Zero-Aware Pattern Databases with 1-Bit Compression for Sliding Tile Puzzles

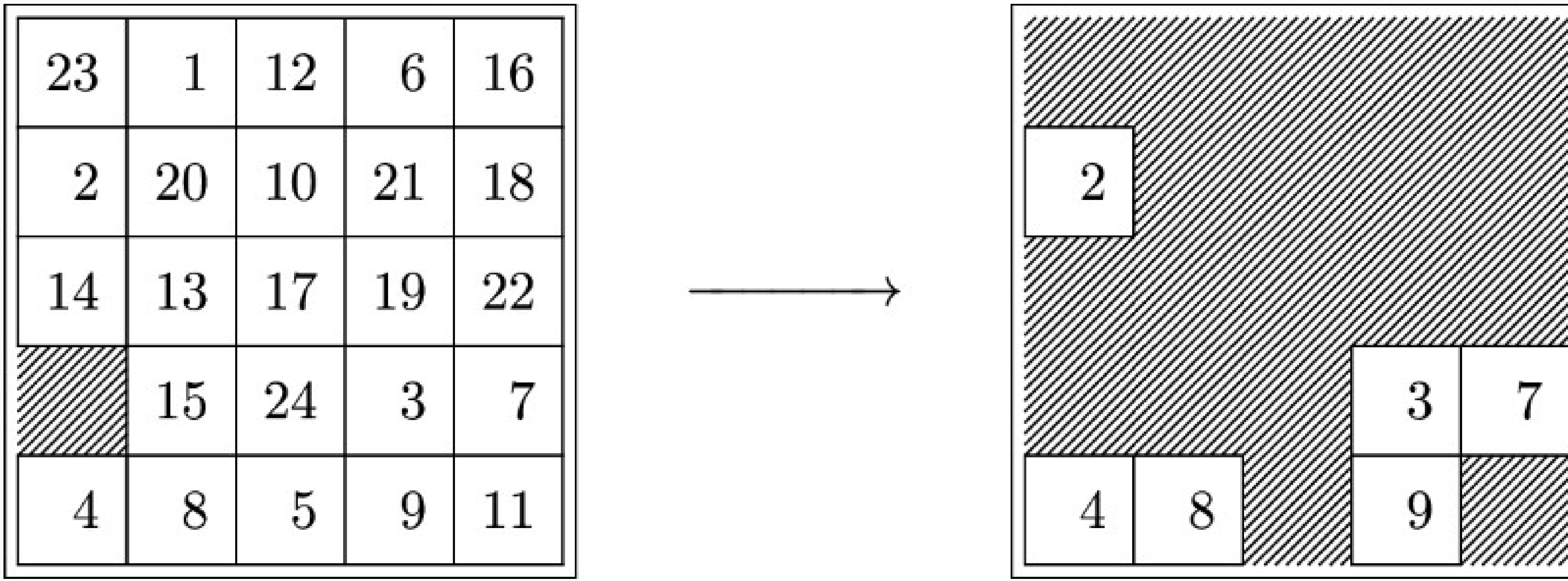
Robert Clausecker, **Alexander Reinefeld**

<clausecker@zib.de> *<ar@zib.de>*

motivation

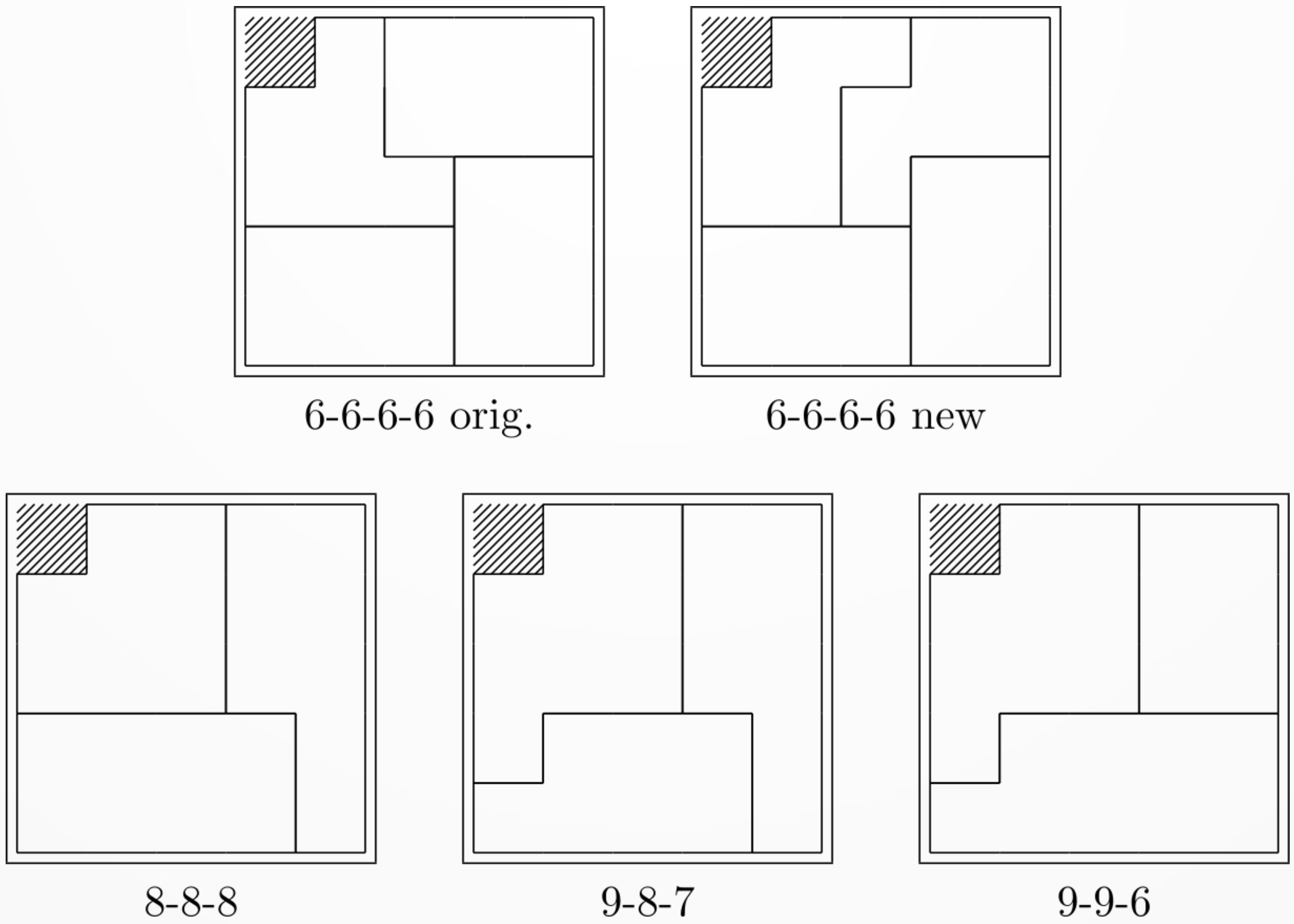
- sliding tile puzzles are a testbed problem for heuristic search
- currently best known heuristics:
  additive pattern databases (APDBs)
- can we do better?

additive pattern databases



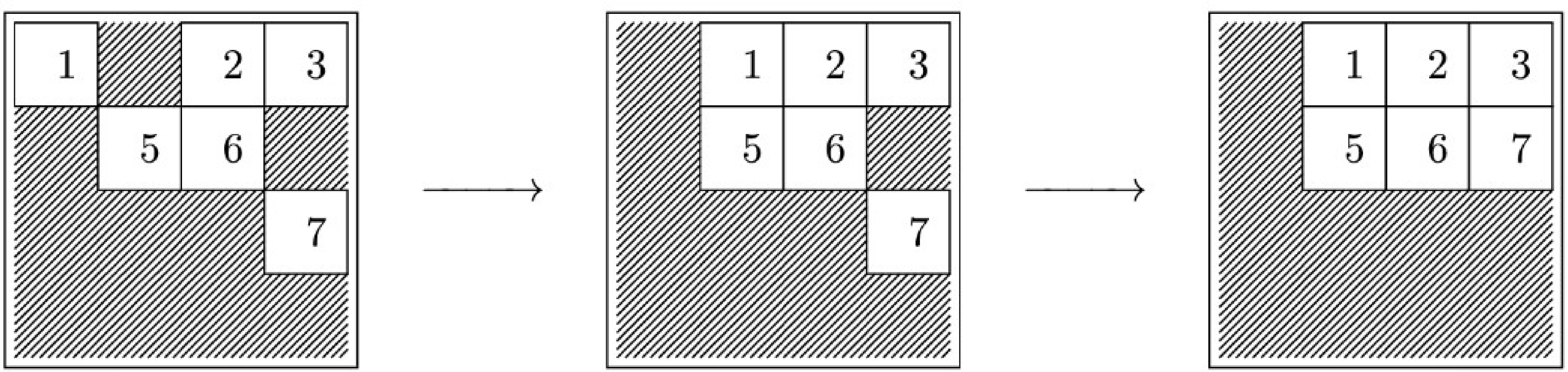
a 24 puzzle and its {2, 3, 4, 7, 8, 9} APDB

additive pattern databases



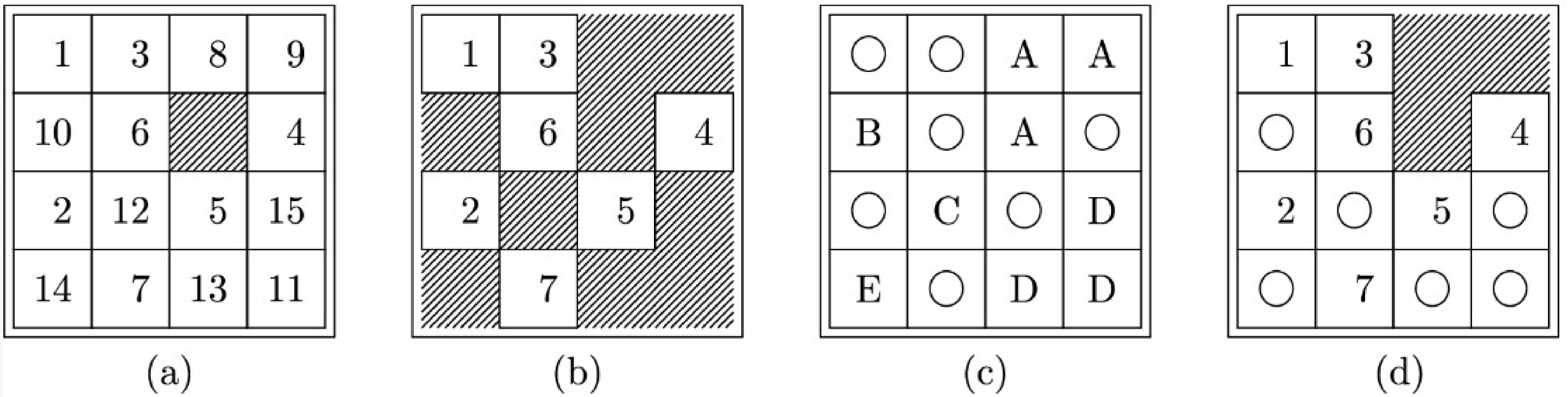
some partitionings of the 24-puzzle

limitations



h = 2 is predicted by the APDB but 2 moves are not sufficient

tracking the blank (zero tile)



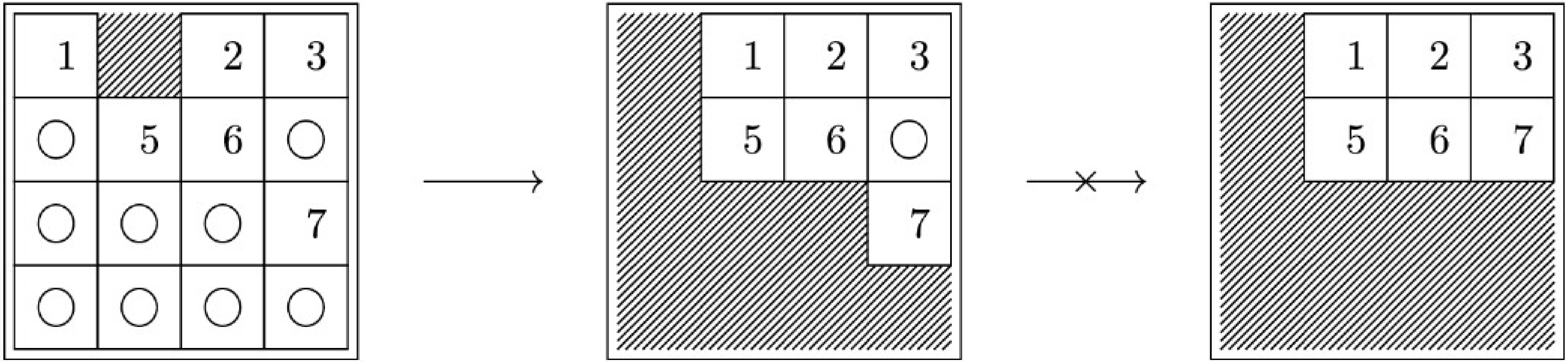
(a) a configuration of the 15 puzzle

(b) as seen by the {1, 2, 3, 4, 5, 6, 7} APDB

(c) its possible zero-tile regions A–E

(d) as seen by the {1, 2, 3, 4, 5, 6, 7} ZPDB

limitations revisited

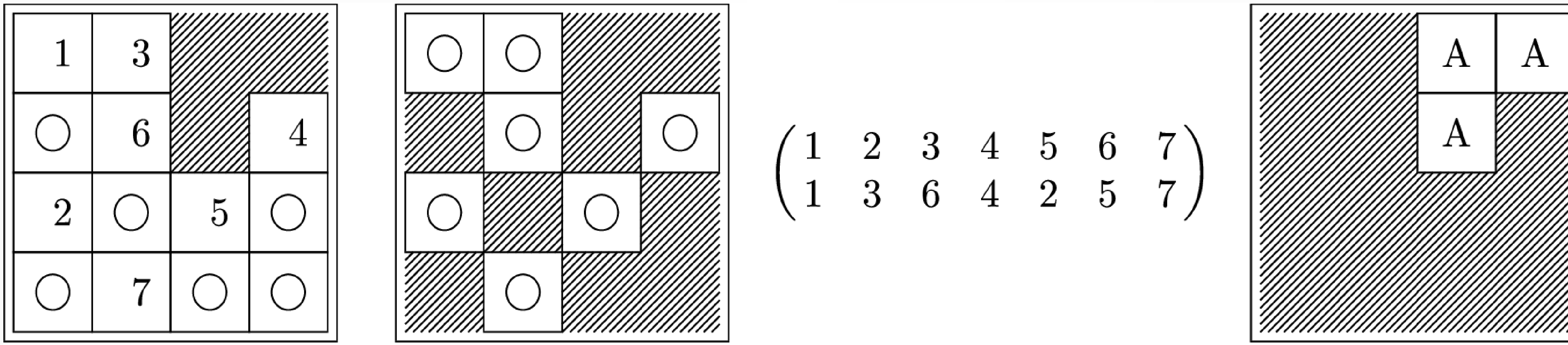


the ZPDB does not predict a 2 move solution

space considerations

k	APDB size	ZPDB size	avg	max
2	600	608	1.01	2
3	13 800	14 472	1.04	2
4	303 600	339 048	1.12	3
5	6 375 600	7 871 280	1.23	4
6	127 512 000	181 008 000	1.42	5
7	2 422 728 000	4 066 655 040	1.68	6
8	43 609 104 000	87 358 400 640	2.00	7
9	741 354 768 000	1 759 513 674 240	2.37	8
10	11 861 676 288 000	32 787 717 580 800	2.76	10
11	177 925 144 320 000	560 680 553 664 000	3.15	11
12	2 490 952 020 480 000	8 749 801 518 796 800	3.51	13

representation



$e = (m, p, r)$ $m = 2027$ $p = 198$ $r = 0$

representing a ZPDB entry by tile ***m****ap,* ***p****ermutation,*
and zero tile ***r****egion*

enhanced compression

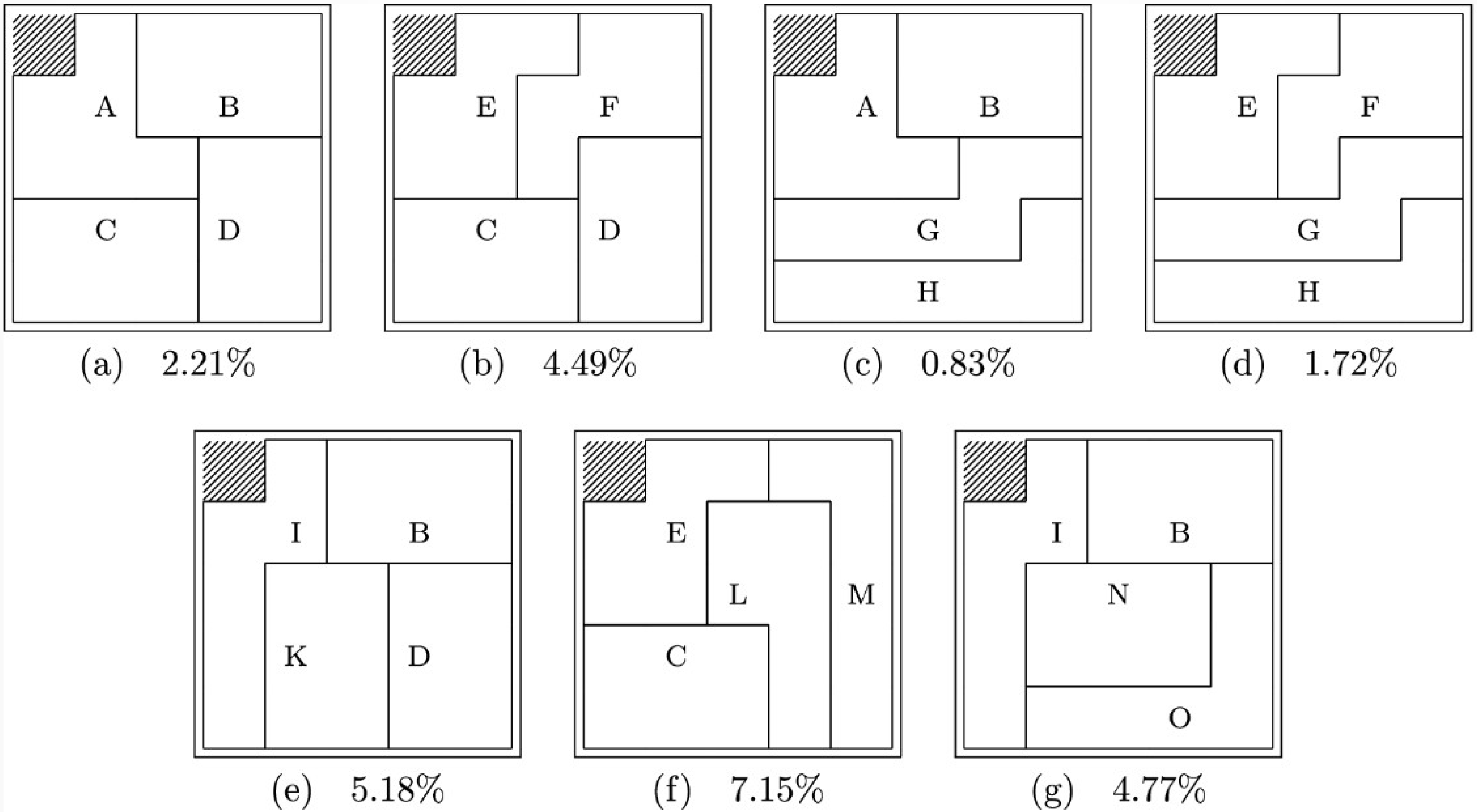
- ZPDBs heuristics are consistent with unit weight
- the difference between adjacent *h*-values is ±1
- knowing the change of *h*-value is sufficient for search
- [Breyer2010a] represented PDB entries mod 3 in $\log_2 3 \approx 1.6$ bits per entry using this idea.
- but we can do better
  - if we store entries mod 4, we need 2 bits per entry
  - the least bit can be omitted for bipartite spaces
  - giving us a representation with 1 bit per entry

building collections

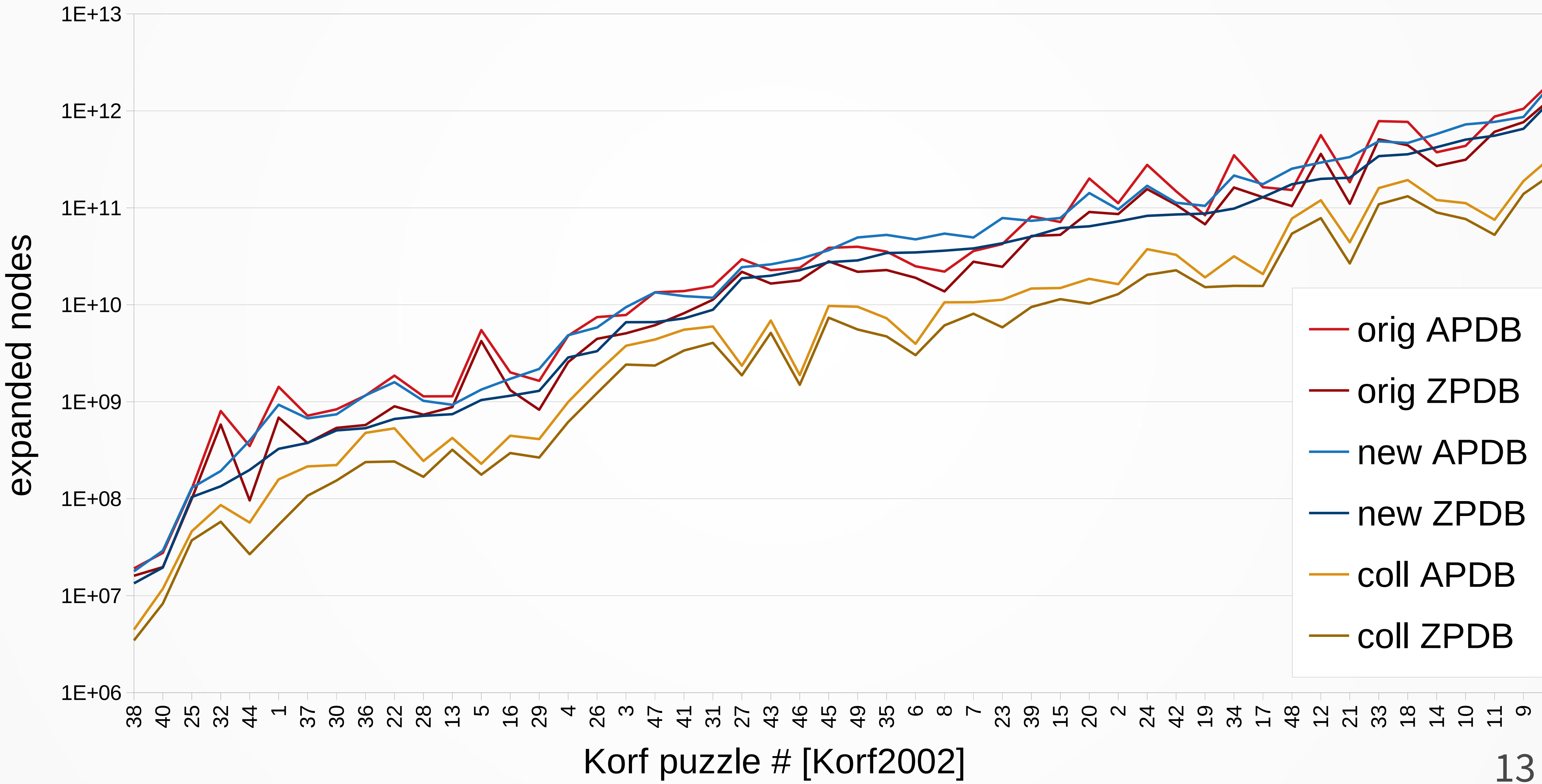
the general approach:

1. start with a set of known good partitionings
2. sample the *h*-values of $n = 10^8$ random puzzles
3. for each partitioning, count how often it was an *h*-values sole support
4. remove partitionings from the collection which rarely supported the maximum *h*-value
5. add new partitionings to replace them
6. repeat steps 2–5 until no improvements are found

building collections



results



1E+13
1E+12
1E+11
1E+10
1E+09
1E+08
1E+07
1E+06
expanded nodes
38 40 25 32 44 1 37 30 36 22 28 13 5 16 29 4 26 3 47 41 31 27 43 46 45 49 35 6 8 7 23 39 15 20 2 24 42 19 34 17 48 12 21 33 18 14 10 11 9 50
Korf puzzle # [Korf2002]
orig APDB
orig ZPDB
new APDB
new ZPDB
coll APDB
coll ZPDB

conclusions

- the zero tile can be tracked explicitly at reasonable memory and performance costs
- tracking the zero tile explicitly reduces
  IDA* nodes by 35%
- such PDBs can be represented with 1 bit per entry
- a small catalogue of pattern databases additionally reduces
  IDA* nodes by 80.2%
  on average