A RICCI-FLOW APPROACH TO THE SCHOOL BUS ROUTING PROBLEM

ALEX YUK

DEERFIELD ACADEMY

ZYUK21@DEERFIELD.EDU

ADVISOR: SHULIANG BAI & KYLE LUH

Abstract. The school bus routing problems seek to select a set of bus stops that will be visited by the buses and determine which stop each student should walk to and develop a set of tours that minimize the total distance travelled by the school bus. In this paper, we give an approach to the school bus routing problem by applying a novel method called discrete Ricci-flow at the stage of clustering students locations. The Ricci flow is originally a geometric approach that is used to decompose smooth manifolds. With the development of discrete Ricci curvature, many geometric ideas and methods have been applied to discrete settings such as the complex network. Very recently, [10] has applied Ricci flow method to detect the community structure and experimentally confirmed the effectiveness of this geometric approach. This clustering algorithm plus a graph center algorithm help to estimate the optimal location of the bus stops and eventually increase the efficiency of school bus routing. Computational experiments are performed using real data.

Date: August 2019.

Schoolbus routing, Stop selection, Ricci-flow, Cluster Algorithm

1. Introduction

The School Bus Routing Problem (SBRP) has been intensively studied since its introduction in 1969 [9]. It is an important example of a theoretical challenge that also has immediate real-word applications. There are various modifications and subproblems one can consider, however the basic problem is to design a bus route and bus-stop locations to minimize either the distance traveled by the buses or the number of buses [13]. Of course, if a single bus is to pick up all the students from their homes then the problem is identical to the notorious Traveling Salesman Problem. The key difference is that in the SBRP formulation, the children are allowed to travel a certain distance to meet the bus at a designated bus stop. Traditionally, the SBRP can be divided into the following subproblems.

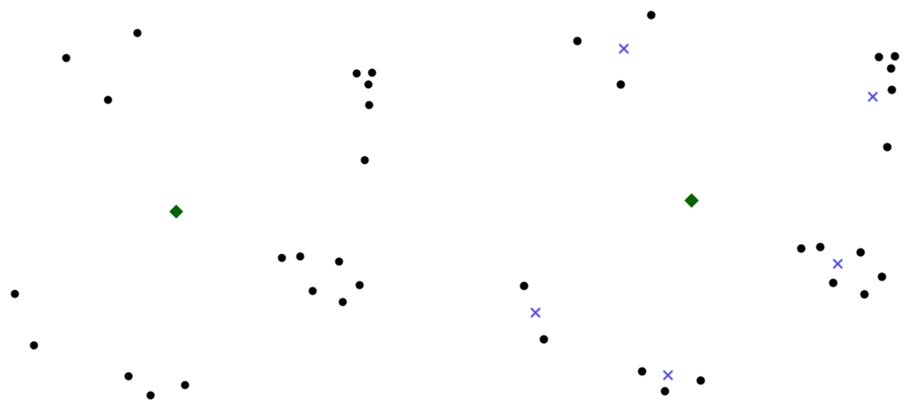
(1) *Data Preparation*: In this stage, the goal is to manipulate the raw data of student locations, schools and school times into a coherent model for the remaining subproblems.

(2) *Bus stop selection*: The goal of this sub-problem is to specify the location of the bus stops and to assign students to these stops. Possible constraints can include the distance students must travel to the stops and the total number of stops.

(3) *Bus route generation*: In this subproblem, one designs efficient routes for the buses to pick up the students at the various stops and drop them off at their respective schools.

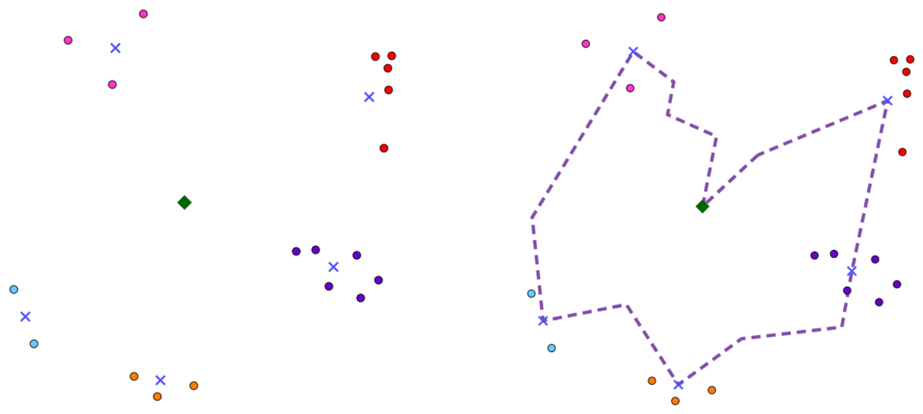
(4) *School bell time adjustment and route scheduling*: In some versions of the SBRP, there are added constraints of the various school starting and ending times. Therefore, one would like to establish starting and ending times and possibly re-use buses for different routes.

In this technical report we utilize a novel algorithm to fix bus stops and assign students to those stops. Our data preparation step is to abstract away the actual roads and geographic data and replace this with a weighted graph indicating the distances needed to travel between students. This abstraction is crucial as the Euclidean distance between the houses do not necessarily coincide with the distance a bus must travel between these two houses. One subtlety we ignore during this project is that the distance from A to B is not necessarily the same as the distance from B to A. For example, the presence of one-way roads can render one direction much longer than the other. The next step utilizes a powerful recent development in mathematics. The notion of curvature in geometry dates back to the pioneering work of Gauss and Riemann. In several recent works, various authors have brought these deep geometric notions to bear on graph theory. We use a discrete version of Ricci flow to locate subsets of our graph with positive curvature. The Ollivier Ricci flow increases the weights of those edges with negative curvature and we exploit this phenomenon by removing edges with weights larger than a certain threshold. This partitions the graph into positive curvature clusters. We use these as the students assigned to a single stop.



(A) SBRP (B) Assigning bus stops

Figure 1



(A) Assigning students to stops (B) Bus route

Figure 2

As there are various possible complications one can add to these models, we begin, as an illustration, by applying our ideas to the simple setting of a single school with a single bus. Later on, we will explain the necessary modifications to handle multiple buses.

Even in the mathematics community, the idea of curvature and Ricci flow for graphs is relatively new. There have been very few works applying these results to problems in computer science (e.g. [10, 18]). We believe this approach offers new insights into the SBRP problem and vehicle routing problems in general. Furthermore, this approach is substantially different from all the previous methods.

The remainder of the report is organized as follows. In Section 2 we describe the current state of the SBRP. We survey the milestones in the research on all the subproblems of the SBRP. In Section 3, we give an introduction to some basic differential geometry that motivates the Ollivier-Ricci curvature on graphs. Next, we describe the details of the Ollivier-Ricci curvature and Ollivier Ricci flow . Also in this section, we specify the algorithmic aspects of discrete Ricci curvature and clustering nodes. Having established the mathematical background, in Section 4, we give an in-depth description of our SBRP algorithm. In this section, we also provide some empirical results from various simulations and discuss some possible metrics to evaluate the quality of our algorithm. In Section 5 we consider possible extensions of this work and point out several shortcomings of our analysis that could be rectified. Finally, we highlight and reiterate the main contributions of this work.

2. Previous Work

The SBRP has a long history and we refer the reader to the surveys [13, 15, 9] for a more thorough discussion of prior work. The SBRP falls under the classification of Vehicle Routing Problemms, which dates back to 1950's when Dantzig and Ramser used the mathematical programming formulation and algorithmic approach to solve the delivering gasoline to service stations problems. The first studies regarding SBRP were carried out by Newton and Thomas in 1969 then followed by numerous studies. SBRP is a combined problem of bus stop selection and bus route generation, it was shown in [1] that not only a single school SBRP is an NP-hard problem, also a single sub-problem such as bus stop selection and bus route generation are shown to be NP-hard problems. There have been various methods applied in these sub-problems. In this project, we use the "cluster-first, route-second" approach whose first phase is to groups the students into clusters satisfying the constraint: in each cluster the diameter is restricted according to the maximum allowable walking distance from students' home to the bus stop. In the phase of clustering, there have been some methods applied from the network clustering approaches into SBRP.

For example, a method called genetic sectoring based on a genetic algorithm is used in [19], a clustering based algorithm to generate a feasible solution for SBRP in an urban context [16].

At the phase of selecting bus stops, we know it can be easier to choose the best bus routes if there are less bus stops can be selected, thus an efficient clustering method at the bus stop selection stage is very important. In [5], the authors used a heuristic algorithm based on the Dominating Set Problem to find the minimum number of bus stops. In [17], the authors used a strategy to discretize the network points so that bus stops can be created in a large number of points and they use a pseudo-random algorithm based on a GRASP [5] Construction Phase to tackle the problem. As SBRP is an optimization problem, many optimization techniques has been applied, such as linear programming, non-linear programming, and dynamic programming. For example, the authors in [2] used a multi-objective districting algorithm to group students into clusters. The authors in [6] proposed a model for SBRP which is formulated as a mixed-integer programming problem and used a heuristic algorithm to solve the proposed model.

2.1. **Data Preparation work.** The SBRP in our project is formulated on a weighted graph $G = (V, E, w)$ with V representing the student locations, school and bus stops, E representing the relation between any pair of vertices and W characterizing the distance required to traverse from one vertices to another. At the data preparing stage, we delete edges with long travel distance between the endpoints.

Through the Boston Public Schools Transportation Challenge website, we are able to obtain over 20,000 real public school student addresses as they provided them in an excel file. In it, we choose a number of schools of different bus-taking student sizes to see how our algorithm works with different scenarios.

As the excel file only gave street addresses and geographic coordinates, we use Google Maps to calculate the distance between each of these locations. We write an algorithm on Google Apps Scripts to calculate the road distance between the locations and formatted the results into a weighted adjacent matrix. Another issue is that one way roads make traveling from one location to another different if going in the opposite direction. So we decided to set each of the weights to the larger of the two driving distances.

3. Discrete Ollivier Ricci Flow

3.1. **Differential Geometry Motivation.** The notion of curvature stems from the seminal work of Gauss and Riemann [4]. They developed the idea to capture how much space is curved. We work in the setting of Riemannian manifolds. Recall that manifolds are simply higher-dimensional generalizations of curves and surfaces such that locally, the manifold behaves like Euclidean space. A Riemannian manifold is a manifold endowed with a metric that allows one to define distance on the manifold. This allows one to define a geodesic as a curve such that for two points on the curve sufficiently close, the path along the geodesic is the shortest path between these two points. We refer the reader to [3] for a more complete account of these classical developments. We can now convey a basic idea behind *Ricci curvature*. Consider two points a and b near each other. Let v be a tangent vector at a and v' a tangent vector at b in the same direction as v (this can be defined rigorously). Ricci curvature measures the change in the distance between a and b as we travel along the geodesics along v and v' (See Fig. 3). Positive curvatures, for example a sphere, indicate that the geodesics get closer while negative curvatures result in growing distances. The key characterization of Ricci curvature that is most convenient for generarlization is as a condition on the average distance points need to travel to map one sphere in the tangent space to another [21]. It will serve no purpose for us to state this precisely. We only emphasize that there is a formulation of Ricci curvature involving transport of mass. There are other ways to interpret Ricci curvature, but these will not be necessary for our discussion [7].

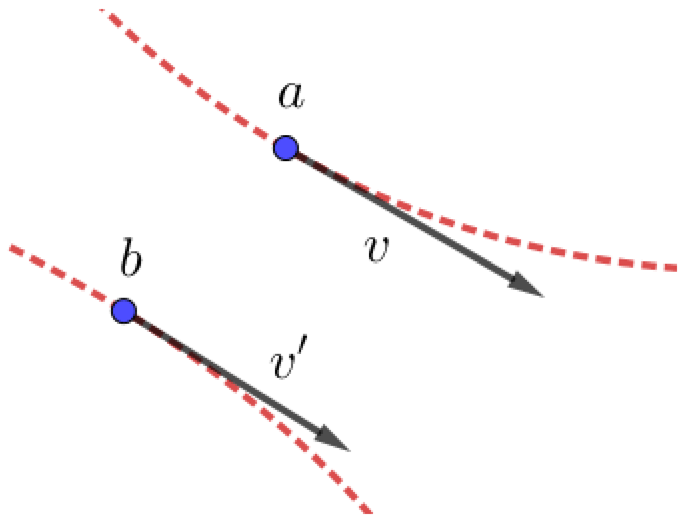


FIGURE 3. The intuition behind Ricci curvature. The geodesics in the directions of v and v' are shown as dotted red lines. The image above shows a surface with a negative curvature near a and b.

Our next goal is to convey the intuition behind Ricci flow. We can relate our story via the heat equation. Recall that in one dimension, the heat equation is given by

$$\partial_t u = \Delta u$$

where $\Delta = \partial_x^2$. This equation models the dissipative behavior of heat: high temperatures tend to drop and low temperatures absorb the heat and rise. This smooths out and flattens the temperature profile [12]. Ricci flow satisfies a similar equation with the metric in place of heat. Intuitively, Ricci flow smooths the metric, but can lead to singularities that can be removed. This procedure is known as surgery. Ricci flow and surgery were used in an astonishing manner in the landmark work of Perleman [14], which resolved a long-standing conjecture on the classification of 3-manifolds. Our algorithm will make use of discrete analogues of the geometric ideas from this section and even an analogous surgery procedure.

3.2. **Ollivier-Ricci Curve and Flow.** In this section, we introduce the Ollivier Ricci flow algorithm that is used in [10] for community detection on various network. We start with the definition of Ollivier's Ricci curvature[11]. Let $G = (V, E, w)$ represent an undirected connected graph with vertex set V and edge set E without multiple edges or self loops, the $w : e \in E \to \mathbb{R}^+$ is the weight function that gives a positive value to each edge of G. Let $N(x)$ represent the set of all neighbors of x, P_{xy} represent a sequence of edges which joins a sequence of distinct vertices with starting vertex x and ending vertex y. The length of a path is the sum of each edge weight on the path. Let $d(x, y)$ represent the distance between x, y, that is, is the weighted length of the shortest path P_{xy}. For each node $x \in V$, we define a mass distribution $\mu_x(v)$ on graph $G = (V, E, w)$ that assigns a certain amount of mass at vertex x and its neighbors. Usually the sum of amount of the mass is scaling to be one. Given a path P_{xy}, we want to move the mass distribution μ_x to μ_y and calculate the cost. Moving mass distribution is a transportation plan which is an ordered function $A : V \times V \to [0, 1]$ such that $A(u, v)$ is the amount of mass at vertex u to be moved to vertex v. Function A is called the coupling function and satisfies the following constrains:

$$\sum_{v \in V} A(u, v) = \mu_x(u) \text{ and} \sum_{u \in V} A(u, v) = \mu_y(v).$$

The cost of any transportation plan is the sum of costs for each mass-movement where the cost of a single movement is calculated by $A(u, v) \times d(u, v)$. The transportation distance between two probability distributions is then the smallest possible cost which is called the Wasserstein distance $W(\mu_x, \mu_y)$, defined as follows:

$$W(\mu_x, \mu_y) = \inf_A \sum_{u,v \in V} A(u, v) d(u, v),$$

where the infimum is taken over all coupling A between μ_x and μ_y.

By the Kantorovich-Rubinstein Duality, there is another way to calculate the Wasserstein distance. Given a locally finite graph $G = (V, E)$, a function $f : V \to \mathbb{R}$ is called 1-Lipschits if $f(x) - f(y) \leq d(x, y)$, for each $x, y \in V$. Then the transportation distance can also be written as follows:

$$W(\mu_1, \mu_2) = \sup_f \sum_{x \in V} f(x)[\mu_1(x) - \mu_2(x)],$$

where the supremum is taken over all 1-Lipschits functions f.

The Ricci curvature on any pair of vertices $\{x, y\}$ is then defined as

$$\kappa(x, y) = 1 - \frac{W(\mu_x, \mu_y)}{d(x, y)}.$$

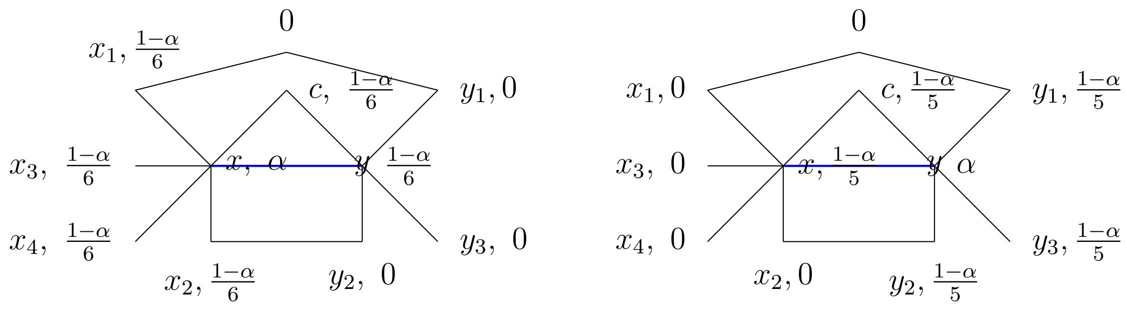
Definition 3.1 (Ollivier's α-Ricci Curvature). [11] *Let $G = (V, E)$ be a simple graph, for any $x, y \in V$ and $\alpha \in [0, 1]$, the α-Ricci curvature κ_α is defined to be*

$$\kappa_\alpha(x, y) = 1 - \frac{W(\mu_x^\alpha, \mu_y^\alpha)}{d(x, y)},$$

where the probability distribution μ_x^α is defined as:

$$\mu_x^\alpha(z) = \begin{cases} \alpha, & \text{if } z = x, \\ \dfrac{1-\alpha}{d_x}, & \text{if } z \sim x, \\ 0, & \text{otherwise.} \end{cases}$$

Example 1. *Given a graph G as shown in Figure 4 where the weight on each edge is set to one. We will consider the transport problem from μ_x^α where masses distributed at x for α unit and at its neighbors for $\frac{1-\alpha}{6}$ unit each to μ_y^α where masses distributed at y for α unit and at its neighbors for $\frac{1-\alpha}{5}$ unit each. In this example, $\alpha \geq \frac{1}{7}$.*



0
$x_1, \frac{1-\alpha}{6}$
$c, \frac{1-\alpha}{6}$
$y_1, 0$
$x_3, \frac{1-\alpha}{6}$
x, α
y $\frac{1-\alpha}{6}$
$x_4, \frac{1-\alpha}{6}$
$y_3, 0$
$x_2, \frac{1-\alpha}{6}$
$y_2, 0$
0
$x_1, 0$
$c, \frac{1-\alpha}{5}$
$y_1, \frac{1-\alpha}{5}$
$x_3, 0$
$x, \frac{1-\alpha}{5}$
y α
$x_4, 0$
$y_3, \frac{1-\alpha}{5}$
$x_2, 0$
$y_2, \frac{1-\alpha}{5}$

FIGURE 4. μ_x^α and μ_y^α

Since $\alpha \geq \frac{1}{7}$, we can define the coupling function:

$$A(u,v) = \begin{cases} \frac{1-\alpha}{6} & (u,v) \in \{(x_1,y_1),(x_2,y_2),(x_3,x),(x_4,x)\} \\ \frac{1-\alpha}{5} - \frac{1-\alpha}{6} & (u,v) \in \{(x,c),(x,y_1),(x,y_2)\} \\ \alpha - \frac{1-\alpha}{6} & (u,v) = (x,y) \\ \frac{1-\alpha}{5} & (u,v) = (x,y_3) \\ 0 & others. \end{cases}$$

Then

$$\begin{aligned} W(\mu_x^\alpha, \mu_y^\alpha) &\leq \frac{1-\alpha}{6} \cdot (2+1+1+1) + (\frac{1-\alpha}{5} - \frac{1-\alpha}{6}) \cdot (1+2+2) \\ &\quad + (\alpha - \frac{1-\alpha}{6}) \cdot (1) + \frac{1-\alpha}{5} \cdot (2) \\ &= \alpha + \frac{7(1-\alpha)}{5} - \frac{1-\alpha}{6}. \end{aligned}$$

In order to prove that this upper bound is in fact an equality, we construct a function f with values: $f(u) = -1$ for $u \in \{x_1, x_2, y\}$; $f(u) = 1$ for $u \in \{x_3, x_4\}$; $f(u) = -3$ for $u = y_1$ $f(u) = -2$ for $u \in \{y_2, y_3\}$; $f(u) = 0,$ for others. One can easily check f is 1-Lipschitz function. Then

$$\begin{aligned}W(\mu_x^\alpha, \mu_y^\alpha) &\geq \frac{1-\alpha}{6} \cdot (f(x_1) + f(x_2) + f(x_3) + f(x_4)) + (0 - \frac{1-\alpha}{5}) \cdot (f(y_1) + f(y_2) + f(y_3)) \\ &\quad + (\frac{1-\alpha}{6} - \frac{1-\alpha}{5}) \cdot f(c) + (\frac{1-\alpha}{6} - \alpha) \cdot f(y) \\ &\geq \frac{1-\alpha}{6} \cdot (-1-1+1+1) + (0 - \frac{1-\alpha}{5}) \cdot (-3-2-2) \\ &\quad + (\frac{1-\alpha}{6} - \frac{1-\alpha}{5}) \cdot 0 + (\frac{1-\alpha}{6} - \alpha) \cdot (-1) \\ &= \alpha + \frac{7(1-\alpha)}{5} - \frac{1-\alpha}{6}.\end{aligned}$$

Together with the upper bound , we can then conclude that

$$\kappa_\alpha(x, y) = 1 - \frac{7}{5} + \frac{1}{6} \approx -0.2333.$$

The value of α-Ricci Curvature depends the value of α. Y. Lin, L. Lu, and S.-T. Yau modified Ollivier's definition of Ricci curvature to be the negative derivative when α approaches 1 in Ollivier's definition, thus eliminating the idleness parameter [8].

Definition 3.2. [8] *Let $G = (V, E)$ be a simple graph, for any $x, y \in V$, the Lin-Lu-Yau Ricci curvature $\kappa(x, y)$ is defined as*

$$\kappa(x, y) = \lim_{\alpha \to 1} \frac{\kappa_\alpha(x, y)}{1-\alpha},$$

where $\kappa_\alpha(x, y)$ is the α-Ricci curvature defined in above definition.

By Ollivier's Ricci curvature, if $W(\mu_x, \mu_y)$ is less than $d(x, y)$, the average distance between neighbors of x and neighbors of y is less than the distance between x, y which implies that the two neighborhood sets are closer to each other. Thus the positive value of k_{xy} means the two neighborhoods are closer and the larger of value k_{xy}, the closer between $N(x)$ and $N(y)$. Alternatively, the positive value of k_{xy} means the two neighborhoods are closer relatively and the smaller of value k_{xy}, the farther between $N(x)$ and $N(y)$.

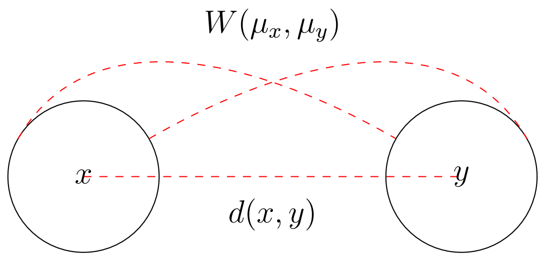


FIGURE 5. Distance from one ball to another compared with the distance between their centers.

Applying this natural idea to the clustering problem, we can imagine that the edge curvatures are mostly positive valued in one cluster and are mostly negative valued between different clusters, by appropriately removing the negatively valued edges (or in other words, edges with large weight), we can separate the nodes and therefore obtain a partition of the network. This idea is applied in [18]. The basic process stated in this paper is to remove the most negative curvatured edge then re-calculate the edge curvature only for those affected nodes/edges due to prior edge removals, then repeat the process until a good partition of nodes. In another almost simultaneously published paper [10], the author used the approach called discrete Ricci-flow to detect the communities in various network including Zachary's

Karate Club graph, Network of American football games, Facebook Ego Network, etc. The experimental results demonstrate the effectiveness of this approach.

Given a weighted graph and the metric w_{xy}, we can calculate the Ricci curvature k_{xy}. On the other hand, if we consider the metric w_{xy} to be functions of a variable which is usually called "time", then the Ricci flow is defined by the following geometric evolution equation:

$$\frac{\partial}{\partial t} w_{xy} = -w_{xy} k_{xy}.$$

A solution to the Ricci fow is a one-parameter family of metrics $w_{xy}(t)$ on a graph satisfying the above partial differential equation. Ricci flow is an iterative process that aims to smooth out the curvatures of the input graph by adjusting the edges weight, it stretches edges of large negative Ricci curvature and shrinks edges of large positive Ricci curvature over time. After i-th Ricci flow iterations, the edge weights are simultaneously updated by the following flow process:

$$w_{xy}^{(i+1)} = d^i(x, y) - \epsilon \cdot k_{xy}^i \cdot d^i(x, y),$$

where ϵ is the step size or learning rate of the gradient decent process.

After "enough" Ricci flow iterations Ricci curvature of the graph will be converged to some value. To detect the clusters of the graph, all edges with weight greater than a threshold are removed.

In [10], they used a more general family of probability distribution $m_x^{\alpha,p}$ that depending on two parameters $\alpha \in [0,1]$ and power $p \geq 0$:

$$m_x^{\alpha,p}(v) = \begin{cases} \alpha & \text{if } v = x \\ \frac{1-\alpha}{\sum\limits_{v\sim x} exp(-d(x,v)^p)} exp(-d(x,v)^p) & \text{if } v \sim x \\ 0 & \text{otherwise.} \end{cases}$$

Such probability distribution assigns amount α of mass to the center vertex x and the rest of $1-\alpha$ to all its neighbors proportionally with respect to a specified function of the edge distance.

[10] also proved rigorously that the Ollivier Ricci flow can successfully detect community structure for the $G(a,b)$ family of graphs, where graph $G(s,t)$ is obtained by taking the complete graph on $t+1$ vertices $p_1,\ldots,p_{t+1}$ and $t+1$ complete graphs $C_1,\ldots,C_{t+1}$ on $s+1$ vertices and then identify a vertex u_i from each C_i and identify u_i with p_i. As we can see the effectiveness of this geometric approach, we will apply this method in our project to cluster the students' locations.

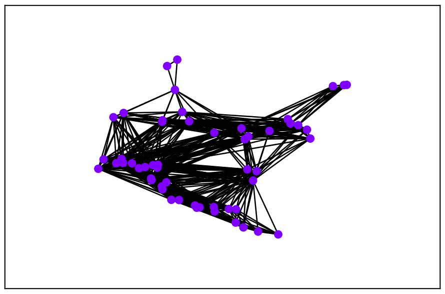
4. Algorithm

We use 400 meters as our maximum distance between a child and his/her bus stop. At a reasonable pace, this distance can be traversed in under 5 minutes. However, one can adjust this constraint without significantly altering the algorithm. We assume the raw data is either the students' addresses or their longitude/latitude. The first step is for us to convert this input into a weighted graph.

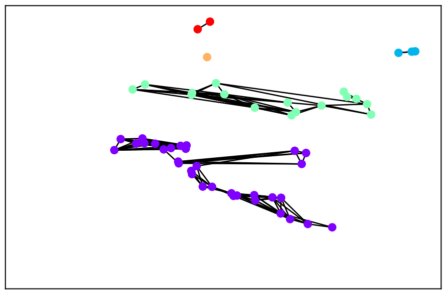
The goal is to use the notion of Ollivier Ricci flow to partition the students and then choose a single stop for each partition. There are two competing objectives. One, we must satisfy the constraint that no student walks too far. Two, we would also like to keep the number of stops to a minimum. Although this does not translate necessarily into a shorter bus route, in practice and simulation, this reduces the distance the bus travels. Therefore, the idea is to avoid further partitioning sets that already meet the diameter restriction. Note, our notion of diameter is induced by the edge weights, not the geometric (Euclidean) distance. Explicitly, as we partition the graph, any connected component of radius less than 400 meters is frozen.

Our final algorithm proceeds in the following stages. If after removing the large weight edges, our graph is disconnected, we apply the algorithm to each connected component. If the connected component is of radius less than 400, we simply gather all its vertices into a single cluster and do no proceed any further in the clustering phase on this set. If the connected component is large, we compute the Ollivier-Ricci curvature and then allow the Ollivier Ricci flow to iterate several times. Our program for this stage utilizes code written by Chien-Chun Ni that he has made available online. This alters the edge weights and we remove all edges above a certain threshold. A demonstration can be seen in Figure 6.

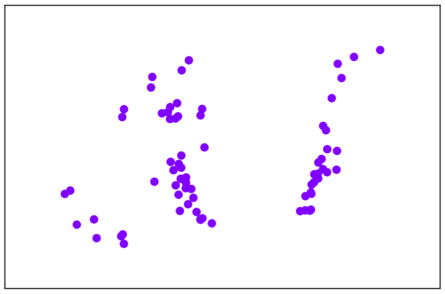
For each connected component of this new graph, we check whether the radius is sufficiently small. If not, we apply our algorithm recursively to this subset. Finally, having



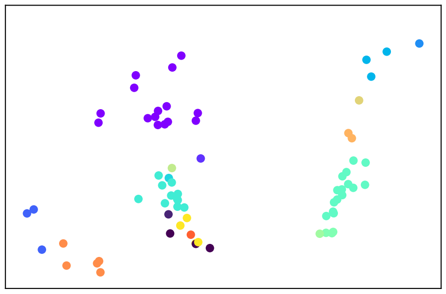
(A) Above is a connected component of our distance graph.



(B) The above is the result of running our Ricci flow algorithm for several iterations and then removing all edges above a certain threshold. Each color corresponds to a connected component.



(C) Another example of a connected component without the edges drawn.



(D) After one iteration of the partitioning scheme (again without edges drawn). Since some clusters have radii larger than 400m, we again partition those connected components.

Figure 6. These figures depict various stages of our partitioning algorithm.

partitioned our set of nodes into sufficiently small clusters, we choose a central node to be our bus stop. We define a center to be any node whose distance to all the other nodes in the cluster is less than the radius. Such a stop can easily be found in polynomial time. For a cluster of size n, we can apply the Floyd–Warshall algorithm which runs in time $O(n^3)$.

So far, we have been vague about several key parameters: the cutoff thresholds for the large edge weights after each Ricci flow application and the number of iterations we allow Ricci flow to perform before removing edges. These parameters are data dependent necessarily as they are certainly not independent of the scaling. For example, in a rural community where the separation of the children may be much larger, we expect the algorithm's output to be different since even though the distances are longer, our 400 meter constraint does not change. We provide several examples of effective parameter choices from our data sets and provide heuristics for choosing such parameters.

Once our partitioning phase terminates, this implies all the clusters have radius less than 400 meters. There exist simple polynomial time algorithms to check the radius of graphs and to locate the (a) center. We designate the center of each cluster to be our bus stop.

From here, we generate an auxiliary graph that is simply the complete weighted graph with nodes corresponding to bus stops (and a single node for the school) and edge weights corresponding to shortest path by road. Now, finding an optimal route starting at the school, passing through each bus stop and returning to the school is exactly the classical Traveling Salesman Problem. For this, we have available many commercial packages that generate approximations to the optimal path. Recall that optimizing the path exactly is an NP-hard problem. We chosen to use Algorithm 2 to approximate the TSP solution. This is a classical algorithm [20] and we include the short proof that this algorithm approximates the real solution to within a multiplicative factor of 2.

Lemma 4.1. *Algorithm 2 is a 2-approximation of the TSP solution.*

Proof. Note that the weight of a minimum spanning tree is a lower bound on the optimal circuit, since removing a single edge from the tour creates a spanning tree. For an upper-bound, note that the cost of the Eulerian tour is an upper bound and this cost is twice the weight of the minimum spanning tree since each edge is included twice. Finally, by the triangle inequality, skipping a vertex cannot increase the cost. □

Algorithm 1 Ricci Flow Algorithm

```
1: procedure Ricci Flow
        Input: An undirected graph $G$ and a real number $\delta > 0$.
        Output: A weighted graph with updated Ricci flow metric on each edge.
2: Normalize the edge weights such that $w_e^{(i)} \leftarrow d^i(e) \cdot \frac{|E|}{\sum_{e \in E} d^i(e)}$.
3: Computer the Ricci curvature of $G$.
4: Update the edge weight by $w_e^{(i+1)} \leftarrow d(e)^{(i)} - \epsilon \cdot \kappa_e^{(i)} \cdot d(e)^{(i)}$.
5: Repeat (1)-(3) $|\kappa_e^{(i)} - \kappa_e^{(i-1)}| < \delta$ for every $e \in E(G)$.
```

Our algorithm is summarized in pseudo-code in Algorithms 1, 2 and 3. Below are several outputs of our algorithm on some data from local Boston high schools (See Fig. 7 and Fig. 8).

4.1. **Choosing Parameters.** As the rigorous behavior of our algorithm is not yet understood, we can only apply heuristic justifications for our choice of parameters. The first

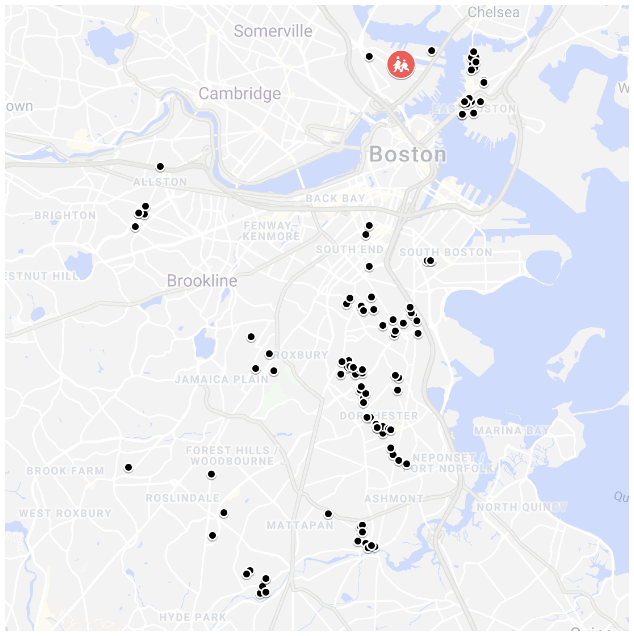
Algorithm 2 Approximate Traveling Salesman Problem

```
1: procedure APPROXIMATE TSP
       Input: A weighted subgraph graph $G[c_1, \dots, c_k]$ induced by centers $\{c_1, \dots, c_k\}$ in each cluster.
       Output: The cycle of minimum cost visiting all of the vertices of $G[c_1, \dots, c_k]$ exactly once.
2: Find a minimum spanning tree of $G$
3: Duplicate edges of the spanning tree $T$
4: Construct Eulerian tour, $U$, of the spanning tree $T$
5: Change $U$ into a tour by following the order in $U$ and skipping vertices that have already been traversed
6:
```

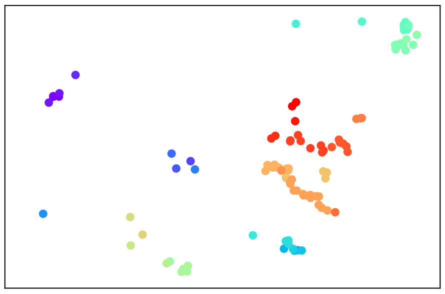
Algorithm 3 Bus Stop Algorithm

```
1: procedure PARTITIONING
2: Run Ricci Flow Algorithm on weighted graph $G$
3: For each connected component $G_i$:
4:     if diam($G_i$) $> \alpha$ then
5:         Run Ricci Flow Algorithm on $G_i$
6: Output partitions of $G$
7: For each cluster of nodes $N_i$ :
8:     Designate the center to be a bus stop
9: Finally, run Approximate TSP on the bus stops (including the school)
```

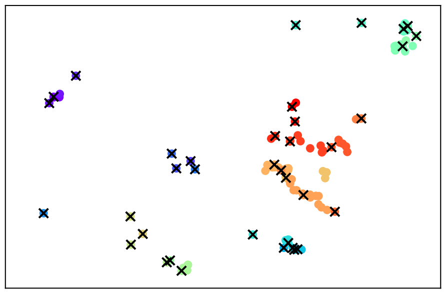
parameter that we are free to choose is the surgery threshold for the edge weights. Generally, we output the weights after the Ricci flow algorithm and search for a natural division in the edge weights (See Fig. 9). For our Billy Goodman high school data set, we found that a reasonable division was 1400 while for the Jason Veritek Highschool, the cutoff we chose was 2000. It would be an interesting direction to automate this choice of cutoff.



(A) The geographic locations of Jason Veritek highschool students

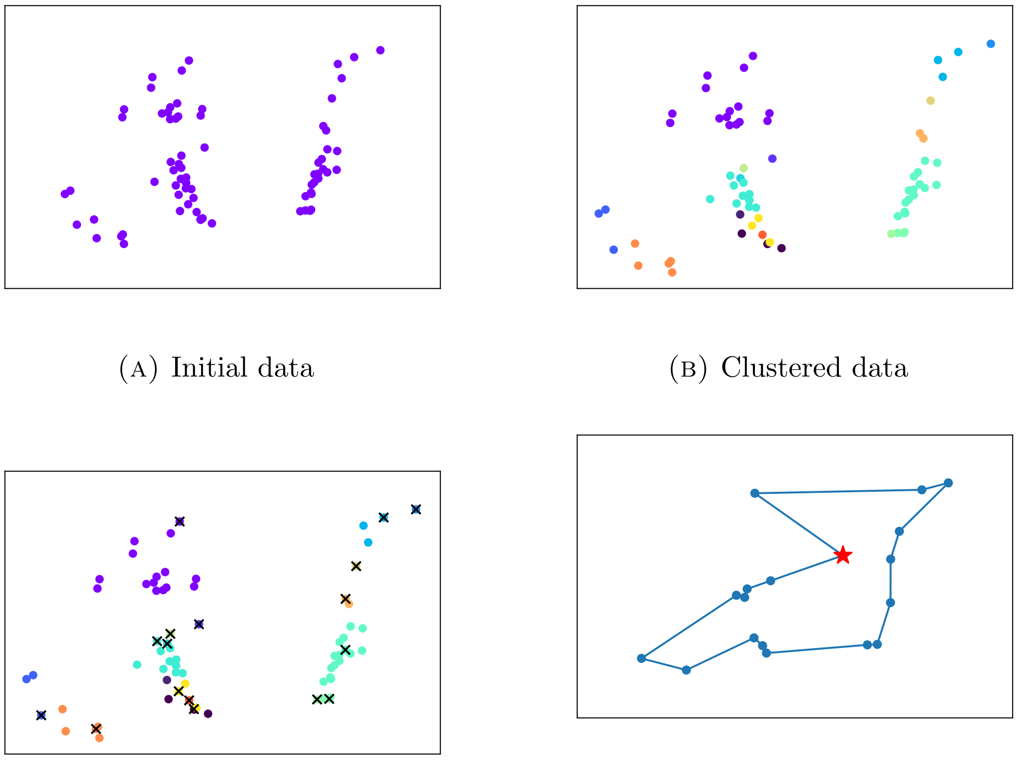


(B) The clustered data from James Veritek Highschool



(C) Our algorithm has assigned bus stops to centers of clusters

FIGURE 7. We show the first two steps of our algorithm for the Jason Veritek high school.



(A) Initial data

(B) Clustered data

(C) Bus stops assigned

(D) Approximate TSP route through stops and school (red star)

FIGURE 8. Data from Billy Goodman High school.

The last parameter that we fix is the number of iterations in the Ricci flow algorithm. Ultimately, we expect after enough iterations that all the edge weights will converge. Therefore, the goal was to choose the number of iterations to be enough that a separation of clusters appears but not so much that all the edge weights are identical. On our data sets,

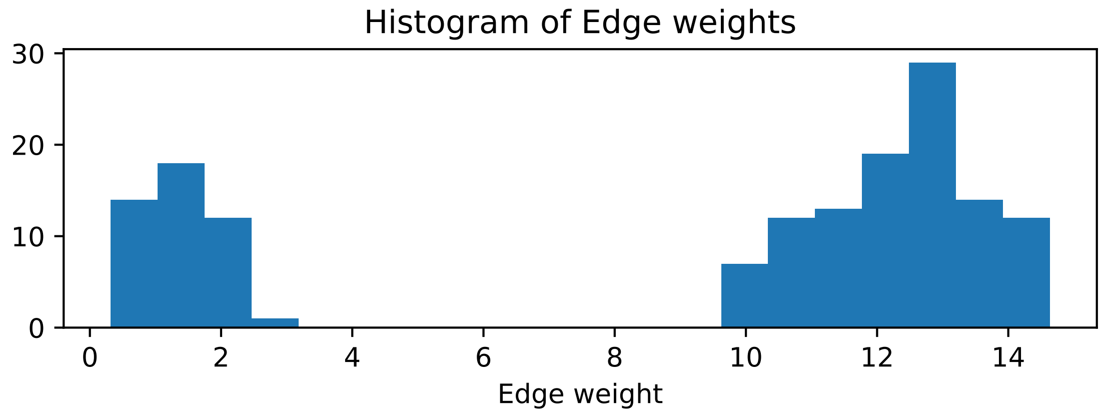


FIGURE 9. Histogram of edgeweights after several iterations of Ricci flow. The initial data was that in Fig. 10.

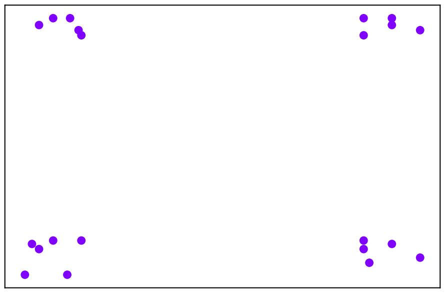
we found that 20 iterations served our purposes. We imagine a more rigorous understanding of the algorithm will provide a way to choose this number in an unsupervised fashion.

4.2. **Evaluating Our Algorithm.** As with many other Vehicle Routing (VR) problems, our single-school formulation of the SBRP is NP-hard in general. Therefore, an effective gauge of the functionality of our algorithm can only be done through comparison with other algorithms or on data sets in which the optimal route is known. With the goal of comparison, we have contacted several research groups in an effort to obtain their data and algorithm implementations (due to imprecise descriptions, we have been unable to implement the leading algorithms ourselves). We are currently awaiting responses. In lieu of a direct comparison with other algorithms, we consider a simple generated data set in which the optimal route is obvious by inspection. Our algorithm is able to recover the optimal route exactly (See Fig. 10).

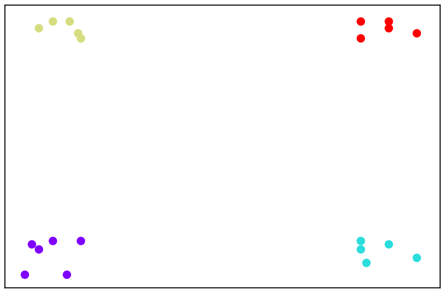
Of course, for such a transparent data set, there are simpler algorithms, but this gives us confidence that are heuristics are on track.

4.3. **Running-time Analysis.** Since the inception of complexity theory, algorithms running in polynomial time have been the gold standard of computer science. As mentioned before, the SBRP is an NP-hard problem and therefore, providing a polynomial time algorithm solving SBRP exactly would be equivalent to proving $P = NP$, a notoriously difficult open problem in mathematics and theoretical computer science. Our algorithm is the composition of several polynomial time algorithms and is therefore, polynomial time. We have not performed an in-depth analysis of the dependence on parameters as this would require provable bounds on the Ricci-flow clustering which are currently out of reach.

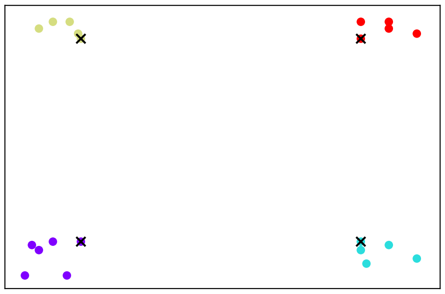
For a data set with 70 nodes, our algorithm runs in 62 seconds on a desktop running with an Intel Core i7. For about 100 nodes, it takes 280 seconds. These are reasonable times and the number of nodes are substantially larger than a real single school bus routing problem.



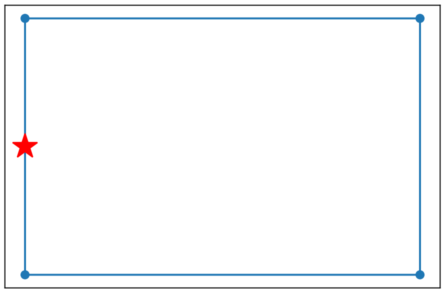
(A) Our initial data



(B) Partitioned data



(C) Bus stop assigned to each partition



(D) Optimal route through bus stops and school (red star)

FIGURE 10. A contrived example showing our algorithm can output the optimal path for simple clusters.

5. Future Work and Concluding Thougts

At this stage, our algorithm is a proof of concept. Further analysis into real-world busing is required. Due to privacy reasons, we were not granted access to the actual bus routes of the Boston public schools. A comparison of our generated routes with that of the actual routes would provide the empirical evidence to justify a policy based around our Ricci-flow algorithm.

Our technique of utilizing Ricci flow to designate students to stops can likely generalize to multiple schools and schedules. We believe one interesting generalization is to define Ricci curvature and Ricci flow for directed graphs. This would be a fruitful mathematical pursuit and furthermore, when applied to the SBRP problem, it captures the situation in which there are one-way roads or heavy traffic in one direction so that the "distance" between two points may not be symmetric.

From a theoretical computer science standpoint, it would be valuable to provide rigorous justification of the performance of this algorithm. This would require a thorough understanding of the interaction between the Ricci flow and the surgery procedure beyond our heuristic justification. This is likely a difficult mathematical problem.

Acknowledgement

I would like to thank Chien-Chun Ni for making his Ricci Curvature code available online. I also thank my mentors Shuliang Bai and Kyle Luh for their suggestions and guidance.

References

[1] Robert Bowerman, Brent Hall, and Paul Calamai. A multi-objective optimization approach to urban school bus routing: Formulation and solution method. *Transportation Research Part A: Policy and Practice*, 29:107–123, 03 1995.

[2] Robert L. Bowerman, Brent Hall, and Paul H. Calamai. A multi-objective optimization approach to urban school bus routing: Formulation and solution method. 1995.

[3] Manfredo Perdigao do Carmo. *Riemannian geometry*. Birkhäuser, 1992.

[4] Manfredo P Do Carmo. *Differential Geometry of Curves and Surfaces: Revised and Updated Second Edition*. Courier Dover Publications, 2016.

[5] Marcelo Fonseca Faraj, João Sarubbi, Cristiano Silva, Marcelo Porto, and Nilson Nunes. A real geographical application for the school bus routing problem. 10 2014.

[6] Taehyeong Kim and Bum jin Park. Model and algorithm for solving school bus problem. 2013.

[7] John M Lee. *Riemannian manifolds: an introduction to curvature*, volume 176. Springer Science & Business Media, 2006.

[8] Yong Lin, Linyuan Lu, and Shing-Tung Yau. Ricci curvature of graphs. *Tohoku Mathematical Journal - TOHOKU MATH J*, 63, 12 2011.

[9] Rita M Newton and Warren H Thomas. Design of school bus routes by computer. *Socio-Economic Planning Sciences*, 3(1):75–85, 1969.

[10] Chien-Chun Ni, Yu-Yao Lin, Feng Luo, and Jie Gao. Community detection on networks with ricci flow. *Scientific Reports*, 9, 07 2019.

[11] Yann Ollivier. Ricci curvature of markov chains on metric spaces. *Journal of Functional Analysis*, 256:810–864, 02 2009.

[12] Peter J Olver. *Introduction to partial differential equations*. Springer, 2014.

[13] Junhyuk Park and Byung-In Kim. The school bus routing problem: A review. *European Journal of operational research*, 202(2):311–319, 2010.

[14] Grisha Perelman. The entropy formula for the ricci flow and its geometric applications. *arXiv preprint math/0211159*, 2002.

[15] Je rey Braca, Julien Bramel, Bruce Posner, and David Simchi-Levi. A computerized approach to the new york city school bus routing problem. 1994.

[16] David Ripplinger. Rural school vehicle routing problem. *Transportation Research Record: Journal of the Transportation Research Board*, 1922:105–110, 01 2005.

[17] João Sarubbi, Caio M. R. Mesquita, Vinicius dos Santos, and Cristiano Silva. A strategy for clustering students optimizing the number of bus stops for solving the school bus routing problem. 04 2016.

[18] Jayson Sia, Edmond Jonckheere, and Paul Bogdan. Ollivier-ricci curvature-based method to community detection in complex networks. *Scientific Reports*, 9, 12 2019.

[19] Sam Thangiah and Kendall Nygard. School bus routing using genetic algorithms. *Proc SPIE*, 03 1992.

[20] Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[21] Max-K von Renesse and Karl-Theodor Sturm. Transport inequalities, gradient estimates, entropy and ricci curvature. *Communications on pure and applied mathematics*, 58(7):923–940, 2005.