A Appendix - For Online Publication

Contents

A.1 Appendix Figures 50
A.2 Appendix Tables 65
A.3 Data Processing 72
A.3.1 Bus GPS Data Processing 72
A.3.2 Bus Travel Times 73
A.3.3 Bus Wait Time Distribution 75
A.3.4 TransJakarta Ridership Data Processing 77
A.3.5 Veraset Smartphone Location Data Trip Processing 79
A.4 Reduced Form Estimation 82
A.5 Attention Probabilities 83
A.6 Demand Model Derivations 83
A.7 Optimal Network Design 85
A.7.1 Optimization Environment: Predicted Bus Travel Times 85
A.7.2 Ridership Equilibrium With Bus Capacity Penalties 87
A.7.3 Analytic Results for Optimal Allocations Model 89
A.7.4 The Simulated Annealing (SA) Algorithm Asymptotic Result 90
A.7.5 The Candidate Network Proposal Function Ψ 91

A.1 Appendix Figures

A.3 Data Processing

A.3.1 Bus GPS Data Processing

This section explains how we process bus GPS data to find bus station arrival times, which we use when describing the wait time distribution and when assigning bus transactions to bus stations. We use GPS data every 5 or 10 seconds available for most TransJakarta busses between January 2017 and March 2020. We also use bus trip logs entered manually by bus dispatchers. For each trip, this data contains the bus code, the bus route code (with direction), and the trip start time. **2,798** TransJakarta busses appear in the GPS data.

Combined data on GPS and trip logs is significantly better starting in 2018. In 2017, **22.2%** of bus days contain both GPS data and trip logs, **76.2%** contain only GPS data but no logs, and the rest contain only trip logs without GPS data. However, in 2018 - March 2020, **70.7%** of bus days contain both GPS data and trip logs, **22.8%** contain only GPS data but no logs, and the rest contain only trip logs without GPS data.

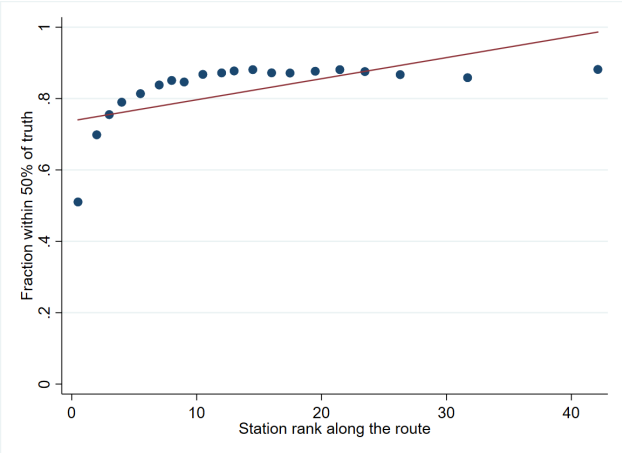
We developed three algorithms to identify when a bus arrives at a given bus station. When both GPS data and trip logs are available, we map match the bus GPS locations to the path of the bus route (from separate data), starting from the trip start time recorded in the trip log, and find arrival times for all bus stations along that route. (The algorithm automatically identifies bus trips where the "return" trip log is missing, which happens **15.4%** of the time.)

When only GPS data is available, the algorithm proceeds in two steps. First, given a bus and a date, it ranks bus routes in decreasing order of overlap with the GPS traces for that bus day and generates a set of candidate routes where the traces overlap at least **30%** with the route. Second, the algorithm map matches the GPS traces to the best-fit bus route, trying multiple candidates, starting from the first GPS point that is near to the first station of a candidate bus route.

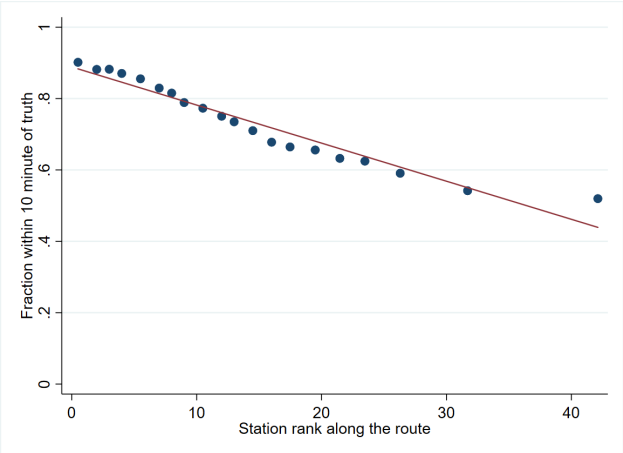
When only trip log data is available, the algorithm proceeds in two steps. First, we predict *station* arrival time as a function of trip start time, bus route, and time of the day. We estimate this model using the output of the first algorithm for the same route, on bus-days when both GPS and trip log data is available. Figure A.15 shows that we achieve high accuracy using this model. Second, we apply these predicted times using the trip start time from the log.

Figure A.15: Accuracy of Predicting Station Arrival Time Based Only on Trip Start Time

(a) Relative Error



(b) Absolute Error



We end up with a total of **7,315,854** trips out of which **5,568,890 (76.1%)** are identified when both GPS data and trip logs are available, **859,776 (11.8%)** are identified when only GPS data is available and the remaining **12.1%** are identified when only trip log data is available.

A.3.2 Bus Travel Times

In this section, we describe how we compute travel times between an origin station o, a destination station d, along a route r. These travel times are used in the reduced form analysis to compute bus route travel times for defining Events 1 and 2, and in the model to characterize the bus network choice set of any given traveler.

For every triplet (r, o, d) we consider all trips along r and the time they take to go from o and d. We then take the median travel time within this set, over all trips between 7 AM and 7 PM in our study period.

Figure A.16 shows that there is only a very small amount of variation in "delay" (median travel time per kilometer, or inverse speed) for trips starting at different times of the day between 7 AM - 7 PM. The variation is even smaller for BRT routes. Moreover, we can also observe that delay is mostly stable over the years in our sample. This supports our choice of using medians of travel times of trips starting between 7 AM - 7 PM, over the entire data period (January 2017 - March 2020). Travel time and delay in different years are strongly correlated after including route, origin and destination fixed effects (Table A.10).

Figure A.16: Bus Travel Delay by Departure Time and Year

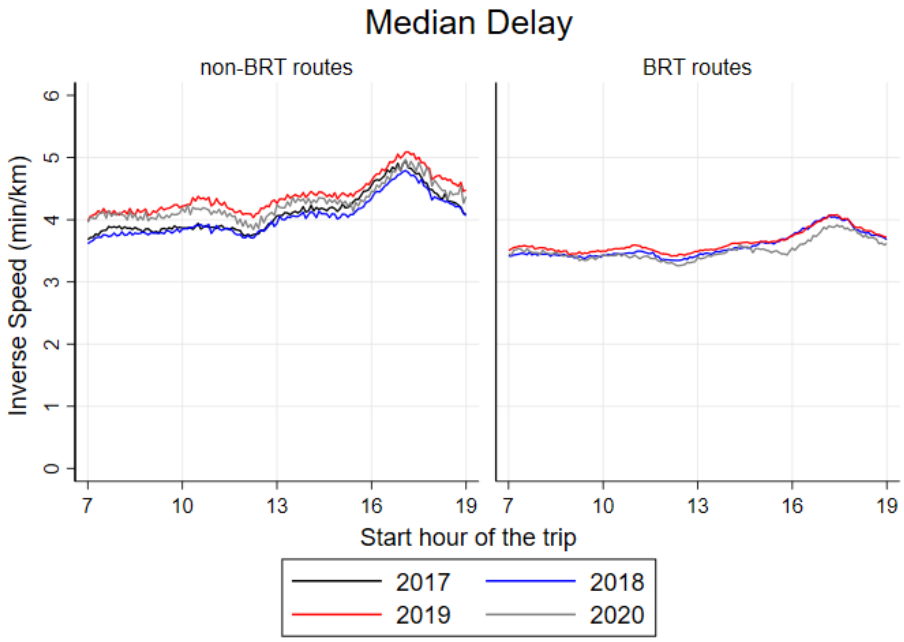


Figure Notes: This figure reports median delay (inverse speed, in minutes per km) by departure time, separately for BRT routes and for non-BRT routes using data from January 2017 - March 2020 for routes that were active throughout this period. To construct this figure, we assign each trip to 5-minute bins by their departure time at first station on the route. We then find the time taken by the bus to complete each trip and the distance traveled for the entire route (start to end). Using this, we calculate the inverse speed for each trip (time/distance). Within each 5-minute bin, we finally plot the median of inverse speed of trips throughout the day, grouping the trips by year.

Table A.10: **Bus Travel Time Correlation Across Time**

	Log(Median Delay) 2017	Log(Median Delay) 2018	Log(Median Travel Time) 2017	Log(Median Travel Time) 2018
	(1)	(2)	(3)	(4)
Log(Median Delay) 2019	0.716***	0.794***		
	(0.048)	(0.038)		
Log(Median Travel Time) 2019			1.031***	1.009***
			(0.007)	(0.002)
Constant	0.368***	0.222***	-0.193***	-0.083***
	(0.062)	(0.047)	(0.043)	(0.018)
R^2	0.476	0.685	0.961	0.983
N	26990	38624	26990	38624

Table Notes: We organize the data at the route-origin-destination level. The outcomes variables are the log of median delay (inverse speed) and log of median travel time. Standard errors are clustered three-way by route, origin and destination, and reported in parentheses: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

A.3.3 Bus Wait Time Distribution

In the model, we assume that wait times are exponentially distributed. In this section we analyze how wait times for different TransJakarta routes are distributed in reality, using the GPS data from 2019.

We first calculate bus headways (difference in minutes between two consecutive buses) for every station, route, and direction. Then, we calculate the frequency of headway occurrence by route and minute. Assuming that TransJakarta passengers are "non-planning" and thus arrive at bus stations at a (locally) constant rate, we calculate the implied histogram of wait times by taking the reverse cumulative total of headway frequency. We compute the wait time distribution for 45 routes (18 BRT and 27 non-BRT) that have only two main route versions (i.e. where the top two trip variants account for $> 95\%$ of all trips on that route).

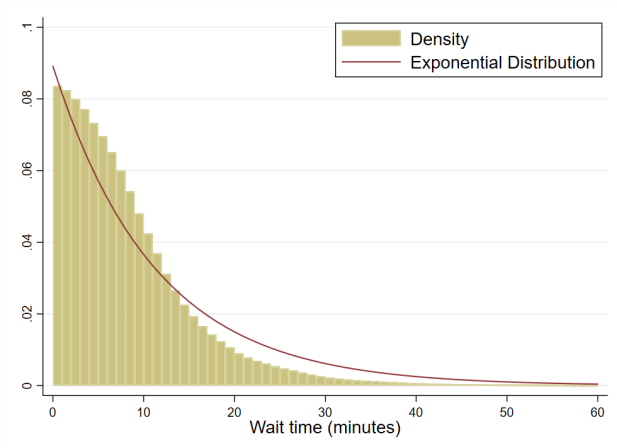
Panels A and B in Figure A.18 plot the wait time distribution for non-BRT route 1E. The wait time distribution is approximately exponential, except for wait times over 30 minutes, where the empirical distribution puts lower weight than the exponential. Panel B plots results for all the routes in our sample, showing that there is a tight linear relationship between a route's mean wait times and the coefficient of variance of wait time. BRT and non-BRT routes share the same relationship. In other words, BRT routes are not less (or more) uncertain than non-BRT routes, conditional on average wait time.

Figure A.17: Sample Arrival Time Monitor at a TransJakarta Station

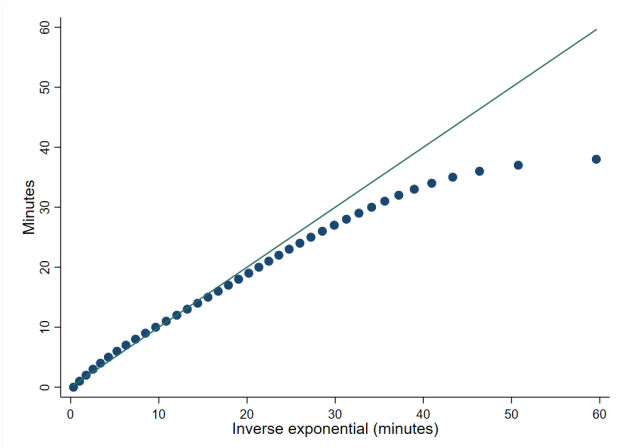
Notes: During the study period, all BRT bus stops were equipped with screens that displayed real-time estimated bus wait times. (The estimates were based on real-time bus GPS location data.)

Figure A.18: Wait Time Distribution Within and Across Routes

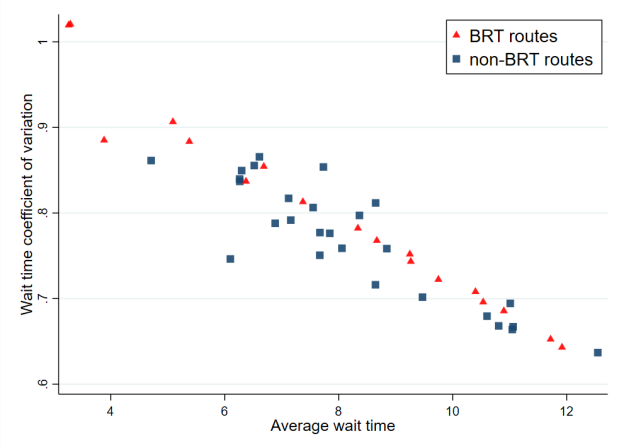
(a) Wait Time Distribution (route 1E)



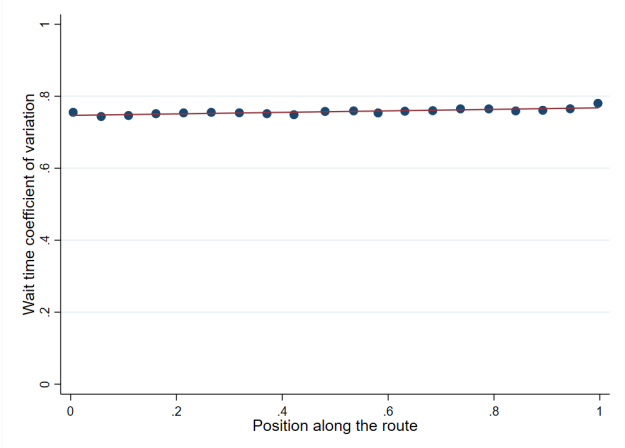
(b) Wait Time Q-Q Plot (route 1E)



(c) Wait Time Reliability on BRT and Non-BRT Routes



(d) Within-Route Reliability



Notes: Panel (A) and (B) report the wait time distribution for non-BRT route 1E. To create these graphs, we calculate bus headways (defined as the duration in minutes between two consecutive buses of the same route and direction at a particular station) using the bus GPS data. To obtain the wait time distribution from the headway distribution, we assume that passengers arrive at the station at a constant rate. In panel (A), we fit an exponential distribution to the empirical wait time distribution. Panel (C) reports the mean of wait time (in minutes) and the coefficient of variation of wait time, separately for BRT routes and for non-BRT routes. Panel (D) reports the coefficient of variation of wait time over position along the route (0 for first station, 1 for last station). For all panels, we use data from January to October 2019 and restrict the sample to only include arrival times during morning peak hours (8 AM-12 PM). Each Transjakarta route may have several route variants that differ slightly in length and stations reached. Therefore, for panels (C) and (D), we restrict the sample to routes where two trip variants account for above 95% of all trips on that, consisting of 45 routes (18 BRT and 27 non-BRT routes).

A.3.4 TransJakarta Ridership Data Processing

We use two main TransJakarta ridership data sources. First, we use transactions ("taps") in BRT bus shelters, where passengers pass through turnstiles to enter the bus shelter. In theory, passengers also need to "tap out" when they exit a bus shelter. This is only enforced in **35.9**% of bus shelters (accounting for **34.0**% of shelter ridership).

Second, we use transaction data from non-BRT busses. When a passenger gets on the bus from a (non-BRT) bus station on the side of the road, they pay inside the bus using their card or cash. When a passenger uses cash, the bus attendant uses their own card on the card reading machine. In 2019, 57% of transactions were cash. To assign these transactions to bus station, we use the bus station arrival time from the GPS or trip log data (section A.3.1).

Origin-Destination Ridership Flows. To construct origin-destination ridership flows at each point in time, we proceed in several steps. We focus on a sample of cards for which we observe ridership behavior over time. At each step, we construct weights assuming that the sample of cards and transactions that we use is representative.

First, we drop "administrative" cards that are likely used by bus attendants or other TransJakarta employees. We label a card as administrative on a given day if it is used repeatedly throughout the day on the same bus (for non-BRT transactions), or at a BRT shelter. Administrative cards account for **2.1%** of all BRT shelter transactions, and **63.2%** of all non-BRT bus transactions. We assume that travel behavior for non-admin cards is representative of all trips in the system.

Second, we process "serial" taps. We combine consecutive taps using the same card into a single transaction, likely capturing groups traveling together and using a single card. **8.1%** of transactions have two or more consecutive taps.

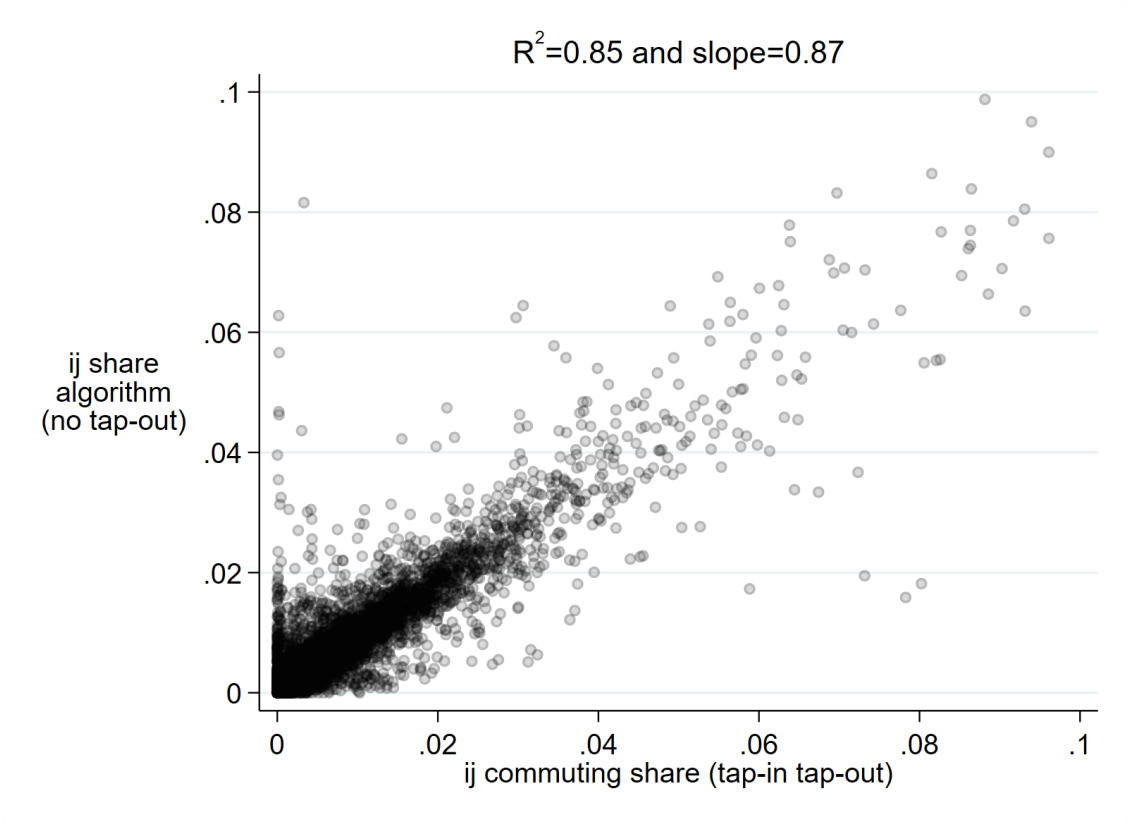
We exclude 15 non-BRT routes from ridership coding for reasons related to the way payments were made. Each non-BRT bus accepts payments with smart cards provided by a specific bank. We lack detailed micro-transaction data corresponding to one of the banks. Overall, tap data from dropped routes accounts for 14.96% of total tap data in 2017-2020. Coverage is 44.11% of total routes in 2017 and 72.14% in 2020.

Algorithm to Infer Trip Destination and Validation. Ideally, we want to determine a commuter's origin-destination based on the tap-in and tap-out locations of their trip. However, tap-out information is not always available across all routes and stations in TransJakarta. Commuters were not required to tap-out on non-BRT routes, and tap-out was enforced only at some BRT bus stops. Given these limitations, we developed an algorithm

to infer a commuter's destination for each ridership tap-in. The algorithm consists of three main rules for assigning a trip's destination. The first rule is to use the actual tap-out location as the destination, when the tap-out occurs within 4 hours after the tap-in. For the second rule, we proxy for the destination using the next trip's tap-in, if that next trip takes place on the same day or the next day. For the third rule, for each smart card, each month, we computed the top 2 stations with the most taps. If the trip origin is one of the top 2 stations, we proxy for the trip destination using the other top 2 stations. This method is only applied for frequent commuters, defined as those with more than 10 taps in a month and more than 75% of total taps located within its top-2 stations.

We validate the algorithm by comparing ridership flow destination shares originating from station i towards station j (destination) using data with only tap-in entry (algorithm) and data with both tap-in and tap-out entry (actual). Figure shows that the two measures are highly correlated.

Figure A.19: Comparison of Actual and Inferred (o, d) Ridership Shares



A.3.5 Veraset Smartphone Location Data Trip Processing

We use the smartphone trip processing algorithm from Kreindler (2023) to convert raw GPS data into individual trips and common location for each device in the data.

For each individual i, the algorithm first classifies individual trips and "stays," periods of time when the device is observed to be stationary in a given area. Then, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is used to cluster all locations for any given devices. The most common cluster is labeled as "home."

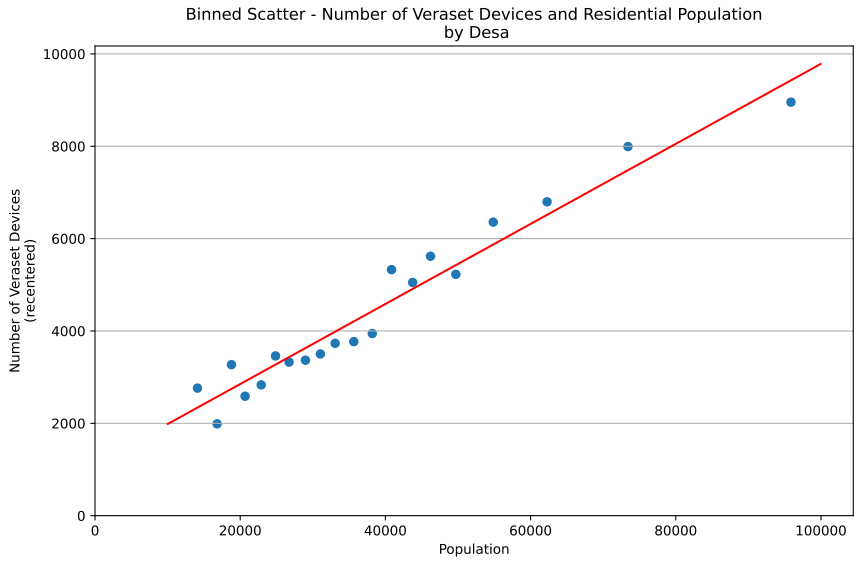
The sample of trips used in the analysis is all weekday trips starting and ending at a known locations (i.e. locations classified by DBSCAN as belonging to a cluster), excluding trips starting before 5AM or after 11PM. We also drop trips that are unusually long or short (in both duration and distance), "swiggly" trips (where ratio of the largest distance between any two points on the trip to path length is less than 0.3), and "short loop" trips (trips that are less than 2km where the ratio of origin-destination distance to path length is less than 0.3).

Given the selection concerns when using smartphone location data (Blanchard et al., 2021), we examine how representative the users in our data are of the general Jakarta population. First, we compare the distribution of users' home locations obtained from the data to that of residential population from the PODES survey (Figure A.20). The number of devices in our data with home locations in each desa is correlated with populations in the desa, suggesting that distribution of Veraset devices is consistent with population distribution. However, the coverage of our data (number of devices per total population) is slightly lower in areas with higher population density and proportion of population under poverty.

We construct two main data sets using the Veraset trips data. First, we construct a panel of trip flows at the origin grid by destination grid by week level. Each week, for each device in the data, we re-weight all its weekday trips that week to represent a single typical weekday. (This is essentially using inverse weights given by the number of weekdays when we observe the decide, except that we also account for partial days.) We also use a single overall adjustment factor such that the sum of all device weights equals approximately 14 million, the number of individuals over 15 years old in the study area. To construct trips, we consider any individual-day with sufficient GPS data frequency and include all the trips that can be identified in the data on that day, and use the trip detection algorithm from Kreindler (2023).

Second, we construct a cross-section of typical (o, d) trip flows throughout the period. We apply the same procedure as for the panel, except that we first pool all data for the entire two year period.

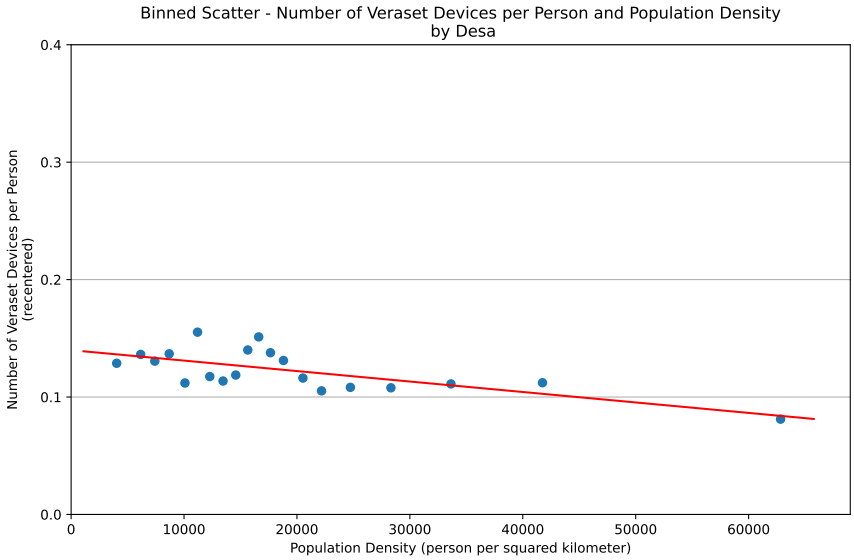
Figure A.20: Binned Correlation Between Veraset Devices and Residential Population



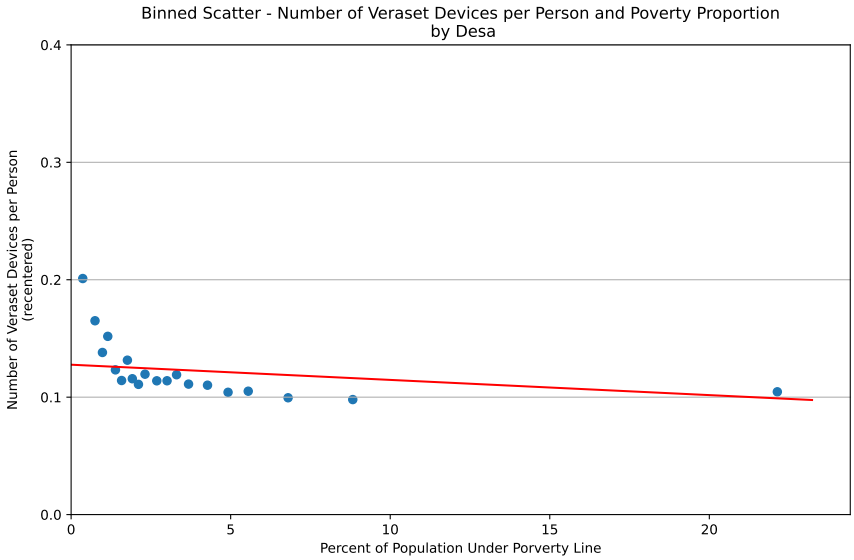
Notes: Each observation is an urban neighborhood (kelurahan or desa), $N = 538$. Population data is from the 2011 PODES survey, the most recent source for population at this level of geographical detail. For each desa, we count all devices which have their "home" location assigned inside the desa.

Figure A.21: Representativeness of the Veraset Trip Data

(a) Population Density



(b) Poverty



Notes: Binscatter plots analyzing the representativeness of the Veraset trip data at the kelurahan level. The sample excludes kelurahans below the 10th percentile of the distribution of PODES population. These locations tend to represent commercial or leisure areas. In these locations, the ratio of Veraset to PODES population is high. Panel A compares the ratio of Veraset home locations in a kelurahan to the kelurahan's population density. Panel B uses a poverty indicator from SMERU (2014).

A.4 Reduced Form Estimation

This section describes the differences-in-difference specifications.

For BRT events, we define $Post^{iB}_{odt}$, $i = 1, 2$, to be a time-varying dummy for the new direct connection between BRT grid cells o and d having been switched on in the past 10 months.[52] The coefficient α^{iB} in equation 1 captures the overall effect of a new direct connection being added between o and d in the first 10 months after its launch. The sample of observations is all odt that have a direct or transfer connection between BRT grid cells o and d, and o and d each have at least one treated observation.[53]

For non-BRT events, $Post^{iN}_{odt}$, $i = 1, 2$, is a time-varying dummy for the new direct connection between grid cells o and d having been switched on in the past 10 months. We restrict to origin grids o that are never BRT, o and d are connected either directly or by transfer, and o and d each have at least one treated observation. (In the latter case, the first leg is necessarily non-BRT, while the second leg can be BRT or non-BRT.)

In Event 3, grid-cell pairs o and d that are already directly connected get more busses from a new route because it overlaps with the existing route for the portion between o and d. Specifically, $Post^{3}_{odt}$ is a dummy for the first event of an additional direct route launched between o and d taking place, in the ten months before week t.

A positive coefficient α^{3N} captures the degree to which ridership or all trips between o and d increases after more busses are added to the route between o and d due to an additional direct route.[54] The sample is all odt such that BRT grid cells o and d are directly connected at time t, o and d each have at least one treated observation, and o and d are only connected by direct connections before the event.[55]

For non-BRT events, $Post^{3N}_{odt}$ is defined analogously. The sample is all odt that are connected directly at time t, o is never a BRT grid cell, o and d each have at least one treated observation, and o and d are only connected by direct routes before the event.

[52] We impose that the origin and destination grids have BRT stations, while the new route may be a non-BRT route traveling between these locations. It is often the case that non-BRT route travel along BRT corridors and stop at BRT stations for a portion of the route.

[53] This sample includes o, d pairs that are treated at some point, as well as o, d pairs that are never treated, yet there exist o', d' such that o, d' and o', d are treated.

[54] As for event types 1 and 2, the route itself can be non-BRT as long as it passes through BRT grid cells o and d.

[55] We make this last restriction to focus on the case where the change in the choice set is very simple. Results where we include o, d pairs that have both direct and transfer connections before the event are similar.

A.5 Attention Probabilities

We incorporate partial inattention by assuming that the agent notices each arrival from option k with independent probability p_k. In other words, with probability $1 - p_k$ the agent fails to notice the first arrival for option k. This leads to an "effective" arrival rate for option k of $\widetilde{\lambda}_k = \phi(p_k)\lambda_k$ with $0 \leq \phi(p) \leq 1$. $\phi(p)$ is given by $-\frac{p\log(p)}{1-p}$, which is increasing and concave in p, $\phi(0) = 0$, and $\phi(1) = 1$. To see this, note that the effective arrival rate is a weighted sum of the first arrival rates, the second arrival, etc. Dropping the k subscript, we can write:

$$\begin{aligned}\widetilde{\lambda} &= p\lambda + (1-p)p\frac{\lambda}{2} + (1-p)^2 p\frac{\lambda}{3} + ... \\ &= p\lambda(\sum_{k=0}^{\infty} k^{-1}(1-p)^k) \\ &= -\frac{p\log(p)}{1-p}\lambda.\end{aligned}$$

The expression on the second line is the Mercator series for $p - 1$.

A.6 Demand Model Derivations

Proof of Proposition 2. Here we derive expressions for the probability π_k to choose option k, and for expected utility $\mathbb{E}\max_k u_k$. We also derive expressions for expected travel time and expected wait time.

In general, assume that we have independent random variables $X_1, X_2, \ldots, X_N$, then

$$\Pr(k \in \arg\max_j X_j) = \int_{-\infty}^{\infty} f_k(x) \prod_{i \neq k} F_i(x) dx, \tag{9}$$

where f denotes a pdf function, and F denotes a cdf function. Expected utility is

$$\mathbb{E}\max_k X_k = \int_{-\infty}^{\infty} x \sum_k f_k(x) \prod_{i \neq k} F_i(x) dx. \tag{10}$$

In our model, have $u_k = v_k - \alpha_{\text{wait}} w_k$ where w_k is exponentially distributed with parameter λ_k. Assume that $\alpha_{\text{wait}} = 1$, if necessary replacing all λ_k by $\lambda_k/\alpha_{\text{wait}}$. The pdf and cdf for exponential variables are given by

- $F_k(u) = \exp(\lambda_k(u - v_k))$ for $u \leq v_k$ and 1 above, and
- $f_k(u) = \lambda_k \exp(\lambda_k(u - v_k))$ for $u \leq v_k$ and 0 above.

Choice Probabilities. Assume that options are ranked such that $v_1 < v_2 < \cdots < v_N$. Replacing the pdf and cdf in (9) and separating the integral by intervals delimited by the v_k's, the probability that option k is optimal is:

$$\begin{aligned}
\pi_k = \Pr(k \in \arg\max) &= \int_{-\infty}^{\infty} f_k(u) \prod_{i \neq k} F_i(u) du \\
&= \lambda_k \int_{-\infty}^{v_1} e^{\lambda_k (u - v_k)} \times \prod_{i \geq 1, i \neq k} e^{\lambda_i (u - v_i)} du \\
&+ \lambda_k \int_{v_1}^{v_2} e^{\lambda_k (u - v_k)} \times \prod_{i \geq 2, i \neq k} e^{\lambda_i (u - v_i)} du \\
&\cdots \\
&+ \lambda_k \int_{v_{k-1}}^{v_k} e^{\lambda_k (u - v_k)} \times \prod_{i > k} e^{\lambda_i (u - v_i)} du.
\end{aligned}$$

(For any $u > v_k$, the probability that k is optimal is zero.)

We use the notation: $\Lambda_i = \sum_{\ell \geq i} \lambda_\ell$ and $M_i = \sum_{\ell \geq i} \lambda_\ell v_\ell$. We have

$$\begin{aligned}
\lambda_k^{-1} \pi_k &= e^{-M_1} \int_{-\infty}^{v_1} e^{u \Lambda_1} du + \cdots + e^{-M_k} \int_{v_{k-1}}^{v_k} e^{u \Lambda_k} du \\
&= \sum_{i=1}^{k} e^{-M_i} \int_{v_{i-1}}^{v_i} e^{u \Lambda_i} du \\
&= \sum_{i=1}^{k} e^{-M_i} \frac{e^{v_i \Lambda_i} - e^{v_{i-1} \Lambda_i}}{\Lambda_i},
\end{aligned}$$

were we use the convention $v_0 = -\infty$.

Expected Utility. Plugging the exponential pdf and cdf formulae in (10) we get

$$\begin{aligned}
\mathbb{E} \max_k u_k &= \sum_{i=1}^{N} \int_{v_{i-1}}^{v_i} u \sum_{k \geq i} f_k(u) \prod_{j \geq i, j \neq k} F_j(u) du \\
&= \sum_{i=1}^{N} \int_{v_{i-1}}^{v_i} u \sum_{k \geq i} \lambda_k \exp\left(\sum_{j \geq i} \lambda_j (u - v_j) \right) du.
\end{aligned}$$

The exp(·) term can be factored out of the sum as it does not depend on k, so we get

$$\begin{aligned}
\mathbb{E}\max_k u_k &= \sum_{i=1}^{N} \int_{v_{i-1}}^{v_i} u\Lambda_i e^{u\Lambda_i - M_i} du \\
&= \sum_{i=1}^{N} \Lambda_i^{-1} e^{-M_i} \int_{v_{i-1}}^{v_i} u\Lambda_i e^{u\Lambda_i} d(u\Lambda_i) \\
&= \sum_{i=1}^{N} \Lambda_i^{-1} e^{-M_i} \left[e^{\Lambda_i v_i}(\Lambda_i v_i - 1) - e^{\Lambda_i v_{i-1}}(\Lambda_i v_{i-1} - 1) \right] \\
&= \underbrace{\sum_{i=1}^{N} e^{-M_i} \left[e^{\Lambda_i v_i} v_i - e^{\Lambda_i v_{i-1}} v_{i-1} \right]}_{v_N} - \underbrace{\sum_{i=1}^{N} \Lambda_i^{-1} e^{-M_i} \left[e^{\Lambda_i v_i} - e^{\Lambda_i v_{i-1}} \right]}_{\lambda_N^{-1}\pi_N}
\end{aligned}$$

The first sum is telescopic and evaluates to $e^{-M_N + \Lambda_N v_N} v_N = v_N$. This concludes the proof of part 2 in Proposition 2.

Expected Travel Time and Expected Wait Time. Travel time is non-random, so expected travel time is given by:

$$\mathbb{E}T_k^{\text{time}} \equiv \mathbb{E}\left(T_k^{\text{time}} \mid k \in \arg\max\right) = \sum_k \pi_k T_k^{\text{time}}. \tag{11}$$

By a similar argument, $\mathbb{E}v_k = \sum_k \pi_k v_k$. We use this result to derive expected wait time:

$$\begin{aligned}
\mathbb{E}u_k &= \mathbb{E}v_k - \alpha_{\text{wait}} \mathbb{E}T_k^{\text{wait}} \\
v_N - \pi_N \alpha_{\text{wait}}/\lambda_N &= \sum_k \pi_k v_k - \alpha_{\text{wait}} \mathbb{E}T_k^{\text{wait}} \\
\Rightarrow \mathbb{E}T_k^{\text{wait}} &= \pi_N \lambda_N^{-1} - \alpha_{\text{wait}}^{-1} \left(\sum_k \pi_k (v_N - v_k) \right).
\end{aligned}$$

A.7 Optimal Network Design

A.7.1 Optimization Environment: Predicted Bus Travel Times

In this section, we describe how we estimate bus travel times for every edge in the grid cell environment, in a manner that makes these edge costs consistent with our entire GPS data on bus travel times. The key challenge is that certain origin-destination pairs are not connected by bus in the current network.

We proceed in two steps. First, we use data that we collected on driving times between

every pair of grid cells, and we project bus log travel time – for those origin-destination-route combinations available – onto log driving time for that origin-destination pair. Second, we estimate edge-specific bus travel times that "micro-found" these predicted bus travel times. We estimate edge times in the routing model from Allen and Arkolakis (2022), which allows us to express the predicted bus travel time between any origin and destination as the noisy shortest route in our grid cell environment network. We repeat this exercise separately for BRT and non-BRT.

Step 1. Predicted bus travel time for all origin-destination pairs. We obtain driving travel times between tens of thousands of pairs of locations in Jakarta from a commercial provider of route data derived from smartphones and other GPS-enabled devices. We obtained data for the entire Jakarta region in the year 2020.[56] We then spatially interpolate this data to construct driving time T_{od}^{drive} between every origin and every destination.

We then predict bus travel times between all (o, d) pairs. We start with our data on median bus travel times T_{odr}^{bus} between o and d on route r (only for o, d pairs where this data is available). We then estimate the following linear regression

$$\log(T_{odr}^{\text{bus}}) = \alpha_0 + \alpha_1 \log(T_{od}^{\text{drive}}) + \epsilon_{odr}.$$

We then construct the prediction $\widehat{T}_{od}^{\text{bus}}$ for all origin-destination pairs, separately for BRT and non-BRT bus travel times.

Step 2. Estimate edge-level bus travel times. In our counterfactual simulations, we need to predict bus travel time for routes following any path. The model primites are *edge-specific* bus travel times, which we sum up over any possible route. We now describe how we estimate edge-specific bus travel times using our data on predicted bus travel times for all origin-destination pairs.

Given that we do not have specific bus routes for $\widehat{T}_{od}^{\text{bus}}$, we model these travel times as noisy shortest routes over the underlying network (Allen and Arkolakis, 2022).

For every edge ij in the grid cell network – this includes horizontal, vertical and diagonal neighboring grids – denote by T_{ij} the bus travel time parameter in the model. We denote by p a path between o and d, and by T_p the sum of edge-wise travel times along the path. Assume that a bus commuter traveling from o to d can choose any finite path between o and d and faces realized travel time $\widetilde{T}_p = T_p + \epsilon_p$, where ϵ_p is an iid Gumbel-distributed idiosyncratic shock with parameter ν. These shocks capture in a stylized way the fact that bus routes

[56] Through additional partial city coverage from earlier years, we confirmed that this data captures pre-COVID traffic congestion patterns and that aggregate congestion appears flat since 2016.

between o and d do not always follow the shortest path, with small ν corresponding to larger deviations. The commuter selects the path with the shortest realized travel time. By properties of multinomial logit, the expected bus travel time between o and d is

$$\widetilde{T}_{od} \equiv \mathbb{E} \max_p \widetilde{T}_p = -\nu^{-1} \log \Big(\sum_p \exp(-\nu T_p) \Big)$$

Allen and Arkolakis (2022) prove that the matrix $B = (\exp(-\nu \widetilde{T}_{od}))_{od}$ is given by $B = (I - A)^{-1}$ where $A = (\exp(-\nu T_{ij}))_{ij}$ is the travel time adjacency matrix, which has entries equal to zero whenever ij is not an edge.

We next estimate T_{ij} for all ij edges, as well as ν, using data on $\widehat{T}^{\text{bus}}_{od}$. We minimize the following objective function

$$\min_{\nu, (T_{ij})_{ij}} \sum_{o,d} \psi_{od} \Big(\log(\widehat{T}^{\text{bus}}_{od}) - \log(\widetilde{T}_{od}) \Big)^2$$

using weights ψ_{od} equal to the inverse squared distance between grid cells o and d. We estimate $\widehat{\nu} = 118$ for BRT and $\widehat{\nu} = 116$ for non-BRT, suggesting that bus routes are best explained as nearly shortest-route paths. (After this point, we no longer use ν.) We use the resulting $\widehat{T}_{ij}$ for BRT and non-BRT as bus travel time in all our counterfactual network exercises.

Figure A.22 prints the empirical fit of this exercise.

A.7.2 Ridership Equilibrium With Bus Capacity Penalties

Here we set up the formal model where commuters incur time penalties when bus occupancy exceeds bus capacity. Consider a route r and two consecutive stations s and s' on r. The route-edge occupancy level $B_{ss'r}$ is given by the expected number of passengers on a bus on route r between s and s'

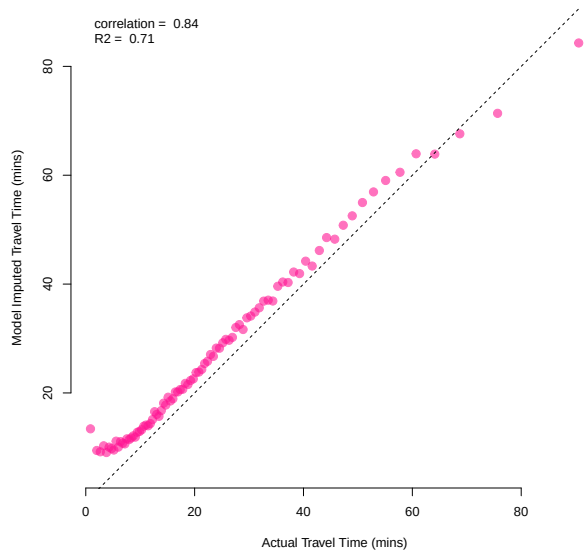
$$B_{ss'r} = \frac{1}{D_r} \sum_{o \leq s < s' \leq d} R_{odr}, \tag{12}$$

where R_{odr} is the daily number of commuters traveling on route r between any pair of stations o and d, and D_r is the daily number of bus departures on route r. The sum is over all stations o on route r that precedes s (in the direction from s to s'), and all d that come after s'.

We assume that the perceived travel time on route r and edge ss' is given by

$$T^p_{ss'r} = T_{ss'r} \cdot \kappa(B_{ss'r}), \tag{13}$$

Figure A.22: Bus Travel Time Model Fit



Notes: This binscatter plot assesses GMM model fit. It correlates our data on bus travel times T^{bus}_{odr} at the origin-destination-route level, restricting to origin-destination pairs for which this data is available, on the x-axis, with model-predicted expected bus travel time $\widetilde{T}_{od}$ on the y-axis.

where $T_{ss'r}$ is the baseline (objective) travel time, and $\kappa(B)$ is a function equal to 1 when bus occupancy is less than bus capacity, $B \leq \overline{B}$, and convex and increasing for $B > \overline{B}$. We parameterize κ as

$$\kappa(B) = \max\left(1, \quad \exp\left(0.2 \cdot (B - 84)\right)\right). \tag{14}$$

We assume there is no penalty up to $\overline{B} = 84$ persons per bus (the average bus capacity in TransJakarta's fleet at the end of our study period), and the penalty increases rapidly afterward. Perceived travel time with 90 people on the bus is already more than three times as high as objective travel time.

We define the perceived travel time between any two stations o, d on route r as the sum of edge-wise perceived travel times along that route, i.e. $T^p_{odr} = \sum_{k=1}^{n} T^p_{s_k s_{k+1} r}$ where $s_1 = o$ and $s_n = d$. As commuters get on and off, crowding on a given route fluctuates at different points on the route. Our model assumes that commuter disutility is additive over route segments, based on the crowding level at each point.

A ridership equilibrium is defined by a set of crowding levels (for each route r and each pair of consecutive stations s, s') such that optimal commuter choices lead to ridership patterns and a set of crowding levels that is exactly the assumed one. That is, it involves a

fixed point, whereby crowding $B_{ss'r}$ is determined by equation (12) based on route-specific ridership flows R_{odr}, and ridership patterns R_{odr} are given by choice probabilities based on equations (2), (3) and (4), where we replace travel time T_{odr} with perceived travel time T^p_{odr} given by (13), which depends on $B_{ss'r}$.

We compute a ridership equilibrium by iterating until we reach a fixed point. For computational reasons, in our simulated annealing algorithm, we approximate the equilibrium by a single one-step updating rule. We compute route choices and ridership assuming no penalties, then compute occupancy levels and re-compute ridership once more. This allocation is very close to the equilibrium. For the 200 bus networks we obtain from the simulated annealing algorithm, we compare the one-step allocation with the fixed-point equilibrium. 170 of the 200 networks are already at equilibrium because all edge-occupancy levels are below $\overline{B}$. Even when they differ, the maximum deviation is negligible, equivalent to less than a second of wait time for every original TransJakarta passenger. The maximum ridership difference is 0.03% of daily bus ridership.

A.7.3 Analytic Results for Optimal Allocations Model

Formula for Local Comparative Statics. Here we derive the expression for the derivative of the expected optimal property $f^*(\theta)$ with respect to a scalar θ, defined in equation (7).

Using the logit formula we can show that

$$D_\theta \pi(N,\theta) = \beta \pi(N,\theta)\Big(D_\theta W(N,\theta) - \mathbb{E}_{N'} D_\theta W(N',\theta)\Big) \tag{15}$$

The term in parentheses says that when θ changes, the probability of a given network N increases if and only if the derivative of welfare W with respect to θ at N is higher than its expectation over all possible networks. Using this expression and rearranging sums several

times we get

$$
\begin{aligned}
D_\theta f^*(\theta) &= \sum_N D_\theta \pi(N,\theta) f(N,\theta) + \pi(N,\theta) D_\theta f(N,\theta) \\
&= \sum_N \beta \pi(N,\theta) \Big(D_\theta W(N,\theta) - \mathbb{E}_{N'} D_\theta W(N',\theta) \Big) f(N,\theta) + \pi(N,\theta) D_\theta f(N,\theta) \\
&= \Big[\sum_N \beta \pi(N,\theta) D_\theta W(N,\theta) f(N,\theta) \Big] - \beta \Big[\mathbb{E}_{N'} D_\theta W(N',\theta) \Big] \Big[\underbrace{\sum_N \beta \pi(N,\theta) f(N,\theta)}_{f^*(\theta)} \Big] + \\
&\qquad + \Big[\sum_N \pi(N,\theta) D_\theta f(N,\theta) \Big] \\
&= \sum_N \beta \pi(N,\theta) D_\theta W(N,\theta) f(N,\theta) - \beta \Big[\sum_{N'} \pi(N',\theta) D_\theta W(N',\theta) \Big] f^*(\theta) + \sum_N \pi(N,\theta) D_\theta f(N,\theta) \\
&= \sum_N \pi(N,\theta) \Big[\beta D_\theta W(N,\theta) \Big[f(N,\theta) - f^*(\theta) \Big] + D_\theta f(N,\theta) \Big].
\end{aligned}
$$

A.7.4 The Simulated Annealing (SA) Algorithm Asymptotic Result

This section reports the sketch of the proof of Proposition 3. The proof uses standard Metropolist-Hastings algorithm arguments. It mirrors a classic proof that as the number of steps increases, the outcome of SA converges to the global optimum.

The key observation is that replacing the variable inverse temperature β_k with a fixed temperature β in the acceptance probability (8) yields the Metropolis-Hastings (MH) algorithm. Because Ψ is irreducible and aperiodic, MH is an irreducible and aperiodic stationary Markov chain. Its stationary distribution at N is proportional to $\exp(\beta W(N))$ and hence is exactly π. This follows from checking that the *detailed balance* condition holds for any N, N' such that $\Psi(N' \mid N) > 0$:

$$
\pi(N) \Pr(N' \mid N) = \pi(N') \Pr(N \mid N').
$$

Returning to the SA algorithm, as the number of steps grows, the algorithm can be approximated by a sequence of MH algorithms with increasing inverse temperatures. In particular, the number of steps with an inverse temperature nearly equal to β grows to infinity, and hence the endpoint of the SA algorithm is asymptotically distributed according to π. (These arguments can be made precise, see Nikolaev and Jacobson (2010).)

A.7.5 The Candidate Network Proposal Function Ψ

How to produce new candidate networks is critical for the success of the SA algorithm. Starting from a network N_k, we obtain a proposed network N' by applying one of the "modifier" operations described below. We first select one of the four categories with equal probability, then a type of modifier within the category with equal probability, and finally, we generate a network N' according to the selected modifier, uniformly randomly.

Local Bus Modifiers

1. Exchange one bus between two randomly drawn bus routes.
2. Exchange a randomly drawn 10% of busses between two randomly drawn bus routes.

Global Bus Modifiers

1. Give or take away busses from a randomly selected bus route. Redistribute busses among randomly chosen other lines to stay at the constraint of 1,500 total busses.

Local Route Modifiers

1. Draw a bus route at random at add one random new adjacent stop to one end of the route.
2. Draw a bus route at random and take away one random stop at one end of the route.
3. Draw a bus route at random, pick two locations A and B on the route and "straighten" the route by replacing the intermediate stops by the shortest path between A and B.
4. Draw a bus route at random and add a detour to it. Pick two stops on the bus route, pick one new location on the map at random and let the route go between the two stops through the new location.

Global Route Modifiers

1. Create a random new bus route. Pick two locations at random and create a bus route on the shortest path between these locations. Assign a random number of busses to the new route, redistributed from randomly chosen other bus routes.
2. Delete a randomly drawn route and replace it with a random new bus route: pick two locations at random and create a bus route on the shortest path between these locations. Assign a random number of busses to the new route, redistributed from randomly chosen other bus routes.
3. Delete a randomly drawn existing bus route, where routes with less ridership are more likely to be chosen. Redistribute the busses among other, randomly chosen, bus routes.
4. Delete a randomly chosen set of 10 bus routes at once. Redistribute the busses among other, randomly chosen, bus routes.
5. Add 10 random bus lines at once. Assign a random number of busses to the new routes, redistributed from randomly chosen other bus routes.

Approximating Proposal Probabilities Ratio In our implementation of the simulated annealing algorithm, we use the following approximation when using equation (8)

$$\frac{\Psi(N_k \mid N')}{\Psi(N' \mid N_k)} = 1.$$

We make this approximation because computing the proposal function ratio term precisely is computationally intensive. $\Psi(N' \mid N_k)$ is the probability that network N' is proposed when starting from network N_k. To compute this, we need to enumerate all possible modifications starting from N_k. (Given that we stratify, we only need to do this within each category and type of modification.)

For most modifiers described above, this ratio is likely to be close to 1. For example, this ratio is approximately equal to 1 for the proposal function that exchanges a bus between two randomly drawn bus routes. There are two types of exceptions. First, certain modifiers that we use are not reversible. Second, for the modifiers that add or delete routes, the ratio is systematically far from 1. We argue that approximating the ratio with 1 will lead our algorithm to converge to a stationary distribution that puts more weight on expansive networks. This would only strengthen our main results that optimal networks are more expansive than the current TransJakarta network.

To see this more clearly, denote by R the (very large) number of possible routes in our square grid cell environment, and by $r(N)$ the number of routes in network N. For ease of exposition, assume that only modifiers that add or remove one route are allowed, and assume that N' is obtained from N_k by adding a randomly chosen route. Then the proposal function ratio is given by

$$\frac{\Psi(N_k \mid N')}{\Psi(N' \mid N_k)} = \frac{\frac{1}{r(N_k)+1}}{\frac{1}{R-r(N_k)}} = \frac{R - r(N_k)}{r(N_k) + 1}.$$

This is significantly larger than 1, meaning that transitions towards networks with many routes is favored in equation (8). It easy to prove that if we use the ratio equal to 1, the Markov chain will converge to a modified stationary distribution given by

$$\widetilde{\pi}(N) \propto \pi(N) \cdot \binom{R}{r(N) + 1}^{-1}$$

In other words, the modified stationary distribution puts less weight on expansive networks. Note that this adjustment effectively compensates for the very large number of expansive networks, which is of the same magnitude as the adjustment factor.