Algorithm Complexity Analysis

1. Performance Comparison of the Proposed Algorithm and Evolution Strategies Algorithm

The paper on the Evolution Strategies Algorithm does not explicitly provide an analysis of its computational complexity. This algorithm allows each region to have multiple layout options; for example, in a rectangular segmentation map, a region can be positioned to the north or east of its adjacent region. If the dataset has m layout options, then the number of possible rectangular segmentation maps that can be generated may be as large as 2^m. In order to find the optimal cartogram, the algorithm needs to calculate the corresponding cartogram for each of these rectangular segmentation maps. However, this process involves running the algorithm multiple times to generate different layouts, which leads to redundant computations, significantly increasing the computational load and reducing efficiency. As the number of spatial units increases, not only does the computational load grow due to the larger dataset, but the number of layout options m also increases, causing the number of layouts to grow exponentially. This results in a surge in computational load and a substantial decrease in efficiency.

In contrast, the algorithm proposed in this paper introduces a specific method to calculate the relative positioning relationships, thereby clearly defining the positions between regions and simplifying the construction process. While maintaining the effectiveness of the visualization, it generates only a single rectangular cartogram, shortening the runtime and improving efficiency. Although the algorithm in this paper may not perform as well as the Evolution Strategies Algorithm in maintaining adjacency and positional relationships, the cartogram generated in a shorter time has relatively small errors, and the adjacency and positional errors do not significantly affect the data visualization. Furthermore, the computation time of the proposed algorithm increases only with the number of spatial units, resulting in a more stable growth in computation time and higher efficiency.

2. Time Complexity Analysis of the Proposed Algorithm

We conducted a comprehensive complexity analysis of the proposed algorithm, which revealed that its overall time complexity is $O(n^3)$. The following is the detailed process for calculating the time complexity of the proposed algorithm in this paper.

2.1 Time Complexity Analysis of the Rectangular Segmentation Map Construction Algorithm

(1) Calculation of Relative Positional Relationships of Rectangular Segmentation Units

This operation first calculates the angle between two regions using the arctangent function. For each pair of regions, the time complexity of this calculation is O(1). Assuming there are n units, the relative positional relationships need to be calculated for all pairs of units. Therefore, the time complexity of calculating the relative positional relationships is $O(n^2)$.

(2) Calculation of Initial Length and Width of Rectangular Segmentation Units

In this part, it is necessary to traverse each rectangular segmentation unit and, for each unit, calculate the number of adjacent units. The lookup operation for the number of adjacent units for each unit is O(1). Therefore, for n units, the overall time complexity is O(n).

(3) Adjustment Order of Weighted Breadth-First Search

In this part, we use weighted breadth-first search (BFS) to determine the adjustment order. The time complexity of BFS is typically O(V + E), where V is the number of vertices in the graph and E is the number of edges. In this algorithm, the number of vertices in the graph is the number of map units, n. We

use an adjacency matrix to store the adjacency relationships between units, so the number of edges can be at most n^2. Therefore, the time complexity of this part is $O(n + n^2) = O(n^2)$.

(4) Dynamic Adjustment Strategy

For the adjustment process of each unit's length, the time complexity is O(1). The time complexity for adjusting the length of all units at once is O(n), but multiple iterations are required to determine the final length. Assuming that the length adjustment requires at most T_L iterations, the time complexity for length adjustment is $O(T_L * n)$. Similarly, assuming that the width adjustment requires at most T_W iterations, the time complexity for width adjustment is $O(T_W * n)$. Therefore, the overall time complexity for the dynamic adjustment process is $O(T_L * n + T_W * n) = O((T_L + T_W) * n)$. If the number of iterations is small, the minimum time complexity is O(n). If the number of iterations is large, the maximum time complexity is $O(n^2)$.

In conclusion, the overall time complexity of the rectangular segmentation map construction algorithm is $O(n^2)$.

2.2 Time Complexity Analysis of the Rectangular Cartogram Construction Algorithm

(1) Expected Area Calculation of Cartogram Units

The calculation here is simply a basic proportional calculation. For each unit, calculating its expected area is a constant-time operation, with a time complexity of O(1). Assuming there are n units, the total time complexity is O(n).

(2) Position Determination Strategies for Cartogram Units

The time complexity for calculating the positions of all units needs to be analyzed according to the process described in Section 5.3.

Step 1: Determine the Initial Rectangular Cartogram Unit

Select the initial unit and determine its position. The selection of the initial unit and calculation of its position are constant-time operations, O(1).

Step 2: Calculate the Priority of Adjacent Units

In this step, we first iterate through the set of placed units. During the iteration, we select a placed unit and then iterate through its adjacent units. Assuming there are n units, the time complexity for this part is $O(n^2)$.

Step 3: Determine the Unit to be Placed

In this part, we may need to check the priority of all combinations of unplaced units. Therefore, in the worst case, the time complexity is O(n).

Step 4: Calculate the Position of the Unit to be Placed

This part is described in detail in Section 5.2. In this step, we need to determine the adjacent placed units for the unit to be placed. Therefore, the time complexity for this part is O(n).

Therefore, the time complexity from Step 2 to Step 4 is $O(n^2)$. We also need to repeat Step 2 to Step 4 until all units are included in set C. In the worst case, this process may need to be repeated n times. Thus, the time complexity for calculating the positions of all units is $O(n^3)$.

The overall time complexity of the rectangular cartogram construction algorithm is $O(n^3)$.

Based on the above analysis, the overall time complexity of the algorithm in this paper is $O(n^3)$.