Constructing the Shortest Hamiltonian Path via Bijection Ranges

Mark Wesley Harris

December 13, 2018

1 Introduction

This document breifly explains the theoretical concepts behind the Shortest Path Algorithm developed in the GitHub pixarninja/shortest_path_algorithm repository. This algorithm attempts to solve the Traveling Salesman Problem, with the input of a 2D graph of datapoints represented on a Cartesian coordinate system.

2 Definitions

- (x, y): the definition of a point with coordinates (x, y).
- $< p, q >$: the definition of a vector from point p to point q.
- $[s_1, s_2, ..., s_k]$: the definition for a path through a known set of edges.
- n: the number of points in the dataset.
- $\mathcal{V}$: the set of all vertices in the dataset. Formally, $\mathcal{V} = \{p_1, p_2, ..., p_n\}$
- $\mathcal{E}$: the set of all edges in the dataset. The size of $\mathcal{E}$ is $n \times n$.
- $\mathcal{H}$: the convex hull of the dataset. Formally, $\mathcal{H} = \{h_1, h_2, ..., h_k\}$, where $k \leq n - 1$. Given any set of points, there is only one convex hull.
- $\mathcal{S}$: the shortest path of a dataset as the set $\mathcal{S} = \{s_1, s_2, ..., s_n\}$ of edges which are contained in the shortest path.
- $\mathcal{E}'$: the set of all edges in the dataset that are in $\mathcal{E}$ but not $\mathcal{S}$. Formally, $\mathcal{E}' = \mathcal{E} - \mathcal{S}$.
- $\mathcal{W}$: the set of edges produced by the construction phase of the algorithm. Formally, $\mathcal{W} = \{w_1, w_2, ..., w_m\}$, where $3 < m < n^2$.
- $\mathcal{S}' \equiv \mathcal{S}$: the path $\mathcal{S}'$ has the same perimeter as path $\mathcal{S}$ (they may or may not have the same edges).

3 Theoretical Concepts

My algorithm is based upon many theoretical concepts I have discovered through research and experimentation. They are explained below Sections 3.1 and 3.2.

3.1 Properties of Shortest Paths

I begin by describing the properties I have found to be true for all shortest paths. These properties have been used to create the foundations for my algorithm.

1. For any two edges $s_i, s_j \in \mathcal{S}$ where $i \neq j$, s_i and s_j do not cross.
2. For any two edges $s_i, s_j \in \mathcal{S}$ where $i \neq j$, s_i and s_j do not overlap.
3. No edge $s \in \mathcal{S}$ intersects a point $p \in \mathcal{V}$ where $s_1, s_2 \neq p$. I.e. if an edge e intersects a point p that isn't an end point of e, then $e \notin \mathcal{S}$.
4. If we were to add an arbitrary point p_{n+1} to $\mathcal{V}$, the form of $\mathcal{S}$ will not change iff p_{n+1} is a point on the line of any $s \in \mathcal{S}$. Thus $\exists s \in \mathcal{S}$ such that p_{n+1} lies on s.

3.2 Invalidating a Segment

My algorithm is based upon the concept of invalidating as many segments of $\mathcal{E}$ as possible, but keeping all segments which are probably in $\mathcal{S}$. In order to do this, I propose a method of edge construction that begins with a complete graph and invalidates edges based upon the properties for shortest paths which I have produced above. Any segment which fails these properties is not likely in $\mathcal{S}$, and so it should not be included in $\mathcal{W}$.

4 Bijection Range Method

The properties discussed in Section 3.1 are not easily enforced on a given dataset. I have interpretted them as illustrated by the following guidelines:

1. We are given a generic segment $e \in \mathcal{E}$ with endpoints $e_1, e_2 \in \mathcal{V}$.
2. Take any two points $p, q \in \mathcal{V}$ such that $p \neq e_1, e_2$ and $q \neq e_1, e_2$.
3. Take all points x on the continuous line e where $x \notin \mathcal{V}$.
4. Test if the distance of $[p, x]$ plus the distance of $[q, x]$ is less than the distances of $[e_1, x]$ plus the distances of $[p, x]$, $[q, x]$, or $[x, e_2]$. In other words, test if the path from p to q through x is shorter than any path from a point in the test set $\mathcal{T} = \{e_1, e_2, p, q\}$ to any other point in $\mathcal{T}$ through x.
5. If true, then $e \notin \mathcal{W}$. Otherwise it is shown that e statisfies the properties in Section 3.1, so $e \in \mathcal{W}$.

My goal is to enforce the rule that if a new point p_{n+1} was added to $\mathcal{V}$ and it were to lie on any edge $w \in \mathcal{W}$, the form of $\mathcal{S}$ would not change.

It is obvious that checking all possible points for all possible segments presents a challenge in time complexity. Even assuming that checking all points x on segment $e \in \mathcal{E}$ takes a polynomial amount of time — let's say this value is some polynomial $f(n) = a_n p^n + a_{n-1} p^{n-1} \cdots + a_1 p + a_0$ — it would be multiplied by checking each edge against each other edge, which without any simplification is $O(n^4)$. Thus even if the degree of function f is relatively low, running the solution on large datasets is infeasible in practice.

The following subsections discuss possible solutions to this complexity problem. What is implmented in the algorithm is the Bijection Range Abstraction Method, defined in Section 4.2.

4.1 Intersection Method

The Bijection Range Method works for determining ranges on a segment e for which points $p, q \in \mathcal{V}$ are closer to e than the endpoints e_1 or e_2. However, since we are only interested in invalidating the line with respect to this property, it is worth looking at the possibility of using the intersection of any $e' \in \mathcal{E}$ with e instead.

Theorem 4.1. *Given a line segment $e \in \mathcal{E}$ with endpoints $e_1, e_2 \in \mathcal{V}$, for each $p, q \in \mathcal{V}$ where $p \neq e_1, e_2$ and $q \neq e_1, e_2$, if p and q have overlapping Bijection Ranges and form a convex quadralateral with their lower and upper bound intersection points, then the segment e' passing through p and q intersects e.*

Proof. The proof is shown below:

(1) Given that p and q have overlapping Bijection Ranges, it is evident that there are 2 intersection points formed by the lower and upper bounds of the intersection of ranges. Let these points be l and u, respectively.

(2) By definition, these points form a quadralateral. As given, the qudralateral is convex. Thus the source point of a Bijection Range will never extend passed a lower or upper bound. Let the quadralateral p, l, q, u be denoted as Q.

(3) Since Q is a convex quadralateral, a line segment can be drawn between p and q that lies within the inner region formed by Q. Let that line be denoted as e'.

(4) Since l and u both intersect e, and e' is a continuous line that lies between l and u, then e' must also intersect e.

□

The Intersection Method is a valid approach to determine if two Bijection Ranges intersect, however the requirement that the two ranges form a convex quadralateral is not a trivial calculation. Therefore another method was developed to determine if two Bijection Ranges overlap. This method is called Bijection Range Abstraction, and was used in the final implementation of the algorithm.

4.2 Bijection Range Abstraction

Calculating both bijections of each possible point $v \in \mathcal{V}$ for each edge $e \in \mathcal{E}$ is a very costly method of validating a segment. Instead of having to do the calculations each time, it would be nice if there were a mathematical area to represent when a point's bijection ranges will overlap for a given segment.

Theorem 4.2. *The area in which there is a bijection range created from point $p \in \mathcal{V}$ using segment $w \in \mathcal{E}$ can be represented as a semicircle, M, where w is the segment opposite its arc.*

Proof. The proof is shown below:

(1) Let w be the edge we are testing, with points $w_1, w_2 \in \mathcal{V}$.

(2) Take a point $p \in \mathcal{V}$ where $p \neq w_1, w_2$ and p does not lie on w.

(3) Let M be the unique semicircle in the direction of p formed using w as the flat segment of M.

(4) Let A, B, and C represent the points w_1, p, and w_2, respectively. Similarly, let D, E, and F represent the midpoints of segments AC, BC, and AC, respectively.

(5) Now we assume the 3 cases for the position of p relative to M:

(i) Assume that p lies on the arc of semicircle M: See Figure 1 as a reference for the

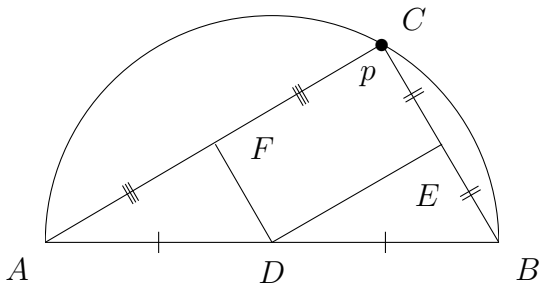

Figure 1: p lies on M

steps discussed below.

(a) Points A, B, and C satsify the requirements of Thale's Theorem: "if segment AB is a diameter, then the angle at C is a right angle." Thus $\angle ACB = 90$.

(b) Let vector V be the bijection of BC, so that V passes through E and is perpendicular to BC. Let the intersection point of V and AC be point I.

(c) Since $\angle ABC = \angle IBE$ and $\angle BCA = \angle BEI$, $\triangle ABC$ and $\triangle IBE$ are similar triangles.

(d) Given that BE is half the length of BC, then $BI = \frac{BA}{2} = BD$. Thus I is halfway between A and B, so $I \equiv D$.

(e) Now let V be the bijection of AC, so that V passes through F and is perpendicular to AC. Let the intersection point of V and AC be point I.

(f) Since $\angle BAC = \angle IAF$ and $\angle ACB = \angle AFI$, $\triangle ABC$ and $\triangle AIF$ are similar triangles.

(g) Given that AF is half the length of AC, then $AI = \frac{BA}{2} = AD$. Thus I is halfway between A and B, so $I \equiv D$.

(h) Since both bijections for p meet in the center of w, the Bijection Range is a single point.

This shows that the Bijection Range for points along the arc of M are equal to the center w.

(ii) Now assume that p lies inside the area of M but not on the arc of M: See Figure

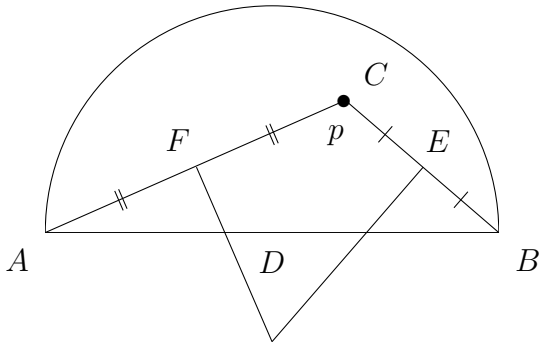

Figure 2: p lies inside M

2 as a reference for the steps discussed below.

(a) As shown above, if p is on the arc of M then $\angle ACB = 90$. Thus if p is inside the arc, $\angle ACB < 90$ or $\angle ACB > 90$.

(b) As p approaches w, $\angle ACB$ approaches 180. So $\angle ACB > 90$.

(c) Since $\angle ACB > 90$, the two bijection vectors will intersect somewhere below w, opposite to M.

(d) The area between the intersections of these vectors with w is the Bijection Range found previously through iteration.

This shows that the Bijection Range for points inside the area of M is equal to the intersection of the bijection vectors with w.

(iii) Now assume the final case, that p lies outside the area of M but not on the arc of M: See Figure 3 as a reference for the steps discussed below.

(a) If p is outside of the arc, then $\angle ACB < 90$.

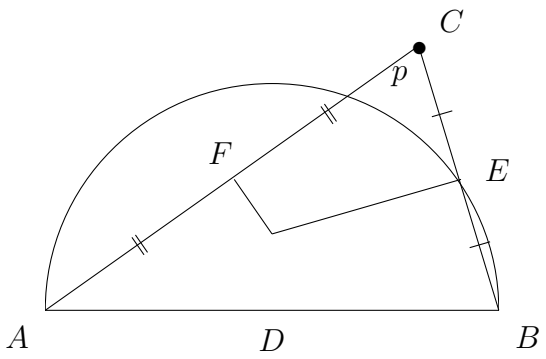

Figure 3: p lies outside M

(b) Since $\angle ACB < 90$, the two bijection vectors will intersect somewhere in the region above w, on the same side as M.

(c) The area between the intersections of these vectors with w is not the Bijection Range, since the inequalities will not contain the same ranges.

Thus the Bijection Range for points outside the area of M is undefined.

(6) This shows that the only possible way a Bijection Range is formed by point p on segment w is if p is inside or on the arc of semicircle M.

□

5 Final Segment Invalidation

I have decided to classify a segment w as a part of $\mathcal{E}'$ instead of $\mathcal{W}$ (i.e. as "invalid") in the following manner:

$$\mathcal{E}' = \{e' \mid \exists\ e \in \mathcal{E} \text{ where } e \neq e' \text{ and } e_1, e_2 \text{ have overlapping Bijection Ranges on opposite sides of } e'\}$$

This means that for each edge $e' \in \mathcal{E}'$ there are two points, $p, q \in \mathcal{V}$ which lie inside opposite semicircles (by Theorem 4.2). Since $\mathcal{W} = \mathcal{E} - \mathcal{E}'$, $\mathcal{W}$ therefore contains all segments on which there are no two other points that have overlapping Bijection Ranges on segment $w \in \mathcal{W}$. The set $\mathcal{W}$ for a random set of datapoints in $[-10, 10]$ is produced below. I have overlaid the actual shortest Hamiltonian path ($\mathcal{S}$) in transparent blue. Notice that the segment between points 2 and 8 was proven to be invalid even though it actually belongs to $\mathcal{S}$. Thus, the Bijection Range method is not full-proof, but it does provide a hueristic result for $\mathcal{S}$.

5.1 Hueristic Analysis

Using approximately 300 tests, only about 1% of paths have a segment in $\mathcal{S}$ that is not in $\mathcal{W}$. Although I have not excessively tested the Bijection Range Method on other distributions nor with datasets larger than 10 points, for small (i.e. brute-forceable) Uniform Distributions

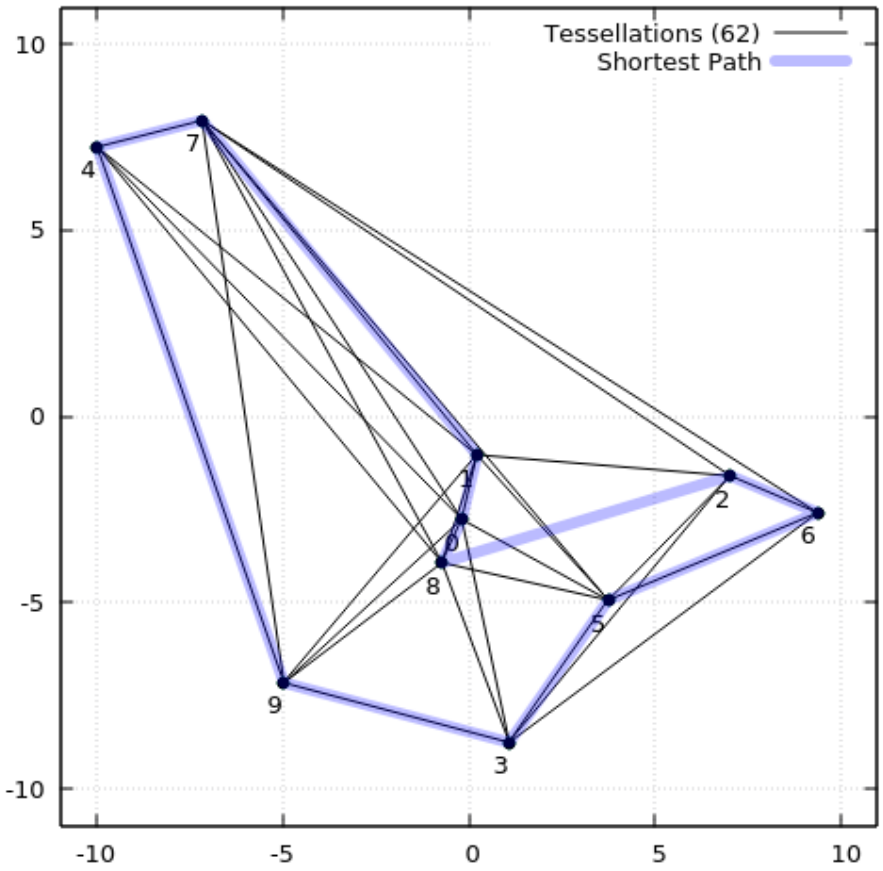

Figure 4: $\mathcal{W}$ (black) and $\mathcal{S}$ (blue)

it is clear that an edge in $\mathcal{S}$ is missing from $\mathcal{W}$ less than 1% of the time. When the distance of the shortest path $\mathcal{S}$ is compared to the distance of $\mathcal{S}'$ (the actual shortest path contained in $\mathcal{W}$), the distances vary less than 1% as well. This means the algorithm gives output that is marginally accurate regarding segment inclusion and precise regarding shortest distance.

5.2 Improving Performance

Not all segments in $\mathcal{E}$ need to be tested. A segement $e \in \mathcal{E}$ is only tested if there is no point $v \in \mathcal{V}$ that lies on e, where $v \neq e_1, e_2$. Similarly, any edge in the Convex Hull, $\mathcal{H}$, is automatically included in $\mathcal{W}$ since no two points in $\mathcal{V}$ will ever invalidate it. These two cases are used in the final implementation in order improve performance.

6 Expectations for $\mathcal{W}$

Based upon the properites of shortest paths, $\mathcal{W}$ should have the following expectations:

1. $\mathcal{H}$ will always be included, since there are no paths which could prove to invalidate

any $h \in \mathcal{H}$.

2. Any two edges $w_i, w_j \in \mathcal{W}$ where $i \neq j$ can cross each other. However any path $\mathcal{S}'$ can only contain one of these edges, otherwise it would invalidate the properties in Section 3.1.

3. If an arbitrary point x was added on any edge $w \in \mathcal{W}$, no path from two points $p, q \in V$ through x would be shorter than any other path from w_1 to p, q, or w_2 (as discussed in Section 4.2).

   Derived from the construction phase of the algorithm, $\nexists\ w \in \mathcal{W}$ such that $w \in \mathcal{E}'$ given that $\mathcal{E}'$ is defined in the following way:

   $$\mathcal{E}' = \{e' \mid \exists\ e \in \mathcal{E} \text{ where } e \neq e' \text{ and } e_1, e_2 \text{ have overlapping Bijection Ranges on opposite sides of } e'\}$$

   Thus, if it were true that $p_{n+1} \in \mathcal{V}$, the only path in $\mathcal{W}$ passing through p_{n+1} would be the path $[w_1, p_{n+1}, w_2]$. All other paths have been proven to be invalid, and thus unlikely to be part of the path $\mathcal{S}$.

4. A path $\mathcal{S}'$ can be made from $\mathcal{W}$ which goes through all vertices in $\mathcal{V}$ only once and starts and ends on the same point. If $\mathcal{W}$ has multiple paths that fit these requirements, then all such paths $\{\mathcal{S}'_1, \mathcal{S}'_2, ..., \mathcal{S}'_k\}$ should be collected. The shortest path in this set is likely to be equivalent to $\mathcal{S}$.

7 The Algorithm

The algorithm is made up of 3 consecutive phases, Edge Invalidation, Path Generation, and Decision. The Edge Invalidation Phase is responsible for invalidating edges from $\mathcal{E}$ to generate the set $\mathcal{W}$. The Path Generation Phase uses a greedy strategy to combine the resultant shapes inside of $\mathcal{W}$ into a set of possible shortest paths, $\mathcal{S}' = \{\mathcal{S}'_1, \mathcal{S}'_2, ...\mathcal{S}'_k\}$. The Decision Phase chooses the shortest of these paths and returns it as the found shortest path, $\mathcal{S}$.

7.1 Input Requirements

There are some requirements that must be met by the input to this algorithm. These are outlined below:

1. If the input is a graph, it must be a complete graph.

2. Data must be represented on a 2D Cartesian plane, or there must be a bijective function to translate the coordinates to and from Cartesian space.

3. The weights between nodes in this Cartesian space must be the distance between them (i.e. the weight between pionts $p : (x_1, y_1)$ and $q : (x_2, y_2)$ is $\sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)}$).

7.2 Edge Invalidation Phase

The goal of the Edge Invalidation Phase of the algorithm is to generate $\mathcal{W}$, i.e. a set of edges $\mathcal{W} = \{w_1, w_2, ..., w_m\}$ such that $\mathcal{W} \subseteq \mathcal{E}$ and $\mathcal{S} \subseteq \mathcal{W}$. These edges must hold the properties of shortest paths, discussed in section 3.1. They must also give us some kind of meaningful information, so that we are able to construct a path $\mathcal{S}'$ from them with the claim that $\mathcal{S}' \equiv \mathcal{S}$. The Edge Invalidation Phase is broken up into steps below:

1. Add all edges $e \in \mathcal{H}$ to $\mathcal{W}$.
2. Choose an edge e from $\mathcal{E}$ with the points e_1, e_2 such that no point $v \in \mathcal{V}$ lies on e where $v \neq e_1, e_2$. We will test this edge to see if it is possible that e is part of a shortest path, and subsequently that $e \in \mathcal{W}$.

   Take each edge $e' \in \mathcal{E}$ with points p, q such that p and q straddle e and where p is on the right side of e and q is on the left side of e. We perform the following tests to see if e' invalidates $e \in \mathcal{W}$:

   (a) Let V_1 be the vector $< p, e_1 >$.
   (b) Let V_2 be the vector $< p, e_2 >$.
   (c) Find the bijection range of equality between V_1, V_2 and e, represented as $r[lower, upper]$, as follows:
      i. Let vector B start from the bijection of V_1 in the direction of e with length $|V_2|$.
      ii. Let the upper bound of r be the intersection of B and e.
      iii. Let B start from the bijection of V_2 in the direction of e with length $|V_1|$.
      iv. Let the lower bound of r be the intersection of B and e.
   (d) Repeat steps (a) through (c) with q instead of p.
   (e) If any of the intersections failed, e' does not invalidate e, so choose the next e' and repeat from the beginning.
   (f) If all intersections suceeded, find the overall intersection area between both r ranges generated from steps (a) through (c) using p and q.
   (g) If this intersection area is greater than the threshold, then e is proven invalid by e'. Choose the next e and repeat from the beginning.
3. If no e' invalidated e, then $e \in \mathcal{W}$.

7.3 Path Generation Phase

Path Generation is where data from the edges in $\mathcal{W}$ is used to contruct a Hamiltonian path. This path represents $\mathcal{S}'$, a possible shortest path. There could be multiple unique Hamiltonian paths created from the dataset, in which case $\mathcal{S}'$ might represent a set of paths $\{\mathcal{S}'_1, \mathcal{S}'_2, ..., \mathcal{S}'_k\}$. The output of this phase is as many shortest paths (S) as wanted. The Path Generation Phase is broken up into steps below:

1. Check the completed case, $\mathcal{W} = \mathcal{H}$. If true, stop processing and return $\mathcal{S}' = \mathcal{H}$.

2. For as many paths as wanted and are available, do steps (a) through (*).

   (a) Choose the smallest edge $w \in \mathcal{W}$ as a starting point.

   (b) Find the shortest Dijkstra Polygon, P, using a modified Dijkstra's Algorithm. The implementation can be any such that the polygon returned is not already part of $\mathcal{S}'$ and it is the shortest polygon that can be formed from w_1 to w_2 through one or more points.

   (c) Perform the following if/else checks on P:

      i. Check if all points of P are covered by edges in $\mathcal{S}'$. If they are, add P to the list of processed polygons, but do not alter $\mathcal{S}'$. Mark w as processed.

      ii. Otherwise, check if any edges are shared between edges in $\mathcal{S}'$ and P. If so, find each polygon P' in $\mathcal{S}'$ from largest to smallest that contains the shared edge(s) and attempt to remove it; if it is possible to remove P' while retaining all points currently covered in $\mathcal{S}$, then remove P' and add P to $\mathcal{S}'$. Otherwise mark this itteration as invalid and backtrack to a valid spot in the program and begin processing again (avoiding steps in creating the invalid state).

      iii. If not, add P to $\mathcal{S}'$ and remove all edges in $\mathcal{W}$ that intersect edges in P. Mark w as processed.

      Note: in order to add P to $\mathcal{S}'$, take all edges shared by P and $\mathcal{S}'$ and remove them from $\mathcal{S}'$. Then add the remaining edges in P to $\mathcal{S}'$. The above checks ensure that Hamiltonian path is created on the set of points in $\mathcal{S}'$ unioned with the set of points in P.

   (d) Once edge w has been processed, check if the points in $\mathcal{S}'$ contain all points in $\mathcal{V}$. If so, exit the loop and store path $\mathcal{S}'$ as a possible shortest path. If further paths are required, start the process at the second shortest path, third shortest path, and so on until all shortest paths required are generated.

   (e) Once edge w has been processed, find the next shortest edge in S' that has not already been processed and set w equal to this edge. Repeat steps (b) through (e) until the check in step (d) passes.

Each path $S'_i \in \mathcal{S}'$ (if there are multiple) may or may not be unique. It is likely that the shortest of all paths will come from using the shortest path in set $\mathcal{E}$ (i.e. the first generation), however this is not necessarily true.

7.4 Decision Phase

If more than one path is stored in $\mathcal{S}'$, choose the path with the shortest distance. This involves picking the smallest set in $\mathcal{S}'$ based on distance, which optimally is a $O(n)$ operation (since the distances of each path can be stored in the Path Generation Phase).

8 Future Work

The code developed thus far on GitHub (pixarninja/shortest_path_algorithm) implements up until the Path Generation Phase (Section 7.3). Completion of the algorithm is planned as future work for this project. Once that is completed, I would like to compare the results and runtime of the algorithm to current shortest Hamiltonian path (Traveling Salesman) algorithms. I plan to also apply the Bijection Range Method to problems such as Concave Hull generation and other problems from Computational Geometry.

To conclude my research in the future will involve the collection of more proofs, alternative algorithms, and past research in the field which I hope to culminate in an official publication.