Bounded Suboptimal Path Planning with Compressed Path Databases

Shizhe Zhao[1], Mattia Chiari[2], Adi Botea[3], Alfonso E. Gerevini[2], Daniel Harabor[1], Alessandro Saetti[2], Peter J. Stuckey[1]

[1]Monash University, [2]University of Brescia, [3]Eaton

Intro: the problem

We study the shortest path problem, where the graph is:

- Two-dimensional grid map
- Each cell has up to 8 neighbors:
  - four straights (N, S, W, E)
  - four diagonals (NW, NE, SW, SE)
- No corner cutting

Intro: the problem

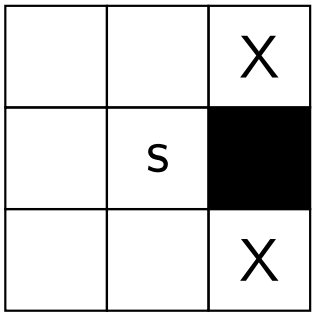
We study the shortest path problem, where the graph is:

- Two-dimensional grid map
- Each cell has up to 8 neighbors:
  - four straights (*N, S, W, E*)
  - four diagonals (*NW, NE, SW, SE*)
- No corner cutting

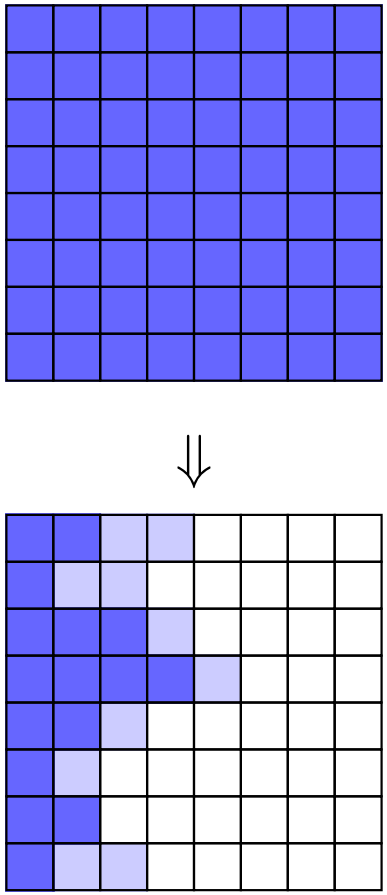
Intro: the problem

We study the shortest path problem, where the graph is:

- Two-dimensional grid map
- Each cell has up to 8 neighbors:
  - four straights (*N, S, W, E*)
  - four diagonals (*NW, NE, SW, SE*)
- No corner cutting



Intro: what are CPDs (Compress Path Databases)?



Compress First move matrix by:
RLE, h-symbol

- Precompute all-pair first moves: $O(n^2)$
- **Compress:** $\ll O(n^2)$
- Search-free path extraction

Contribution: Bounded suboptimal CPD

Motivation:

- Existing approaches have achieved good compression and hard to improve.
- Trade-off between space and suboptimality.

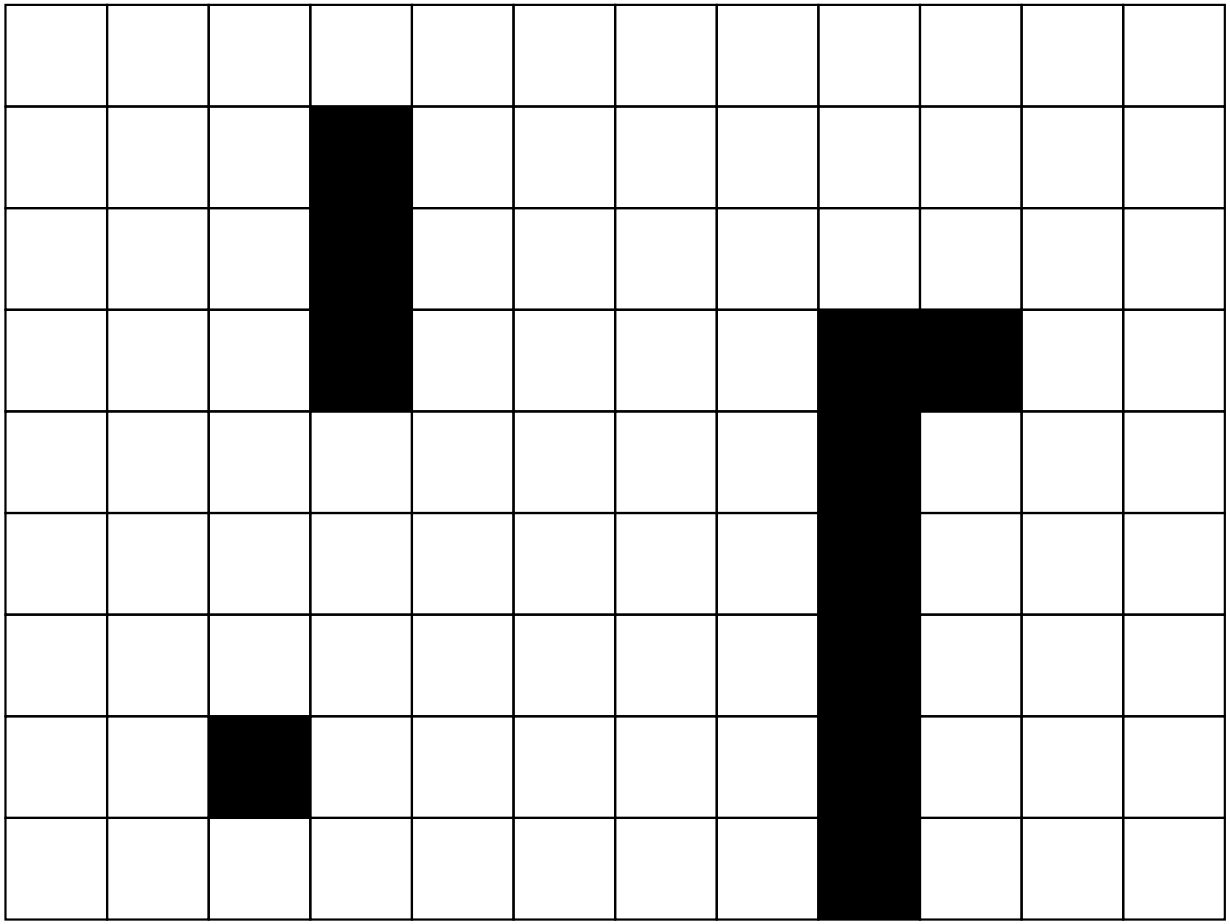
Idea:

- For each node, only store first-move data for a subset of grid nodes C, called *centroid*
  - First-move matrix: $N \times N \rightarrow N \times |C|$
- Each node i belongs to a centroid $C(i)$
- Centroid path (cp for short)

$$cp(s,t) = shortestPath(s, C(t)) + +shortestPath(C(t), t)$$

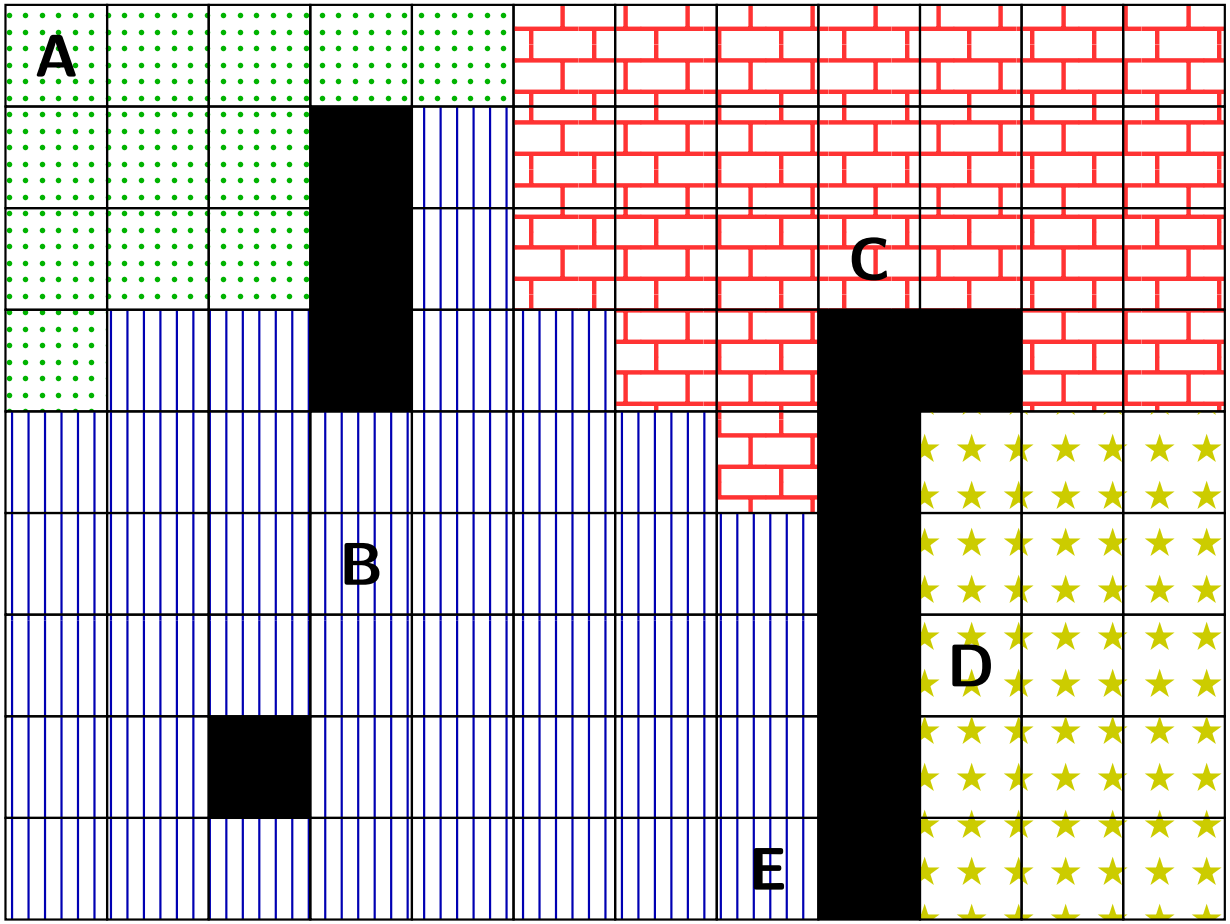
- Bounded suboptimal: $|t, C(t)| \leq \delta$

How to compute centroids?



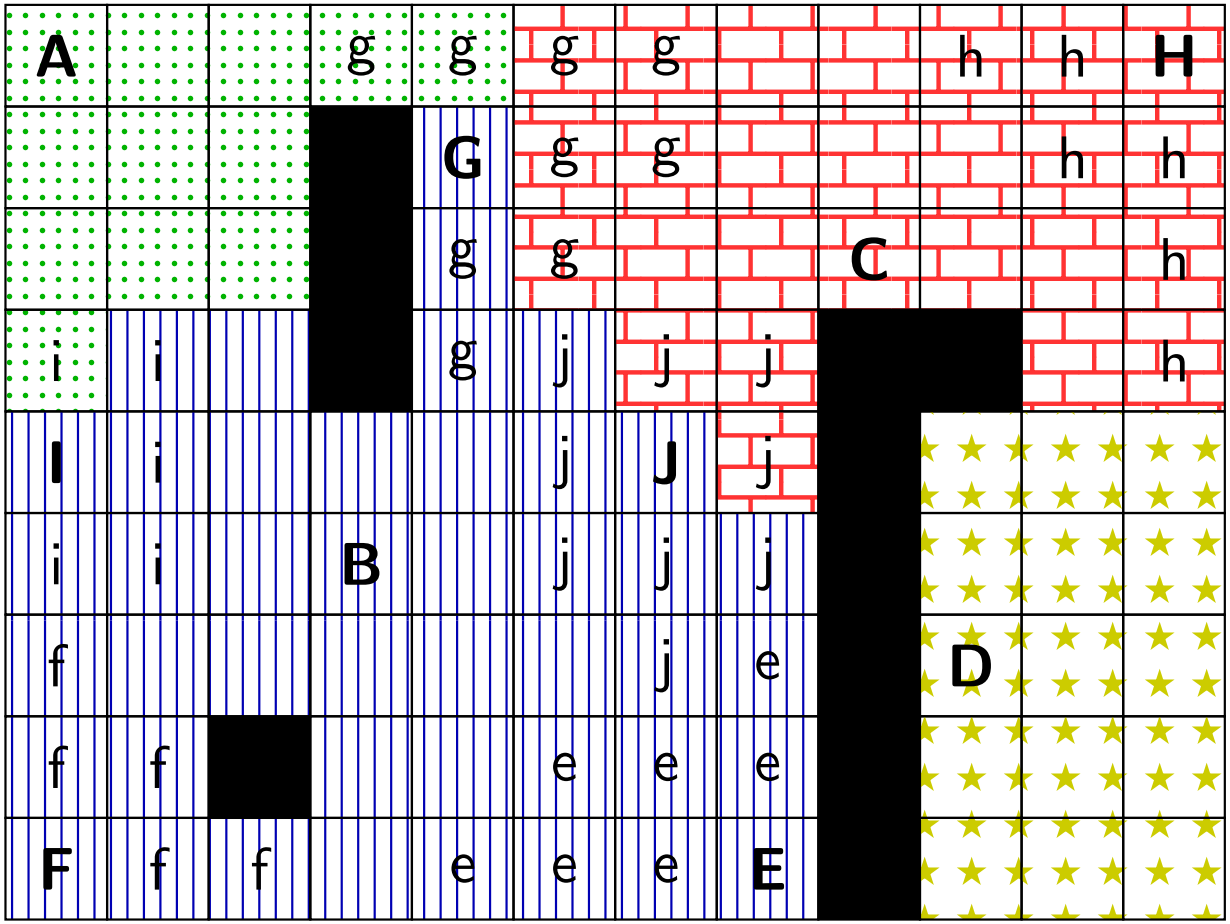
- let $\delta = 3$
- S1: $d(c_i, c_j) \leq 2\delta$
- S2: generate centroids on "borders"
- $\delta \leq d(c_i, c_j) \leq 2\delta$
- $|C_i| \geq \frac{\delta}{2}$
- thus $i \leq \frac{2V}{\delta}$

How to compute centroids?



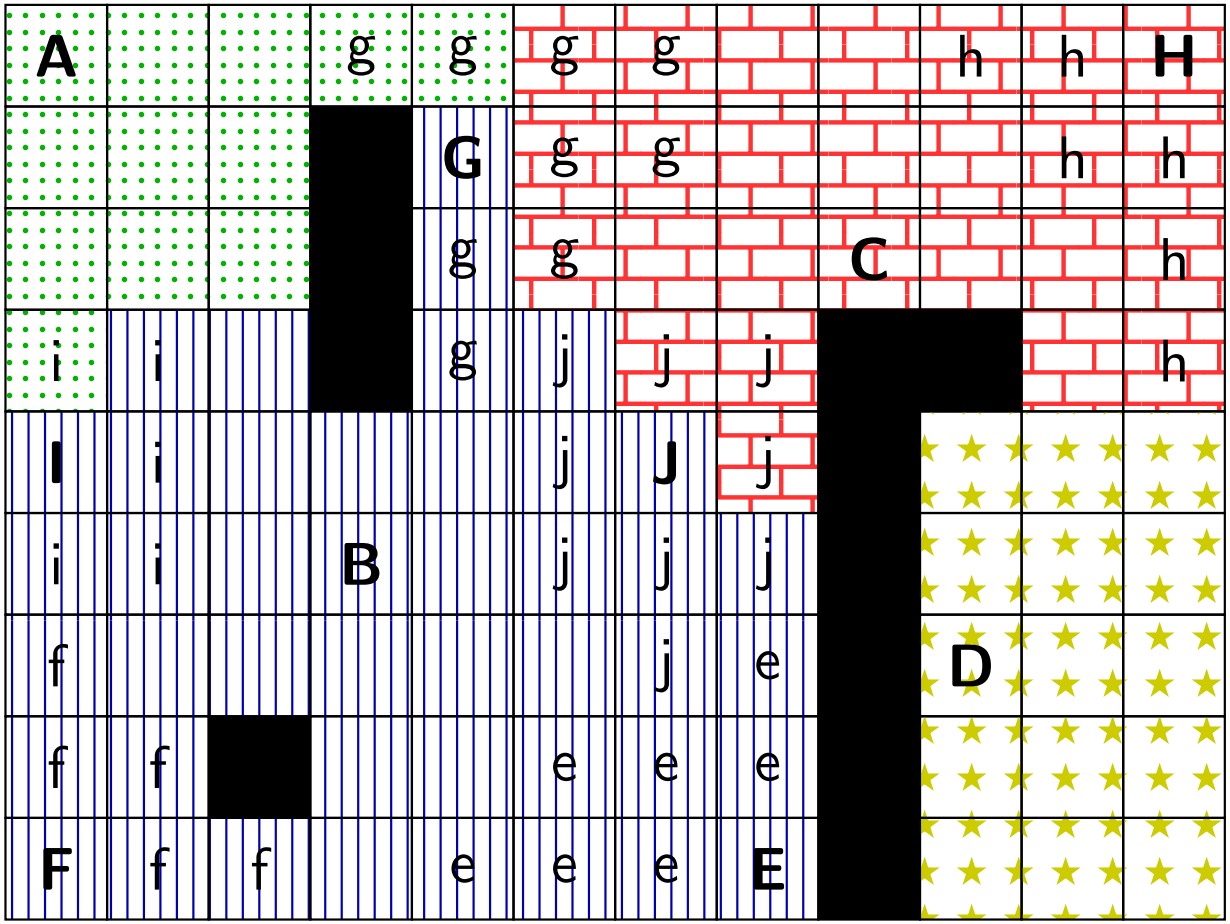
- let $\delta = 3$
- S1: $d(c_i, c_j) \leq 2\delta$
- S2: generate centroids on "borders"
- $\delta \leq d(c_i, c_j) \leq 2\delta$
- $|C_i| \geq \frac{\delta}{2}$
- thus $i \leq \frac{2V}{\delta}$

How to compute centroids?



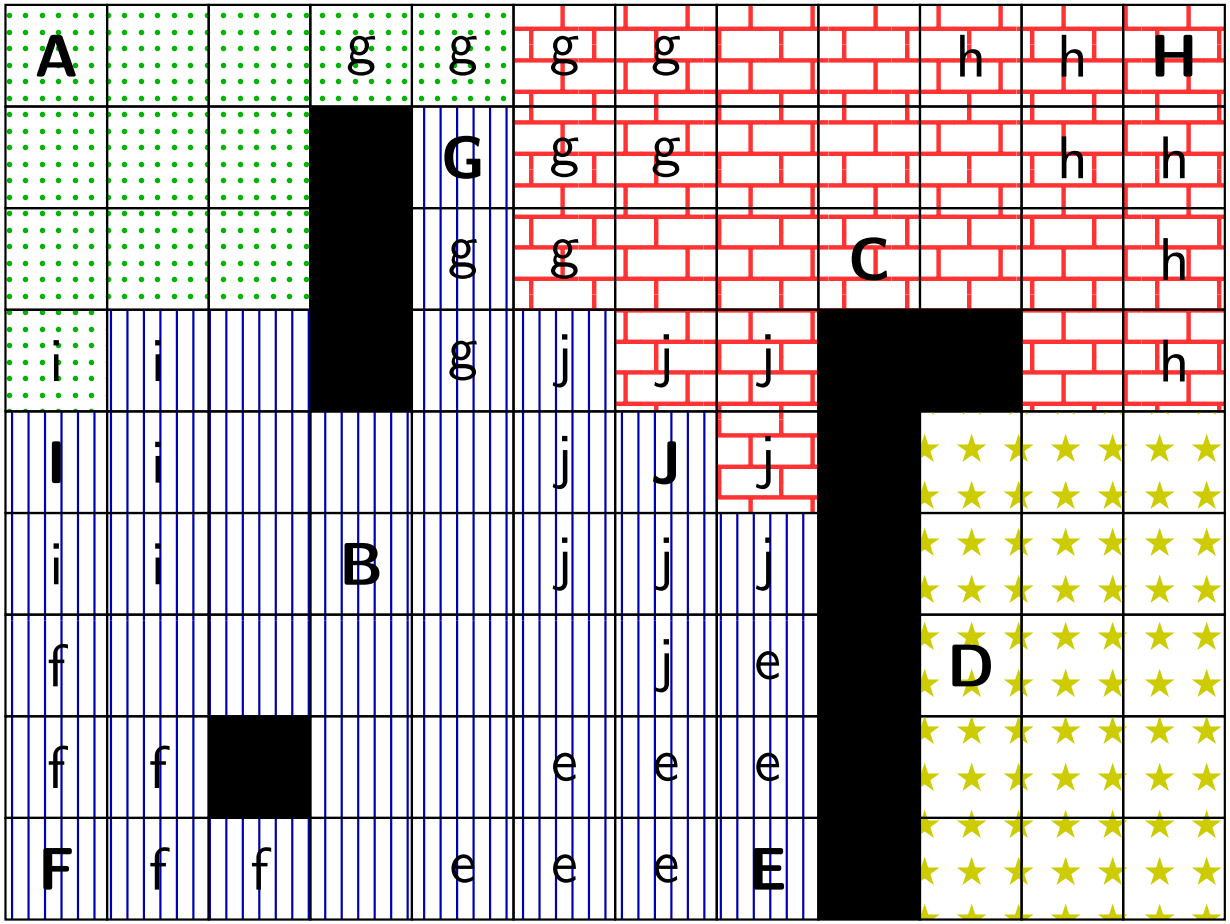
- let $\delta = 3$
- S1: $d(c_i, c_j) \leq 2\delta$
- S2: generate centroids on "borders"
- $\delta \leq d(c_i, c_j) \leq 2\delta$
- $|C_i| \geq \frac{\delta}{2}$
- thus $i \leq \frac{2V}{\delta}$

How to compute centroids?



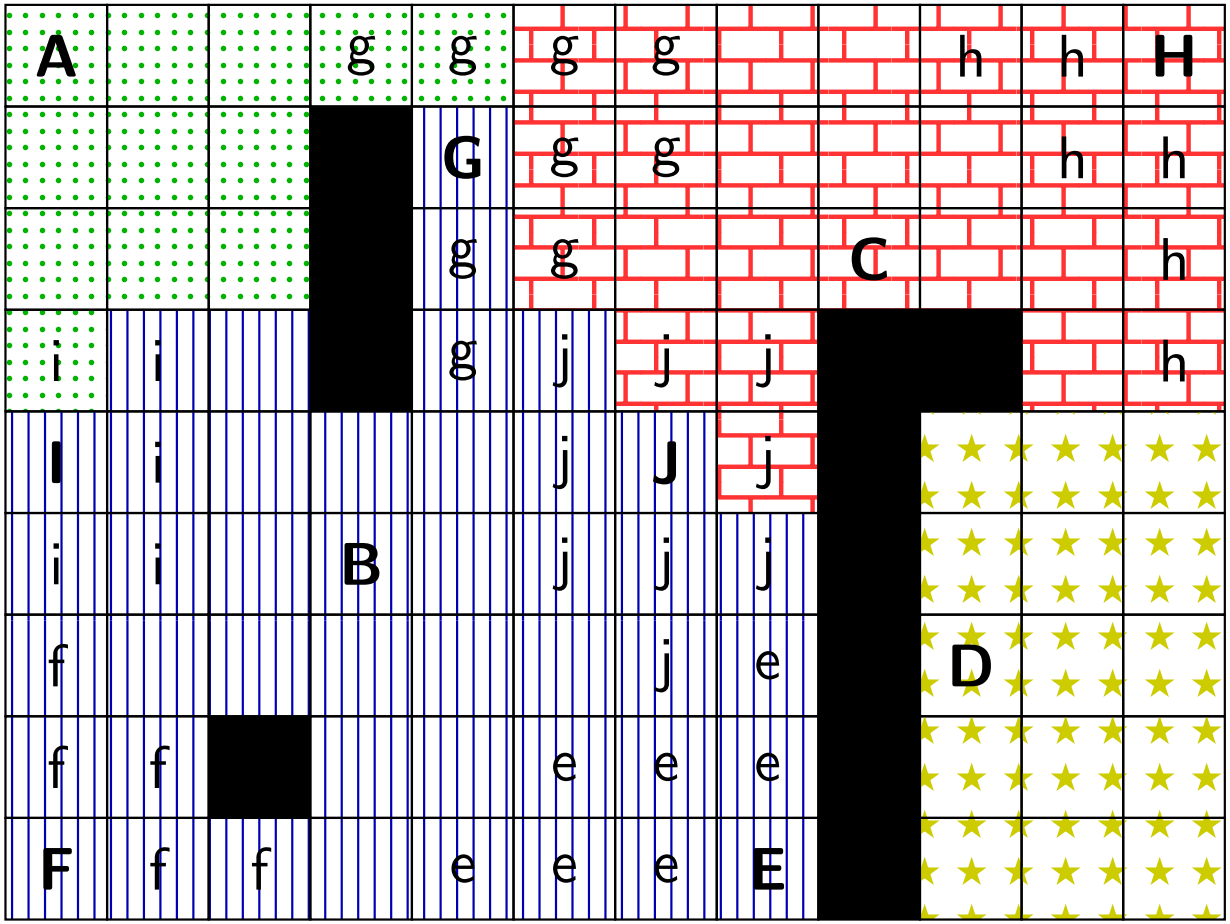
- let $\delta = 3$
- S1: $d(c_i, c_j) \leq 2\delta$
- S2: generate centroids on "borders"
- $\delta \leq d(c_i, c_j) \leq 2\delta$
- $|C_i| \geq \frac{\delta}{2}$
- thus $i \leq \frac{2V}{\delta}$

How to compute centroids?



- let $\delta = 3$
- S1: $d(c_i, c_j) \leq 2\delta$
- S2: generate centroids on "borders"
- $\delta \leq d(c_i, c_j) \leq 2\delta$
- $|C_i| \geq \frac{\delta}{2}$
- thus $i \leq \frac{2V}{\delta}$

How to compute centroids?



- let $\delta = 3$
- S1: $d(c_i, c_j) \leq 2\delta$
- S2: generate centroids on "borders"
- $\delta \leq d(c_i, c_j) \leq 2\delta$
- $|C_i| \geq \frac{\delta}{2}$
- thus $i \leq \frac{2V}{\delta}$

Optimizations: Reverse CPD

- Original CPD (forward): $T(i, j)$ - the first move from i to j;
- Reverse CPD: $T'(i, j)$ - the first move from j to i;

Advantages of reverse-CPD:

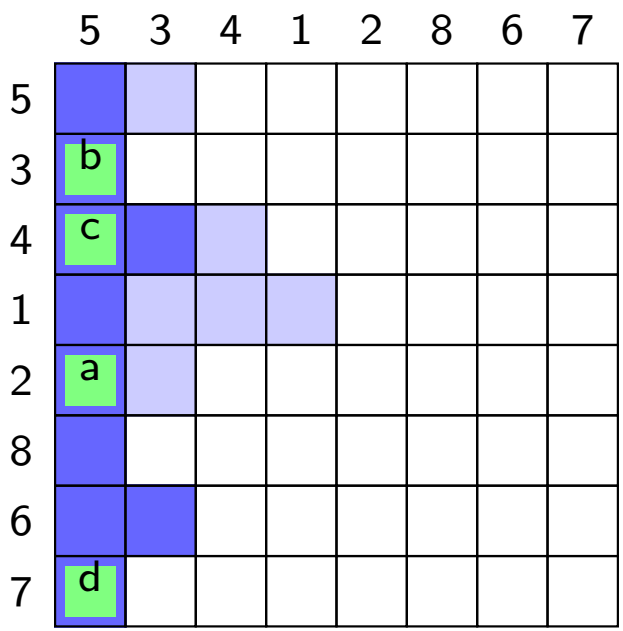
- get more space reduction from the centroids idea
- faster path extraction

Optimizations: Reverse CPD

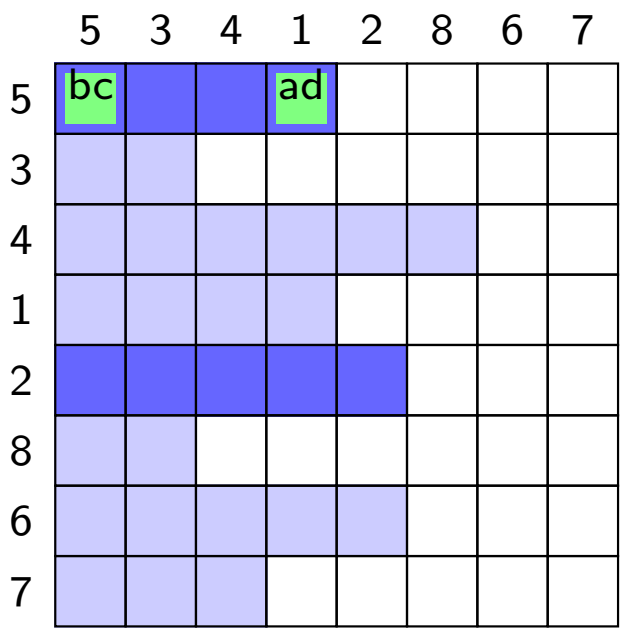
Advantages of reverse-CPD:

- get more space reduction from the centroids idea
- faster path extraction

Forward: 16, 10 (with centroids)

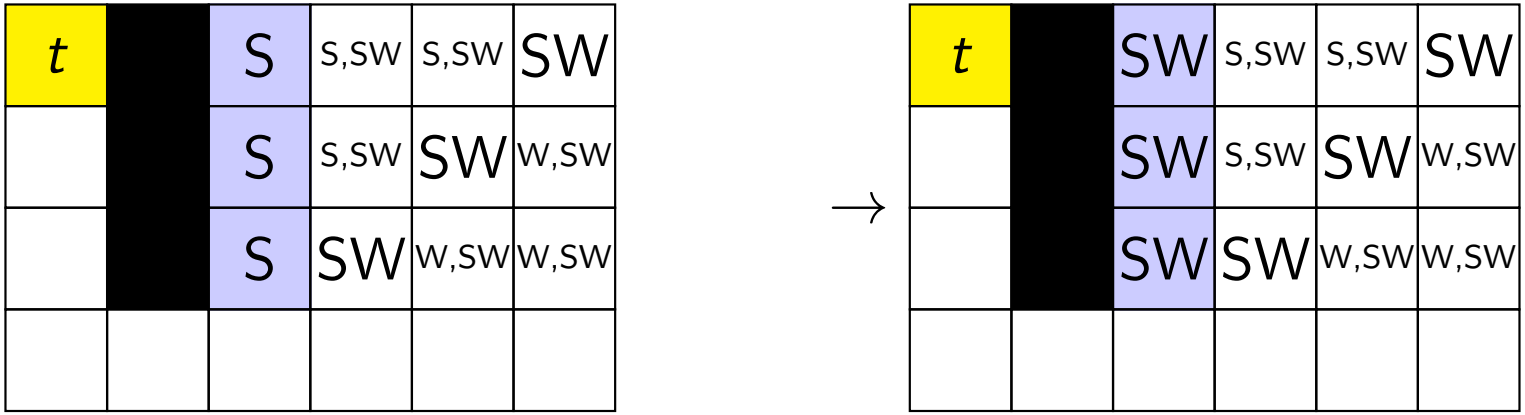


Reverse runs: 31, 9 (with centroids)



- $C = \{2, 5\}$
- Path: (2, 3, 4, 7, 5)

Optimizations: encode "illegal" Moves



- Encoding S to SW allows us to compress the rectangle region to SW.
- When look-up column-3 symbols, we know SW is illegal, thus we can decode it to a valid move S.

Experiment: Set up

- Benchmark GPPC 2012: 105 game maps
- $\delta = 0, 2, 4, 8, 16, 32, 64$
- Notation
  - Forward Centroid-CPD: fwd_δ
  - Reverse Centroid-CPD: rev_δ
  - Full CPD (competitor): fwd_0
- Evaluation: size reduction ratio - $\frac{|fwd_0|}{|rev_\delta|}$ or $\frac{|fwd_0|}{|fwd_\delta|}$

Experiment: Result (reduction)

Size reduction ratio: $\frac{|fwd_0|}{|rev_\delta|}$ or $\frac{|fwd_0|}{|fwd_\delta|}$

δ	type	mean	min	25%	50%	75%	max
2	`rev`	0.33	0.01	0.12	0.20	0.35	1.57
	`fwd`	0.97	0.85	0.93	0.98	1.01	1.19
4	`rev`	0.82	0.02	0.45	0.72	1.05	2.34
	`fwd`	1.13	0.85	1.02	1.12	1.24	1.68
8	`rev`	1.54	0.04	1.13	1.56	1.99	3.45
	`fwd`	1.34	0.86	1.08	1.27	1.56	2.90
16	`rev`	2.59	0.09	1.87	2.48	3.12	7.39
	`fwd`	1.60	0.86	1.14	1.36	1.86	4.90
32	`rev`	3.81	0.21	2.39	3.18	4.16	17.88
	`fwd`	1.84	0.87	1.19	1.42	2.10	7.20
64	`rev`	4.93	0.44	2.70	3.64	5.12	32.43
	`fwd`	2.08	0.87	1.24	1.50	2.21	9.65

Experiment: Result (speed up)

Speed up ratio: $\frac{Time(fwd_0)}{Time(rev_\delta)}$ *or* $\frac{Time(fwd_0)}{Time(fwd_\delta)}$

	mean	min	25%	50%	75%	max
$\texttt{rev}_{16}$	1.839	0.061	1.311	1.747	2.162	235.606
$\texttt{rev}_{32}$	1.738	0.031	1.194	1.666	2.091	229.882
$\texttt{rev}_{64}$	1.580	0.008	0.998	1.490	1.953	207.992
$\texttt{fwd}_{16}$	1.209	0.013	1.038	1.125	1.312	175.824
$\texttt{fwd}_{32}$	1.233	0.033	1.041	1.144	1.355	163.937
$\texttt{fwd}_{64}$	1.230	0.012	1.013	1.139	1.389	184.361

The speedup decreases as the compression increases due to overheads from checking the heuristic direct path in centroid-region.

Conclusion and Future Work

- Reverse CPDs shrink more aggressively, eventually overpassing forward CPDs
- Reverse CPDs are faster on path extraction
- In future work:
  - Improve the compression of reverse CPD
  - Use suboptimal-CPD as an admissible heuristic in search