Node–attributed Spatial Graph Partitioning

Daniel Bereznyi
dbereznyi2016@fau.edu
Florida Atlantic University
Boca Raton, Florida

Ahmad Qutbuddin
aqutbuddin2017@fau.edu
Florida Atlantic University
Boca Raton, Florida

YoungGu Her
yher@ufl.edu
University of Florida
Homestead, Florida

KwangSoo Yang
yangk@fau.edu
Florida Atlantic University
Boca Raton, Florida

ABSTRACT

Given a spatial graph and a set of node attributes, the Node-attributed Spatial Graph Partitioning (NSGP) problem partitions a node-attributed spatial graph into k homogeneous sub-graphs that minimize both the total $RMSE_{rank1}$ and *edge-cuts* while meeting a size constraint on the sub-graphs. $RMSE_{rank1}$ is the Root Mean Square Error between a matrix and its rank-one decomposition. The NSGP problem is important for many societal applications such as identifying homogeneous communities in a spatial graph and detecting interrelated patterns in traffic accidents. This problem is NP-hard; it is computationally challenging because of the large size of spatial graphs and the constraint that the sub-graphs must be homogeneous, i.e. similar in terms of node attributes. This paper proposes a novel approach for finding a set of homogeneous sub-graphs that can minimize both the total $RMSE_{rank1}$ and *edge-cuts* while meeting the size constraint. Experiments and a case study using U.S. Census datasets and HP#6 watershed network datasets demonstrate that the proposed approach partitions a spatial graph into a set of homogeneous sub-graphs and reduces the computational cost.

CCS CONCEPTS

• **Information systems → Geographic information systems**.

KEYWORDS

spatial graph partitioning, node-attributed spatial graph, matrix rank-one decomposition

ACM Reference Format:
Daniel Bereznyi, Ahmad Qutbuddin, YoungGu Her, and KwangSoo Yang. 2020. Node–attributed Spatial Graph Partitioning. In *28th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '20), November 3–6, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397536.3422198

SIGSPATIAL '20, November 3–6, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-8019-5/20/11…$15.00
https://doi.org/10.1145/3397536.3422198

1 INTRODUCTION

In this work, we propose a new problem of spatial graph partitioning, namely Node-attributed Spatial Graph Partitioning (NSGP). Given a spatial graph and a set of node attributes, the NSGP problem partitions the node-attributed spatial graph into k homogeneous sub-graphs that minimize both the total $RMSE_{rank1}$ and *edge-cuts* while meeting a size constraint. Fig. 1(a) shows an example input of NSGP consisting of a graph with 15 nodes with 4 attributes and 23 edges. Assume that $k = 3$ and that each sub-graph should contain at least 4 nodes. Figure 1(b) shows an example output of NSGP where the sub-graphs minimize the total $RMSE_{rank1}$ and *edge-cuts* while meeting the size constraint. The NSGP problem is NP-hard (a proof is provided in Section 1.3). Intuitively, the problem is computationally challenging because of the large size of spatial graphs and the constraint that the sub-graphs must be homogeneous, i.e. similar in terms of node attributes.

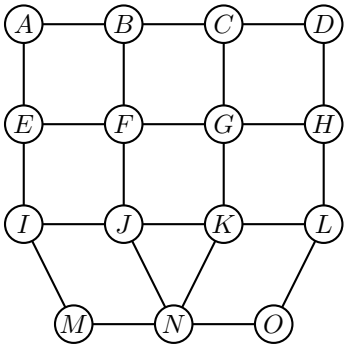
1.1 Application Domain

NSGP is important in many societal applications such as identifying patterns in spatial graph data where each node is associated with additional information. One such application is discovering communities in the spatial graph of a city by finding partitions where attributes of the population (e.g., average age, income, education, etc.) are similar and the nodes are spatially close to one another. This information can then be used to better identify and target these communities [4]. Another application is for Hydrological Response Unit delineation that helps us to understand the spatial variability of the watershed among soil, land use, and topographic characteristics [12, 15]. The size constraint allows for flexibility in the partitioning, allowing for more focus to be placed on homogeneity over balanced-size partitions if desired [3]. The high-level intent of NSGP is to identify groups of nodes that are related both structurally in the graph and in their attributes.

1.2 Problem Definition

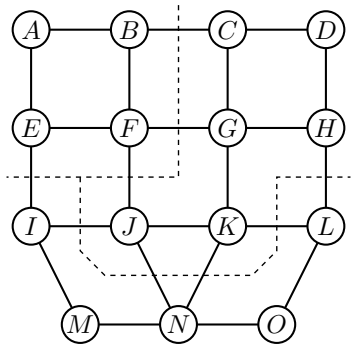
In our formulation of the NSGP problem, a node-attributed spatial graph is represented as a graph composed of nodes, edges, and node attributes. Each node represents a spatial location in geographic space and each edge represents the topological connectivity between two nodes. Each node has a set of numerical attributes that can characterize the aspects of the spatial location. The $NSGP(N, E, A, k, s, \lambda)$ problem is defined as follows:
Input: A node-attributed spatial graph G with

Node	Attributes			
A	0	50	51	0
B	0	49	52	1
C	48	2	0	51
D	100	3	100	2
E	1	49	50	1
F	1	50	49	0
G	101	2	99	1
H	99	3	100	1
I	50	0	0	49
J	99	1	101	2
K	102	1	100	3
L	41	0	1	50
M	49	0	1	49
N	50	1	0	51
O	51	1	1	49



(a) Input (Node-attributed Spatial Graph)

Node	Attributes			
A	0	50	51	0
B	0	49	52	1
E	1	49	50	1
F	1	50	49	0
C	48	2	0	51
D	100	3	100	2
G	101	2	99	1
H	99	3	100	1
J	99	1	101	2
K	102	1	100	3
I	50	0	0	49
L	41	0	1	50
M	49	0	1	49
N	50	1	0	51
O	51	1	1	49



(b) Output (best in color)

Figure 1: Example of the Input and Output of NSGP (k = 3, s = 4, and λ = 1)

- a set of nodes N and a set of edges E,
- a set of node attributes $A = \{a_1, a_2, \ldots, a_m\}$ for each node $n \in N$,
- the number of sub-graphs k,
- the minimum number of nodes in a sub-graph (i.e., size constraint) s, and
- the weight multiplier λ for $RMSE_{rank1}$

Output: k homogeneous sub-graphs

Objective:

- Minimize $\lambda \cdot RMSE_{rank1}$ + *edge-cuts*.

Constraints:

- Size Constraint (s): Every sub-graph has at least s nodes

The objective of the NSGP problem is minimizing both total $RMSE_{rank1}$ and *edge-cuts*. In this paper, we formulate the objective of NSGP as a single objective by assigning a weight (i.e., λ) to $RMSE_{rank1}$ [18]. The value of λ is used to control the importance of homogeneous sub-graphs. The number of partitions (i.e., k) is used for the resolution of partitions. The size constraint (i.e., s) is used to remove a trivial solution (e.g., a partition with a few nodes).

Definition 1. *Rank-one decomposition: Given a matrix M, the rank-one decomposition factorizes M into a product of two vectors: u and v (i.e., $M = u \cdot v^T$).*

Definition 2. *$RMSE_{rank1}$: Given a matrix M and $u \cdot v^T$, $RMSE_{rank1}$ is the standard deviation of the difference between M and $u \cdot v^T$.*

Definition 3. *Edge-cuts: Given a set of sub-graphs, edge-cuts is the number of edges with endpoints in different sub-graphs.*

1.3 Problem Hardness

The NP-hardness of NSGP follows from a well-known result about the NP-hardness of the balanced min-cut graph partitioning problem.

Theorem 1. *The NSGP problem is NP-hard.*

Proof. The NP-hardness of NSGP follows from a well-known result about the NP-hardness of the following balanced min-cut graph partitioning (BMGP) problem [3]. Given a graph $G = (N, E)$, where N denotes a set of nodes and E a set of edges, the goal of BMGP is to partition N into k equal-sized parts $N_1, N_2, \ldots, N_k$ while minimizing *edge-cuts*. Let $X = (N, E, k)$ be an instance of BMGP. Let $Y = (N, E, A, k, s, \lambda)$ be an instance of NSGP, where N is a set of nodes, E is a set of edges, A is a set of node attributes, k is the number of sub-graphs, s is the minimum number of nodes in a sub-graph, and λ is the weight multiplier for $RMSE_{rank1}$. Then it is easy to show that the instance of BMGP is a special case of NSGP, where $s = |N|/k$ and $\lambda = 0$. Since X is constructed from Y in polynomial-bounded time, the proof is complete. □

1.4 Our Contributions

In this paper, we propose a novel algorithm, Clustering and Local Refinement (CLR), that partitions a node-attributed spatial graph into k homogeneous sub-graphs that can minimize both the total $RMSE_{rank1}$ and *edge-cuts* while meeting a size constraint. The proposed approach consists of three main components: 1) Initial solution based on a hierarchical clustering strategy, 2) Homogeneity measurement using $RMSE_{rank1}$, and 3) Local refinement using the generalized k-way Fiduccia–Mattheyses (FM) algorithm [11]. Our contributions are as follows:

- We introduce a new spatial graph partitioning problem, namely the Node-attributed Spatial Graph Partitioning (NSGP) problem.
- We prove that the NSGP problem is NP-hard.
- We propose the Clustering and Local Refinement (CLR) approach for the NSGP problem.
- We provide a cost model for our proposed approach.
- We experimentally evaluate our proposed approach using U.S. Census datasets [1] and HP#6 watershed network datasets [16]. Experimental results and a case study demonstrate that the proposed algorithm outperforms the baseline algorithm and creates a solution of NSGP.

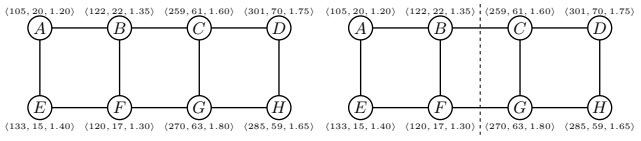
1.5 Related Work

Approaches to partitioning or clustering node-attributed graphs range from converting node attributes into edge weights [8, 24], defining distance functions to apply traditional distance-based clustering techniques [6], random walk distance [29], and statistical inference [27]. Each edge can be weighted by the similarity between the attributes of its endpoint nodes. Afterwards, an existing algorithm for partitioning edge-weighted graphs is applied. This requires the selection of a similarity function, such as the extended matching coefficient [24]. Traditional distance-based clustering techniques can be used by defining a distance function that

combines structural and attribute similarity [6]. A neighborhood random walk distance on node-attributed graphs can be defined by counting the number of attributes that two nodes share [29]. However, this approach is only applicable for categorical attributes. Statistical inference approaches can be used to partition node-attributed graphs by treating the input graph and the attributes as observations and attempting to predict a partition class for each node using a statistical method. For example, a generative Bayesian model that produces samples of all possible partitionings of a graph can be used to find desirable partitionings [27]. Although such models combine topological features and attributes, they often require costly parameter optimization and non-trivial expertise to choose the required a priori distributions [5]. However, no existing approach incorporates the min-cut objective, the size constraint, and the group homogeneity measurement into the spatial graph partitioning problem. The min-cut objective is important for identifying spatially connected regions. The size constraint is useful for excluding a trivial solution, where a single node becomes one partition (i.e. a sub-graph). In addition, the group homogeneity measurement with noise and irrelevant/redundant attributes is critical for discovering meaningful sub-graphs. In this work, we propose a novel approach for NSGP that honors the size constraint and minimizes both the total $RMSE_{rank1}$ and *edge-cuts*.

1.6 Basic Concepts

1.6.1 Node-attributed spatial graph. A node-attributed spatial graph is a graph where each node represents a location in geographic space, each edge represents the topological connectivity between two nodes, and each node has a set of numerical attributes that can characterize the aspects of the spatial location. Consider a network representing houses in a neighborhood. Each node represents a spatial location (e.g. a house) and each edge represents a road segment. Assume that the property value of a house is affected by its age and square footage. We can collect datasets regarding the property values of houses (in thousands of dollars), their ages (in years), and their square footage (in thousands of square feet). Let a = ⟨ property value, age, square footage ⟩. Then we can represent the network as a node-attributed spatial graph.



(a) Input (k = 2, s = 4, and λ = 1) (b) Output

Figure 2: Example of the Input and Output of NSGP

Figure 2(a) shows an example of a node-attributed graph with 8 nodes, 10 edges, and 3 attributes for each node. The attributes associated with each node are indicated by the vector displayed above or below the node. The objective of NSGP is to partition the spatial graph into k sub-graphs such that $\lambda \cdot RMSE_{rank1} + edge\text{-}cuts$ is minimized. This means each sub-graph should be topologically well-connected and the set of attributes within a sub-graph should be close to one another in terms of homogeneity (i.e. $RMSE_{rank1}$). Let k = 2, s = 4, and λ = 1. Figure 2(b) shows an example output of NSGP. The graph is partitioned into two sub-graphs, separated by the dashed line. The left sub-graph contains the nodes A, B, E, and F and the right sub-graph contains the nodes C, D, G, and H. The value of the objective function $\lambda \cdot RMSE_{rank1} + edge\text{-}cuts$ for this partitioning is $1 \cdot 4 + 2 = 6$. The computation of $RMSE_{rank1}$ will be explained in the following subsection.

1.6.2 Rank-one decomposition. Given a sub-graph G_{sub}, the attribute matrix of G_{sub} is defined as a matrix where each row is the attributes of one of the nodes in G_{sub}. Figure 3 shows an example attribute matrix for a given sub-graph of nodes in a partition. Note that the order of the rows in the attribute matrix is not important [25].

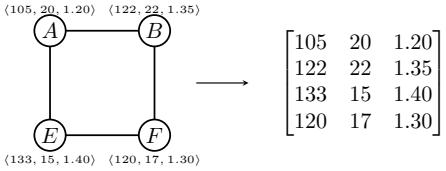


Figure 3: A node-attributed sub-graph and a corresponding attribute matrix

The Root Mean Square Error (RMSE) measures the difference between two matrices [28]. Given two matrices $A_{n\times m}$ and $B_{n\times m}$, the RMSE between A and B is defined as:

$$RMSE(A_{n\times m}, B_{n\times m}) = \sqrt{\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}(a_{ij} - b_{ij})^2}, \quad (1)$$

where a_{ij} is the element of A at the ith row and jth column and b_{ij} is the element of B at the ith row and jth column.

Given a matrix M, the rank-one decomposition factorizes M into a product of two vectors: u and v (i.e., $M = u \cdot v^T$). $RMSE_{rank1}(M)$ is defined as $RMSE(M, u \cdot v^T)$. $RMSE_{rank1}(M)$ can be used to measure the similarity of the row vectors in M (see Lemma 2.1). Furthermore, $RMSE_{rank1}(M)$ can efficiently identify homogeneous groups even with the presence of irrelevant, redundant, and noisy attributes [25]. As the value of $RMSE_{rank1}(M)$ decreases, the homogeneity of the row vectors in M increases. This is because M can be decomposed into u and v with a lower $RMSE_{rank1}$ when the row vectors (or column vectors) in M are similar to one another.

2 PROPOSED APPROACH

In this section, we introduce our novel approach, Clustering and Local Refinement (CLR), to the NSGP problem. CLR consists of three main components: (1) Construction of an initial solution based on a hierarchical clustering strategy, (2) Homogeneity measurement using $RMSE_{rank1}$, and (3) Local refinement using the generalized k-way Fiduccia–Mattheyses (FM) algorithm.

2.1 Initial solution based on hierarchical clustering strategy

CLR starts by constructing an initial solution based on a hierarchical clustering strategy. Consider the example node-attributed graph shown in Figure 4 (reproduced from Figure 1(a)). The nodes and edges are illustrated on the right-hand side while the corresponding attributes for each node are listed on the left-hand side. For

instance, node A is adjacent to nodes B and E and has the attributes $\langle 0, 50, 51, 0\rangle$.

Node	Attributes			
A	0	50	51	0
B	0	49	52	1
C	48	2	0	51
D	100	3	100	2
E	1	49	50	1
F	1	50	49	0
G	101	2	99	1
H	99	3	100	1
I	50	0	0	49
J	99	1	101	2
K	102	1	100	3
L	41	0	1	50
M	49	0	1	49
N	50	1	0	51
O	51	1	1	49

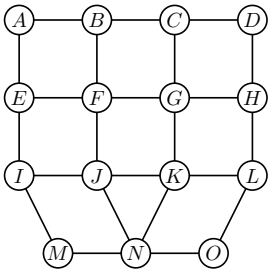
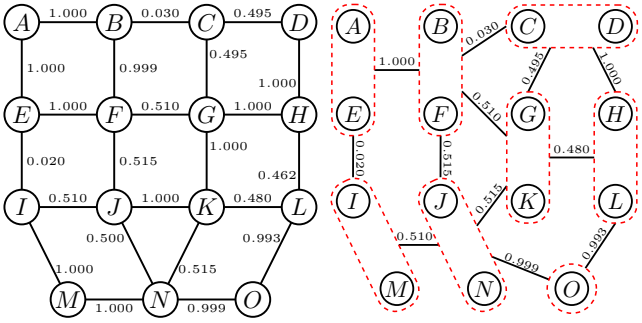


Figure 4: An example input graph with corresponding node attributes

Let the number of sub-graphs be 3 (i.e., $k = 3$) and let the size constraint s be 4. First, CLR converts the node-attributed graph into an edge-weighted graph where the weight of each edge is the cosine similarity between the attributes of its two nodes. Cosine similarity is defined as

$$cosine{-}similarity(a, b) = \frac{a \cdot b}{\|a\|\|b\|}, \tag{2}$$

where a and b are vectors.

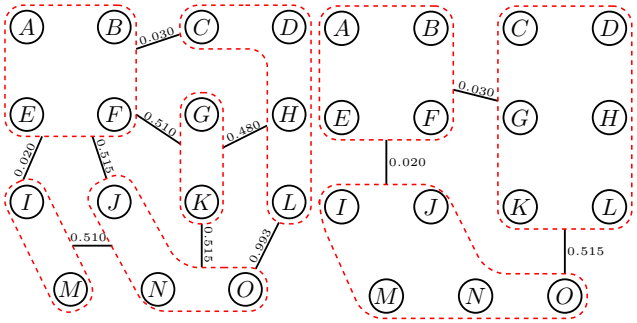


(a) Step 1: Edge-weighted Spatial Graph (b) Step 2: 1st-level matching

Figure 5: Initial Solution Construction (Steps 1-2)

Consider the example input in Figure 4. Figure 5(a) shows the edge-weighted graph that results from assigning each edge the cosine similarity between its two nodes as a weight. Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space [26]. However, it is important to note that cosine similarity cannot directly measure the similarity between groups because it compares only two non-zero vectors. To remedy this, CLR groups similar nodes and measures similarities between groups in a hierarchical fashion.

CLR groups pairs of nodes into a single node by using the highest-weighted edge between them. We refer to this single node as a super-node. This grouping process can be generalized to the maximal matching problem [17]. Figure 5(b) shows the output of the first-level matching. Every node can be matched with at most one node to form a super-node (represented as dashed ovals). If a node has no other node to match with, then it becomes a super-node with a single node (see node O).



(a) Step 3: 2nd-level matching (b) Step 4: 3rd-level matching (Output)

Figure 6: Initial Solution Construction (Steps 3-4)

After the first-level matching, CLR defines the similarity between two super-nodes as the smallest-weighted edge between them and continues to group super-nodes based on the maximal matching. The size of every super-node should be bounded and approximately balanced. Figure 6(a) shows the second-level matching. In this example, $\{A, E\}$ is merged with $\{B, F\}$, $\{C, D\}$ is merged with $\{H, L\}$, and $\{J, N\}$ is merged with $\{O\}$. Figure 6(b) shows the output of the third-level matching. $\{C, D, H, L\}$ is merged with $\{G, K\}$, and $\{I, M\}$ is merged with $\{J, N, O\}$. Since the number of groups is now 3 (i.e., $k = 3$), which is the desired number of partitions, the initial solution is complete.

2.2 Homogeneity measurement with $RMSE_{rank1}$

Cosine similarity can only compare two non-zero vectors. In addition, it has a limited ability to measure the homogeneity of groups that consist of more than two vectors. The core idea of CLR is to utilize the rank-one decomposition to measure group homogeneity.

The rank-one decomposition factorizes a matrix M into two vectors (i.e., u and v), aiming to minimize $RMSE_{rank1}$. $RMSE_{rank1}$ is used as a measure of homogeneity of the attributes in a sub-graph (Lemma 2.1). When the attributes of a sub-graph are close in value or follow similar patterns, the rank-one decomposition of the attribute matrix is able to better approximate the original attribute matrix and so the $RMSE_{rank1}$ will be lower, indicating higher homogeneity. Consider the following example of a rank-one decomposition.

$$M = \begin{bmatrix} 2 & 2 & 2 \\ 3 & 3 & 3 \\ 4 & 4 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 \\ 3 & 3 & 3 \\ 4 & 4 & 4 \end{bmatrix}$$

In this example, the matrix M can be decomposed into two vectors such that $RMSE_{rank1} = 0$. Since M can be completely represented by the product of two vectors, all vectors in M are considered homogeneous in terms of $RMSE_{rank1}$.

The Singular Value Decomposition (SVD) with the highest singular value can be used to find the rank-one decomposition [25]. However, since computing the SVD is expensive, CLR utilizes the Coordinate Descent (CD) optimization technique to factorize a matrix M into two vectors (i.e., u and v). The CD method starts with

an initial decomposition where the elements of the column vector and row vector are randomly chosen [7]. Then, it alternatingly and iteratively estimates the value of each element in the two vectors to minimize $RMSE_{rank1}$. Consider the following example of the CD method for the rank-one decomposition.

$$M = \begin{bmatrix} 2 & 4 & 7 \\ 3 & 6 & 9 \\ 4 & 8 & 12 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

Here, the elements of u and v are each set to 1. The Coordinate Descent (CD) method estimates the value of each element in the two vectors (i.e., u and v) using the following equations (see Lemma 2.2 and 2.3).

$$u_i = \frac{v^T \cdot r_i}{v^T \cdot v}, \tag{3}$$

where u_i is the ith element of vector u and r_i is the ith row vector of matrix M.

$$v_j = \frac{u^T \cdot c_j}{u^T \cdot u}, \tag{4}$$

where v_j is the jth element of vector v and c_j is the jth column vector of matrix M.

First, CD estimates the value of the first element of the column vector (i.e., u_1).

$$\begin{bmatrix} 2 & 4 & 7 \\ 3 & 6 & 9 \\ 4 & 8 & 12 \end{bmatrix} \rightarrow \begin{bmatrix} \alpha \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix},$$

$$\alpha = u_1 = \frac{v^T \cdot r_1}{v^T \cdot v} = \frac{(1,1,1) \cdot (2,4,7)}{(1,1,1) \cdot (1,1,1)} = 4.33$$

Next, CD estimates the value of the first element of the row vector (i.e., v_1).

$$\begin{bmatrix} 2 & 4 & 7 \\ 3 & 6 & 9 \\ 4 & 8 & 12 \end{bmatrix} \rightarrow \begin{bmatrix} 4.33 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} \alpha & 1 & 1 \end{bmatrix},$$

$$\alpha = v_1 = \frac{u^T \cdot c_1}{u^T \cdot u} = \frac{(4.33,1,1) \cdot (2,3,4)}{(4.33,1,1) \cdot (4.33,1,1)} = 0.75$$

Then, CD estimates the value of the second element of the column vector (i.e., u_2).

$$\begin{bmatrix} 2 & 4 & 7 \\ 3 & 6 & 9 \\ 4 & 8 & 12 \end{bmatrix} \rightarrow \begin{bmatrix} 4.33 \\ \alpha \\ 1 \end{bmatrix} \begin{bmatrix} 0.75 & 1 & 1 \end{bmatrix},$$

$$\alpha = u_2 = \frac{v^T \cdot r_2}{v^T \cdot v} = \frac{(0.75,1,1) \cdot (3,6,9)}{(0.75,1,1) \cdot (0.75,1,1)} = 6.73$$

This alternating estimation process continues until the value of $RMSE_{rank_1}$ no longer decreases. The final decomposition for this example is shown below.

$$\begin{bmatrix} 2 & 4 & 7 \\ 3 & 6 & 9 \\ 4 & 8 & 12 \end{bmatrix} \rightarrow \begin{bmatrix} 4.75 \\ 6.43 \\ 8.57 \end{bmatrix} \begin{bmatrix} 0.46 & 0.92 & 1.41 \end{bmatrix} = \begin{bmatrix} 2.19 & 4.37 & 6.70 \\ 2.96 & 5.92 & 9.07 \\ 3.94 & 7.88 & 12.08 \end{bmatrix}$$

This decomposition produces a matrix similar to the original matrix with $RMSE = 0.18$.

2.3 Local refinement using the generalized k-way FM algorithm

In this subsection, we describe the generalized k-way FM algorithm for the NSGP problem. CLR uses the following objective function to measure the *cost* of a partitioning:

$$cost = \lambda \cdot \sum_{i=1}^{k} RMSE_{rank1}(i) + edge\text{-}cuts, \tag{5}$$

where $RMSE_{rank1}(i)$ is the $RMSE_{rank1}$ of the ith partition, $edge\text{-}cuts$ is the number of edge cuts, and λ is a user-specified parameter that controls the importance of homogeneity over edge cuts. CLR begins with an initial partitioning of a node-attributed graph that meets the size constraint and iteratively moves nodes between partitions in order to minimize the cost of the partitioning. To reduce the potential increase in edge cuts, CLR moves only boundary nodes, i.e. nodes adjacent to nodes in a different partition. The *gain* of a move is defined as the amount of decrease in the objective function after the move is made:

$$gain = cost_{old} - cost_{new}, \tag{6}$$

where $cost_{old}$ and $cost_{new}$ are the costs before and after, respectively, the move.

Figure 7 shows an initial partitioning based on a hierarchical clustering strategy. Let the number of sub-graphs be 3 (i.e., $k = 3$), let the size constraint s be 4, and let the weight multiplier λ be 1. The Moves table shows all possible moves of a boundary node to an adjacent partition along with the gain of each move. The gain is computed as $-1 \cdot (\lambda \cdot \Delta RMSE + \Delta Cuts)$, so that the decreases in the objective function result in positive gains. The History table is used to record moves that have been made as well as the running net gain. The Locked table records nodes that have already been moved during this iteration, and so cannot be moved again until the next iteration [11, 14, 17].

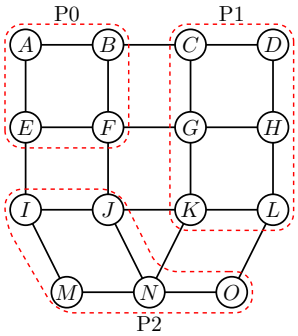
After all gains have been computed, CLR makes the move with the highest gain. Moves that violate the size constraint are not considered. Figure 8 shows that node J was moved into $P1$ (i.e., partition 1). After the move, node J is considered *locked*, and so it cannot be moved again for the remainder of the iteration [14]. Locked nodes are illustrated as gray instead of white. After the move, all nodes incident to the moved node must have their gains recomputed. The Moves table shows the updated available moves and their respective gains, and the History table records that node J has been moved to $P1$ for a net gain of 21.

Next, CLR makes the move with the highest gain. Figure 9 shows that the move with the highest gain is moving node L to $P2$, which has a gain of +3. Node L is then locked and the gains for incidents of node L are recomputed. The move is recorded in the History table, bringing the running net gain to 24.

This process repeats until no possible moves are left. Figure 10(a) shows the History table after the last possible move has been made. CLR uses the History table to identify that the net gain was highest

Node	Attributes			
A	0	50	51	0
B	0	49	52	1
C	48	2	0	51
D	100	3	100	2
E	1	49	50	1
F	1	50	49	0
G	101	2	99	1
H	99	3	100	1
I	50	0	0	49
J	99	1	101	2
K	102	1	100	3
L	41	0	1	50
M	49	0	1	49
N	50	1	0	51
O	51	1	1	49

(a) Attributes



(b) Sub-graphs

Attributes				
Node	Target	ΔRMSE	ΔCuts	Gain
B	P1	+2	+1	-3
C	P0	+10	+1	-11
E	P2	+2	+1	-3
F	P1	+2	+1	-3
F	P2	+2	+1	-3
G	P0	+22	+2	-24
I	P0	+15	+1	-16
J	P0	0	+1	-1
J	P1	-22	+1	+21
K	P2	+2	0	-2
L	P2	-4	+1	+3
N	P1	+2	+2	-4
O	P1	+2	0	-2

(c) Moves Table

History	
Move	Net Gain

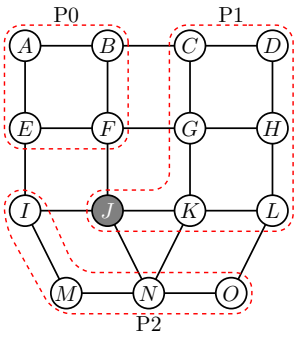
Locked

(d) History & Locked Tables

Figure 7: Initial solution (k = 3, s = 4, and λ = 1)

Node	Attributes			
A	0	50	51	0
B	0	49	52	1
C	48	2	0	51
D	100	3	100	2
E	1	49	50	1
F	1	50	49	0
G	101	2	99	1
H	99	3	100	1
I	50	0	0	49
J	99	1	101	2
K	102	1	100	3
L	41	0	1	50
M	49	0	1	49
N	50	1	0	51
O	51	1	1	49

(a) Attributes



(b) Sub-graphs

Attributes				
Node	Target	ΔRMSE	ΔCuts	Gain
B	P1	+2	+1	-3
C	P0	+10	+1	-11
E	P2	+2	+1	-3
F	P1	+2	+1	-3
F	P2	+2	0	-2
G	P0	+22	+2	-24
I	P0	+15	0	-15
I	P1	+2	0	-2
K	P2	+22	+2	-24
L	P2	-4	+1	+3
N	P1	+2	0	-2
O	P1	+2	0	-2

(c) Moves Table

History	
Move	Net Gain
J → P1	21

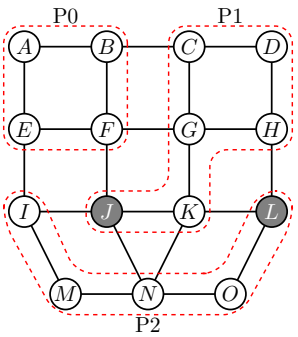
Locked
J

(d) History & Locked Tables

Figure 8: After 1st move (k = 3, s = 4, and λ = 1)

Node	Attributes			
A	0	50	51	0
B	0	49	52	1
C	48	2	0	51
D	100	3	100	2
E	1	49	50	1
F	1	50	49	0
G	101	2	99	1
H	99	3	100	1
I	50	0	0	49
J	99	1	101	2
K	102	1	100	3
L	41	0	1	50
M	49	0	1	49
N	50	1	0	51
O	51	1	1	49

(a) Attributes



(b) Sub-graphs

Attributes				
Node	Target	ΔRMSE	ΔCuts	Gain
B	P1	+2	+1	-3
C	P0	+10	+1	-11
E	P2	+2	+1	-3
F	P1	+2	+1	-3
F	P2	+2	0	-2
G	P0	+22	+2	-24
H	P2	+20	+1	-21
I	P0	+15	0	-15
I	P1	+2	0	-2
K	P2	+20	0	-20
N	P1	+2	0	-2

(c) Moves Table

History	
Move	Net Gain
J → P1	21
L → P2	24

Locked
J
L

(d) History & Locked Tables

Figure 9: After 2nd move (k = 3, s = 4, and λ = 1)

after node L was moved to $P2$, highlighted in red [14]. The partitioning after this move is chosen as the output of the iteration, shown in Figure 10(b).

Algorithm 1 shows the pseudocode for CLR. Line 1 starts by computing an initial solution using the hierarchical clustering strategy described in Section 2.1. Lines 2-11 improve the solution using the generalized k-way FM algorithm. Line 3 creates a copy of the current solution. Line 4 checks that there are unlocked boundary nodes to move. Line 5 examines all possible moves of an unlocked boundary node to an adjacent partition. Lines 6-7 make the best move and add the moved node to the History and Locked tables. Line 9 finds the point at which the net gain was maximum. Line 10 applies the moves up to the maximum point to the current solution. After t iterations of the local improvement, Line 12 returns the solution.

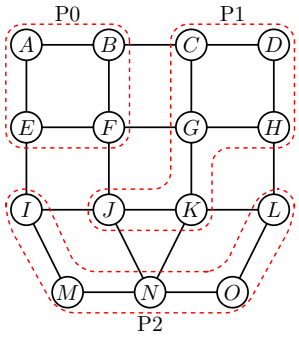
2.4 Analysis of CLR

In this section, we prove that CLR is correct, i.e., CLR creates a solution of NSGP.

History	
Move	Net Gain
J → P1	21
L → P2	24
N → P1	22
E → P2	19
C → P0	8
F → P1	3
G → P0	-5
H → P0	-5
D → P0	-3
K → P0	-4
B → P1	-6
A → P1	-1
M → P1	-4
I → P1	-6

Locked
J
L
N
E
C
F
G
H
D
K
B
A
M
I

(a) History & Locked Tables



(b) Output

Figure 10: After last move (k = 3, s = 4, and λ = 1)

Algorithm 1: CLR Algorithm (Pseudocode)

Input:
- A spatial graph G with a set of graph-nodes N and a set of edges E.,
- A set of node attributes $A = \{a_1, a_2, \ldots, a_m\}$ for each node $n \in N$,
- The number of sub-graphs k,
- The size constraint s,
- The maximum number of iterations, t

Output: k homogeneous, complete, and non-overlapping sub-graphs

Step:

```
Compute an initial solution, Π, using the hierarchical clustering strategy.
for up to t iterations do
    Create a copy of the current solution (i.e., Π_copy ← Π).
    while there are unlocked boundary nodes do
        Compute the gain of each possible move of an unlocked boundary
          node to an adjacent partition.
        In Π_copy, move the boundary node with the highest gain.
        Add the moved node to the History and Locked tables.
    end
    Identify the point at which the net gain was maximum in the History
      table.
    Apply all moves up to the maximum point to the current solution Π.
end
return Π (i.e., NSGP).
```

Lemma 2.1. *When all row vectors in a matrix M have the same direction but different magnitudes, $RMSE_{rank1}(M)$ becomes* 0.

Proof. Let r_i be the ith row vector of M. Assume that $r_i = c_i \cdot v$ and c_i is a constant. Then the rank-one decomposition of M is $(c_1, c_2, \ldots, c_n) \cdot v$. Therefore, $RMSE_{rank1}(M)$ is 0. □

Lemma 2.2. *Given a matrix M and two vectors u and v, $RMSE_{rank1}$ can be minimized when $u_i = \frac{v^T r_i}{v^T v}$, where r_i is the ith row vector of M.*

Proof. Let a_{ij} be the element of matrix $M_{n\times m}$ from the ith row and jth column, let u_i be the ith element of vector u, and let v_j be the jth element of vector v. According to Equation 1, $RMSE_{rank1}(M, u \cdot v^T) = \sqrt{\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}(a_{ij} - u_i \cdot v_j)^2}$. Minimizing $RMSE_{rank1}$ is equivalent to minimizing $\sum_{i=1}^{n}\sum_{j=1}^{m}(a_{ij} - u_i \cdot v_j)^2$. Assume that u_i is unknown. Since the function becomes strictly convex, $RMSE_{rank1}$ can be minimized if the first derivative of the function becomes 0 (i.e., $\frac{d}{du_i}((\sum_{i=1}^{n}\sum_{j=1}^{m}(a_{ij} - u_i \cdot v_j)^2) = 0$). Then we see that $\sum_{j=1}^{m}(a_{ij} - u_i \cdot v_j) \cdot v_j = v^T r_i - u_i v^T v = 0$. Therefore, $u_i = \frac{v^T r_i}{v^T v}$ can minimize $RMSE_{rank1}(M, u \cdot v^T)$. □

Lemma 2.3. *Given a matrix M and two vectors u and v, $RMSE_{rank1}$ can be minimized when $v_j = \frac{u^T c_j}{u^T u}$, where c_j is the jth column vector of $M_{n\times m}$.*

Proof. Let a_{ij} be the element of matrix $M_{n\times m}$ from the ith row and jth column, let u_i be the ith element of vector u, and let v_j be the jth element of vector v. Minimizing $RMSE_{rank1}$ is equivalent to minimizing $\sum_{i=1}^{n}\sum_{j=1}^{m}(a_{ij} - u_i \cdot v_j)^2$. Assume that v_j is unknown. Then, $RMSE_{rank1}$ can be minimized if $\frac{d}{dv_j}((\sum_{i=1}^{n}\sum_{j=1}^{m}(a_{ij} - u_i \cdot v_j)^2) = 0$. We see that $\sum_{i=1}^{n}(a_{ij} - u_i \cdot v_j) \cdot u_i = u^T c_j - v_j u^T u = 0$. Therefore, $v_j = \frac{u^T c_j}{u^T u}$ can minimize $RMSE_{rank1}(M, u \cdot v^T)$. □

2.4.1 Computational Complexity of CLR. Let n be the number of nodes, let m be the number of edges, let a be the number of attributes, let k be the number of partitions, let i be the number of iterations of the Coordinate Descent (CD) method, and let t be the number of passes (or iterations) for the k-way FM algorithm. Since a spatial graph is a sparse graph, m = $O(n)$ [4, 13]. First, CLR constructs an initial solution based on a hierarchical clustering strategy. The construction of the edge-weighted graph takes $O(n \cdot a)$. The hierarchical grouping process takes $O(n^2)$. Then, CLR computes $RMSE_{rank1}$ to measure the homogeneity of all groups. This takes $O(n \cdot a \cdot i)$. Afterwards, CLR uses the k-way Fiduccia-Mattheyses (FM) algorithm to re-optimize the partitions. This requires multiple passes (i.e., t) to identify the near-optimal solution. In each pass, CLR identifies the best boundary node and moves it to an adjacent partition to reduce the cost of the objective function. Next, it recomputes $RMSE_{rank1}$ for the updated group. This takes $O(n^2 + n \cdot a \cdot i)$. Since the number of boundary nodes is bounded by $O(n)$, each pass takes $O(n^3 + n^2 \cdot a \cdot i)$. Therefore, the cost model of CLR is $O((n^3 + n^2 \cdot a \cdot i) \cdot t)$. In practice, the number of iterations of both CD and FM is small, so we can set the two parameters (i.e., i and t) as constants.

3 EXPERIMENTAL EVALUATION

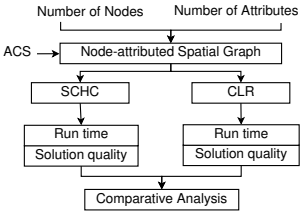
We conducted experiments to evaluate the performance of CLR. The experiments set out to answer: (1) What is the effect of the number of nodes? (2) What is the effect of the number of attributes? (3) What is the effect of the number of partitions, k? (4) What is the effect of the weight parameter, λ? (5) Is solution quality preserved? (6) Is CLR scalable?

3.1 Experiment Layout

Figure 11(a) shows our experimental setup. We used ACS 2016 data (see Figure 11(b)) and constructed the nearest neighbor spatial graph [1]. We fixed the size constraint to 30% less than the balanced partition size across all experiments.

Ideally, we would test our proposed algorithm against comparable algorithms from related work. Unfortunately, we found no algorithms in the literature that handle multiple numeric attributes, enforce a size constraint, and incorporate the min-cut objective. Instead, we provide a rough baseline comparison by applying hierarchical clustering using cosine similarity as the distance metric with

the single linkage criterion [20, 26]. We enforce a size constraint by disallowing merges that would result in clusters that violate the maximum allowed size. After all clusters have been formed, we incrementally move nodes from excess clusters (clusters whose sizes are greater than the minimum allowed size) to deficit clusters (clusters whose sizes are less than the minimum allowed size) [11]. When choosing a deficit cluster to move a node to, we choose the cluster with the closest centroid to the excess cluster's centroid. This results in a size-constrained hierarchical clustering (SCHC), which we used as a baseline comparison against CLR.



(a) Experiment Layout

Area	No. of Nodes ($\|N\|$)	No. of Edges ($\|E\|$)
Miami	682	2175
West Florida	1254	4004
Southeast Florida	3395	11260

(b) Datasets (Source: ACS 2016)[1]

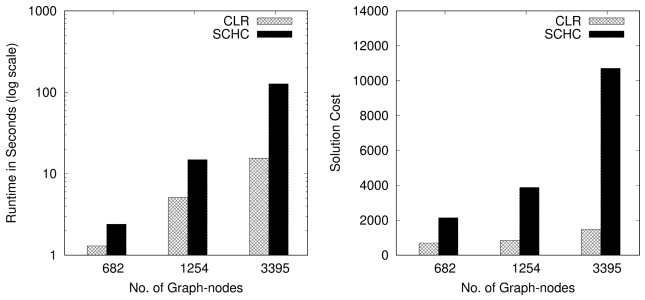
Figure 11: Experiment Setup

We evaluated CLR by comparing the impact on performance and solution cost of (1) the number of nodes, (2) the number of attributes, (3) the number of partitions, and (4) the value of the weight parameter λ. The algorithms were implemented in Java 1.8 with a 1 GB memory runtime environment. All experiments were performed on an Intel i5-6600K CPU machine running Windows 10 with 16 GB of RAM.

3.2 Experiment Results and Analysis

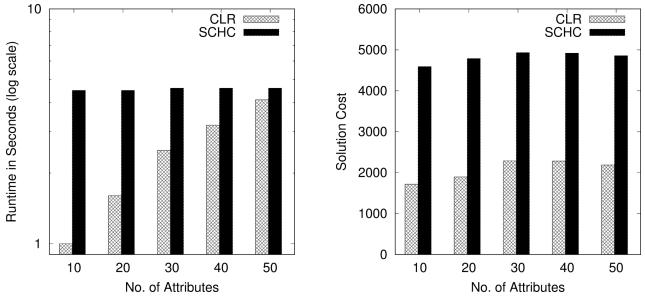
3.2.1 Effect of the number of nodes. The first set of experiments evaluated the effect of the number of nodes on the performance of CLR compared to SCHC. We used the three datasets described in Section 3.1 which had node counts of 682, 1,254, and 3,395, respectively. Parameters k, λ, and the number of attributes (i.e., a) were fixed to 25, 100, and 30, respectively. Figure 12 shows the execution times and solution costs for the different network sizes. CLR outperforms SCHC in both execution time and solution quality. SCHC starts with each node in its own cluster and incrementally merges similar clusters. SCHC considers all possible merges to find the optimal one whereas CLR moves only boundary nodes. SCHC is unaware of graph edges, so its partitionings have high numbers of edge cuts, leading to degradation of solution quality.

3.2.2 Effect of the number of attributes. The second set of experiments evaluated the effect of the number of attributes on the performance of CLR compared to SCHC. We varied the number of attributes (i.e., a) from 10 to 50. We used the West Florida dataset ($|N|$= 1,254) and fixed k and λ to 25 and 1,000, respectively. Figure 13 shows that CLR outperforms SCHC in both execution time and solution quality. As the number of attributes increases, the execution time of CLR increases because the attribute matrix will have more columns, making the computation of $RMSE_{rank1}$ more expensive. SCHC only needs to compute the cosine distance between each pair of node attributes, which does not take substantially longer from 10 attributes to 50 attributes.



(a) Effect on Execution Time (b) Effect on Solution Quality

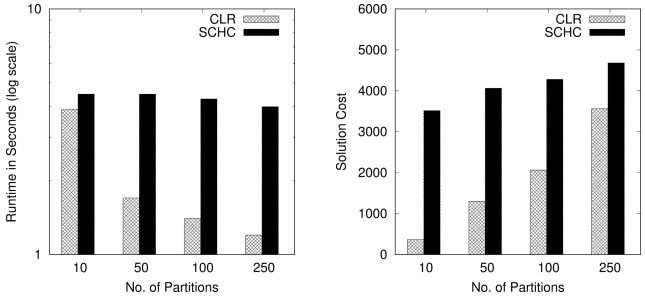
Figure 12: Effect of the number of graph-nodes (k = 25, λ = 100, a = 30)



(a) Run-time Comparison (b) Comparison of Solution Quality

Figure 13: Effect of the number of attributes ($|N|$= 1,254, k = 25, λ = 1,000)

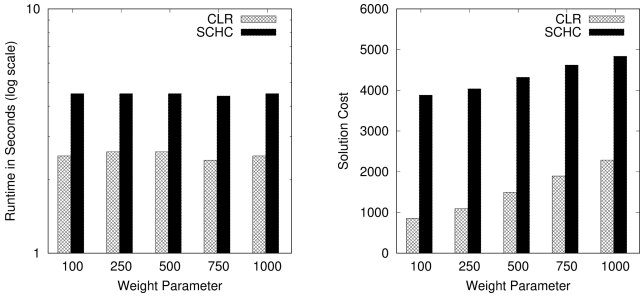
3.2.3 Effect of the number of partitions. The third set of experiments evaluated the effect of the number of partitions, k, on the performance of CLR compared to SCHC. We varied the number of partitions (i.e., k) from 10 to 250. We used the West Florida dataset ($|N|$= 1,254) and fixed λ and the number of attributes (i.e., a) to 100 and 30, respectively. Figure 14 shows that CLR outperforms SCHC in both execution time and solution quality. As the number of partitions increases, the execution time of CLR decreases. This is because the size of the attribute matrices decreases as the number of partitions increases. The solution quality degrades as k increases because the number of *edge-cuts* increases.



(a) Run-time Comparison (b) Comparison of Solution Quality

Figure 14: Effect of the number of partitions ($|N|$= 1,254, λ = 100, a = 30)

3.2.4 Effect of the weight parameter. The fourth set of experiments evaluated the effect of the weight parameter, λ, on the performance of CLR compared to SCHC. We varied the weight parameter from 100 to 1000. We used the West Florida dataset ($|N| = 1,254$) and fixed k and the number of attributes (i.e., a) to 25 and 30, respectively. Figure 15 shows CLR outperforms SCHC in both execution time and solution quality. The execution times of both algorithms were not affected by the value of λ. The solution qualities of both algorithms degrade slightly as the value of λ increases. This is because increasing the value of λ causes the value of the objective function to increase as well.

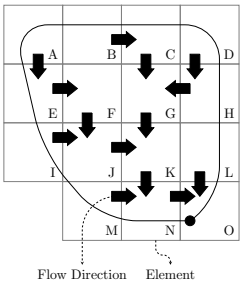


(a) Run-time Comparison (b) Comparison of Solution Quality

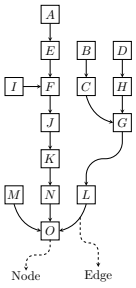
Figure 15: Effect of the weight parameter ($|N| = 1,254$, $k = 25$, $a = 30$)

3.3 Case Study

In our case study, we look at how NSGP partitions a watershed into homogenous sub-graphs to delineate hydrological response units (HRUs). A node-attributed spatial graph can efficiently represent a watershed. The watershed area can be decomposed into cells. Each edge signifies the direction of flow movement betwen two cells (i.e., nodes) (see Figure 16(a) and 16(b)). Each cell has multiple spatial attributes, such as land use and land cover (LULC), soil, and slope (see Figure 16(c)).



(a) Grid Cells



(b) Spatial Graph

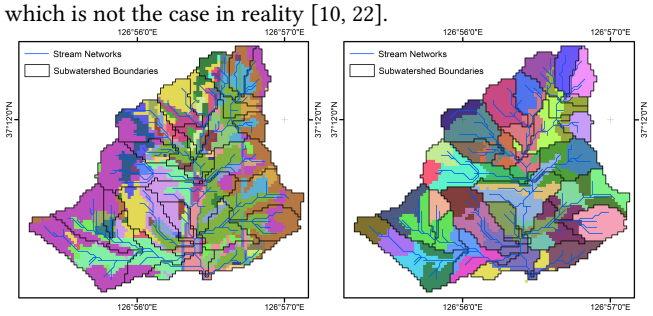
Node ID	Node Attribute			
	To	LULC	Soil	Slope
A	E	41	SL	5
B	C	41	SL	6
C	G	41	SL	7
D	H	41	SL	8
E	F	41	SL	6
F	J	82	SL	3
G	K	82	SL	4
H	G	41	SL	6
I	J	82	SIL	4
J	K	82	SIL	2
K	N	81	SIL	2
L	O	22	CL	3
M	N	81	CL	3
N	O	90	SIC	1
O	-	90	SIC	0

(c) Node Attributes

Figure 16: Graph representation of a watershed

A hydrological response unit (HRU) is the smallest spatial unit that has "common land use and pedo-topo-geological associations generating and controlling their homogeneous hydrological dynamics" [12]. An HRU is often described as a group of individual cells that have the same or similar hydrological characteristics (or attributes). The HRU delineation is important for understanding the spatial heterogeneity and complexity of a watershed as well as for improving the hydrological modeling efficiency [12, 15].

The traditional way of delineating HRUs is to find common areas of overlap for LULC and soil layers [12]. Since the overlapping method does not consider the topological relationships between cells, the nodes in the sub-graph may not be hydrologically connected to each other. Thus, no interaction between HRUs is assumed, which is not the case in reality [10, 22].



(a) Overlapping Method (b) NSGP

Figure 17: Delineating hydrological response units (HRUs) using the traditional overlapping method and NSGP.

In our case study, we compared the results of the overlapping method and CLR. We used the geospatial layers of the HP#6 watershed in South Korea [16]. The spatial graph of the watershed landscape consists of 4,201 nodes, 4,209 edges, and 3 attributes. The 30-m resolution digital elevation model (DEM) was rasterized from 1 : 5000 vector maps developed from the National Geographic Information Institute [21]. The DEM was used to derive slope, define stream flows, and delineate the watershed boundary. The soil texture was rasterized from 1 : 25,000 vector maps and the dataset was provided by the Rural Development Administration [23]. The land use information was obtained from the Ministry of Environment [19]. The number of HRUs (i.e., k) was set to 42 derived from the number of unique combinations of LULC and soil classes. The slope values were considered in CLR but not in the overlapping method because we wanted to test if CLR can remove noisy or redundant information for identifying homogeneous sub-graphs. We set the size constraint s to 0.90 to accommodate the variations found in the sizes of HRUs based on the overlapping method. We set the weight multiplier λ for $RMSE_{rank_1}$ to 1,000.

Figure 17 shows the output of the overlapping method and NSGP. The overlapping method created HRUs that were often divided by the ridge lines (the subwatershed boundary lines in black in Figure 17(a)), which violates the fundamental assumption of similar hydrological behavior. Surface runoff generated in an HRU is assumed to be transported to common downstream areas, but HRU parts divided by the ridge are connected to different downstream areas. On the other hand, NSGP delineated HRUs along the water flows (or paths) and identified homogeneous sub-graphs even though the node attributes included noisy information (i.e., slope). The result clearly demonstrates that CLR can effectively identify HRUs based on the homogeneity of sub-graphs and topological connectivity.

3.4 Discussion

CLR achieves a significant computational performance gain over SCHC. This improvement was obtained by three key components: (1) Initial Solution based on hierarchical clustering strategy, (2) homogeneity measurement using $RMSE_{rank1}$, (3) Local refinement

using the generalized k-way FM algorithm. Since the problem is a discrete optimization problem and is not convex, we limit the number of iterations to a threshold (t) that can minimize the objective function sufficiently. The parameter λ is crucial to achieving a tradeoff between connectivity (or contiguity) and homogeneity. The correct choice of λ is not trivial because the best output depends on shape (i.e., *edge-cuts*), homogeneity (i.e., $RMSE_{rank1}$), desired partition resolution (i.e, k), and size (i.e., s) for the users. CLR is an exploratory method for the Node-attributed Spatial Graph Partitioning problem. If we have no prior information (e.g., domain expert opinions), we can perform sensitivity analysis to identify the best parameters. In our experiments, we see that 100 or $1,000$ are the best choices for λ to clearly show the contiguous and homogeneous sub-graphs using ACS datasets. The case study demonstrates that CLR produces partitions with semantic value, i.e. they identify HRU boundaries present in the input spatial network. When the datasets contain null or missing attributes, we can use mean imputation to estimate the values. A better approach is to modify $RMSE_{rank1}$ to account for only available attributes [2]. The CLR method uses node-attributes to identify homogeneous sub-graphs. For edge-attributes, we can convert the graph to a line graph and construct a node-attributed graph. For both node and edge attributes, we can split a node (or edge) into two nodes (or two edges) and create a virtual edge (or node) between the two nodes (edges) to construct a node-attributed graph [9]. These approaches do not lose information regarding topological connectivity. However, the graph transformation increases the size of the input spatial graph.

4 CONCLUSION AND FUTURE WORK

We presented the problem of Node-attributed Spatial Graph Partitioning (NSGP). An important potential application of NSGP is identifying well-connected homogeneous groups in spatial networks by identifying partitions where the nodes have similar attributes and are spatially close to one another. NSGP is challenging because of the large size of spatial graphs and the constraint that the sub-graphs must be homogeneous, i.e. similar in terms of node attributes. In this paper, we proposed the Clustering and Local Refinement (CLR) approach to the NSGP problem, which partitions a node-attributed spatial graph such that the attribute similarity between nodes in the same partition is maximized and the number of edge cuts is minimized. We presented experiments and a case study using real-world spatial network datasets.

In future work, we will explore new initialization methods for various spatial graphs to improve the performance of CLR. We will also study the effect of adding a contiguity constraint to the output partitioning. Lastly, we will investigate parallel formulations of the CLR algorithm. The rank-one decomposition can be parallelly executed using multiple threads or big data processing platforms (e.g., Hadoop and Spark). Multi-level graph partitioning techniques can be utilized to reduce the size of a spatial-graph. We plan to study scalable techniques for CLR to efficiently handle big spatial graph datasets.

ACKNOWLEDGMENTS

We would like to thank the National Science Foundation under Grant No. 1844565. We are particularly thankful to the ACM SIGSPATIAL reviewers for their helpful comments.

REFERENCES

[1] ACS. 2016. "American Community Survey (ACS)". https://www.census.gov. Online; accessed Dec. 2019.
[2] Charu C Aggarwal. 2020. *Linear Algebra and Optimization for Machine Learning: A Textbook*. Springer Nature.
[3] Konstantin Andreev and Harald Racke. 2006. Balanced graph partitioning. *Theory of Computing Systems* 39, 6 (2006), 929–939.
[4] Marc Barthélemy. 2011. Spatial networks. *Physics Reports* 499, 1-3 (2011), 1–101.
[5] Cécile Bothorel, Juan David Cruz, Matteo Magnani, and Barbora Micenkova. 2015. Clustering attributed graphs: models, measures and methods. *Network Science* 3, 3 (2015), 408–444.
[6] David Combe, Christine Largeron, Elöd Egyed-Zsigmond, and Mathias Géry. 2012. Combining relations and text in scientific network clustering. In *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE, 1248–1253.
[7] Pierre Comon, Xavier Luciani, and André LF De Almeida. 2009. Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics: A Journal of the Chemometrics Society* 23, 7-8 (2009), 393–405.
[8] Juan David Cruz, Cécile Bothorel, and François Poulet. 2014. Community detection and visualization in social networks: Integrating structural and semantic information. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 1 (2014), 1–26.
[9] Reinhard Diestel. 2017. *Graph Theory*. Springer-Verlag Berlin Heidelberg.
[10] Zachary M Easton, M Todd Walter, Daniel R Fuka, Eric D White, and Tammo S Steenhuis. 2011. A simple concept for calibrating runoff thresholds in quasi-distributed variable source area watershed models. *Hydrological Processes* 25, 20 (2011), 3131–3143.
[11] Charles M Fiduccia and Robert M Mattheyses. 1982. A linear-time heuristic for improving network partitions. In *19th Design Automation Conference*. IEEE, 175–181.
[12] Wolfgan-Albert Flügel. 1997. Combining GIS with regional hydrological modelling using hydrological response units (HRUs): An application from Germany. *Mathematics and Computers in Simulation* 43, 3-6 (1997), 297–304.
[13] Michael T Gastner and Mark EJ Newman. 2006. The spatial structure of networks. *The European Physical Journal B-Condensed Matter and Complex Systems* 49, 2 (2006), 247–252.
[14] Fred Glover. 1990. Tabu search: A tutorial. *Interfaces* 20, 4 (1990), 74–94.
[15] Younggu Her, Jane Frankenberger, Indrajeet Chaubey, and Raghavan Srinivasan. 2015. Threshold effects in HRU definition of the soil and water assessment tool. *Transactions of the ASABE* 58, 2 (2015), 367–378.
[16] MS Kang, SW Park, JJ Lee, and KH Yoo. 2006. Applying SWAT for TMDL programs to a small watershed containing rice paddy fields. *Agricultural Water Management* 79, 1 (2006), 72–92.
[17] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. 1999. Multilevel hypergraph partitioning: applications in VLSI domain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 7, 1 (1999), 69–79.
[18] R Timothy Marler and Jasbir S Arora. 2004. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization* 26, 6 (2004), 369–395.
[19] MOE. 2019. "Ministry of Environment (MOE)". https://eng.me.go.kr/eng/web/main.do. Online; accessed Dec. 2019.
[20] Fionn Murtagh. 1983. A survey of recent advances in hierarchical clustering algorithms. *The computer journal* 26, 4 (1983), 354–359.
[21] NGII. 2019. "National Geographic Information Institute (NGII)". https://www.ngii.go.kr/eng/main.do. Online; accessed Dec. 2019.
[22] H Rathjens, K Bieger, I Chaubey, JG Arnold, PM Allen, R Srinivasan, DD Bosch, and M Volk. 2016. Delineating floodplain and upland areas for hydrologic models: a comparison of methods. *Hydrological Processes* 30, 23 (2016), 4367–4383.
[23] RDA. 2019. "Rural Development Administration (RDA)". http://www.rda.go.kr/foreign/ten/. Online; accessed Dec. 2019.
[24] Karsten Steinhaeuser and Nitesh V Chawla. 2008. Community detection in a large real-world social network. In *Social computing, behavioral modeling, and prediction*. Springer, 168–175.
[25] Gilbert Strang. 2019. *Linear algebra and learning from data*. Wellesley-Cambridge Press.
[26] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. 2016. *Introduction to data mining*. Pearson Education India.
[27] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. 2012. A model-based approach to attributed graph clustering. In *Proceedings of the 2012 ACM SIGMOD international conference on management of data*. 505–516.
[28] Jieping Ye. 2005. Generalized low rank approximations of matrices. *Machine Learning* 61, 1-3 (2005), 167–191.
[29] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment* 2, 1 (2009), 718–729.