Algorithm Report for Week 9

Hatem Feckry (120220264)
Seifeldeen Mohamed Galal (120220252)
Abdelrahman Ashraf (120220292)
Ahmed Elsherbeny (120220271)

April 10, 2025

1 Overview

This week, our tasks focused on rigorously testing the true performance of our bottom-up dynamic programming solution. We examined the effects of imposing an initial set bounding strategy on both the success rate and the runtime behavior of our algorithm. In addition, we attempted to extend our approach to push the boundary to 32 points that are free of heptagons, building on our earlier work with 16-point configurations. Finally, we enhanced the resolution of the heat map that was presented in last week's report in order to provide more detailed insights.

2 Part 1: Code Setup for Each Task

Task 1: Baseline Performance of 16-Point Configurations:
To obtain a baseline performance metric of our algorithm when generating 16-point sets that are void of hexagons, we executed 100 iterations. For each run, we recorded both the resulting output and the number of iterations needed to generate a valid configuration (if a valid configuration was generated). The relevant implementation is contained in the `16pointGenerator.cpp` file.

Task 2: Bounding the Initial Seed Set:
In this task, we experimented with constraining the initial seed to only a quarter of the original grid. The rationale, which is discussed further in the analysis section, was that such a restriction might provide additional free space for incremental growth. This was achieved by modifying the `threadFunctionEmptySet` function to generate only points in the top left quarter of the grid. Additionally, we developed a variation in which we shifted the bounding box to the center of the grid by applying a diagonal translation to each point in the set.

Task 3: Extending the Algorithm Beyond the Proven Boundary:
We attempted to extend our dynamic programming approach to a new, unverified boundary by using an initial seed of 16 points and then expanding this set to 32 points. Due to the

noticeable decline in performance (as detailed in the results section), we inserted an intermediate output to report the size of the final empty set after all viable extensions. Moreover, we implemented multithreading in the construction process by continuously partitioning the grid into 14-point batches, aligned with the number of available threads.

Task 4: Enhancing Heat Map Fidelity:
To improve our previous visualization, we increased the number of data points used for the heat map while also narrowing the range of the output. This process involved initially generating the number of iterations required for random points located on the previously defined boundary, and then augmenting the dataset with an additional 20 points. For these additional points, we imposed a time-limit on the generation function for points on the boundary defined by $(n-1) \times (n-2)$, with the constraint that the generated points exceed the pigeonhole theorem's threshold.

3 Part 2: Results

Result 1: Baseline 16-Point Configuration:
Out of 100 iterations, we achieved 28 successful configurations. The evaluation process took approximately 31 minutes in total, yielding a success rate of 0.28 and an average execution time of 18.44 seconds per successful run.

Result 2: Bounding the Initial Seed (Top-Left):
In this configuration, only 18 out of 100 runs were successful. The overall evaluation period took about 39 minutes, resulting in a success rate of 0.18 and an average runtime of 23.2 seconds per successful instance.

Result 3: Extending to 32-Point Sets:
The algorithm did not succeed in generating a complete set of 32 points in any of the runs. The largest set generated in any of these attempts reached a size of 30 points. However, we believe this represents an extreme outlier, with 27 points being a more realistic upper bound for the expected maximum. This task took around 7 minutes to evaluate, and given the exponential scaling involved—particularly with the insertion of new points—the current algorithm appears infeasible for set sizes larger than 28 points.

Result 4: Heat Map Enhancement:
The process of increasing the number of data points and reducing the output range proved straightforward, resulting in a notably improved heat map.

4 Part 3: Analysis

Analysis Point 1: Success Probability and Monte Carlo Potential:
While the overall probability of success is not optimal, we note that if the exact probability could be determined, it might be incorporated into a Monte Carlo method to effectively test configurations for 17-point sets. Furthermore, our experiments suggest that significant performance improvements might be achievable via GPU acceleration. Our current approach

of dividing the grid into many small batches resulted in CPU threads not being fully utilized, as each task was too brief. Exploiting parallelism on GPUs could mitigate this bottleneck.

Analysis Point 2: Effect of Bounding the Initial Seed:
The primary hypothesis behind bounding the initial seed was that it would provide additional free space for subsequent point insertions, thereby preventing spatial constraints from stifling growth. Despite this reasoning and the prior knowledge that such bounding should not impact computational complexity when the bounds exceed $(n-1)(n-2)$ (coupled with the pigeonhole theorem's stipulations), the observed success rate dropped significantly. The underlying cause of this decline in performance, as well as the slight increase in runtime, remains unclear and requires further investigation.

Analysis Point 3: Exponential Growth of the Problem:
Our experiments suggest that the probabilistic element in the growth phase has now been successfully minimized, transforming the core problem into one characterized by exponential growth. Although this exponential behavior is an improvement over the previous model, it still renders the use of Monte Carlo methods impractical for larger set sizes due to the steep increase in computational requirements.

Analysis Point 4: Special Characteristics of the Boundary:
We have confirmed our suspicion that the current boundary possesses unique characteristics. Nevertheless, the fundamental reason behind this behavior has not been fully uncovered, indicating a need for additional research into the underlying geometric or combinatorial properties.

5 Part 4: Future Work

Based on the results and analysis, several directions for future work have been identified:

1. Code Optimization and GPU Acceleration:
We plan to further optimize our code and explore leveraging GPU acceleration to overcome the CPU bottleneck encountered during batch processing.

2. Investigate the Bounding Strategy:
Further exploration is required to understand why the initial set bounding strategy did not yield the expected improvements in success rate. Detailed statistical analysis and parameter tuning may provide insights that could improve performance.

3. Exact Probability Estimation for Monte Carlo Methods:
Developing a method to precisely calculate the probability of successful empty set generation could enable the integration of Monte Carlo methods into our approach, particularly for the generation of 17-point configurations.

4. Transformer Model Implementation:
With a working algorithm that reliably generates 16-point sets, we plan to experiment with

implementing a transformer model. This model would be designed to "shift" a random 16-point set into one that is void of hexagons, potentially opening up new avenues for improved configuration generation.

6 Part 5: Figures

Figures are an essential component of our report, providing a visual insight into the performance and behavior of the algorithm. The heat map generated during this week's analysis is presented in Figure 1.

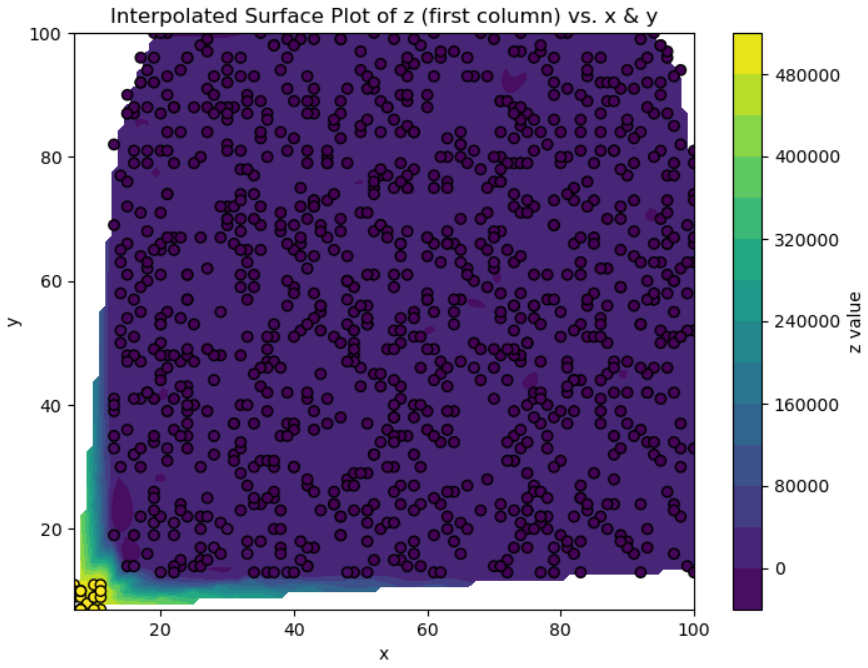

Figure 1: Heat map showing the distribution of iterations required for random points along the specified boundary. This enhanced heat map features increased data points and a reduced output range for improved fidelity.

7 Conclusion

The work undertaken during week 9 has provided valuable insights into the limitations and potential of our bottom-up dynamic programming approach. Although challenges remain—particularly in extending the algorithm to 32 points and in understanding the effects of initial set bounding—the experimental data and analysis have laid a solid foundation for future investigations. The planned future work, including code optimizations, deeper statistical analysis, and the exploration of GPU acceleration and transformer models, promises to guide us toward more efficient and scalable solutions.