

# EHL\*: Memory-Budgeted Indexing for Ultrafast Euclidean Pathfinding

Jinchun Du, Bojie Shen, Muhammad Aamir Cheema  
Faculty of Information Technology, Monash University, Australia

## Problem and Motivation

### Problem (ESPP):

We need optimal obstacle-avoiding shortest paths in a Euclidean plane with polygonal obstacles for large-scale applications (games, robotics, indoor navigation).

### Why existing SOTA is limited:

Euclidean Hub Labeling (EHL) delivers ultra-fast queries but requires very large memory (up to tens of GB), and its memory footprint is only known after construction, which is painful for memory-constrained devices.

### Our contribution :

EHL\* is the first memory-budgeted Euclidean Hub Labeling method with provable memory-runtime trade-offs. It supports workload-aware compression, achieving 10–20× memory reduction while maintaining near-EHL query performance in practice.

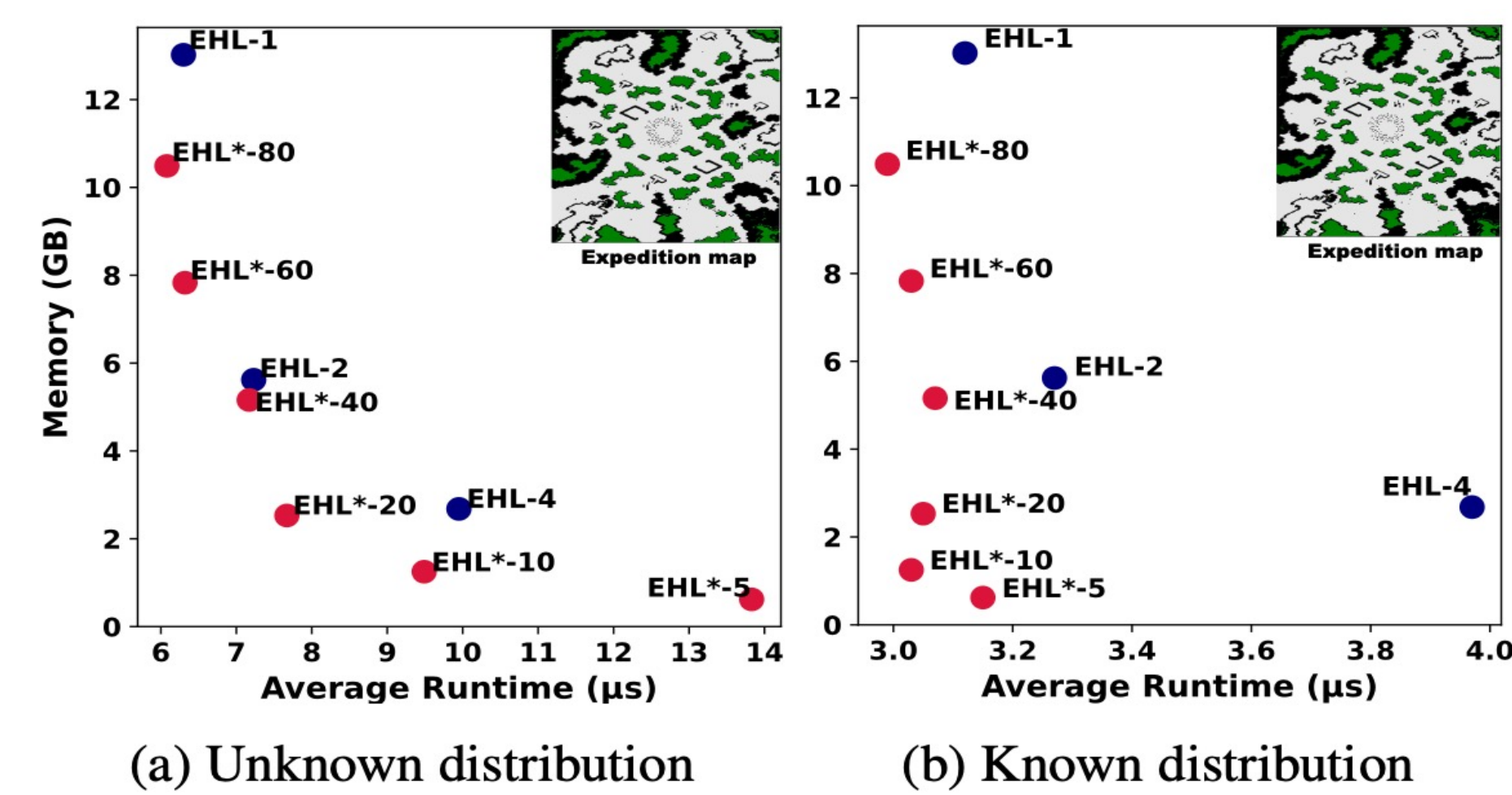


Figure 1: Memory-runtime tradeoff for EHL and EHL\*

## Theoretical Contributions

### Memory Reduction Model:

- Let:  $N$ : number of grid cells.  $H$ : average labels per cell in EHL.  $\theta$ : average Jaccard similarity between merged regions. EHL memory:  $M = HN$ .
- After  $i$  merge rounds:  $M_i = \frac{HN}{(1+\theta)^i}$
- To achieve budget  $B = bM$ :  $i = \frac{\log(1/b)}{\log(1+\theta)}$
- Memory decreases exponentially with merge depth, governed by label overlap.

### Query-Time Complexity Bound:

- EHL query complexity:  $O(H)$ .
- EHL\* query complexity after  $i$  merges:  $O\left(\frac{2^i H}{(1+\theta)^i}\right)$ .
- Query slowdown is provably bounded and grows smoothly as memory is reduced.

### Index Construction Overhead:

- Compression performs  $O(N)$  merges.
- Cost of merging two level- $i$  regions:  $O\left(2^{i/2} + \frac{2^i H}{(1+\theta)^i}\right)$ .
- Total compression cost:  $O\left(N\left(2^{i/2} + \frac{2^i H}{(1+\theta)^i} + \log N\right)\right)$ .
- Index construction remains polynomial and predictable, even for small budgets.

## Experimental Results

### Memory-Runtime Trade-off:

- EHL\* enables explicit memory budgeting.
- Achieves 10–20× memory reduction vs. EHL.
- Maintains near-EHL query performance down to 20–40% memory.
- At 5–10% memory, runtime increases modestly.

### Workload-Aware Benefits:

- With known query distributions:
- Memory reduced to 5% with minimal runtime impact.
  - Fine-grained regions preserved in high-query areas.
  - Compression concentrated in low-query areas.

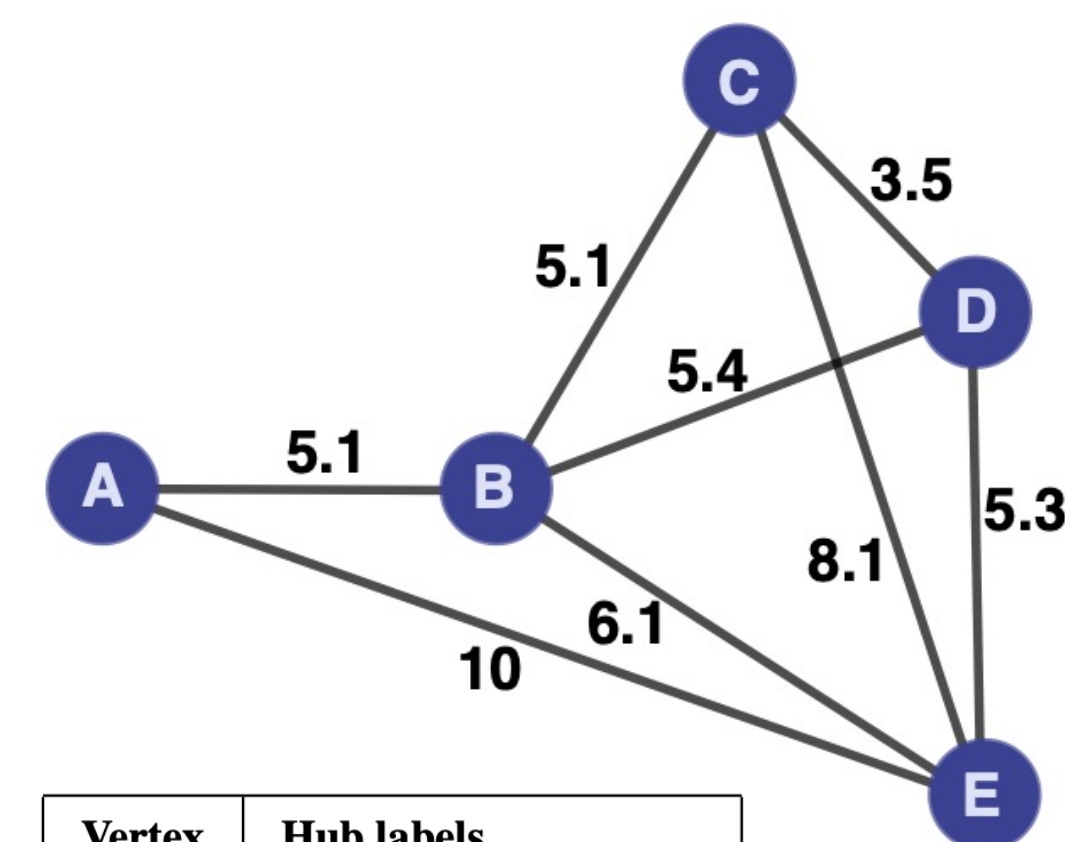
Map	Query Set	EHL*					Competitors					
		80%	60%	40%	20%	10%	5%	EHL-1	EHL-2	EHL-4	EPS	Polyanya
DAO	Memory(MB)	105.5	78.8	52.4	26.3	13.6	7.5	134.8	57.8	27.6	0.21	-
	Build Time(Secs)	6.63	7.21	7.98	8.99	9.43	9.68	4.57	1.24	0.36	0.02	-
	Unknown	2.02	2.08	2.20	2.49	3.12	4.40	1.84	2.28	3.19	25.94	176.50
	Cluster-2	1.00	0.99	0.98	0.98	1.03	1.15	0.98	1.27	1.88	7.22	17.51
	Cluster-4	1.22	1.21	1.20	1.23	1.33	1.59	1.21	1.55	2.23	10.01	32.66
DA	Memory(MB)	36.0	27.0	18.2	9.4	5.1	3.0	47.2	19.6	9.0	0.06	-
	Build Time(Secs)	2.76	2.88	3.05	3.26	3.37	3.43	1.67	0.51	0.18	0.11	-
	Unknown	1.64	1.68	1.75	1.93	2.25	2.86	1.54	1.85	2.48	12.17	37.85
	Cluster-2	0.89	0.88	0.90	0.91	0.93	1.00	0.88	1.09	1.47	5.29	9.45
	Cluster-4	1.10	1.09	1.12	1.14	1.20	1.38	1.09	1.33	1.82	7.13	14.10
BG	Memory(MB)	150.8	112.8	75.0	37.6	19.3	10.3	193.0	71.2	31.8	0.12	-
	Build Time(Secs)	10.13	10.92	11.85	13.04	13.72	14.00	6.91	1.80	0.51	0.04	-
	Unknown	1.34	1.37	1.43	1.55	1.79	2.25	1.27	1.48	1.93	9.77	13.25
	Cluster-2	0.75	0.74	0.78	0.80	0.79	0.82	0.73	0.85	1.11	4.32	6.84
	Cluster-4	0.96	0.97	1.03	1.02	1.04	1.11	0.95	1.10	1.42	6.32	10.12
SC	Memory(MB)	2060.7	1538.5	5515.1	2715.1	1348.1	676.2	14211.1	5845.9	2716.6	5.7	-
	Build Time(Secs)	178.96	189.44	207.40	226.60	241.39	243.18	118.06	30.62	8.01	3.49	-
	Unknown	3.43	3.50	3.96	4.29	5.05	6.47	3.34	4.13	5.40	68.20	261.16
	Cluster-2	1.83	1.93	1.79	1.94	1.81	1.89	1.92	2.22	2.93	30.56	83.23
	Cluster-4	2.48	2.43	2.49	2.41	2.43	2.63	2.57	2.96	3.89	43.74	104.74
CITY	Memory(MB)	11298.9	8389.5	5515.1	2715.1	1348.1	676.2	14211.1	5845.9	2716.6	5.7	-
	Build Time(Secs)	1894.12	1966.11	2090.59	2236.74	2347.68	2417.96	1693.2	425.5	110.4	10.9	-
	Unknown	4.96	4.75	5.07	5.48	6.29	7.98	4.89	6.57	8.09	45.7	119.5
	Cluster-2	1.75	1.71	1.69	1.69	1.75	1.83	2.33	2.79	3.50	4.52	4.67
	Cluster-4	3.75	3.77	3.22	3.16	3.27	3.45	4.24	4.76	5.76	21.7	27.21
Cluster-8	4.20	4.23	4.15	4.21	4.51	5.12	5.36	6.03	7.87	28.43	44.69	

Table 1: Memory, build time and runtime comparison between EHL\* and the competitors for different scenarios.

## Euclidean Hub Labeling (EHL)

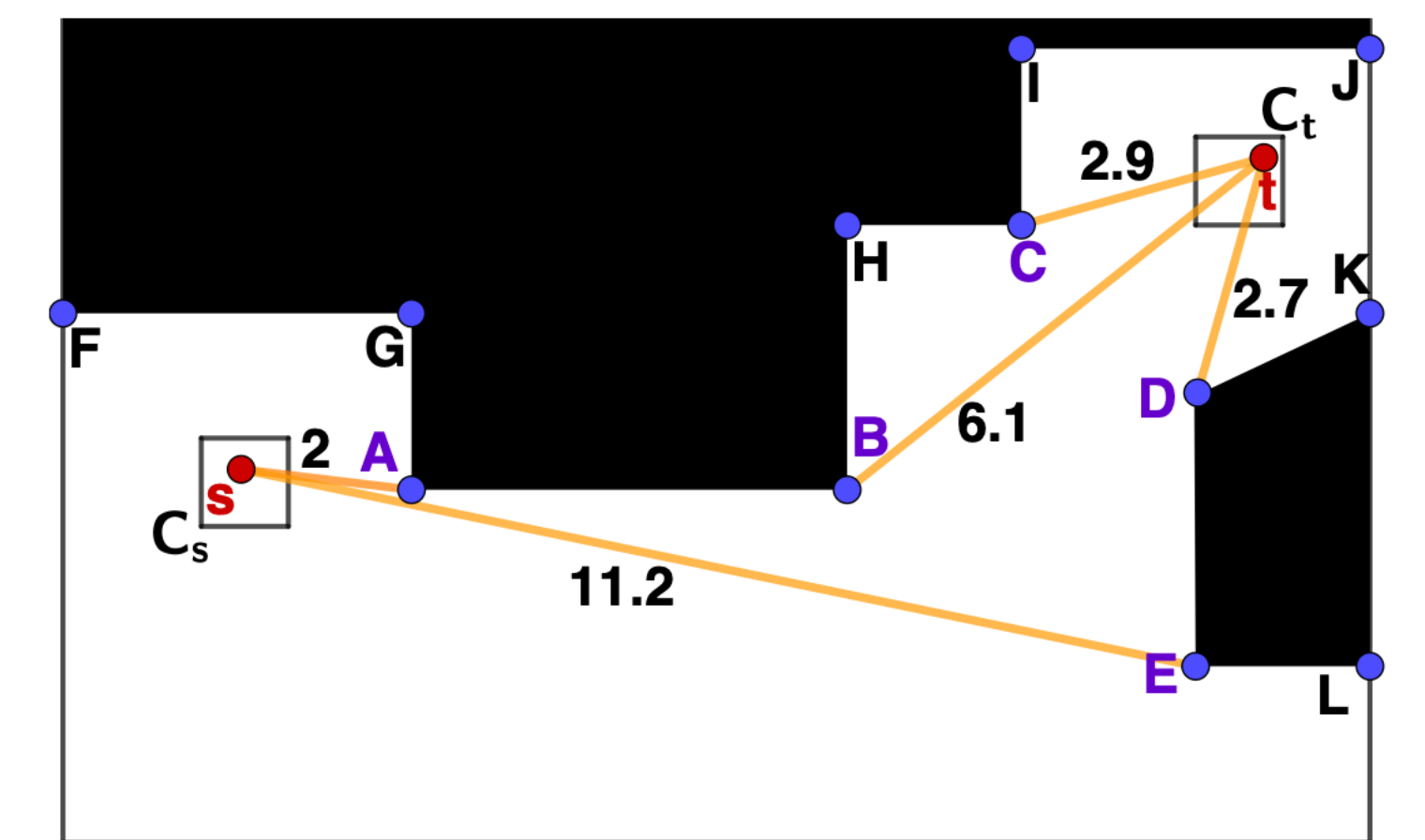
### Offline Preprocessing:

- A visibility graph is constructed on the convex vertices. Hub labels are computed on this visibility. (See the left figure below)
- We superimpose a uniform grid covering the whole map. For each grid cell, we store a list called its visibility list, which consists of every convex vertex such that at least some part of the grid is visible from it.
- For each cell, we create its labels called via labels using its visibility list.
- We sort the hub nodes according to the hub node ID to efficiently join the hub labels of two different cells in online query phase. (See the right figure below)



Vertex	Hub labels
A	(A, 0), (B, 5.1), (E, 10)
B	(B, 0)
C	(B, 5.1), (C, 0)
D	(B, 5.4), (D, 0)
E	(B, 6.1), (D, 5.3), (E, 0)

Figure 2: The visibility graph and the corresponding hub labelling for the figure on the right.



$H(c_s)$	Via Labels $VL_{h_i}(c_s)$	$H(c_t)$	Via Labels $VL_{h_i}(c_t)$
A: (A, 0)		B: (B, 0), (C, 5.1), (D, 5.4)	
B: (A, 5.1), (B, 0), (E, 6.1)		C: (C, 0)	
D: (E, 5.3)		D: (D, 0)	
E: (A, 10), (E, 0)			

Figure 3: In the Euclidean Plane with polygonal obstacles, we show EHL constructed using hub labelling from the left figure.

### Online Query Processing:

- The via-distance for each hub node in both start and target cell is computed. The aim of via-distance is to find the via-label with the minimum distance.
- The shortest distance computation is calculated by using the common hub nodes in the start and target cell. The distance with the minimum cost is then returned as the shortest distance between a start and target.

## Euclidean Hub Labeling Star (EHL\*)

### Motivation:

- Neighboring grid cells often store highly overlapping hub labels.
- Storing duplicate labels across cells wastes memory.
- Merging carefully chosen cells can dramatically reduce memory, with minimal runtime impact.

### Core Idea:

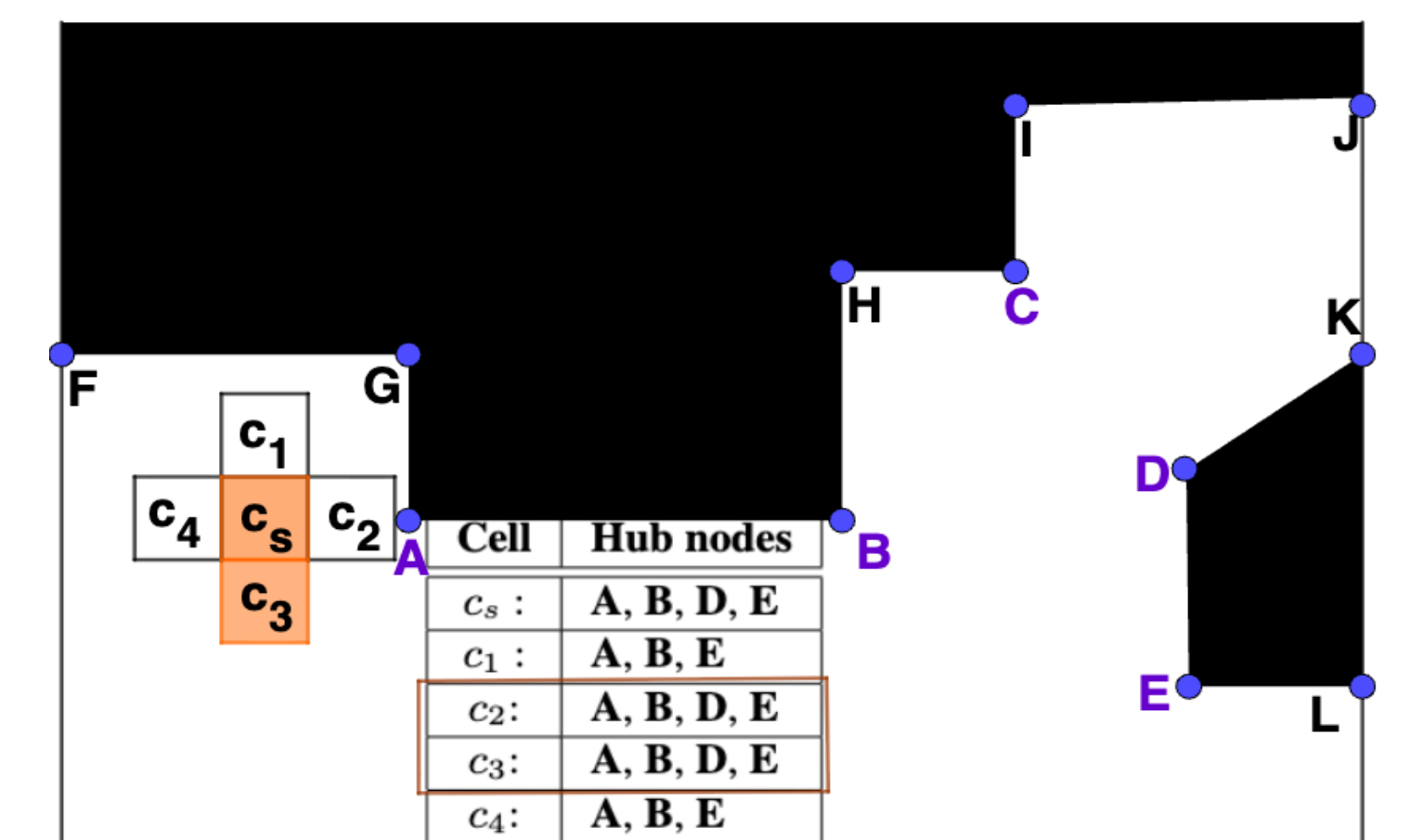
- EHL\* introduces a compression phase that:
- Iteratively merges adjacent grid cells into arbitrary-shaped regions.
  - Maximizes label sharing.
  - Stops automatically when memory usage falls below budget  $B$ .

### Region Selection:

For a region  $e$ , merge with adjacent region  $r$  that maximizes:

$$\frac{|H(r) \cap H(e)|}{|H(r) \cup H(e)|}$$

where  $H(x)$  is the hub set of region  $x$   
→ Maximizes label sharing, minimizes added labels.



$H(c_s)$	Via Labels $VL_{h_i}(c_s)$
A: (A, 0)	
B: (A, 5.1), (B, 0), (E, 6.1), (D, 5.4)	
D: (E, 5.3), (D, 0)	
E: (A, 10), (E, 0)	

Figure 4: Grid cell  $c_s$  with its adjacent neighbors. Via-labels for  $c_s$  after merging  $c_3$  into it. Newly inserted labels are shown in red.

## Workload-Aware EHL\*

### Motivation:

- In many applications (e.g., games), queries are not uniform.
- Some map regions are queried far more frequently.
- Standard EHL/EHL\* treats all regions equally.

### Core Idea:

Exploit known or predicted query distributions to:

- Keep hot regions small.
- Merge cold regions aggressively.
- Achieve better runtime at the same memory budget.

### Workload Scoring:

Each cell  $c$  is assigned:

$$s(c) = 1 + w_c$$

where  $w_c$  is the expected number of queries in  $c$   
→ Higher score = less likely to merge.

### Region Selection:

When merging region  $e$ , choose adjacent region  $r$  maximizing:

$$(1 - \alpha) \frac{|H(r) \cap H(e)|}{|H(r) \cup H(e)|} + \alpha \frac{1}{s(r)}$$

First term: label similarity.

Second term: low workload preference.

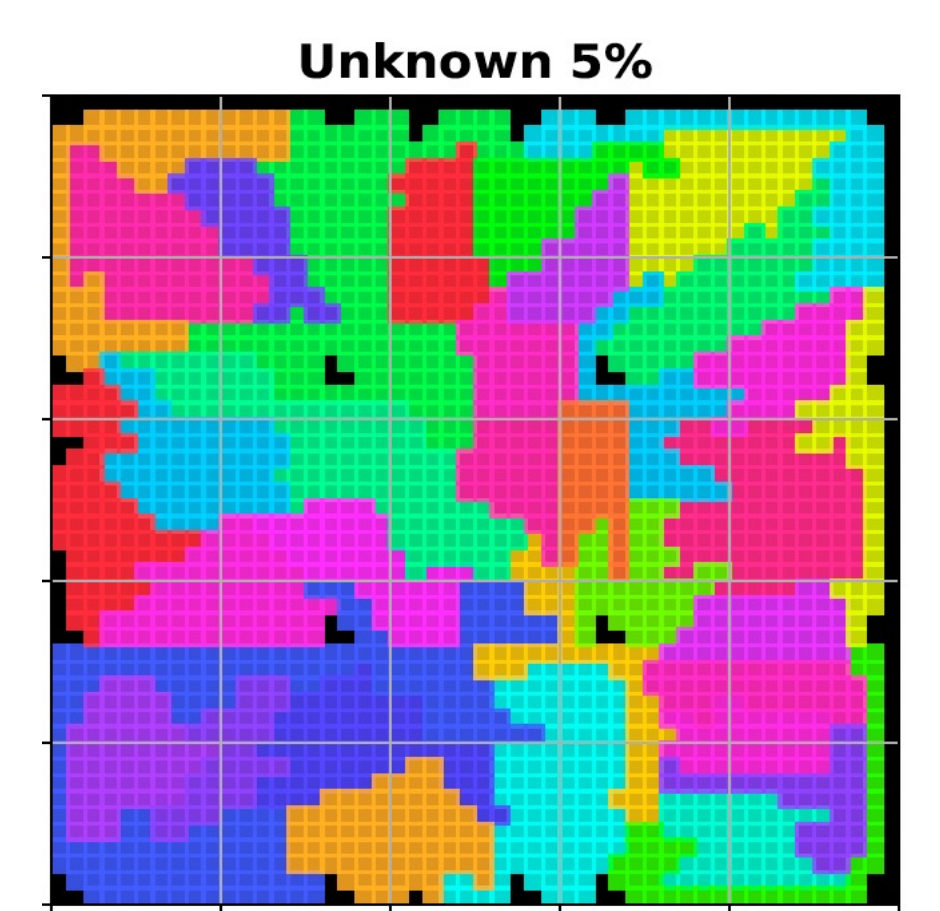


Figure 5: Distribution of merged grids for the arena map for different cluster scenarios at 5% memory. The red rectangles indicate cluster regions on the arena map, while black polygons represent obstacles.