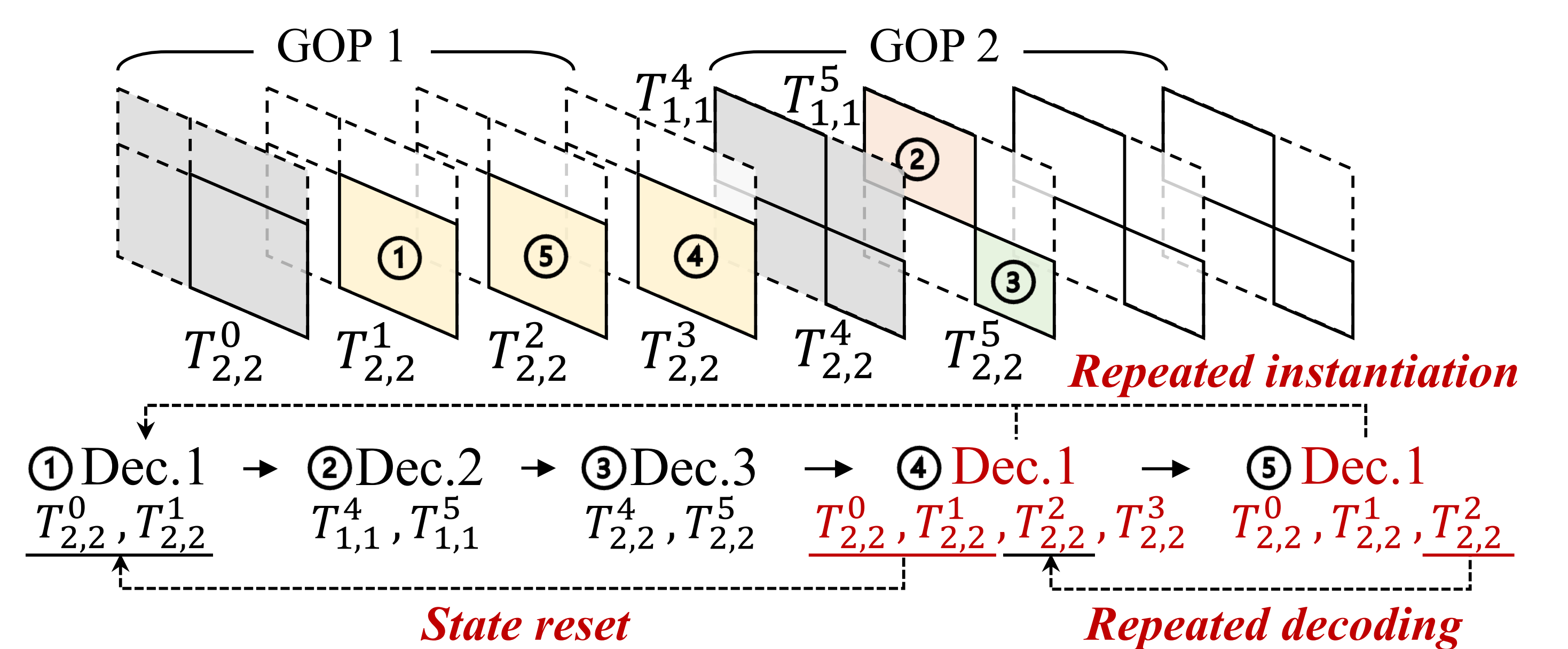
- We model the decoding/DNN cost as a function of the number of pixels which can be used to quantify the pros and cons of different layouts.
- To speed up the search for the optimal layout, we also propose DP and greedy algorithms.

Query-Specific Acceleration Algorithms

☐ Tiles   ✓ Meets query condition   → Inference sequence

Sort by frame's proxy score   Sort by frame's proxy score

$B = 4$ f_0 f_1 f_2 f_3 f_0 f_1 f_2 f_3

Sort by tile's proxy score

T_0 b_0 b_1 b_2 b_3 b_0 b_0 b_0 b_0
T_1 b_0 b_1 b_3 b_1 b_1
T_2 b_0 b_1 b_2 b_3 b_2
T_3 b_0 b_1 b_2 b_3

(a) Column-wise parallel   **(b) Row-wise parallel**

Parallel decoding algorithm for Limit queries to avoid access tiles that do not need to be processed.

GOP 1 ——————— GOP 2

$T_{1,1}^4$ $T_{1,1}^5$

① ⑤ ④ ③ ②

$T_{2,2}^0$ $T_{2,2}^1$ $T_{2,2}^2$ $T_{2,2}^3$ $T_{2,2}^4$ $T_{2,2}^5$

Repeated instantiation

①Dec.1 → ②Dec.2 → ③Dec.3 → ④Dec.1 → ⑤Dec.1
$T_{2,2}^0, T_{2,2}^1$ $T_{1,1}^4, T_{1,1}^5$ $T_{2,2}^4, T_{2,2}^5$ $T_{2,2}^0, T_{2,2}^1$ $T_{2,2}^2$ $T_{2,2}^3$ $T_{2,2}^0, T_{2,2}^1, T_{2,2}^2$

State reset   *Repeated decoding*

Cache decoders, decoding states, and dependent frames to alleviate the resource waste in track queries.

Experiment

■ Decode   ■ Detection   ■ Propagation/Scoring   ■ Other

(y-axis: seconds)

night-street (car): 1.0x, 1.49x
amsterdam (car): 1.0x, 1.59x
canal (boat): 1.0x, 2.91x
archie (car): 1.0x, 5.63x

(a) Aggregation Queries

night-street (car): 1.0x, 1.74x
amsterdam (car): 1.0x, 1.69x
canal (boat): 1.0x, 1.64x
archie (car): 1.0x, 1.69x

(b) Limit Queries

(x-axis labels: Frame, Ours, TASM, TASM+)

Learn more

VLDB2024 GUANGZHOU