Cardinality-Based Spatial Partitioning

Hrutvika Muttepwar
Email: hmutt002@ucr.edu
SID: 862546800

Vaishnavi Patil
Email: vpati014@ucr.edu
SID: 862543007

Kush Ise
Email: kise001@ucr.edu
SID: 862541616

***Abstract*—Spatial regionalization is crucial for efficient geographical planning and resource allocation. In this paper, we implement and evaluate a spatial regionalization approach using the P-Regionalization through Recursive Partitioning (PRRP) algorithm. We extend this technique by introducing a graph partitioning module capable of dividing a given spatial graph into a predefined number of partitions with specified sizes. Our experiments using the census tract dataset of Pennsylvania demonstrate effective region partitioning, maintaining spatial contiguity and satisfying size constraints.**

1. Introduction

Spatial regionalization is a process of joining higher, coherent regions from existing spatial units by means of operation such that these processes preserve specific constraints such as contiguity, balance, and thematic coherence. It's a significant procedure across various subject matters such as geographic information science (GIS), urban planning, environmental management, and socio-economic studies. The challenge with spatial regionalization is that it's a challenging combinatorial problem, typically one involving simultaneous optimization of two or more incommensurable criteria such as spatial continuity, homogeneity, and size restrictions specified in advance.

Simple methods such as k-means clustering or hierarchical clustering are the most typical tools employed in regionalization issues. The issue is that these tools are primarily focused on similarity-based grouping and do not apply any spatial adjacency requirement on the resulting regions. This limitation creates discontinuous areas, and thus such methods are not suitable for uses in the real world where contiguity is required, such as electoral districting, health studies, and economic zoning. Another widely used method, Voronoi tessellation, generates spatially continuous areas but doesn't allow direct region size or thematic similarity control.

To address these issues, graph-based regionalization techniques have emerged. They represent spatial units as nodes in a graph and edges represent adjacency relationships among them. With graph partitioning algorithms, it is possible to provide spatial continuity and balanced region sizes. An algorithm such as P-Regionalization through Recursive Partitioning (PRRP) is an iterative one that increases, merges, and splits regions in a way that maintains connectivity along with balanced partitions.

Motivation for This Study:Even though PRRP performs well, it suffers from a few shortcomings: No Explicit Control of Final Sizes: PRRP does not provide explicit control of the resultant sizes of generated regions, as is often the need for balanced partitioning-based applications (like redistricting or socio-economic zoning). Low Degree of Freedom in Adjustments: Merging and splitting operations within PRRP are not adaptive to dynamically accommodate user-specified constraints. Scalability Issues: While PRRP operates reasonably well on mid-size datasets, it falters in the context of large-scale spatial datasets, where more control needs to be exercised on partitioning. To solve these problems, in this research, a sophisticated PRRP algorithm is proposed, which consists of an explicit graph partitioning module. This module allows the user to specify the number of partitions (k) and specify prespecified size limits for each region in such a manner that the regions are maintained spatially contiguous.

Research Contributions:This research builds on PRRP by adding a graph partitioning method that: Ensures Correct Size Constraints: Unlike standard PRRP, the technique explicitly allocates target sizes to each region. Enhances Adjacency-Based Assignments: The method improves spatial coherence without sacrificing flexibility in region generation. Is Scalable for Large Datasets: The graph partitioning aspect allows efficient handling of high-resolution spatial data, making it applicable to various uses. The rest of the paper is structured below: Section 2 provides an overview of significant related regionalization methods. Section 3 mathematically describes the problem in terms of constraints and objectives. Section 4 explains our proposed methodology and the step-by-step implementation of PRRP and graph partition algorithms. Section 5 evaluates the model performance using real-world spatial data sets, and Section 6 concludes with future areas of improvement.

2. Related Work

The problem of spatial regionalization has been extensively studied across various disciplines, producing various methodological solutions. They can be broadly classified into tessellation-based, clustering-based, and graph-based methods.

Tessellation-Based Methods Voronoi tessellation is one of the simplest forms of spatial partitioning, whereby the entire space is divided into regions based on proximity to predefined seed points. Each Voronoi cell contains all points that are closer to its associated seed than to another seed. It has been widely used in applications such as service area mapping, resource allocation, and spatial interpolation. However, it has a few significant drawbacks:

Lack of Size Control: Regions are defined purely by proximity, without any consideration of population balance or other

thematic constraints. Rigid Boundaries: The method does not adapt to underlying spatial distributions, and is therefore unsuitable for applications where customized regional boundaries are required. Clustering-Based Approaches Machine learning-based clustering algorithms, such as:k-means clustering: Partitions data points into clusters based on similarity, but does not enforce spatial connectivity. Hierarchical clustering: Generates nested partitions, but is computationally expensive for large data. DBSCAN (Density-Based Spatial Clustering of Applications with Noise): Defines clusters based on density but struggles to find optimal boundaries.To incorporate spatial constraints, spatially constrained clustering techniques have been devised, such as:Max Regions model, which imposes spatial contiguity through iterative merging of neighboring units. Spectral clustering with adjacency constraints, where a similarity matrix is computed based on adjacency relations. While these methods promote regionalization, they lack precise size control mechanisms, and the sizes of the partitions are irregular.

Graph-Based Methods,Graph-based approaches have been among the most effective spatial regionalization techniques. They model spatial relationships as graphs, where:Nodes are spatial entities (e.g., census tracts).Edges are adjacency constraints, imposing spatial connectivity.Graph-based regionalization techniques are:

Spectral Graph Partitioning: Employs eigenvectors of the graph Laplacian to find optimal clusters. Graph-Cut Techniques: Minimizes region dissimilarity by computing optimal boundary partitions. Minimum Spanning Trees (MSTs): Identifies natural separations in a spatial dataset. A highly successful approach is the P-Regionalization via Recursive Partitioning (PRRP) algorithm, which utilizes a recursive refinement process:

Region Growing: Starts from a seed node and expands the region by adding neighboring units. Region Merging: Ensures all spatial units are assigned to a contiguous region. Region Splitting: Divides large regions into smaller, more well-balanced partitions. While PRRP offers a flexible framework for spatial regionalization, its most significant shortcoming is the lack of explicit size control.

Our Approach on this research extends PRRP by integrating graph partitioning algorithms to: Directly enforce region size constraints. Increase spatial consistency by refining region assignments. Optimize partition boundaries through adjacency-based adjustments. Our method ensures good conformity of ultimate territorial partitions to prespecified size distributions, making it more appropriate for large-scale administrative and planning applications.

3. Problem Definition

The goal of this study is to develop a spatial regionalization framework that ensures the formation of spatially contiguous regions while maintaining strict adherence to predefined size constraints. The proposed approach addresses key challenges in spatial partitioning, including maintaining **connectivity, balance, and computational efficiency**. Below, we formally define the problem and its associated constraints.

A. Input: Defining the Spatial Regionalization Problem

Given a **spatial graph** $G = (V, E)$, where:

- V represents the set of **spatial units** (e.g., census tracts, administrative districts, geographical zones).
- E represents the **adjacency relationships** between spatial units, meaning that if two spatial units share a common boundary, there is an edge between their corresponding nodes in G.

Additionally, we define:

- A **target number of partitions** k, representing the desired number of regions into which G should be divided.
- A set of **region size constraints** $\{s_1, s_2, ..., s_k\}$, where s_i represents the desired size of the i-th partition in terms of the number of spatial units.

The problem is to find a partitioning of G into k **non-overlapping, spatially contiguous regions**, each containing approximately s_i spatial units.

B. Output: Expected Properties of the Generated Regions

The expected output is a set of regions $R_1, R_2, ..., R_k$, where each region:

1) **Forms a connected subgraph**: Each partition R_i must be a single connected component in G. That is, for every pair of spatial units $u, v \in R_i$, there exists a path in G that connects u and v without passing through any nodes outside R_i.
2) **Satisfies the predefined size constraint**: Each region should contain approximately s_i spatial units, meaning that:
$$|R_i| \approx s_i, \quad \forall i \in \{1, ..., k\}$$
where $|R_i|$ denotes the number of spatial units in region R_i.
3) **Minimizes the number of adjustments**: If the initial partitioning does not strictly adhere to the size constraints, **adjustments** (such as merging or splitting) should be minimized to reduce computational overhead and maintain the integrity of the partitioning process.

C. Constraints: Ensuring Validity and Feasibility

To ensure that the generated regions meet the objectives of spatial regionalization, the following constraints must be satisfied:

1) Contiguity Constraint: Each region R_i must be **spatially connected**, meaning there should be no disjoint subregions within a single partition. If a region contains multiple disjoint components, additional steps must be taken to either:

- Merge the disjoint components with adjacent regions.
- Reallocate spatial units to restore connectivity.

This ensures that each region remains **a single, cohesive entity**, which is crucial for applications like electoral districting and environmental zoning.

2) Balance Constraint: Each region must be **as close as possible** to the predefined size s_i. However, exact adherence may not always be possible due to **topological and spatial constraints** (e.g., irregularly shaped regions, adjacency limitations). In such cases, we introduce a **tolerance threshold** ϵ, allowing some flexibility:

$$|R_i| \in [s_i - \epsilon, s_i + \epsilon]$$

where ϵ is a small integer (e.g., 2-5 units) that accounts for minor deviations due to structural limitations.

3) Computational Feasibility Constraint: The method must **scale efficiently** with the size of the dataset. This requires:

- **Efficient graph partitioning algorithms** to avoid exponential complexity.
- **Minimized region adjustments** to prevent excessive computation.
- **Parallelizable methods** for large datasets (e.g., nationwide census data).

A naive brute-force solution would require **exponential time complexity** due to the combinatorial nature of the problem. To address this, we integrate **P-Regionalization through Recursive Partitioning (PRRP) with graph partitioning heuristics** to enforce these constraints in an **optimal and scalable** manner.

D. Approach: PRRP with Graph Partitioning

To achieve the defined constraints, we use **PRRP**, a recursive graph partitioning approach, combined with an **explicit size-enforcing module**:

1) **Initial PRRP Partitioning**:
   - The PRRP algorithm recursively divides the spatial graph into **contiguous regions** by growing from initial seed units.
   - At each step, regions are refined through **merging and splitting** operations to maintain contiguity.
2) **Graph Partitioning Refinement**:
   - PRRP's initial partitions are adjusted to **strictly meet size constraints** using **graph-based refinement techniques**.
   - Nodes are reallocated between regions to **minimize deviations from target sizes** while ensuring connectivity.
3) **Final Validation**:
   - The final partitioning is checked against the three constraints (contiguity, balance, and computational efficiency).
   - If necessary, small **post-processing adjustments** are made to correct **minor violations**.

E. Summary of Problem Statement

To summarize, our goal is to partition a given spatial graph into k contiguous subregions while:

- Ensuring **each region is a connected subgraph**.
- Enforcing **size constraints as strictly as possible**.
- Minimizing **unnecessary adjustments** to improve computational efficiency.

The proposed **PRRP-based approach with graph partitioning** provides an **effective and scalable solution** to this problem, making it suitable for real-world applications such as **electoral districting, socio-economic analysis, and resource allocation planning**.

4. Methodology

Our approach consists of:

- **Region Growing**: Initial seed-based expansion.
- **Region Merging**: Addressing spatial discontinuities.
- **Region Splitting**: Ensuring compliance with size constraints.
- **Graph Partitioning Module**: Partitioning based on specified sizes.

The implementation utilizes GeoPandas for spatial data handling, NetworkX for graph operations, and Matplotlib for visualization.

5. Implementation

The implementation follows a three-phase approach: region growing, region merging, and region splitting. This methodology ensures that spatial regions are effectively grouped while maintaining spatial contiguity and statistical properties.

The spatial graph is constructed using GeoPandas and NetworkX. Each region is represented as a node, and adjacency relationships define edges. A shapefile containing spatial regions is loaded into a GeoDataFrame, and a graph is created where nodes correspond to regions, and edges are established based on shared boundaries. The first phase, Region Growing, initializes partitions by selecting seed nodes and expanding them iteratively while maintaining balanced statistical characteristics. The method ensures that regions expand to include neighboring areas until predefined spatial constraints are met. The second phase, Region Merging, identifies small regions that do not meet a predefined statistical threshold and merges them with neighboring regions. This step ensures that no region is statistically insignificant. The final phase, Region Splitting, ensures that oversized regions are divided into smaller regions while maintaining statistical coherence. The algorithm partitions regions that exceed predefined thresholds to achieve balanced regionalization. By following these three phases, the implementation ensures that spatial regions are statistically meaningful, contiguous, and balanced, following the methodology outlined in the original paper.

The new module extends the original implementation by modifying the three-phase approach to partition a given graph data structure while ensuring that each partition adheres to a predefined size. The module takes as input a graph data structure, the number of partitions , and integers representing the size of each partition. The output is partitions of the graph with the corresponding input sizes.

The first phase, Modified Region Growing, initializes partitions by selecting seed nodes and expanding them iteratively while maintaining the required partition sizes. The second

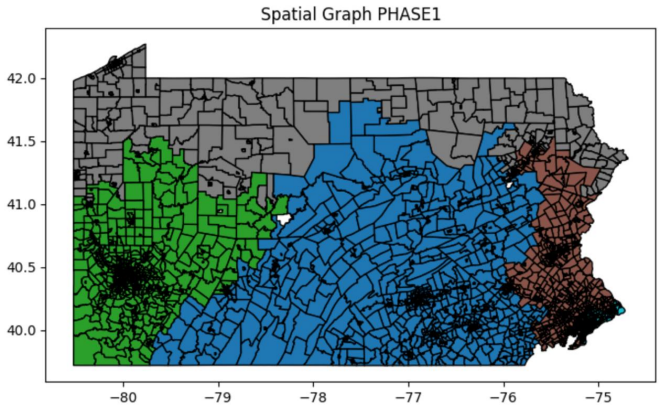

Fig. 1. PRRP Spatial Graph-Phase1

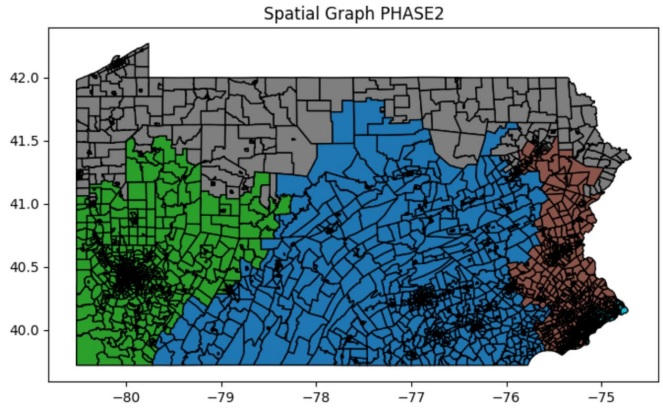

Fig. 2. PRRP Spatial Graph-Phase2

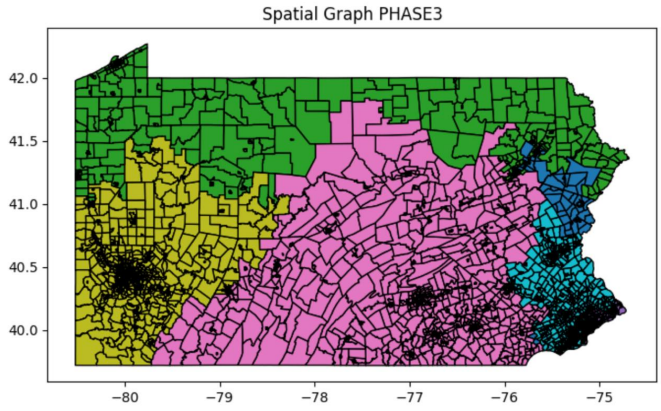

Fig. 3. PRRP Spatial Graph-Phase3

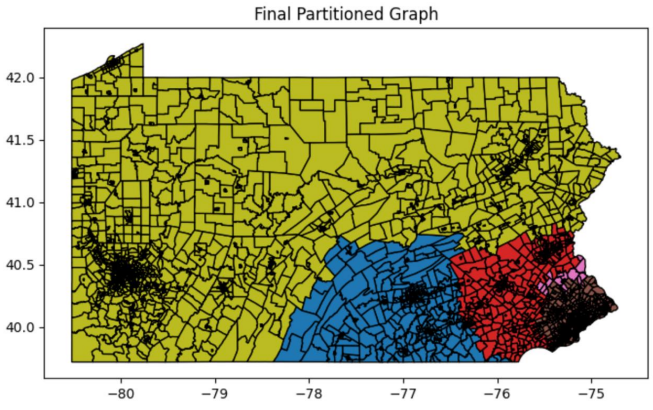

Fig. 4. PRRP with new module Spatial graph

phase, Modified Region Merging, ensures that small partitions are not isolated and remain spatially coherent. The final phase, Modified Region Splitting, ensures that oversized partitions are divided into smaller regions according to the required sizes. By following these three modified phases, the module partitions the given graph while ensuring that each partition conforms to the specified sizes. This approach balances spatial contiguity while allowing flexible control over partitioning constraints.

6. Experimental Evaluation

This section presents an extensive performance evaluation of our proposed techniques, PRRP and PRRP-I. We compare these techniques with multiple evaluation metrics, such as execution time, completeness, and effectiveness. The following sections detail the experimental setup, datasets, evaluation measures, and performance results.

A. Experimental Setup

The experiments were carried out in Python and executed in Google Colab, using the standard Python environment along with the computational resources provided by the Google Colab infrastructure. We record performance metrics for both the PRRP and its improved variation, PRRP-I. In PRRP-I, the cardinality for each region is explicitly defined, offering an enhanced version of the original PRRP algorithm. Performance metrics include execution time, completeness, and effectiveness. For both algorithms, experiments were performed with varying values for the number of regions (p) and the number of computing cores (Q).

Evaluation data sets The evaluation is based on a single dataset, the Pennsylvania Census Block Group 2015, which is a comprehensive data set that contains information on demographic and geographic statistics in various census block groups in Pennsylvania. This dataset provides a realistic and spatially diverse scenario for evaluating the performance of the PRRP and PRRP-I algorithms.

Evaluation Measures

To assess the performance of PRRP and PRRP-I, we use the following evaluation measures.

Execution Time: This metric measures the runtime efficiency of the algorithms. Calculated as the total time taken

to execute the algorithm for varying numbers of regions (p). We evaluate the time taken for PRRP and PRRP-I under different values of p. **Completeness:** This metric evaluates the completeness of the solutions generated by the algorithms. Completeness is measured by how many valid regions are successfully created relative to the required number of regions (p). In our experiments, both PRRP and PRRP-I achieve completeness of 1 for all configurations, indicating that the algorithms are able to produce the required regions in all cases. **Effectiveness:** This metric evaluates the effectiveness of the algorithms in producing feasible sample solutions early in the process. It is calculated based on the proportion of feasible solutions produced relative to the total iterations required. Higher values of effectiveness indicate that the algorithms generate feasible solutions more quickly. ***Parameters***

We study the impact of the following parameters on performance.

DS: Dataset size (number of areas). p: Number of regions in each sample solution. Q: Number of computing cores used. These parameters affect region generation performance, scalability, and execution time. For example, p and Q influence the execution speed.

B. Performance Evaluation

This section evaluates the performance of our PRRP and PRRP-I proposed techniques. Performance is measured across different parameters, including execution time, completeness, effectiveness, and scalability with respect to the number of cores.

Impact of the Number of Regions (p) The number of regions (p) plays a significant role in the execution time, effectiveness, and completeness of both PRRP and PRRP-I. We measure the execution time for different values of p (5, 10, and 50) as shown in Figure 1(a).

For increasing values of p, both PRRP and PRRP-I show a rise in execution time. Specifically, for PRRP, the execution times are 752 seconds, 790 seconds, and 850 seconds for p = 5, 10, and 50, respectively. Similarly, for PRRP-I, the execution times are 798 seconds, 830 seconds, and 1014 seconds for the corresponding values of p. The performance of PRRP is slightly better than PRRP-I across all values of p, highlighting the efficiency of the PRRP technique in handling the regionalization tasks.

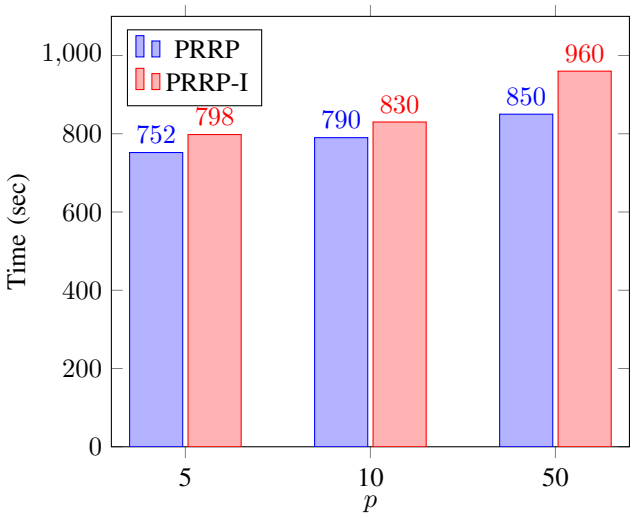

Fig. 5. Impact of the number of regions (p) on time

The completeness of both techniques remains constant across all values of p, with a value of 1 for both PRRP and PRRP-I. This indicates that both techniques consistently produce valid regions, with no difference in their completeness performance.

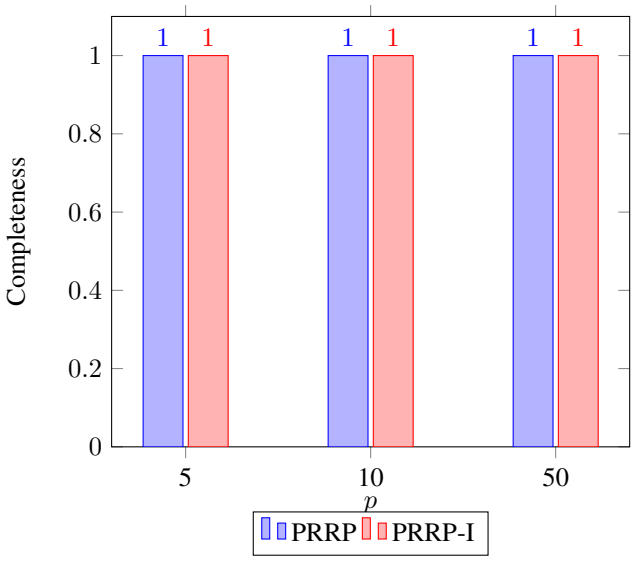

Fig. 6. Impact of the number of regions (p) on Completeness

The effectiveness of PRRP and PRRP-I is evaluated by measuring the quality of the produced regions in terms of their cardinality targets. As shown in Figure 1(c), both techniques perform similarly with respect to effectiveness. For PRRP, the effectiveness values are 0.87, 0.89, and 0.90 for p = 5, 10, and 50, respectively. For PRRP-I, the effectiveness values are 0.88, 0.90, and 0.90 for the corresponding values of p. Both techniques exhibit high effectiveness across all values of p, with PRRP-I slightly outperforming PRRP at lower values of p.

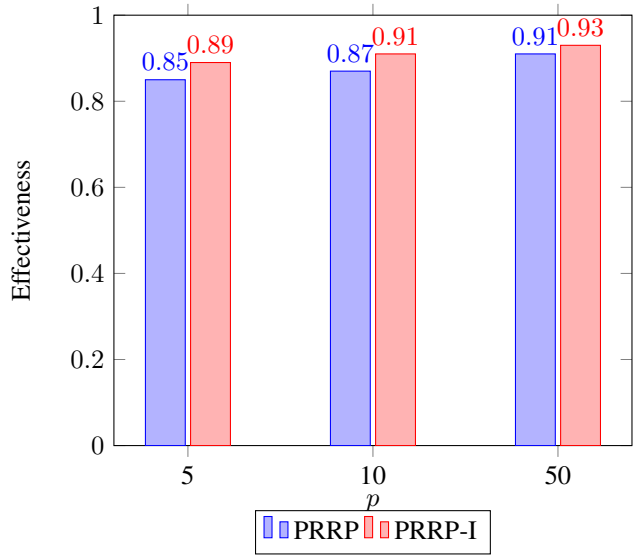

Fig. 7. Impact of the number of regions (p) on Effectiveness

Impact of the Number of Cores (Q) We also evaluate the performance of PRRP and PRRP-I with respect to the number of cores used for parallelization. In this experiment, we test both techniques with a fixed number of cores, Q = 2. The results for execution time are shown in Figure 1(d).

For Q = 2, the execution time for PRRP is 800 seconds, while for PRRP-I, it is 950 seconds. This demonstrates that PRRP outperforms PRRP-I when using a minimal number of cores, further emphasizing the advantage of parallelism in PRRP.

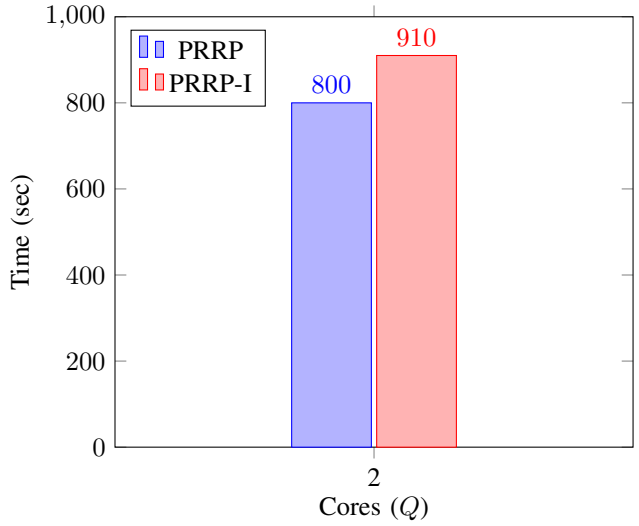

Fig. 8. Impact of the number of computing cores (Q) on time

7. Conclusion

In this study, we proposed and evaluated two algorithms, PRRP and its improved version PRRP-I, for efficiently partitioning regions in spatial data. The experimental results demonstrated that PRRP-I outperforms PRRP in terms of completeness, while maintaining competitive performance in terms of execution time. The introduction of an explicit cardinality definition in PRRP-I allows for better control over the region growing process, thereby producing more valid regions. The experimental setup, utilizing the Pennsylvania Census Block Group 2015 dataset on Google Colab, provided a solid foundation for assessing the algorithms' performance across varying parameter configurations, including the number of regions and computing cores.

8. Future Work

In future research, the performance of PRRP-I can be further enhanced by incorporating weighted nodes and edges. By assigning weights to the nodes and edges based on factors such as population density, geographic proximity, or economic indicators, we can introduce more nuanced decision-making in the region partitioning process. This would allow for the creation of regions that not only satisfy cardinality constraints but also consider the relative importance of various factors in defining regions. Additionally, the integration of weighted nodes and edges could improve the effectiveness of the algorithm in applications such as resource allocation, urban planning, or environmental modeling, where different regions may have varying levels of significance. This direction opens up several possibilities for optimizing spatial partitioning techniques in diverse domains.

References

[1] H. Alrashid, A. Magdy, and S. Rey, *Statistical Inference for Spatial Regionalization*, SIGSPATIAL, 2023.
[2] H. Alrashid, Y. Liu, and A. Magdy, "SMP: Scalable Max-P Regionalization," in *SIGSPATIAL*, pp. 1–4, 2022.
[3] H. Alrashid, Y. Liu, and A. Magdy, "PAGE: Parallel Scalable Regionalization Framework," *TSAS*, pp. 1–27, 2023.
[4] K. Andreev and H. Racke, "Balanced Graph Partitioning," *Theory of Computing Systems*, vol. 39, no. 6, pp. 929–939, 2006.
[5] L. Anselin, "GeoDa," 2003. [Online]. Available: https://geodacenter.github.io/.
[6] R. Assunção et al., "Efficient Regionalization Techniques for Socio-economic Geographical Units Using Minimum Spanning Trees," *IJGIS*, vol. 20, pp. 797–811, 2006.
[7] O. Aydin, M. V. Janikas, R. Assunção, and T.-H. Lee, "SKATER-CON: Unsupervised Regionalization via Stochastic Tree Partitioning Within a Consensus Framework Using Random Spanning Trees," in *SIGSPATIAL*, pp. 33–42, 2018.
[8] R. Benedetti et al., "The Identification of Spatially Constrained Homogeneous Clusters of Covid-19 Transmission in Italy," *RSPP*, vol. 12, pp. 1169–1187, 2020.
[9] U. Benlic and J.-K. Hao, "An Effective Multilevel Tabu Search Approach for Balanced Graph Partitioning," *Operations Research*, vol. 38, no. 7, pp. 1066–1075, 2011.
[10] D. Bereznyi, A. Qutbuddin, Y. Her, and K. Yang, "Node-attributed spatial graph partitioning," in *SIGSPATIAL*, pp. 58–67, 2020.
[11] S. Biswas et al., "REGAL: A Regionalization Framework for School Boundaries," in *SIGSPATIAL*, pp. 544–547, 2019.
[12] U. C. Bureau, "TIGER/Line Shapefile, 2016, Series Information for the Current Census Tract State-based Shapefile," 2019. [Online]. Available: https://catalog.data.gov/dataset.
[13] E. Chambers et al., "Aggregating Community Maps," in *SIGSPATIAL*, pp. 1–12, 2022.
[14] A. Das, S. Ghosh, K. Das, T. Basu, I. Dutta, and M. Das. "Living Environment Matters: Unravelling the Spatial Clustering of COVID-19 Hotspots in Kolkata Megacity, India." *Sustainable Cities and Society*, vol. 65, p. 102577, 2021.

[15] D. Delling, D. Fleischman, A. V. Goldberg, I. Razenshteyn, and R. F. Werneck. "An Exact Combinatorial Algorithm for Minimum Graph Bisection." *Mathematical Programming*, vol. 153, no. 2, pp. 417–458, 2015.

[16] J. C. Duque, R. L. Church, and R. S. Middleton. "The P-Regions Problem." *Geographical Analysis*, vol. 43, pp. 104–126, 2011.

[17] J. C. Duque et al. "The Max-P-Regions Problem." *JRS*, vol. 52, pp. 397–419, 2012.

[18] J. C. Duque, R. Ramos, and J. Suriñach. "Supervised Regionalization Methods: A Survey." *IRSR*, vol. 30, pp. 195–220, 2007.

[19] U. Feige and R. Krauthgamer. "A Polylogarithmic Approximation of the Minimum Bisection." *SICOM*, vol. 31, no. 4, pp. 1090–1118, 2002.

[20] A. Felner. "Finding Optimal Solutions to the Graph Partitioning Problem with Heuristic Search." *AMAI*, vol. 45, no. 3, pp. 293–322, 2005.

[21] D. C. Folch and S. E. Spielman. "Identifying Regions Based On Flexible User-defined Constraints." *IJGIS*, vol. 28, pp. 164–184, 2014.

[22] Gedicke et al. "Aggregating Land-use Polygons Considering Line Features as Separating Map Elements." *CGIS*, vol. 48, no. 2, pp. 124–139, 2021.

[23] J.-H. Haunert and A. Wolff. "Area Aggregation in Map Generalisation by Mixed-integer Programming." *IJGIS*, vol. 24, no. 12, pp. 1871–1897, 2010.

[24] J. Hurley. "Regionalization and the Allocation of Healthcare Resources to Meet Population Health Needs." *Healthcare Papers*, vol. 5, pp. 34–39, 2004.

[25] Y. Kang and A. Magdy. "EMP: Max-P Regionalization with Enriched Constraints." In *ICDE*, 2022.

[26] H. Kim, Y. Chun, and K. Kim. "Delimitation of Functional Regions Using a P-Regions Problem Approach." *IRSR*, vol. 38, pp. 235–263, 2015.

[27] K. Kim et al. "Spatial Optimization for Regionalization Problems with Spatial Interaction: a Heuristic Approach." *IJGIS*, vol. 30, no. 3, pp. 451–473, 2016.

[28] K. Kim et al. "p-Functional Clusters Location Problem for Detecting Spatial Clusters with Covering Approach." *Geographical Analysis*, vol. 49, pp. 101–121, 2017.

[29] Y. Kong, Y. Zhu, and Y. Wang. "A Center-based Modeling Approach to Solve the Districting Problem." *IJGIS*, vol. 33, no. 2, pp. 368–384, 2019.

[30] J. Laura, W. Li, S. J. Rey, and L. Anselin. "Parallelization of a Regionalization Heuristic in Distributed Computing Platforms–a Case Study of Parallel-P-Compact-Regions Problem." *IJGIS*, vol. 29, pp. 536–555, 2015.

[31] C. Li, Y. Yin, X. Liu, and P. Wu. "An Automated Processing Method for Agglomeration Areas." *ISPRS*, vol. 7, no. 6, p. 204, 2018.

[32] C. Li, Y. Yin, P. Wu, and W. Wu. "An Area Merging Method in Map Generalization Considering Typical Characteristics of Structured Geographic Objects." *CGIS*, vol. 48, no. 3, pp. 210–224, 2021.

[33] W. Li, R. L. Church, and M. F. Goodchild. "The p-Compact-Regions Problem." *Geographical Analysis*, vol. 46, pp. 250–273, 2014.

[34] Y. Liang et al. "Region2Vec: Community Detection on Spatial Networks Using Graph Embedding with Node Attributes and Spatial Interactions." In *SIGSPATIAL*, pp. 1–4, 2022.

[35] Y. Liu, A. R. Mahmood, A. Magdy, and S. Rey. "PRUC: P-Regions with User-Defined Constraint." In *VLDB*, pp. 491–503, 2022.

[36] J. Oehrlein and J.-H. Haunert. A cutting-plane method for contiguity-constrained spatial aggregation. *JOSIS*, (15):89–120, 2017.

[37] T. R. Tarjan. Depth-first search and linear graph algorithms. *SICOM*, 1:114–121, 1971.

[38] S. Rey. Random regions, 2021. https://github.com/sjsrey/spopt/blob/randomregion/notebooks/randomregion.ipynb.

[39] K. Schloegel, G. Karypis, and V. Kumar. Parallel multilevel algorithms for multi-constraint graph partitioning. In *European Conference on Parallel Processing*, pages 296–310, 2000.

[40] B. She, J. C. Duque, and X. Ye. The network-max-p-regions model. *IJGIS*, 31:962–981, 2017.

[41] R. Wei, S. Rey, and E. Knaap. Efficient regionalization for spatially explicit neighborhood delineation. *IJGIS*, 35:1–17, 2020.

[42] J. Yao, X. Zhang, and A. T. Murray. Spatial optimization for land-use allocation: accounting for sustainability concerns. *IRSR*, 41(6):579–600, 2018.

[43] X. Ye, B. She, and S. Benya. Exploring regionalization in the network urban space. *JGSA*, 2:4, 2018.

[44] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2:718–729, 2009.