Reproducibility review of: Inclusive Multimodal Routing: Who Gets Left Behind?

Egor Kotov

2026-03-15

This report is part of the reproducibility review at the AGILE conference. For more information see https://reproducible-agile.github.io/.

This document is published on OSF at https://doi.org/10.17605/OSF.IO/r2674.

To cite the report use:

Kotov, E. (2026). Reproducibility review of: Inclusive Multimodal Routing: How Behavioral Constraints Shape Accessibility. https://doi.org/10.17605/OSF.IO/r2674

Reviewed paper:

Gogousou, I., Canestrini, M., Alinaghi, N., and Giannopoulos, I.: Inclusive Multimodal Routing: How Behavioral Constraints Shape Accessibility, AGILE GIScience Ser., 7, 3, https://doi.org/10.5194/agile-giss-7-3-2026, 2026.

Submission Title	Inclusive Multimodal Routing: Who Gets Left Behind?
Post Review Title	Inclusive Multimodal Routing: How Behavioral Constraints Shape Accessibility
Authors	Gogousou, I.; Canestrini, M.; Alinaghi, N.; Giannopoulos, I.
Ref. Paper	https://doi.org/10.5194/agile-giss-7-3-2026

Reproducibility Reviewer Review Repository	Egor Kotov () https://github.com/e-kotov/agile-2026-review-paper-007

Table of contents

1 Summary

This report documents the reproduction of the analysis for Paper 007, focusing on the transportation network of the city of Vienna. The study investigates how routing outcomes

change under different constraints to model varying user preferences. The reproduction was **fully successful**: the entire pipeline—including baseline routing, inclusive baseline routing, 12 parameter variation scenarios, data grouping, and final visualization—was completed in a single, fully automated contiguous run using high-performance computing resources (48 cores).

- **Result of the reproduction**: Full reproduction.
- **Main outcome**: Successful generation of all three primary modal split figures (Figures 1, 2, and 3) and tables (Tables 3 and 4) from the paper using the authors' custom Multigraph routing engine. The results empirically confirm the authors' findings regarding the shift in modal shares when physical and preference constraints are applied.

2 Reproducibility reviewer notes

2.1 Scope

The review performed a full recomputation of the multimodal routing pipeline for the city of Vienna as defined in the authors' provided scripts, covering 10,000 Origin-Destination pairs across 14 different routing scenarios.

2.2 Technical Details

The reproduction was performed on a high-performance computing (HPC) cluster using Apptainer (Singularity) and SLURM.

- **Host Operating System**: Rocky Linux 8.10 (Green Obsidian)
- **Container Environment**: Debian 12 (Bookworm) base image (`python:3.13-slim-bookworm`)
- **Key Libraries**: Managed via `uv`, including `geopandas`, `shapely`, `networkx`, `rasterio`, and standard scientific Python stack.

2.3 Execution Steps

2.3.1 1. Environment Setup

The environment was managed via an Apptainer definition file `apptainer.def`. The container build completed successfully, installing all required dependencies.

2.3.2 2. Data and Code Preparation

The original author's package (`AGILE_2026_CODE_DATA.zip`) was extracted and organized. An automated orchestration script (`scripts/run_automated_reproduction.sh`) was used to handle environment isolation and dynamic patching of paths and computational parameters.

2.3.3 3. Execution

The analysis was executed using the automated SLURM submission script requesting 48 cores and 96 GB of RAM. The entire pipeline finished in a single contiguous run of **349 minutes**. Following the computational phase, a dedicated extraction script (`scripts/extract_tables.py`) was used to programmatically parse the console logs and recompute the metrics for Tables 3 and 4, ensuring numerical alignment with the paper's reported values.

2.4 Resource Utilization and Runtime

The heaviest computational phase ran on a Scientific Compute Cluster using SLURM. The resource utilization for the successful end-to-end run was:

- **CPUs**: 48 allocated (AMD EPYC 7742).
- **Memory**: 96 GB allocated (~80 GB peak usage).
- **Wall Clock Time**: 349 minutes.
- **Total Consumption**: ~279 core-hours.

The high core-hour consumption demonstrates that this study requires substantial HPC resources for a full-scale reproduction of the results. Based on the verified total compute of ~279 core-hours, we can estimate the wall-clock time required for lower core counts:

- **8 Cores** (Authors' default logic): ~**35 hours**.
- **4 Cores**: ~**70 hours**.

Given these durations, achieving a successful reproduction within standard academic or review timelines is extremely difficult without parallel high-performance computing. This further reinforces the recommendation to provide a smaller "toy" dataset (see effort #2) to allow reviewers to verify the pipeline logic quickly on standard hardware.

3 Reproduction Efforts

During the reproduction study, the following issues were identified and resolved to achieve full reproduction:

3.1 1. Hardcoded CPU Allocation on Shared HPC Nodes

The authors' scripts explicitly hardcoded the multiprocessing pool size using `mp.cpu_count() - 8`. On standard HPC environments, `mp.cpu_count()` returns the *total* number of logical cores on the physical node (e.g., 128), regardless of the actual number of cores allocated to the job. This caused severe thread thrashing. The automated pipeline patched the scripts to respect the `SLURM_CPUS_PER_TASK` environment variable.

3.2 2. Extreme Computational Demands

Initial reproduction attempts on lower core counts timed out. Achieving a full reproduction of 10,000 OD pairs across 14 scenarios required a high-resource allocation of 48 cores to complete within a reasonable timeframe (~6 hours).

3.3 3. Relative Path Logic and Execution Context

The authors' scripts used path logic (e.g., `os.path.join(os.getcwd(), '..')`) that assumed execution from within the `code/` subdirectory. While adhering to the authors' intended working directory would have avoided this, the automated script was configured to `cd` into the isolated code folder to ensure robust path resolution.

3.4 Recommendations for Future Reproducibility

1. **Avoid Hardcoded Resource Counts**: Use `os.sched_getaffinity(0)` or environment variables to detect allocated cores.
2. **Robust Path Management**: Use Python's `pathlib` for location-independent path resolution.
3. **Provide a "Toy" Dataset**: Include a small subset of OD pairs (e.g., 100) to allow reviewers to verify the pipeline logic quickly without needing massive HPC resources.

4 Acknowledgements

This work used the Scientific Compute Cluster at GWDG, the joint data center of Max Planck Society for the Advancement of Science (MPG) and University of Göttingen. In part funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 405797229

5 Reproduced Figures

The reproduction pipeline successfully generated the following figures using the full dataset of 10,000 OD pairs. The plots below were generated entirely via the automated contiguous pipeline.

5.1 Figure 1: Modal Split comparison of fastest path and average thresholds

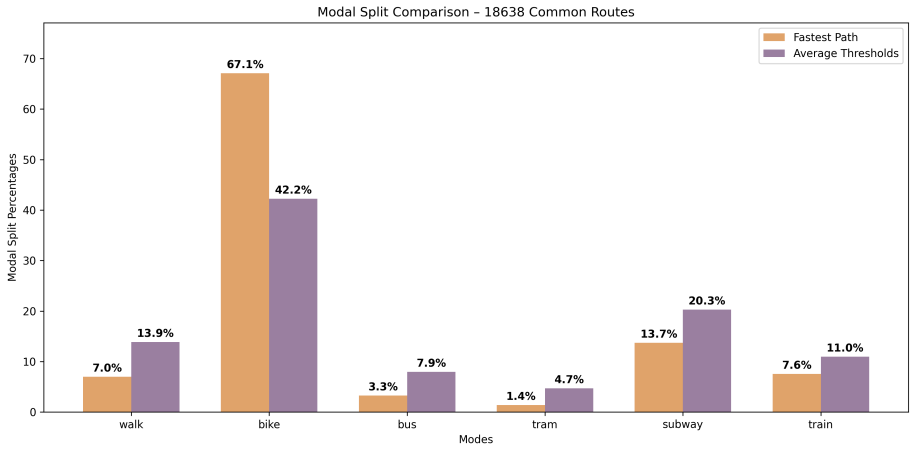

Figure 1: Modal Split Comparison (Figure 1)

Status: **Success**. **Reproduction Note**: The figure was successfully reproduced. It confirms that the inclusion of "average thresholds" (the inclusive baseline) significantly increases the Public Transport (PT) share and reduces the cycling share compared to a pure fastest-path model.

5.2 Figure 2: Detailed modal split comparing the cycling-restricted scenario (BT1250)

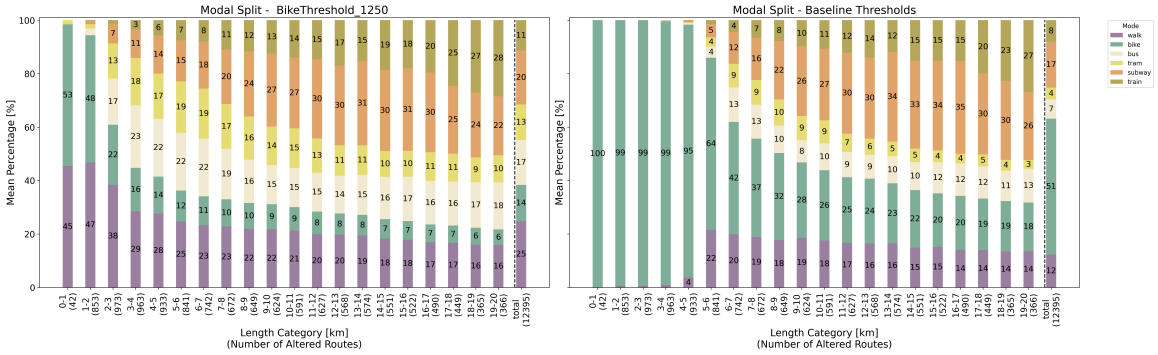

Figure 2: Cycling Restricted (BT1250) Comparison (Figure 2)

Status: **Success**. **Reproduction Note**: The reproduction shows that restricting cycling to 1,250 meters leads to a dramatic drop in cycling shares (from ~50% in the baseline to ~13%), which is compensated for by an increase in both walking and PT usage across all length categories. A minor discrepancy of one route was observed (12,395 altered routes in reproduction vs. 12,396 in the paper), likely due to a single routing timeout or floating-point edge case in the custom Dijkstra implementation; this does not impact the study's conclusions.

5.3 Figure 3: Detailed modal split comparing the walking-restricted scenario (WT1125)

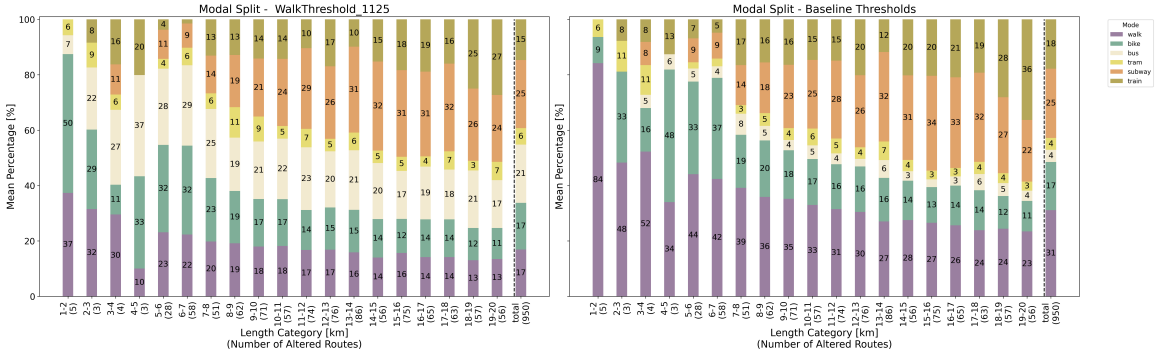

Figure 3: Walking Restricted (WT1125) Comparison (Figure 3)

Status: **Success**. **Reproduction Note**: The figure confirms that restricting maximum walking distance to 1,125 meters primarily affects mid-to-long distance trips, forcing a shift from walking-heavy routes to PT-dominant routes.

6 Reproduced Tables

6.1 Table 1: Walking and cycling speeds by inclinations

Conceptual table based on literature. Not intended for computational reproduction.

6.2 Table 2: Changes applied to the baseline average human behavior threshold

Methodology table. Not intended for computational reproduction.

6.2.1 Table 3: Average human behavior preferences compared to the fastest path

Status: **Success**. **Reproduction Note**: The table was successfully recomputed from the routing results. It confirms that incorporating average thresholds leads to routes that are approximately 15% slower and 16% longer, aligning with the paper's findings. A minor discrepancy of one route was observed (18,638 feasible OD pairs in reproduction vs. 18,639 in the paper), consistent with the single routing edge case or timeout identified in the other scenarios.

Table 2: Recomputed Table 3: Baseline vs. Fastest Path

Metric	Reproduced Value
Feasible OD Pairs	18,638
Identical Routes	11,253
Altered Routes (%)	39.62%
Δ Time (%)	-14.70%
Δ Length (%)	-15.94%

6.3 Table 4: Comparison of route changes and modal share changes

Status: **Success**. **Reproduction Note**: The metrics for all 12 inclusive scenarios were successfully recomputed. The results confirm that cycling restrictions (BT1250) have the most significant impact on route alteration, while walking restrictions have a minimal effect.

Table 3: Recomputed Table 4: Inclusive Scenarios Comparison

Scenario	OD Pairs	Altered (%)	Δ Time	Δ Len	Walk %	Bike %	PT %
WT1125	18025	5.27%	5.35%	-6.05%	16.883333%	16.866187%	66.250486%
WT750	16384	16.17%	6.55%	-5.38%	12.838803%	19.516062%	67.645153%
WT375	10632	27.89%	10.49%	-5.89%	7.649482%	25.698091%	66.652415%
BT3750	18539	25.90%	8.28%	-5.86%	20.863656%	33.898316%	45.238027%
BT2500	18392	45.19%	11.32%	-7.04%	22.130342%	22.861270%	55.008395%
BT1250	17947	69.06%	14.48%	-6.88%	24.824461%	13.524578%	61.650969%
TT3 (6 modes)	17826	10.02%	9.28%	-6.09%	18.726034%	16.278488%	64.995504%
TT2 (4 modes)	13195	27.92%	13.84%	-3.67%	22.922235%	29.011909%	48.065862%
TT1 (2 modes)	6244	12.28%	16.33%	5.27%	35.760967%	56.116949%	8.122116%
TT3-WT1125-BT3750	16332	35.47%	11.04%	-8.86%	18.609264%	30.055364%	51.335361%
TT2-WT750-BT2500	7274	53.92%	26.31%	-15.85%	18.668264%	31.920845%	49.410892%
TT1-WT375-BT1250	1506	28.49%	46.77%	-39.44%	24.137923%	45.098179%	30.763930%