Reimagining Transit Networks: A Data-Driven, Algorithmic Approach for the Washington DC Region

Spencer Jenkins

Abstract

This paper presents a computational framework for the design of a reimagined rapid transit network for the Washington, DC metropolitan area. Motivated by the historical context and limitations of the existing WMATA system, this study leverages geospatial analysis, graph algorithms, and data visualization to generate and evaluate alternative transit networks. The project consists of a transit network development pipeline and a program to visualize and evaluate the generated networks. The resulting networks outperform the real-world WMATA network on key metrics: critical coverage (up to 24.1"), neighborhood coverage (up to 6.6"), and average distance to the nearest station (as low as 0.92 km in DC). These results demonstrate the potential of algorithmic approaches to improve urban mobility and equity.

1 Introduction

Public transportation is a vital component of urban infrastructure, contributing to economic productivity [1], social equity [8], and environmental sustainability [1] in metropolitan areas. Washington, DC, and its surrounding areas in Maryland and Virginia, comprise the second-largest rapid transit network in the United States by daily ridership [19]. The network consists of six heavy rail lines spanning 208 kilometers and connecting 98 stations, operated by the Washington Metropolitan Area Transit Authority (WMATA) [19]. In 2019, the Washington Metro provided over 160 million trips per year, including work commutes, tourism, and other purposes [19]. Figure 1 provides a reference for the existing transit network, which is compared to the generated networks throughout this study.

Figure 1: Map of the real-world WMATA transit network in the Washington, DC region.

A primary goal in designing public transit networks in the U.S. is to reduce car dependency [1]. However, the Washington Metro has struggled to achieve this, with only about 14% of commuters in the region using transit [18]. Comparable metropolitan areas such as Toronto report higher transit commute shares, while many European and Latin American cities also have significantly higher shares [15]. These statistics are typically limited to work commutes, as data for other trip types is less available [18].

The limited success of the Washington Metro in attracting ridership is largely attributable to the priorities at the time of its planning [14]. Unlike older East Coast cities with legacy rail systems, the Metro was conceived in the 1950s, an era of expanding car infrastructure and urban expressways. The Metro was designed as a compromise, connecting car commuters to the city center, with expansions primarily serving suburban areas [14]. As a result, the network remains largely radial and focused on suburban commuters.

This design philosophy limits practical use. The network makes suburb-to-suburb and circumferential travel difficult, often requiring passengers to travel into the city center and transfer, even for short cross-town trips [19]. High-density neighborhoods such as Georgetown and marginalized areas like Anacostia have been historically underserved due to political, financial, and social factors [8]. Despite significant investment, the Metro has not achieved a substantial reduction in vehicle miles traveled (VMT) or car dependency [4]. The static nature of the network has also hindered adaptation to shifting population centers and job clusters.

The persistent gaps in the current network's ability to connect high-density and underserved regions further motivate a new approach. For example, the lack of direct connections between major suburban job centers or marginalized neighborhoods limits economic opportunity and social inclusion [8]. Existing evaluation frameworks may overlook these gaps, focusing on aggregate metrics that mask disparities in access and service quality.

To address these challenges, this study combines computational and geospatial techniques to generate, evaluate, and visualize alternative transit networks for the Washington, DC region.

Source: U.S. Census Bureau, QuickFacts, 2020 Census Redistricting Data (Public Law 94-171) Summary File [17].

2 Literature Review

The design and optimization of transit networks intersects transportation engineering, computer science, social science, and operations research. This section summarizes a review of relevant literature on geospatial data structures, network generation algorithms, and optimization approaches for transit planning.

Efficient representation and querying of spatial data is foundational for transit network design. Classic work by Libera [10] introduced B-tree structures for geographic information systems, enabling rapid spatial queries and hierarchical decomposition of map regions. Samet et al. [13] are well known for introducing quadtree structures for spatial data, supporting hierarchical decomposition in GIS applications. These data structures inspire the recursive spatial partitioning and point selection methods used to identify high-need transit locations. Toman and Olszewska [16] further demonstrate how geographic data can be transformed into

Table 1: Counties and County-Equivalents in the Study Area

County/Equivalent	2020 Population	Density (per km^2)	FIPS Code
Prince George's	967,201	687	24033
Montgomery	1,062,061	849	24031
District of Columbia	689,545	4,457	11001
Arlington County	238,643	3,672	51013
Alexandria City	159,467	4,293	51510
Falls Church City	14,658	2,651	51610
Fairfax County	1,150,309	1,184	51059
Fairfax City	24,146	1,563	51600
Loudoun County	420,959	334	51107

Table 2: Key Data Sources Used in the Study

Source	Description
US Census Bureau	Census blocks
Open Data DC/MD/VA	Points of interest, land use
Arlington County	Neighborhood boundaries
State of Maryland	Neighborhood boundaries
District of Columbia	Neighborhood centroids
WMATA	Existing transit network

network graphs, a process analogous to the use of census and point-of-interest data to construct candidate station and edge sets.

A variety of algorithmic approaches have been proposed for generating and optimizing transit networks. Bast et al. [2] provide a comprehensive overview of route planning algorithms in transportation networks, including shortest-path, greedy, and metaheuristic methods. Greedy algorithms, as surveyed by Davis and Impagliazzo [6], offer computational efficiency for a range of graph problems, though they may yield suboptimal solutions for complex network design tasks. Genetic algorithms (GAs) have been shown to be effective for transit route planning, as demonstrated by Chien et al. [5] and Dib et al. [7], who apply GAs to optimize bus and public transit networks under multiple criteria. This study adapts these approaches to the context of rapid transit, using a GA to evolve networks that balance demand coverage, redundancy, and service diversity.

Recent work has emphasized the importance of equity and real-world constraints in transit planning. Camporeale et al. [3] discuss the quantification of horizontal and vertical equity in route planning, motivating the inclusion of coverage metrics for both high-density and underserved areas. Extensions to other modes, such as smart transit and high-speed rail, are explored by Périvier et al. [11] and Roy and Maji [12], whose methodologies for station location optimization and real-time routing offer inspiration for future work.

3 Network Design

In this section, the network design process is detailed, including data collection, transformation, metric construction, network generation constraints, and the definition of the transit network search space. The network is built within this search space according to the scoring metrics and constraints, followed by post-processing.

3.1 Data Sources and Transformation

The first step is to define the study area. Table 1 lists the counties and county-equivalents included, along with their 2020 population, density, and FIPS code [17]. Figure 2 shows a map of the included regions.

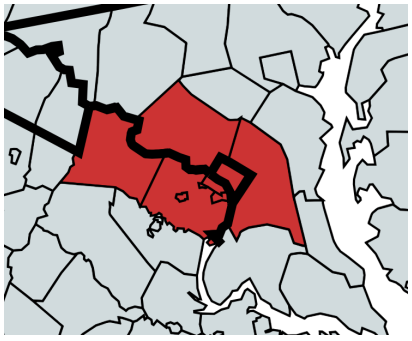

Figure 2: Counties and districts considered as part of the Washington, DC metropolitan area.

Data for the Washington Metropolitan Area is available from Open Data DC, Open Data MD, and the Virginia Open Data Portal [9]. Census block population data is obtained from the US Census Bureau. Transit potential for each block is calculated as population divided by area. Figure 3 shows the census block boundaries used for population and density analysis.

Figure 3: Census block map for the Washington, DC area.

The census block data is transformed by taking the log of the population density. A recursive spatial tree algorithm identifies the highest-need locations for transit, using a depth of 8, and detailed in Figure 4. The resulting points represent locations with the highest need for transit based on population. Figure 5 visualizes the transformed census block data used to identify high-need locations for transit.

This data is augmented with points for important destinations, as shown in Table 2. The selection reflects the availability of data from each source. The study also includes all bus and rail stops in the counties listed, to approximate areas of high transit importance.

A kernel density estimate (KDE) is constructed over the selected points to provide scores for spatial queries at several stages. Figure 6 displays the KDE heatmap used to model spatial demand for transit.

A mesh of candidate transit points is generated to balance spatial coverage with computational tractability. Population-based points (census block centroids) are combined with non-population points and deduplicated. The Gabriel graph, constructed using `libpysal.weights.Gabriel`, connects mutually closest points, resulting in a proximity network suitable for transit planning. The Gabriel graph is converted to a `networkx` graph for further manipulation, including assignment of edge weights based on Euclidean distance. This mesh serves as the substrate for both random walk and genetic algorithm-based network generation.

A Gabriel graph is constructed using the points. The graph is contracted by condensing Louvain communities by a user-defined threshold, forming the overall search space for the network. The network is constructed from the nodes and edges of this graph.

With the data fully preprocessed, one of three network construction algorithms is used to determine which line segments from the contracted Gabriel graph become part of the transit network. Figure 7 shows the contracted Gabriel graph used as the basis for candidate network construction.

function GetPoints(DataFrame D, Box E, Integer L)
  Initialize list $IDs \leftarrow [\,]$
  if $L \leq 0$ or number of rows in $D < 2$ **then**
    return IDs
  end if
  Sort D by `point_likelihood` descending
  $P \leftarrow$ second row in sorted D
  Append P.`SID` to IDs
  Define sub-boxes using P:
    $E_{BL} \leftarrow$ bottom-left quadrant of E below and left of P
    $E_{BR} \leftarrow$ bottom-right quadrant of E below and right of P
    $E_{TL} \leftarrow$ top-left quadrant of E above and left of P
    $E_{TR} \leftarrow$ top-right quadrant of E above and right of P
  Query D for points in each quadrant:
    $D_{BL} \leftarrow D$ in E_{BL}
    $D_{BR} \leftarrow D$ in E_{BR}
    $D_{TL} \leftarrow D$ in E_{TL}
    $D_{TR} \leftarrow D$ in E_{TR}
  Recursively collect points:
    $IDs \leftarrow IDs+$
      GetPoints$(D_{BL}, E_{BL}, L-1)+$
      GetPoints$(D_{BR}, E_{BR}, L-1)+$
      GetPoints$(D_{TL}, E_{TL}, L-1)+$
      GetPoints$(D_{TR}, E_{TR}, L-1)$
  return IDs
end function

Figure 4: Recursive spatial decomposition and top-point selection in a 2-D geographic space.

Figure 5: Transformed census block data representing input to the recursive spatial algorithm.

3.2 Network Creation: Naive (Random Walk) Algorithm

The naive method involves constrained walks through the graph. The walk procedure is detailed in Figure 8. Each walk starts from a random, previously unexplored node and follows the available edge leading to the greatest score. For each subsequent step, the walk chooses an available edge according to a weighted distribution: 75% probability for the straightest angle, 25% for the highest KDE score.

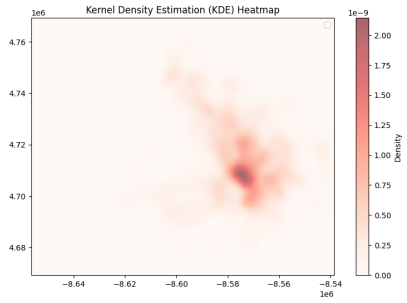

Figure 6: Kernel density estimation (KDE) heatmap of population and transit demand in the Washington, DC region.

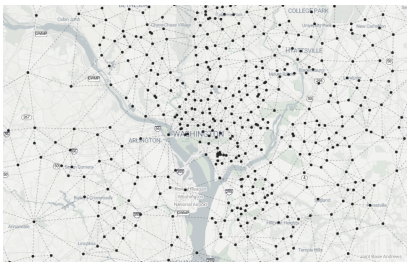

Figure 7: Contracted Gabriel graph representing the search space for network generation.

Only edges forming at least a 130-degree angle from the previous edge can be chosen, and if the overall trajectory deviates by more than 80 degrees, the walk must proceed down edges that bring the angle back. If the walk reaches a node with no options, it terminates and attempts to continue from the original endpoint in the opposite direction. The walk is only kept if it is within a specified length range (45–100 km).

Edges are available if they are unexplored or have been explored up to three times. This reflects the interlined nature of the WMATA Metro, where several lines share alignments.

3.3 Network Creation: Iterative Improvement Algorithm

The iterative improvement method initializes a set of random walks. Walks are scored by the sum of KDE scores for all nodes, indicating transit utility. The lowest-scoring walk is replaced with a higher-scoring walk, and this process is repeated up to 100 times.

3.4 Network Creation: Genetic Algorithm

The genetic algorithm (GA) is a population-based metaheuristic inspired by natural selection. In this project, the GA optimizes rapid transit networks by evolving a population of candidate solutions toward higher fitness according to multiple criteria.

Each individual represents a candidate transit network, consisting of 20 lines, with a population size of 100. The initial population is generated by creating random sets of lines, as in the naive method. The fitness function evaluates how well a candidate network meets the objectives, as a weighted sum of:

function PerformWalks(Graph G, Positions P, Parameters)
  Initialize *walks*, *threeCount*, *traversedEdges*, *completeTraversedEdges*
  for $i \leftarrow 1$ to *numWalks* **do**
    Choose random unvisited *start* node
    Initialize $walk \leftarrow [start]$, $reverseWalk \leftarrow [start]$
    Initialize $distance \leftarrow 0$, $totalTurn \leftarrow 0$, $sign \leftarrow 0$
    while $distance < maxDistance$ **do**
      $(next, deviation, angle) \leftarrow$ GetNextEdge($G, walk[-1], prev, visited, sign$)
      This function chooses an edge based on angle and score
      if no valid *next* and already reversed **then**
        break
      else if no valid *next* **then**
        Reverse walk direction, adjust *sign*
        continue
      end if
      Extend walk with *next*, update *distance* and *totalTurn*
      Update *sign* based on *totalTurn*
    end while
    if $distance > minDistance$ **then**
      Append *walk* to *walks*
      Track traversed edges and update counts
      Add frequently used edges to *traversedEdges*
    else
      Retry with new start node (with timeout)
    end if
  end for
  return *walks*, *traversedEdges*, *completeTraversedEdges*
end function

Figure 8: Pseudocode for directionally-constrained random walks to generate a transit network over a spatial graph.

- **Demand Capture:** Sum of KDE scores for all nodes covered.
- **Coverage:** Rewards networks with more nodes (potential stations).
- **Pattern Bonus:** Rewards networks connecting suburban and dense areas.
- **Redundancy Penalty:** Penalizes networks for lines sharing alignments.
- **Load Penalty:** Penalizes networks with uneven service distribution.
- **Diversity Penalty:** Penalizes networks that are too similar within a generation.

Crossover and mutation produce the next generation. Crossover combines parts of two parent networks; mutation introduces random changes. Operators are designed to respect network constraints (valid paths, feasible line lengths). Elitism ensures the best individuals are retained. The algorithm runs for 30 generations, with efficient execution enabled by Python multiprocessing.

3.5 Network Post-processing

Each walk represents a candidate metro line. Lines are grouped by shared alignment using a pairwise distance matrix and a similarity threshold.

Not all nodes in the contracted Gabriel graph become stations due to varying edge lengths. Stations are selected as all terminal nodes, transfer points between different line groups, and nodes spaced at least 1 km apart. This ensures each station serves a catchment area of roughly 500 meters, corresponding to a 15-minute walk.

Station names are assigned by proximity to neighborhood centroids, using data from Arlington County, Maryland, and the District of Columbia. If multiple stations fall within the same neighborhood, a numeric suffix is added. I was unable to find neighborhood name or boundary data for for other parts of Virginia. Figure 9 illustrates the Voronoi polygons used to analyze neighborhood coverage and spatial equity.

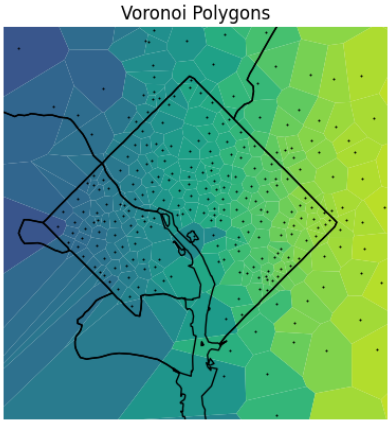

Figure 9: Voronoi polygons constructed around neighborhood centroids in the study area.

3.6 Benchmarks and Evaluation Criteria

Candidate networks are evaluated using benchmarks that reflect practical and theoretical priorities: geographic coverage (fraction of high-demand areas served), network connectivity (linking major regions), efficiency (minimizing redundant or excessively long lines), and demand capture (maximizing KDE-weighted coverage). These metrics are computed for each network and inform both random walk scoring and the genetic algorithm's fitness function.

3.7 Influential Hyperparameters

There were many hyperparameters that contributed to the success or lack thereof of the input. Table 3 shows the hyperparameters and the values used. The Louvain community resolution threshold was a key factor in determining the success of the network creation algorithm. Higher networks lead to stronger community contraction, leading to sparser graphs. The value of 0.07 provided a balance between a network with far too large of a search space and far too long considering short edges of less than a city block, while values that were too great resulted in suburban nodes being greatly reduced or lost. The line minimum and maximum distance were selected to reflect the size of the Washington metropolitan area and the distance of similar rapid transit services across the United States. The angle constraints were chosen so that the course of a given transit line would be preserved while still allowing for options and randomness. A minimum angle constraint of greater than 130 degrees results in premature halting of the walk algorithm as many nodes do not have any edges with angles greater than this. The same applies to the maximum deviation and maximum reset angles, where values too low for both hyperparameters results in walks reaching nodes they cannot proceed from. The station catchment area size, used in determining station scoring and spacing, is set to 500 meters, which reflects an average walking speed of 3 km/hr with Manhattan distance, and indicates the range around a station accessible by walking speed. The genetic algorithm weights were chosen to provide the strongest preference towards networks with coverage of low- and high-density areas on the same line.

4 Network Visualization and Exploration

The Transit Network Viewer application, developed as a part of this project, allows users to visualize and explore the networks created using the algorithms designed in this project. The web application is highly interactive, allowing users to compare networks, examine specific lines, view station names, and determine estimated travel times. Click here to access the Transit Network Viewer.

The web application's interface centers around a mapping tool powered by Leaflet. The interactive map displays the networks outputted by each of the three algorithms. Users can use checkboxes below the map to choose to view or hide specific lines, show the real-world transit network for comparison, view the catchment areas for each station, or view the contracted Gabriel graph used to generate the network. Users can hover over the transit lines, colored by group, to see helpful tooltips including their start and end locations, total distance, and number of stations. Users can hover the mouse over stations to see their name, lines served, and score.

The most important feature of the Transit Network Viewer is the Route Finder, which users can use to calculate the travel time between selected start and end locations. The route is determined using Dijkstra's algorithm, with a penalty incurred for every time a route requires a transfer. If the user selects start and end locations that are not at stations, walking time estimates are included for the parts of the itinerary both before and after the trip through the metro network. If an itinerary is found, a window displays the details of the itinerary, including the route required, the number of stations the route passes through, the total distance traveled, and the estimated travel time. The travel time calculation uses an average speed across the network of 80 km/h, 6 minutes per transfer, and a walking speed of 5 km/h.

5 Network Performance Analysis

I analyze the success of the networks quantitatively using several metrics. First, I determine the critical coverage percentage by calculating the proportion of critical transit points (high-density census blocks, critical services) that fall within the catchment area of any

Table 3: Key Hyperparameters Used in Network Generation

Hyperparameter	Explanation	Value
Louvain community resolution threshold	Controls granularity of community contraction	0.07
Line minimum distance	Minimum allowed length for a line (meters)	45,000
Line maximum distance	Maximum allowed length for a line (meters)	100,000
Minimum angle	Minimum angle between consecutive edges (degrees)	130
Maximum deviation angle	Angle leading to walk behavior forcing return to course	80
Maximum reset angle	Angle that will reset return-to-course behavior (degrees)	30
Group assignment threshold	Similarity threshold for grouping lines	50%
Station radius	Minimum distance between stations (meters)	500
Iterative improvement iterations	Number of iterations for improvement algorithm	100
Genetic: total score weight	Weight for total score in fitness function	1
Genetic: service pattern weight	Weight for service pattern in fitness function	1000
Genetic: redundancy penalty weight	Penalty for redundant lines in fitness function	-50
Genetic: load penalty weight	Penalty for uneven service distribution	-20
Genetic: diversity incentive weight	Incentive for diversity in population	-1

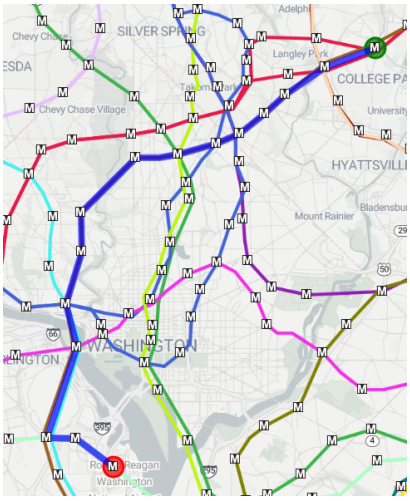

Figure 10: Transit Network Viewer in route finder mode, showing the map interface and selected route.

Figure 11: Transit Network Viewer itinerary details window for the selected route.

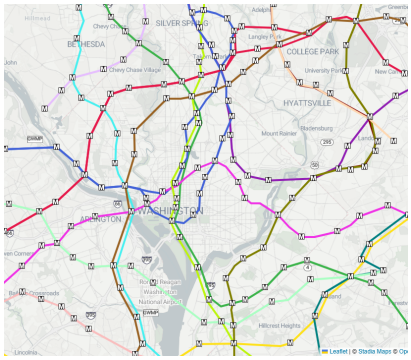

Figure 12: Transit network generated by the genetic algorithm.

station in the network. The neighborhood coverage percentage is the fraction of neighborhood centroids within the catchment areas of at least one station. I also take the average distance from the nearest station across the entire metropolitan area and for the District of Columbia. Table 4 provides the calculated data for the networks I created using each algorithm, and for the real-world WMATA network.

By the metrics established in this table, the networks produced by all three algorithms outperform the real-world network. They exhibit higher critical coverage and neighborhood coverage rates, and provide lower average distances across the region. Each algorithm performs most strongly in a different metric: the genetic algorithm performs most strongly in neighborhood coverage and average distance within DC, while the naive algorithm shows the lowest average distance across the entire region, and the iterative improvement algorithm shows the highest critical coverage.

I also perform some potential use-case evaluation by identifying location pairs and calculating the transit time in each network. I

Table 4: Comparison of Network Designs on Key Metrics

Network	% Critical Coverage	% Neighborhood Coverage	Avg. dist, regional	Avg. dist, DC
Genetic	17.47	6.61	11.03	0.92
Iterative	24.12	4.96	11.19	1.67
Naive	18.95	3.99	10.86	1.59
Real-world	13.39	3.03	14.94	1.59

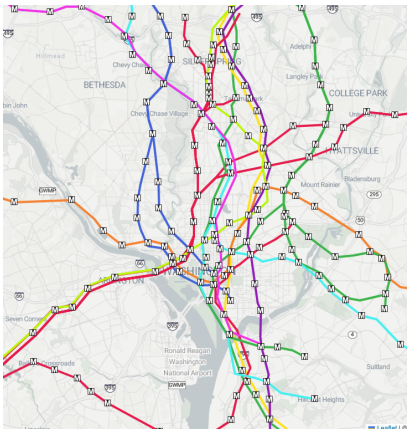

Figure 13: Transit network generated by the iterative improvement algorithm.

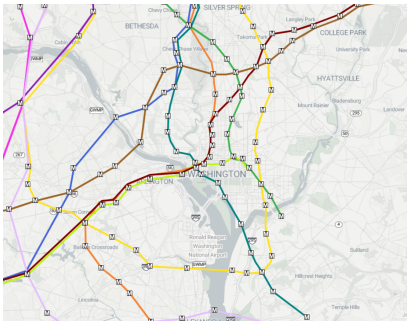

Figure 14: Transit network generated by the naive (random walk) algorithm.

calculate the transit times for my network using my Route Planner tool, and I calculate transit times for the real-world network using the WMATA transit planner application. In this report I compare the results obtained for three sample itineraries chosen to demonstrate a variety of transit uses and to address the shortcomings of the existing WMATA network that I raised in the introduction. The three routes presented in this paper are: College Park to DCA Airport (suburb to important hub), downtown Bethesda to Reston Center (suburb to suburb), and Congress Heights to downtown Silver Spring (disadvantaged neighborhood to suburb). For all three routes, the algorithms developed in this project provide faster transit times than the real-world network, with each algorithm having stronger performance on different routes. Table 5 shows the travel times calculated for each network.

Visual comparison of the three obtained networks can provide additional justification and insight. The network has strong suburb-to-suburb connection, and also has a few strong connections in the urban core, but there are some large underserved areas. This is likely because the search space is simply too large to ensure that random walks will construct a network that covers it sufficiently. This likely explains its low performance in neighborhood coverage and average station distance within DC.

The iterative improvement algorithm takes the most time to execute, but provides a much more robust and connective network, especially in the urban core. The network has strong connections between suburbs and the urban core, but suburb-to-suburb connections are more limited. This is likely because the reward mechanism simply rewards routes with higher score sums for all stations across the line, so routes that gain a high increase in score due to being very long and travelling through DC are selected for. This likely explains its high critical coverage rate.

The genetic algorithm is the most sophisticated, and its output reflects this. There are strong connections between the urban core and suburbs in all directions, and coverage within the District of Columbia is strong as well. This is likely because of the service pattern reward weight, rewarding for routes that serve the urban core with connections to suburban areas. The high neighborhood coverage rate and low average distance within DC are likely due to this.

5.1 Limitations

The quality of the networks generated using my pipeline are only as good as the data they are created from. As discussed earlier, the data portals for each of the states varied wildly with respect to data depth and breadth. Virginia, in particular, did not have nearly as much data containing locations of areas supported by transit. Maryland and Washington, D.C. both had much more comprehensive data resources, which is expected given that Washington, D.C. is a single municipal jurisdiction and Maryland is a relatively small state mostly centered around a single metropolitan area. Virginia, on the other hand, is a large state with many metropolitan areas, and the data reflected this. There were often data of the sort I was interested in, but only for one or two counties and outside the DC area.

Table 5: Sample Transit Times for Selected Origin-Destination Pairs

Network	College Park → DCA	Bethesda → Reston	Congress Heights → Silver Spring
Genetic	29 min	41 min	30 min
Iterative	37 min	1 hr 1 min	19 min
Naive	32 min	32 min	17 min
Real-world	43 min	1 hr 7 min	41 min

The method of combining point data from multiple sources and creating a KDE from multiple sources entailed several limitations: first, each data point was treated as having the same transit importance, and the KDE therefore acted as a map of the amount of points in a given area. This means that the KDE scoring returned high values for regions that were well covered by the existing data, and low values for those with poor coverage, and no weight to whatever relative importance may have existed between points. The overall takeaway is that the transit algorithm happened to produce plausible-looking maps not as a reflection of the data, but because I constrained an otherwise random algorithm to produce results that look plausible. Across the three algorithms, I received wildly different results that often left many densely populated neighborhoods underserved. On the other hand, across algorithms, I obtained similar travel times between destination pairs, which is likely because with sufficient density, the network is large enough that one- or two-transfer routes on fairly straight alignments are possible between most urban destinations. It is also important to note that my network's reported high performance as compared to the real-world network is also likely largely attributable to the length and number of lines. Each algorithm produces a 20-line network, as compared to the real-world network which has six lines.

In general, the method I have created in this project does not largely reflect the priorities or procedures of real-world transit agencies. Transit networks are not decided by random algorithms applied across entire metropolitan areas. The search space of the problem is still too great; there were not enough hard constraints on which areas the network should have connected. The result is a network that reaches most necessary areas because the number of randomly-generated lines is high enough.

5.2 Implications for Urban Transit Planning

Despite the many shortcomings of the procedure inherent to this project, it still serves as a valuable tool at encouraging ambition and imagination in public transit planning. Although a 20-line metro system for the Washington metropolitan area would potentially cost hundreds of billions of dollars to construct, it would not be the first transit network of its size in the United States. The networks generated in this project are similar in density and robustness to the New York City Subway, which operates 28 service patterns across over 1000 km of track. New York City is far beyond all other American cities at reducing car usage, which is largely attributable to the coverage, speed, and dependability of its subway network. Although the New York metropolitan area is nearly three times as large as the Washington metropolitan area, the New York Subway does not expand beyond city limits into the suburbs, and the population within New York city limits of around 9 million is much more comparable to the population of the Washington metropolitan area. Therefore, a rapid transit network for the Washington metropolitan area that is significantly larger than the existing network would not be without comparison.

5.3 Future Work

I hope to apply the general procedure of this work to create similar transit maps for other metropolitan areas in the United States. I also hope to explore deep learning-based approaches to learning a network, which I attempted but was unable to find a model that could converge on a non-trivial solution.

6 Acknowledgments

I thank the instructors and peers of CMSC 725 for their feedback and support. I also acknowledge the open data providers, including the US Census Bureau, city- and state-level open data providers, and transit providers.

References

[1] American Public Transportation Association. 2022. Public Transportation Facts. https://www.apta.com/news-publications/public-transportation-facts/

[2] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck. 2016. Route Planning in Transportation Networks. In *Algorithm Engineering*, L. Kliemann and P. Sanders (Eds.). Lecture Notes in Computer Science, Vol. 9220. Springer, 19–80. doi:10.1007/978-3-319-49487-6_2

[3] R. Camporeale, L. Caggiani, A. Fonzone, and M. Ottomanelli. 2016. Quantifying the impacts of horizontal and vertical equity in transit route planning. *Transportation Planning and Technology* 40, 1 (2016), 28–44. doi:10.1080/03081060.2016.1238569

[4] Mikhail V. Chester and Arpad Horvath. 2009. Environmental assessment of passenger transportation should include infrastructure and supply chains. *Environmental Research Letters* 4, 2 (2009), 024008. doi:10.1088/1748-9326/4/2/024008

[5] Steven Chien and Paul Schonfeld. 2001. Optimization of Grid Transit System in Heterogeneous Urban Environment. *Journal of Transportation Engineering* 127, 4 (2001), 281–290. doi:10.1061/(ASCE)0733-947X(2001)127:4(281)

[6] S. Davis and R. Impagliazzo. 2007. Models of greedy algorithms for graph problems. *Algorithmica* 54, 3 (2007), 269–317. doi:10.1007/s00453-007-9124-4

[7] Nour Dib and Hojjat Adeli. 2017. Optimization of Transit Network Design Using Genetic Algorithms. *Journal of Advanced Transportation* 2017 (2017), 1–12. doi:10.1155/2017/8969352

[8] Mark Garrett and Brian D. Taylor. 1999. Reconsidering Social Equity in Public Transit. *Berkeley Planning Journal* 13 (1999), 6–27. https://escholarship.org/uc/item/1mc9t108

[9] Government of the District of Columbia. 2024. Open Data DC Portal. https://opendata.dc.gov/

[10] G. Di Libera and H. Samet. 1986. B-trees, k-d Trees, and Quadtrees: A Comparison Using Two-Dimensional Keys. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8, 5 (1986), 586–593. doi:10.1109/TPAMI.1986.4767842

[11] N. Périvier, C. Hssaine, S. Samaranayake, and S. Banerjee. 2021. Real-time approximate routing for Smart Transit Systems. In *ACM SIGMETRICS Performance Evaluation Review*. ACM, New York, NY, USA, 73–74. doi:10.1145/3543516.3460096

[12] S. Roy and A. Maji. 2023. High-speed rail station location optimization using customized utility functions. *IEEE Intelligent Transportation Systems Magazine* 15, 3 (2023), 26–35. doi:10.1109/mits.2022.3207411

[13] Hanan Samet. 1984. The Quadtree and Related Hierarchical Data Structures. *Comput. Surveys* 16, 2 (1984), 187–260. doi:10.1145/356924.356930

[14] Zachary M. Schrag. 2006. *The Great Society Subway: A History of the Washington Metro.* Johns Hopkins University Press, Baltimore, MD, USA.

[15] Statistics Canada. 2017. Journey to Work: Key Results from the 2016 Census. https://www150.statcan.gc.ca/n1/daily-quotidien/171129/dq171129c-eng.htm

[16] Jakub Toman and Joanna Olszewska. 2014. Algorithm for Building Graphs from Maps for Route Planning. In *2014 17th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, Qingdao, China, 3022–3027. doi:10.1109/ITSC.2014.6958142

[17] U.S. Census Bureau. 2020. 2020 Census Data. https://data.census.gov/

[18] U.S. Census Bureau. 2022. Commuting Characteristics by Sex: 2022 American Community Survey 1-Year Estimates. https://data.census.gov/table?q=commute+washington+dc&tid=ACSST1Y2022.S0801

[19] Washington Metropolitan Area Transit Authority. 2023. WMATA Facts and Figures. https://www.wmata.com/about/facts/