WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Road Travel Time Saved for Emergency Vehicles: A Journey-Level Analysis of GRU Predictive Routing

Freshia Njoki Macharia[1]*, Gilbert Langat[2], Stephen Mageto[2]

[1]Department of Computer Science and Informatics, Kirinyaga University, Kerugoya, Kenya
[2]School of Pure and Applied Sciences, Kirinyaga University, Kerugoya, Kenya
**Corresponding author: fnjokimacharia@gmail.com | Co-authors: glangat@kyu.ac.ke; snyakiba@kyu.ac.ke*

ABSTRACT :

A published emergency-routing framework that joins a gated recurrent unit forecast, time-dependent A*, and a remaining-time threshold reported mean travel-time reductions of 16.19% under a simulated incident, 1.03% in peak hour, and 0.01% off-peak (Macharia et al., 2026). Those means were taken across matched journeys and, in the regime tables, across four threshold settings on the same origin–destination pairs. This article keeps that framework fixed and reports, at the operational threshold δ = 0.20, how road travel time is distributed across 75 journeys in each regime. A journey counts as a win or a loss only when realised time differs from snapshot Dijkstra by more than five seconds. Under the incident regime the framework was faster on 71 of 75 journeys, slower on none, and tied on four. The median saving was 83.1 seconds and the mean was 127.7 seconds (paired d = 1.22; 95% bootstrap interval 104.7 to 153.0 seconds). Fifty-two journeys saved at least one minute, and 28 saved at least two. The time-weighted reduction was 17.6%. Short incident journeys saved 64.4 seconds on average and long ones saved 220.5 seconds (r = 0.77). Ranked by seconds saved, 17 of the 75 incident journeys account for half of the road time removed. In peak hour the median saving was zero seconds: 22 journeys were faster, 46 stayed within five seconds, and 7 were slower, one of them by 68 seconds. Off-peak removed 10.7 seconds in total across 75 journeys. Against a reactive snapshot, the framework was faster on 11 incident journeys, tied on 60, and slower on 4; the median advantage was zero seconds and the mean advantage was 4.0 seconds. Raising δ from 0.05 to 0.20 left the incident win count at 71 and moved the 90th percentile of seconds saved from 282.5 to 263.2. The published 16.19% describes the incident journeys. Peak-hour and off-peak medians are zero.

Keywords: emergency vehicle routing; journey-level evaluation; practical significance; travel-time distribution; predictive routing; heterogeneity

1. Introduction

Emergency dispatch is judged one call at a time. A mean travel-time reduction can be real and still be the wrong quantity to brief a dispatcher, because the mean can be carried by a minority of long journeys while the typical journey is unchanged. Macharia et al. (2026) designed a decision-support loop that forecasts speeds with a gated recurrent unit, routes with time-dependent A*, and offers a new path only when remaining travel time has deteriorated past a threshold δ. On 900 matched journeys scored with actual future speeds, that paper reported mean reductions versus current-map Dijkstra of 5.74% overall and 16.19% when a corridor incident was injected, with 1.03% in peak hour (p = .0002) and 0.01% off-peak (p = .90). The same paper reported a 0.61% margin over a reactive A* baseline, a 1.69% gap behind an oracle that planned on actual future speeds, and no travel-time difference among $\delta \in \{0.05, 0.10, 0.15, 0.20\}$. It recommended δ = 0.20.

Those figures answer whether the loop works on average. They do not answer how often it works, by how many seconds, on which lengths of trip, or whether the incident gain survives once a dispatcher is allowed to replan from the slowed snapshot without any forecast. The published regime tables also pooled four threshold settings (N = 300 per regime). The source paper argued that the pool was legitimate because mean travel time did not depend on δ. Pooling still repeats the same origin–destination pairs, so it is the wrong sample for a journey-level claim. This article therefore treats δ = 0.20, the published operational default, as the journey a dispatcher would actually run, and it analyses the 75 pairs in each regime once.

The contribution is the journey-level distribution of road time saved: how many journeys win, the median seconds saved, the role of trip length, the share held by a minority of corridors, and the seconds that remain after a reactive replan. Prediction error and inference latency are reported in Macharia et al. (2026).

Five questions organise the analysis. First, at δ = 0.20, on how many journeys is the framework practically faster than snapshot Dijkstra, tied with it, or slower, if a difference of five seconds or less is treated as a tie? Second, how many seconds are saved, and does the mean of percentages agree with the time-weighted reduction? Third, do longer journeys account for the saving? Fourth, is the incident saving spread across origin–destination pairs or held by a few corridors, and do journeys that never replan still save time at departure? Fifth, once a reactive snapshot is allowed, and once δ is varied, what part of the published incident result remains?

2. Literature Review

Macharia et al. (2026) specify the predictor, the search procedure, the graph, and the controller. The points required for the tables below are these. The routing graph is the METR-LA detector adjacency: 207 nodes and 1,515 directed edges, each carrying a sensor speed (Jiang & Luo, 2022; Macharia et al., 2026). Seventy-five origin–destination pairs were fixed. Each pair was run in three regimes. Peak hour used the slowest quartile of test-window mean speed. Off-peak used the fastest quartile. The incident regime placed a 40% speed drop on the planned Dijkstra corridor after 20% of planned Dijkstra time, so the vehicle met the disruption. Four methods chose a path: snapshot Dijkstra (B1), A* on historical mean speeds (B2), a reactive A* allowed one replan from the current snapshot once the incident was visible and with no 30-minute forecast (B3), and an oracle A* that planned on actual future speeds (B4). The framework planned on the forecast, fused with the speeds observed so far, and could replan when remaining time grew by more than δ. Every path was then driven on actual future speeds. Predicted costs chose the route. They did not score it.

The source evaluation crossed the 75 pairs with three regimes and four thresholds, giving 900 rows. This article analyses that archive at journey level. Random seed, incident placement, and the model checkpoint are those reported by Macharia et al. (2026).

Surveys since 2022 still report speed forecasting apart from the route that is driven (Jiang & Luo, 2022; Qi et al., 2025). Emergency-vehicle studies in the same period attach a forecast to signal priority rather than to the road time of each journey (Chowdhury et al., 2023; Feng et al., 2024). This article measures how the published travel-time reduction is spread across the 75 journeys in each regime.

3. Methodology

3.1 Journey archive

The input is the simulation file results/simulation/simulation_results.csv in the public repository that accompanies Macharia et al. (2026) (https://github.com/Freshia-Njoki/Emergency_Routing). The file contains 900 rows: 75 origin–destination pairs, three regimes, and $\delta \in \{0.05, 0.10, 0.15, 0.20\}$. Each row records realised travel time for the framework and for B1–B4, the percentage reduction versus Dijkstra, and the number of framework replans. No network was retrained for this article, and no journey was resimulated.

3.2 Unit of analysis

Unless a threshold comparison is explicitly stated, every result uses $\delta = 0.20$ only. That choice follows the source paper's operational default and yields 75 journeys per regime. The same pair is not counted four times. Threshold comparisons in Section 4.7 return to the incident rows at all four values of δ, still with 75 pairs at each value, and they are described as a sensitivity of the tail rather than as a larger sample.

3.3 Practical tie, effect size, and uncertainty

Seconds saved equal Dijkstra time minus framework time. A positive value means the framework arrived sooner. A journey is a win if it saved more than five seconds, a loss if it lost more than five seconds, and a tie otherwise. Five seconds is a practical band for a dispatch recommendation, not a clinical minimal important difference. It is declared before the counts are interpreted. The mean of journey-level percentages is reported beside the time-weighted reduction, which is total seconds saved divided by total Dijkstra time. These two percentages answer different questions and are not interchangeable.

The paired effect size is the mean within-journey time difference divided by the standard deviation of those differences, the paired form of Cohen's d (Liu, 2023). The interval for the mean seconds saved is a percentile bootstrap from 5,000 resamples of the 75 journeys, with seed 7 (Kang et al., 2023).

3.4 Outcomes estimated in this article

Prediction error, graph construction, and module latency are taken from Macharia et al. (2026). The outcomes estimated here are journey-level wins, ties, losses, seconds saved, and the residual against a reactive replan. The archive contains no personal data and no named emergency call. The ethics position is that of Macharia et al. (2026): public detector speeds, simulated pairs, and no live ambulance dispatched from these tables.

4. Results

4.1 Wins, ties, and losses

Table 1 and Figure 1 classify the 225 operational journeys. Under incidents the framework was faster by more than five seconds on 71 journeys and slower on none. The remaining 4 journeys were exact ties. The sign test on 71 positive differences and 0 negative differences gives $p < .001$. That count is the journey-level content of the published 16.19% incident mean: the gain is present on almost every incident journey, not on a handful of outliers that drag an otherwise null set.

Peak hour is a different distribution. The median saving was 0.0 seconds. 46 journeys stayed within five seconds, 22 were faster, and 7 were slower.

The worst peak-hour journey was 68 seconds slower than Dijkstra. A sign test on the 32 gains and 15 losses that exceeded 0.05 seconds is p = .019, and the paired effect size is only d = 0.27. Statistical direction and practical size part company. The published 1.03% peak-hour mean is compatible with these counts. It should not be briefed as a saving the typical peak-hour call will feel.

Off-peak is a negative control that survives the journey-level cut. 71 of 75 journeys stayed within five seconds. Three were faster by more than five seconds, and one was slower, by 40 seconds. The sign test is p = .002 because tiny positive differences (21) outnumber tiny negative differences (5). The median is still zero and the mean is 0.1 seconds. A significant sign test is not an operational result. The published non-significant mean of 0.01% remains the right summary for off-peak, and the one 40-second loss shows that a forecast is not automatically harmless.

Table 1. Journey outcomes versus snapshot Dijkstra at δ = 0.20 (N = 75 per regime).

Quantity	Incident	Peak hour	Off-peak
Wins (> 5 s faster)	71	22	3
Ties (within 5 s)	4	46	71
Losses (> 5 s slower)	0	7	1
Median seconds saved	83.1	0.0	0.0
Mean seconds saved	127.7	7.7	0.1
95% bootstrap interval	104.7 to 153.0	1.6 to 14.7	-1.3 to 1.3
Paired Cohen's d	1.22	0.27	0.03
Mean of percentages	16.1%	1.2%	0.01%
Time-weighted reduction	17.6%	1.5%	0.04%
Total seconds saved	9580.3	575.9	10.7
Journeys saving ≥ 60 s	52	6	0
Smallest saving (seconds)	0.0	-68.4	-40.1

Note. The bootstrap interval is the 2.5th and 97.5th percentiles of 5,000 resamples of the mean (seed 7). Time-weighted reduction is total seconds saved divided by total Dijkstra time. The published pooled means were 16.19% (incident), 1.03% (peak hour), and 0.01% (off-peak) at N = 300.

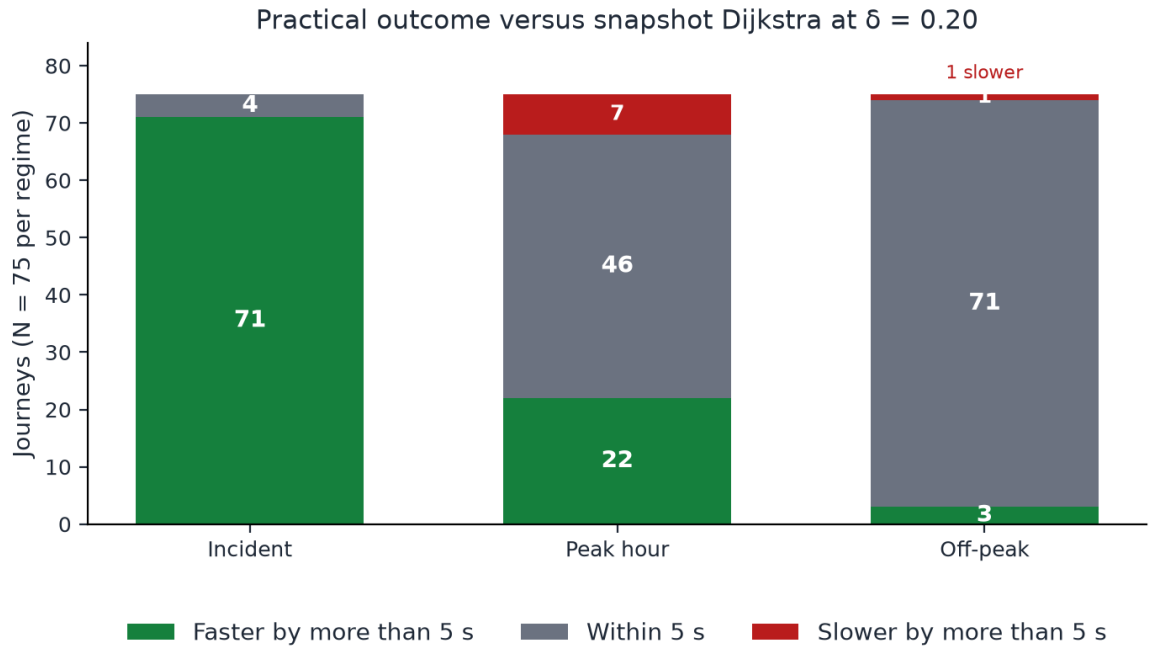


Figure 1. Practical wins, ties, and losses versus snapshot Dijkstra at δ = 0.20. Each bar is 75 journeys.

4.2 Seconds, and why two percentages disagree

Figure 2 shows the shape behind Table 1. Incident savings are large and right-skewed: the 10th percentile is 30.2 seconds, the median 83.1, and the 90th percentile 263.2. The largest single saving was 420.1 seconds. Across the 75 incident journeys the framework removed 159.7 minutes of road travel time relative to Dijkstra, which is 127.7 seconds, or 2.1 minutes, per journey. The mean of the 75 percentages is 16.1%. The time-weighted reduction is 17.6%. The second number is larger because the journeys that last longer also save a greater fraction of their time. Quoting only 16%

slightly understates the share of incident Dijkstra-time removed in this archive, and it hides that the typical journey saves about 83.1 seconds rather than the mean of 127.7.

Peak hour removed 9.6 minutes of road travel time in total across 75 journeys (mean 7.7 seconds; 95% interval 1.6 to 14.7). Ten peak-hour journeys saved at least 30 seconds, and six saved at least a minute. Off-peak removed 10.7 seconds in total. Its bootstrap interval (-1.3 to 1.3 seconds) includes zero. The interval for incidents (104.7 to 153.0 seconds) does not. An overall mean that mixes the three regimes, such as the published 5.74%, averages a two-minute incident effect with a zero off-peak effect. It is not an expected saving for a call whose regime is unknown.

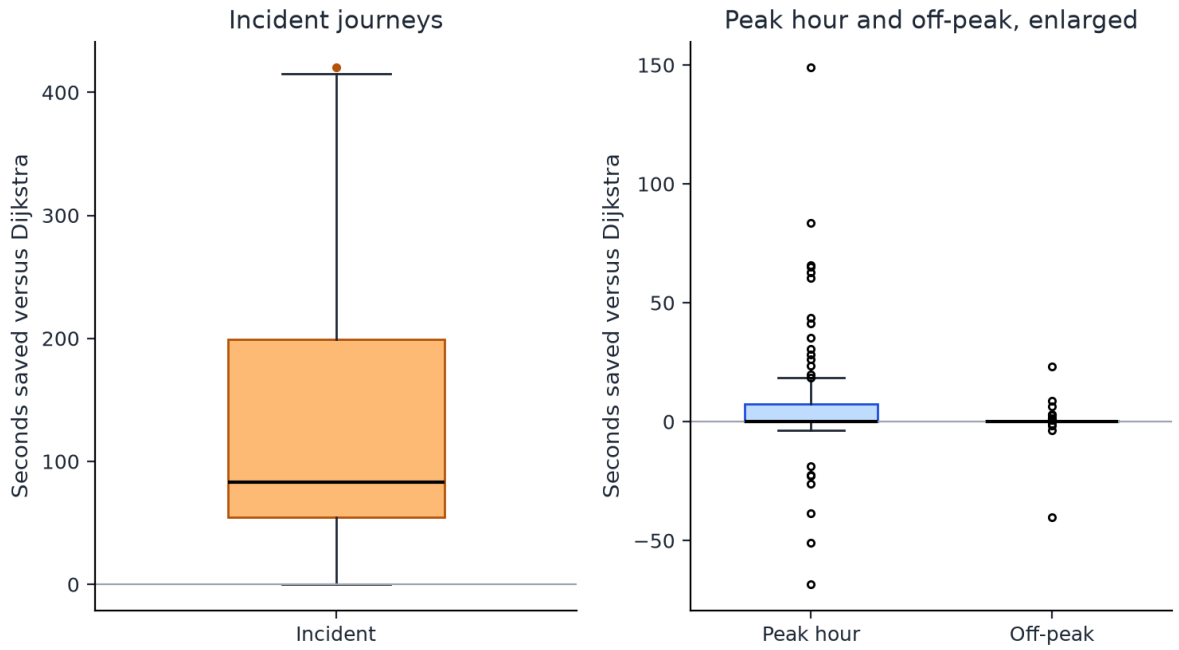


Figure 2. Seconds saved versus snapshot Dijkstra at δ = 0.20. The right panel enlarges peak hour and off-peak so that losses of tens of seconds remain visible. The horizontal line is zero.

4.3 Journey length

Table 2 and Figures 3 and 4 split each regime into tertiles of Dijkstra time. Under incidents, short journeys (Dijkstra 424–644 s) saved 64.4 seconds on average (11.0%; 22 of 25 wins). Middle journeys saved 98.3 seconds (14.2%; 25 of 25). Long journeys (Dijkstra 747–1282 s) saved 220.5 seconds (23.2%; 24 of 25). Pearson's r between Dijkstra time and seconds saved was 0.77 ($n = 75$, $p < .001$). Length does not merely add a constant number of seconds. The percentage itself rises from 11.0% on short incident trips to 23.2% on long ones.

The same gradient appears, much smaller, in peak hour. Short peak-hour journeys saved 0.3 seconds on average (4 wins). Long peak-hour journeys saved 19.6 seconds (13 of 25 wins; median 5.3 seconds). The published peak-hour mean is therefore carried by the longer trips. Off-peak tertiles stay within about one second of zero (short -1.2 s, middle 0.6 s, long 1.0 s). Length is not a reason to expect a saving when the road is already fast.

Table 2. Mean seconds saved by journey-length tertile at δ = 0.20 (N = 25 per cell).

Group	Dijkstra time (s)	Mean seconds	Median seconds	Mean %	Wins
Incident, short	424–644	64.4	67.1	11.0%	22/25
Incident, middle	650–744	98.3	70.5	14.2%	25/25
Incident, long	747–1282	220.5	240.5	23.2%	24/25
Peak hour, short	315–447	0.3	0.0	0.1%	4/25
Peak hour, middle	454–565	3.2	0.0	0.5%	5/25
Peak hour, long	568–768	19.6	5.3	2.9%	13/25
Off-peak, short	278–370	-1.2	0.0	-0.3%	0/25
Off-peak, middle	374–431	0.6	0.0	0.1%	1/25
Off-peak, long	432–616	1.0	0.0	0.2%	2/25

Note. Tertiles are equal counts after sorting each regime by realised Dijkstra time. Wins use the five-second band.

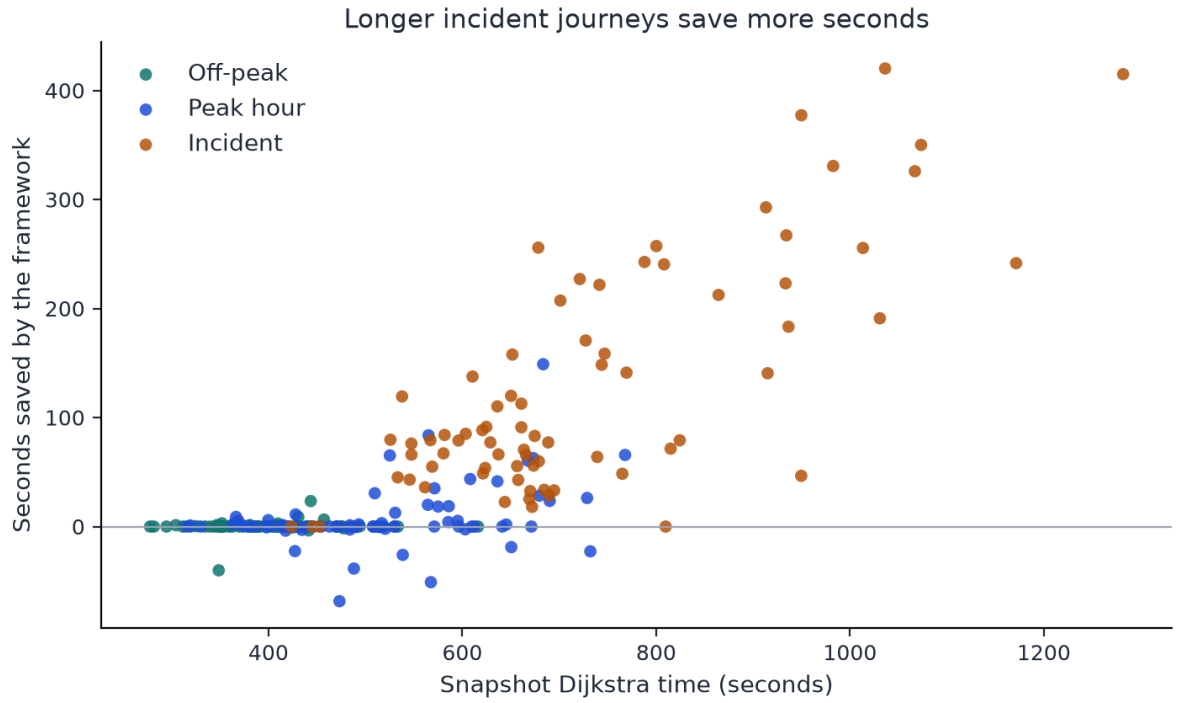


Figure 3. Seconds saved against snapshot Dijkstra time at δ = 0.20. Each point is one journey.

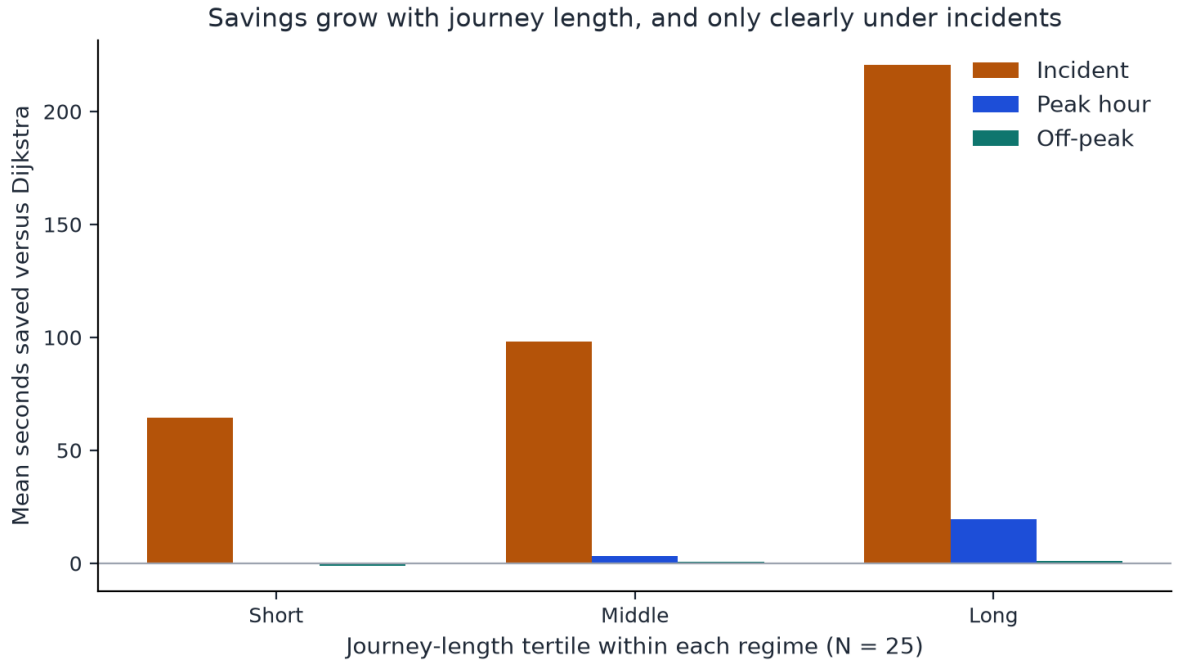


Figure 4. Mean seconds saved by length tertile and regime at δ = 0.20.

4.4 Concentration across pairs

The incident saving is widespread and uneven. No incident journey lost time, so the positive mass equals the total, 159.7 minutes of road travel time. Ranked from the largest saving to the smallest, 17 of the 75 journeys account for half of that road time, and 37 account for 80% (Figure 5). The result is not a single corridor. It is also not an equal gift to every pair. Table 3 lists the five largest and five smallest incident outcomes. The largest, pair 89→29, saved 420.1 seconds on a Dijkstra time of 1036 seconds (41%) with one replan. Four incident journeys saved nothing: the framework and Dijkstra took the same time. Three of those ties were short (Dijkstra 424–453 s). Pair 152→181 was not short — its Dijkstra time was 810 seconds — and it still saved nothing. A long trip is not guaranteed a saving when the snapshot path is already as fast as the forecast path.

Table 3. Largest and smallest incident savings at δ = 0.20.

Pair	Dijkstra (s)	Framework (s)	Seconds saved	Percent	Replans
89→29	1036	616	420.1	40.5	1
34→78	1282	867	414.9	32.4	1
172→95	950	573	377.2	39.7	1
2→185	1073	723	350.0	32.6	1
66→34	983	652	330.7	33.7	0
123→34	672	654	18.1	2.7	1
75→171	445	445	0.0	0.0	0
184→31	424	424	0.0	0.0	0
174→21	453	453	0.0	0.0	0
152→181	810	810	0.0	0.0	0

Note. Sensor identifiers are METR-LA node ids. The first five rows are the largest savings; the last five are the smallest.

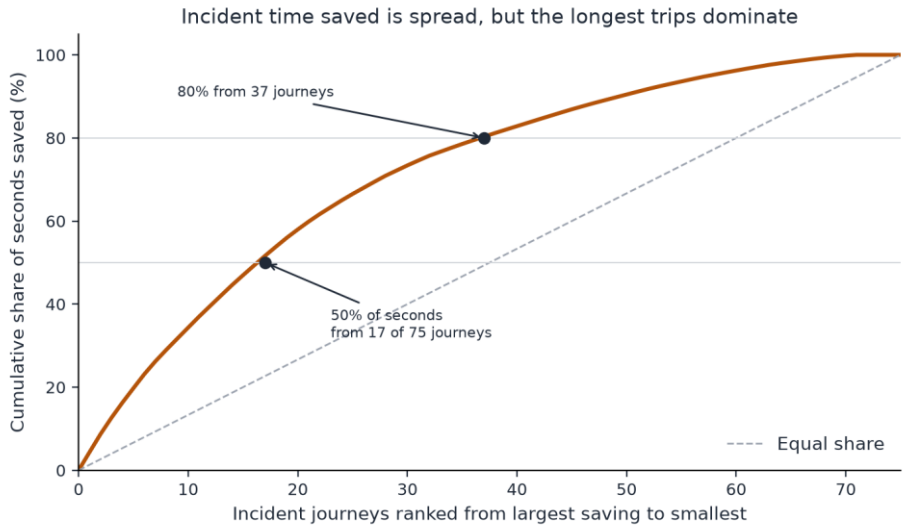


Figure 5. Cumulative share of incident seconds saved when journeys are ranked from the largest saving to the smallest.

4.5 Replanning and the departure forecast

At δ = 0.20, 68 of 75 incident journeys replanned at least once (mean 0.99 replans). Peak hour replanned on 10 journeys and off-peak on 3. Table 4 splits incident savings by replan count. The association between replan count and seconds saved is weak (r = 0.17, p = .141). One replan is the modal incident journey (n = 62, median saving 81.4 s). Six journeys replanned twice (median 185.5 s).

Seven incident journeys never replanned. Their median saving was 0.0 seconds, so the typical no-replan journey tied Dijkstra. Their mean was 91.2 seconds because three of them saved a large amount at departure and never needed an update: pair 66→34 saved 331 seconds, pair 196→86 saved 242 seconds, pair 130→17 saved 66 seconds. On those three calls the forecast changed the path before the vehicle moved. The published account correctly says that incident journeys replanned about once. It does not show that a minority of the incident road time was saved at dispatch, before any replan, with a replan count of zero. Both facts are in the archive.

Table 4. Incident seconds saved by number of replans at δ = 0.20.

Replans	Journeys	Mean seconds	Median seconds	Wins (> 5 s)
0	7	91.2	0.0	3/7
1	62	127.0	81.4	62/62
2	6	178.0	185.5	6/6

Note. The mean at zero replans is pulled by three departure-time savings. The median at zero replans is the typical case.

4.6 What remains after a reactive snapshot

The source paper's 0.61% margin over reactive A* is the residual value of the 30-minute horizon after a current-map replan is allowed. Journey-level counts make that residual concrete. Under incidents the framework's mean advantage over B3 was 4.0 seconds and its median advantage was 0.0 seconds. Using the same five-second band, the framework beat reactive A* on 11 journeys, tied it on 60, and lost to it on 4. Mean framework time was 600 seconds, reactive A* 604 seconds, and Dijkstra 727 seconds. Almost all of the two-minute gap between Dijkstra and the framework is also a gap between Dijkstra and a reactive snapshot. The forecast's extra incident saving, on this particular incident, is about four seconds on average and zero seconds on the typical journey.

That statement does not make the forecast unused. The incident was placed on the Dijkstra corridor and then became visible, which is exactly the situation a reactive replan is built for. Section 4.5 already showed three journeys on which the departure forecast avoided that corridor before any replan. The honest summary is narrower than either "the forecast does the work" or "the forecast does nothing." Snapshot Dijkstra is the weak baseline under this incident. Reactive replanning captures most of the repair. The forecast adds a large saving only when it changes the initial path, and a small saving on the typical journey after the slowdown is already on the map.

The oracle comparison has the same skew. Mean framework time sat 19.3 seconds behind the oracle under incidents, while the median gap was 0.2 seconds. The published 1.69% figure is an all-regime mean, not this incident gap. 42 of 75 incident journeys finished within 10 seconds of the oracle path. 27 were slower by more than 10 seconds. 6 finished more than 10 seconds faster than the oracle path. The oracle is a planning baseline that uses actual future speeds. It is not a mathematical lower bound on every trajectory, because the framework may replan after the vehicle has moved. The mean gap is produced by the 27 slower journeys, not by a uniform lag on every call. In peak hour the picture is closer to a tie with reactive routing, because reactive A* and Dijkstra took the same time on every peak-hour journey in this archive (both means 513 s). The peak-hour residual over B3 is therefore the same 7.7 seconds already reported against Dijkstra.

4.7 Threshold sensitivity of the incident saving

The source ANOVA found no mean travel-time difference among the four thresholds ($F \approx 0.001$, $p = 1.00$) and recommended $\delta = 0.20$ because it issues fewer alerts. Table 5 and Figure 6 ask whether that recommendation hides a change in the tails. The same 71 incident journeys win at every threshold. At every δ from 0.05 to 0.20, 71 incident journeys were faster by more than five seconds and none were slower by more than five seconds. The 10th percentile stayed at 30.2 seconds. The median moved only from 79.6 seconds at $\delta = 0.05$ to 83.1 seconds at $\delta = 0.20$. The 90th percentile fell from 282.5 to 263.2 seconds. Mean replans fell from 1.07 to 0.99. Moving to the quieter threshold gives up about 19 seconds at the extreme upper tail and does not create incident losses. For a decision-support rule that is supposed to avoid needless alerts, that trade is still the one the source paper recommended. The new information is the price of the trade, which the mean ANOVA could not see.

Table 5. Incident distribution of seconds saved at each remaining-time threshold (N = 75).

δ	Mean seconds	10th pct.	Median	90th pct.	Wins	Losses	Mean replans
0.05	129.4	30.2	79.6	282.5	71	0	1.07
0.10	129.3	30.2	79.6	282.5	71	0	1.05
0.15	127.8	30.2	79.6	263.2	71	0	1.03
0.20	127.7	30.2	83.1	263.2	71	0	0.99

Note. Wins and losses use the five-second band against snapshot Dijkstra. The same 75 pairs appear at each δ.

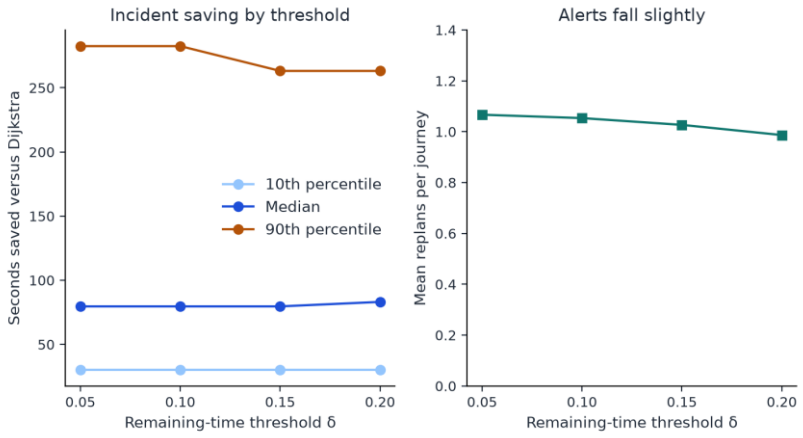


Figure 6. Incident percentiles of seconds saved (left) and mean replans (right) across the four published thresholds. The panels are separate so the small drop in replans is not read on the seconds scale. The win count is 71 at every threshold and is not drawn.

5. Discussion

5.1 Implications for a single dispatch call

Three sentences are supported, and they should be kept apart. If the call resembles the incident regime — a corridor on the snapshot path slows sharply after departure — the published framework was faster than snapshot Dijkstra on 71 of 75 journeys, never slower by more than five seconds, and the typical saving was about a minute and a half of road travel time, with a mean a little over two minutes on the road. If the call resembles peak hour on this freeway graph, the typical saving was zero seconds, about three journeys in ten were faster by more than five seconds, and about one in ten was slower. If the call resembles off-peak, the framework and the snapshot agreed. Quoting 16% for all three situations repeats the quantity the source paper already warned was a mixture. This archive shows how uneven that mixture is.

A second distinction matters for system design. Beating snapshot Dijkstra under this incident is mostly what a reactive replan also does, because the incident was written onto the Dijkstra corridor and then became visible. The forecast's unique residual is about four seconds on average. The forecast's unique large savings are the departure paths that never needed a replan. A deployment that already replans from live speeds should not expect another 16% from adding this horizon. A deployment that still routes once, from the speeds on the screen at the moment of the call, is the setting in which the incident result is a two-minute difference.

5.2 Relation to the source paper

The journey-level percentages agree with Macharia et al. (2026): 16.1% under incidents, 1.2% in peak hour, and 0.01% off-peak, beside the published pooled means of 16.19%, 1.03%, and 0.01%. The small differences come from using $\delta = 0.20$ alone rather than the pool of four thresholds. Mean travel time stays effectively unchanged across thresholds, the win count stays at 71, and the 90th percentile of seconds saved falls from 282.5 to 263.2. The margin over reactive A* remains small, and under incidents its median is zero seconds. These sections supply the distributions behind those published means.

The analysis also explains a misreading that a sign test invites. Off-peak produces more tiny gains than tiny losses, and the sign test rejects a fair coin. The seconds involved do not. Effect size and the bootstrap interval are the quantities that match the published t-test, which did not find an off-peak mean difference. Practical significance and sign counts are reported together so that the surplus of tiny differences is not upgraded into a new claim.

5.3 Transfer

The archive is a fully instrumented freeway detector graph in Los Angeles, with one vehicle and an incident placed on the snapshot path. It does not represent Nairobi mixed traffic, a sparse sensor network, or a disruption that appears off the planned corridor. Those limits are the source study's limits. They bind this article equally, because no new city was simulated. What transfers, if anything transfers, is a way of reporting: win rate, median seconds, length tertile, and the residual against a reactive replan. A mean percentage alone is not that report. Local sensor coverage remains the constraint on any later trial, as the source paper concluded. A later trial should be powered and briefed for incident calls and for longer trips. An across-the-board percentage is the wrong target for that trial.

6. Conclusion

This article reports the journey-level distribution of road travel time in the 900-journey archive of the published GRU predictive routing framework. At the operational threshold $\delta = 0.20$, incident journeys were faster than snapshot Dijkstra by a median of 83.1 seconds, with 71 wins, no losses beyond five seconds, and a time-weighted reduction of 17.6%.

The saving rose from about 64.4 seconds on short incident trips to about 220.5 seconds on long ones. Half of the saved incident road time came from 17 of 75 pairs. Peak-hour and off-peak medians were zero. The same incident archive shows a mean advantage of about four seconds over a reactive snapshot, and a median advantage of zero. The published 16% is the gain from repairing a snapshot path that has been invalidated on its own corridor. A higher alert threshold leaves the win count unchanged and slightly thins the extreme tail. The next result that would change this conclusion is a repeat on a second road graph, or an incident that is not planted on the Dijkstra path. In this archive, the road time saved is incident time on longer trips, and most of that time is also available to a reactive replan.

Data availability

The journey archive, simulation code, and source article materials are available at https://github.com/Freshia-Njoki/Emergency_Routing. The file read for every table in this article is results/simulation/simulation_results.csv. Figures 1–6 were drawn from that file at $\delta = 0.20$, except Figure 6 and Table 5, which use the incident rows at all four thresholds.

References

- Chowdhury, A., Kaisar, S., Khoda, M. E., Naha, R., Khoshkholghi, M. A., & Aiash, M. (2023). IoT-based emergency vehicle services in intelligent transportation system. *Sensors, 23*(11), 5324. https://doi.org/10.3390/s23115324
- Feng, J., Guo, G., Wang, J., Liu, X., Liu, Z., & Ding, Y. (2024). EVHF-GCN: An emergency vehicle priority scheduling model based on

heterogeneous feature fusion with graph convolutional networks. *IEEE Access*. https://doi.org/10.1109/ACCESS.2024.3349591

- Jiang, W., & Luo, J. (2022). Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications, 207*, 117921. https://doi.org/10.1016/j.eswa.2022.117921
- Kang, K., Jones, M. T., Armstrong, K., Avery, S. N., McHugo, M., Heckers, S., & Vandekar, S. (2023). Accurate confidence and Bayesian interval estimation for non-centrality parameters and effect size indices. *Psychometrika, 88*(1), 253–273. https://doi.org/10.1007/s11336-022-09899-x
- Liu, X. S. (2023). Bias correction for Cohen's d. *The Journal of General Psychology*. https://doi.org/10.1080/00221309.2023.2172545
- Macharia, F. N., Langat, G., & Mageto, S. (2026). Design and evaluation of a GRU-based predictive routing framework for emergency vehicles. *International Journal of Research Publication and Reviews, 7*(8), Article IJRPR69191. https://doi.org/10.55248/gengpi.07.0826.2618
- Qi, P., Pan, C., Xu, X., Wang, J., Liang, J., & Zhou, W. (2025). A review of dynamic traffic flow prediction methods for global energy-efficient route planning. *Sensors, 25*(17), 5560. https://doi.org/10.3390/s25175560