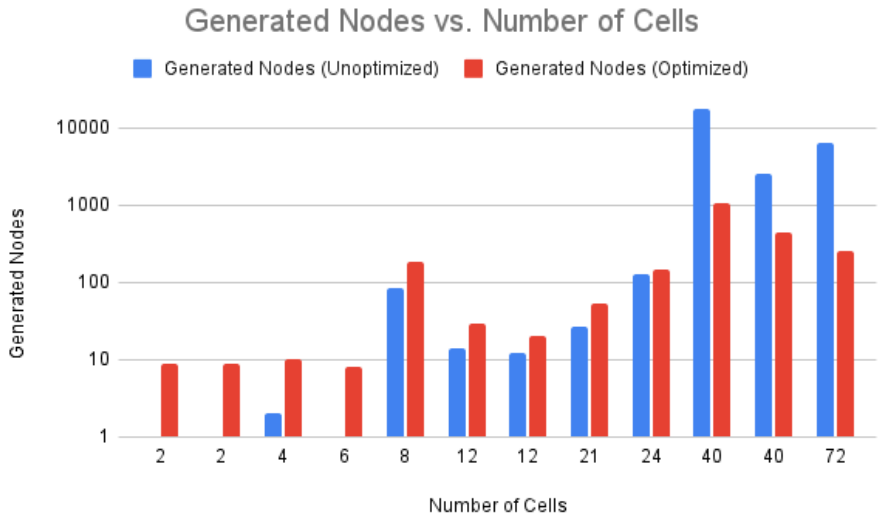
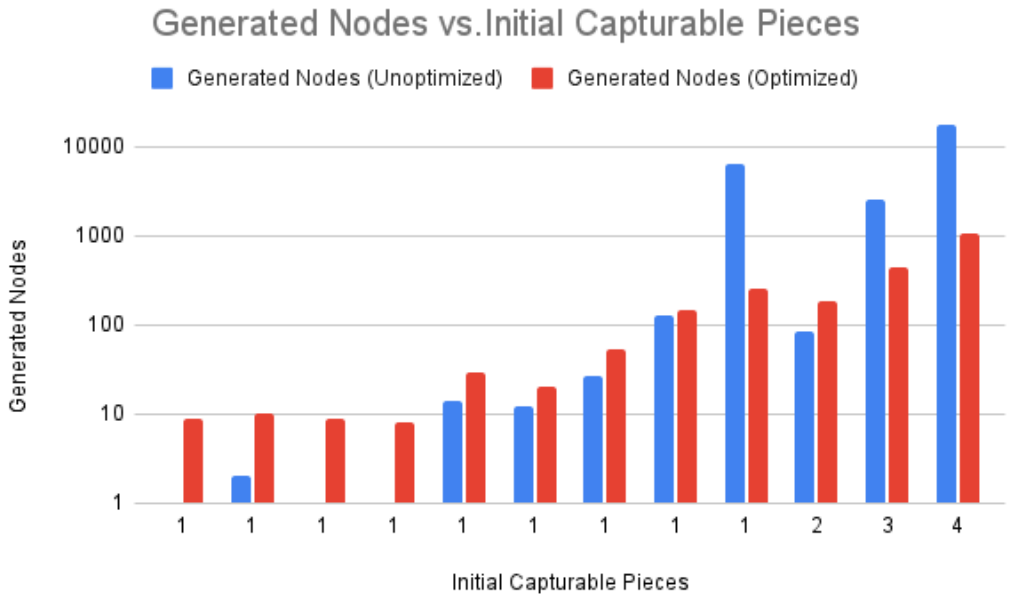
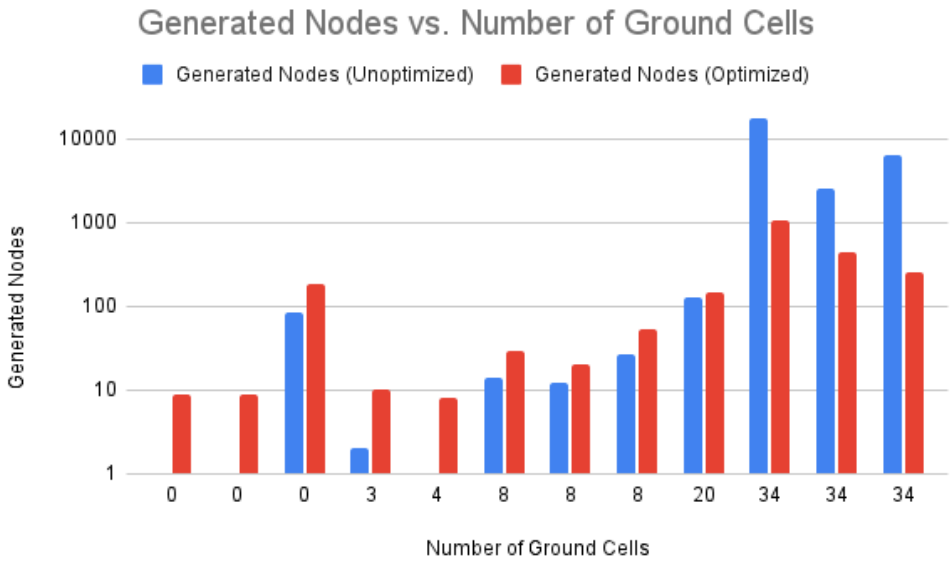
Data

Puzzle	Without optimization	With optimization
1	Expanded nodes: 2 Generated nodes: 1 Duplicated nodes: 0 Time (seconds): 0.000018	Expanded nodes: 2 Generated nodes: 9 Duplicated nodes: 0 Time (seconds): 0.000021
2	Expanded nodes: 2 Generated nodes: 2 Duplicated nodes: 0 Time (seconds): 0.000022	Expanded nodes: 2 Generated nodes: 10 Duplicated nodes: 1 Time (seconds): 0.000024
3	Expanded nodes: 2 Generated nodes: 1 Duplicated nodes: 0 Time (seconds): 0.000012	Expanded nodes: 2 Generated nodes: 9 Duplicated nodes: 0 Time (seconds): 0.000021
4	Expanded nodes: 2 Generated nodes: 1 Duplicated nodes: 0 Time (seconds): 0.000023	Expanded nodes: 2 Generated nodes: 8 Duplicated nodes: 0 Time (seconds): 0.000024
5	Expanded nodes: 1684 Generated nodes: 17523 Duplicated nodes: 0 Time (seconds): 0.097126	Expanded nodes: 73 Generated nodes: 1048 Duplicated nodes: 609 Time (seconds): 0.001484
6	Expanded nodes: 238 Generated nodes: 2592 Duplicated nodes: 0 Time (seconds): 0.011190	Expanded nodes: 31 Generated nodes: 440 Duplicated nodes: 254 Time (seconds): 0.000730
7	Expanded nodes: 5 Generated nodes: 14 Duplicated nodes: 0 Time (seconds): 0.000056	Expanded nodes: 4 Generated nodes: 30 Duplicated nodes: 6 Time (seconds): 0.000038
8	Expanded nodes: 4 Generated nodes: 12 Duplicated nodes: 0 Time (seconds): 0.000098	Expanded nodes: 3 Generated nodes: 20 Duplicated nodes: 4 Time (seconds): 0.000031
9	Expanded nodes: 15 Generated nodes: 27 Duplicated nodes: 0 Time (seconds): 0.000835	Expanded nodes: 7 Generated nodes: 54 Duplicated nodes: 7 Time (seconds): 0.000073
10	Expanded nodes: 14 Generated nodes: 86 Duplicated nodes: 0 Time (seconds): 0.000320	Expanded nodes: 14 Generated nodes: 190 Duplicated nodes: 65 Time (seconds): 0.000130
11	Expanded nodes: 13 Generated nodes: 130 Duplicated nodes: 0 Time (seconds): 0.000648	Expanded nodes: 10 Generated nodes: 150 Duplicated nodes: 90 Time (seconds): 0.000149
12	Expanded nodes: 1151 Generated nodes: 6366 Duplicated nodes: 0 Time (seconds): 0.126177	Expanded nodes: 21 Generated nodes: 260 Duplicated nodes: 133 Time (seconds): 0.000521

Generated Nodes Analysis



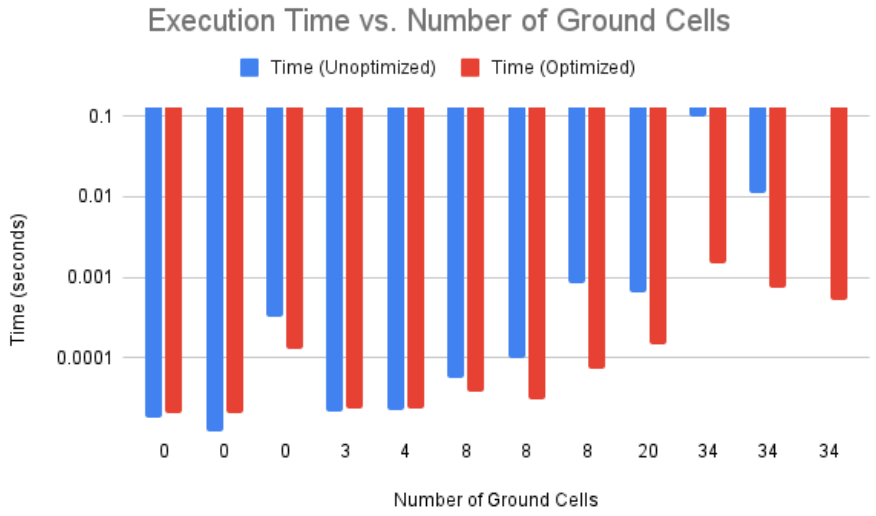


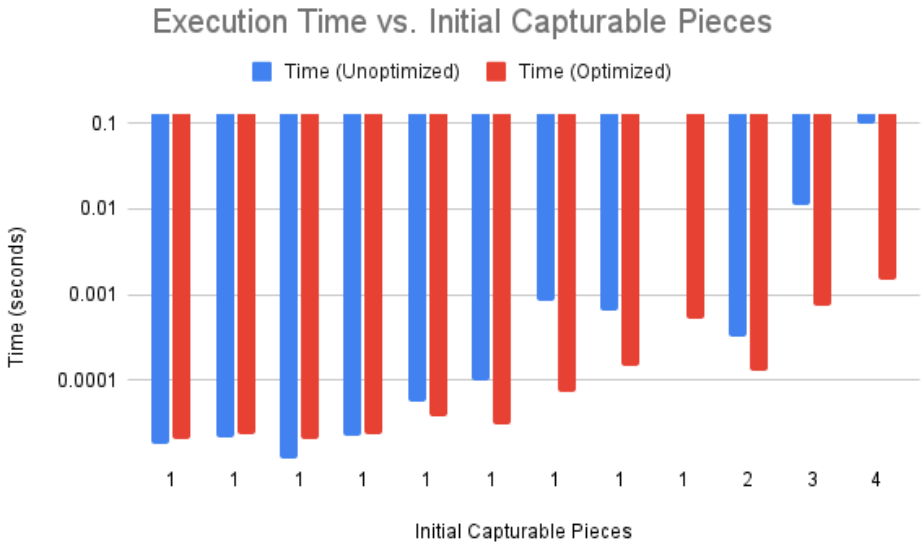


Explanation of Results

- **Trend:**
  Across all metrics (number of cells, ground cells, and initial capturable pieces):
  - The number of generated nodes grows exponentially, particularly in the unoptimized version, as puzzle complexity increases.
  - For example, with 40 cells, the unoptimized solver generates 17,523 nodes, compared to 1,048 nodes in the optimized version.
- **Comparison:**
  The optimized solver consistently generates fewer nodes across all scenarios:
  - The difference becomes more pronounced with increased complexity, such as with 34 ground cells where the unoptimized version generates 6,366 nodes and the optimized version generates only 260 nodes.
  - Similarly, for 4 initial capturable pieces, the unoptimized solver generates 10,000+ nodes, while the optimized solver reduces this to around 4,000 nodes.
- I**mpact:**
  The optimization significantly reduces computational costs by minimizing unnecessary node exploration:
  - The reduction is particularly effective in larger puzzles, such as those with 72 cells or 34 ground cells.
  - This improves the solver's scalability and efficiency in handling more complex configurations of cells, ground cells, and capturable pieces.

Execution Time Analysis





Explanation of Results

- **Trend:**
  Across all metrics (number of cells, ground cells, and initial capturable pieces):
  - Both the unoptimized and optimized versions show relatively constant execution times for simpler puzzles (fewer cells, ground cells, or capturable pieces).
  - As complexity increases, execution time grows, particularly for larger puzzles (e.g., 40 and 72 cells, 34 ground cells, and 4 capturable pieces).
- **Comparison:**
  - Smaller puzzles: Both versions perform similarly, with minimal execution time differences.
  - Larger puzzles: The optimized solver starts showing a clear advantage:
    - For puzzles with 40 cells, execution time is reduced from 0.097 seconds (unoptimized) to 0.0015 seconds (optimized).
    - For 34 ground cells, the optimized version reduces time to 0.0005 seconds, compared to 0.126 seconds in the unoptimized version.
    - Similarly, with 4 capturable pieces, time decreases from 0.01 seconds (unoptimized) to 0.005 seconds (optimized).
- **Impact:**
  The optimization provides only modest improvements for smaller puzzles but significantly reduces execution time as complexity increases:
  - The optimized solver scales better with more cells, ground cells, and capturable pieces, keeping execution times much lower.
  - This demonstrates the optimization's effectiveness in handling complex puzzles, improving efficiency in more challenging scenarios.

Complexity Growth:

The data shows exponential growth in terms of both the number of generated nodes and execution time as puzzle complexity increases (i.e., as the number of cells, ground cells, or initial capturable pieces grows). This is particularly evident in the unoptimized version, where both the number of generated nodes and execution time increase rapidly for larger puzzles. The exponential nature suggests that the solver's complexity is driven by the number of possible states or moves that need to be explored, which increases drastically as the puzzle grid size and complexity rise.

Computational Benefit of the Optimization:

The optimization reduces the computational cost by decreasing the number of generated nodes and execution time. Although the complexity remains exponential, the optimization reduces the base growth rate by minimizing unnecessary node generation and improving the solver's efficiency.

- In the unoptimized version, each additional cell, ground cell, or capturable piece significantly increases the state space that needs to be explored.
- In the optimized version, fewer redundant nodes are generated, which leads to a slower rate of growth in both the number of generated nodes and the execution time.

Theoretically, while the optimization may not change the overall order of complexity (still exponential), it lowers the constants involved and leads to more efficient scaling as puzzle complexity increases, especially for larger grids. This is particularly important in large puzzles, where the difference in node generation and time becomes substantial.