Introducing a novel Location-Assignment Algorithm for Activity-Based Transport Models: CARLA

Felix Petre*[1], Lasse Bienzeisler[1], and Bernhard Friedrich[2]

[1]M.Sc., Institute of Transportation and Urban Engineering, Technische Universität Braunschweig, Germany
[1]Univ.-Prof. Dr.-Ing., Institute of Transportation and Urban Engineering, Technische Universität Braunschweig, Germany

Short summary

This paper introduces CARLA (spacially Constrained Anchor-based Recursive Location Assignment), a recursive algorithm for assigning secondary (or any) activity locations in activity-based travel models. CARLA minimizes distance deviations while integrating location potentials, ensuring more realistic activity distributions. The algorithm decomposes trip chains into smaller subsegments, using geometric constraints and configurable heuristics to efficiently search the solution space.
Compared to a state-of-the-art relaxation-discretization approach, CARLA achieves significantly lower mean deviations, even under limited runtimes. It is robust to real-world data inconsistencies, such as infeasible distances, and can flexibly adapt to various priorities, such as emphasizing location attractiveness or distance accuracy.
CARLA's versatility and efficiency make it a valuable tool for improving the spatial accuracy of activity-based travel models and agent-based transport simulations.
Our implementation is available at https://github.com/tnoud/carla.
Keywords: Activity-based model; agent-based; exhaustive search; location choice; relaxation–discretization algorithm; synthetic population

1 Introduction

Activity-based models (ABMs) have emerged as a more behaviorally realistic alternative to traditional demand modeling paradigms, such as the four-step model (Rasouli & Timmermans, 2014; Bastarianto et al., 2023; Rezvany et al., 2024). Their disaggregate representation enables ABMs to better reflect individual decision-making processes and capture complex travel behavior patterns (Rasouli & Timmermans, 2014). Thus, ABMs require modeling the attributes, intentions, and interactions of each individual traveler (Hörl & Axhausen, 2023).
A key challenge is the difficulty of robustly predicting activity location choices (Miller, 2023). Primary activities, such as commuting to work or attending educational institutions, can often be modeled with relative accuracy, supported by structured data sources like commuter matrices (Matet et al., 2024). Modeling the locations of secondary activities, including shopping, leisure, or dining, presents a significantly greater challenge (Matet et al., 2024). These decisions are influenced by several factors and exhibit variability, making them more complex to predict. Furthermore, the lack of standardized approaches and datasets exacerbates the difficulty of accurately representing these behaviors (Hörl & Axhausen, 2023).
Common methods for the location choice of activities within activity plans use space-time prisms to define feasible activity locations, ensuring they are reachable within defined time or distant constraints (Salvadé et al., 2022). Another approach involves the use of mobile phone data, which can be utilized to generate trip schedules for activity-based traffic simulations, enabling the estimation of global spatial mobility behavior of a population (Cui et al., 2021; Matet et al., 2024).
Hörl & Axhausen (2023) introduced a data-driven algorithm for assigning secondary activity locations, ensuring consistency with fixed points in daily activity plans while maintaining realistic distance distributions. This method eliminates the need for complex choice models and supplementary origin-destination data, providing a practical and user-friendly solution for secondary location assignment. The algorithm has proven effective in various ABMs (Hörl & Balac, 2021; Hörl et

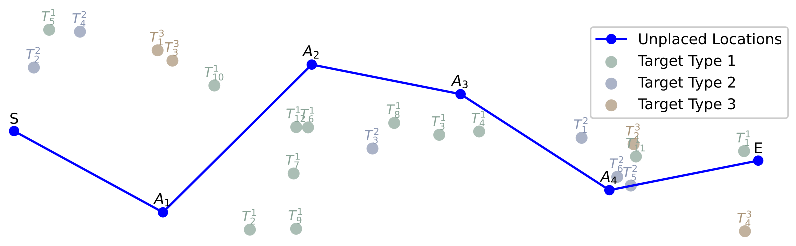

Figure 1: A five-trip activity chain with some illustrated target locations.

al., 2021; Sallard et al., 2021; Pereira et al., 2022; Manout et al., 2024) and represents the current state-of-the-art.

Although computationally efficient, the approach has potential for further improvement. The algorithm iteratively selects a feasible option from the solution space to find a valid result, but it does not guarantee the optimal solution. A more systematic optimization algorithm could reliably achieve the best possible outcome. Additionally, the algorithm does not account for *location potential*, i.e. the prominence or attraction of a certain location. As a result, larger destinations, such as shopping malls, are not selected more frequently than smaller shops, limiting the representation of realistic activity patterns.

In this work, we present a new algorithm for assigning secondary activity locations, or any activity chain, using Euclidean distances from a household travel survey (HTS). The algorithm utilizes a recursive strategy to decompose activity chains into manageable segments and systematically narrows the solution space using tailored heuristics and geometric constraints. It incorporates location potentials, ensuring realistic assignments. Its flexibility allows it to adapt to various scenarios and priorities; for example, when location potentials are more important, the algorithm can prioritize them while using distances solely to define the search space. Furthermore, the algorithm is robust and capable of handling implausible distance inputs without significant degradation in performance.

2 Methodology

Building on the conceptual framework established by Hörl & Axhausen (2023), we adopted a similar terminology to describe the activity assignment problem. An activity chain represents a sequence of activities performed by an individual, connected by trips, which denote the travel between consecutive activities. Within an activity chain, illustrated in Figure 1, a main activity (usually *work* or *education*) is placed using a different algorithm, leaving secondary activities to be located. Activity chains are divided into segments: a segment is a sequence of trips with known locations at its start and end and any number of activities with unknown locations in between.

For each segment bounded by two fixed locations S (start) and E (end), the goal was to assign n variable activities $A_1, A_2, \ldots, A_n$ (e.g. sport, leisure...) to discrete target locations $T_1, T_2, \ldots, T_k$ from a predefined set of locations in the study area (e.g. parks, businesses...), where the desired activity type must fit the target location type. The problem can be divided into two cases:

1. **Two-trip Case**: This involves a single variable activity A located between the two fixed points S and E. It is relatively straightforward to solve.
2. **Complex case**: This involves $n \geq 2$ variable activities $A_1, A_2, \ldots, A_n$ distributed between S and E. The solution space becomes extremely large and complex as n increases, creating a highly interdependent optimization problem. Each activity placement T_j affects the subsequent trips.

The objective was to minimize the total discrepancy between the distances d_{ij}^{HTS} reported in the HTS and the distances d_{ij}^{model} derived from the assigned locations. In order to incorporate location potentials $P(T_j)$ for the target candidate locations T_j into the optimization objective, a weighted potential term was added:

$$\text{Maximize} \quad \alpha \sum_{j=1}^{k} P(T_j) - \beta \sum_{i=1}^{n+1} \left| d_{i-1,i}^{\text{HTS}} - d_{i-1,i}^{\text{model}} \right| \tag{1}$$

where $d_{i-1,i}^{\text{HTS}}$ represents the distance between consecutive activities as observed in the HTS, $d_{i-1,i}^{\text{model}}$ represents the Euclidean distance between their assigned locations in the models, and α and β are tunable parameters controlling the influence of potentials and deviations.

Core Idea

To address the optimization problem, we developed a recursive algorithm that simplifies the trip chain step-by-step until a solution is found. The key idea is to divide a complex segment into two smaller subsegments at a middle activity, the *anchor*, which is placed at a discrete, suitable location. This splits the segment $(S \to E)$ into $(S \to A_m)$ and $(A_m \to E)$. The process is repeated for all subsegments until all activities are assigned. A tree search evaluates multiple anchor placements and subsegments simultaneously. Each *branch* is scored based on distance deviations and location potentials, selecting the best solution. Geometric constraints narrow the search space, and selection heuristics improve efficiency.

Key Components

1. Main Algorithm Loop
For each person, the algorithm processes each segment sequentially in the main loop. The results for each segment are combined to form a complete, placed trip chain.

2. SolveSegment Function
The `SolveSegment` function (see Algorithm 1) is the core of the algorithm. It operates on the given segment by selecting a number of possible anchor locations (*candidates*). Each chosen anchor spawns two new subsegments, unique to this anchor location, which are solved recursively. The algorithm proceeds until it reaches one of two base cases: *(i)* a trivial single-trip subsegment (where no further activity placement is needed), or *(ii)* the *Two-trip Case* (where the best activity location can be found directly). It returns the best identified segment placement.
In the *Two-trip Case*, ideal possible locations for A are at the intersections of the two circles:

- Centered at S with radius $d_{S,A}$,
- Centered at E with radius $d_{A,E}$.

If the input distances are infeasible (i.e., the circles do not intersect because they are too far apart or one being fully enclosed within the other), the algorithm selects a single point that minimizes total deviation.
After determining the ideal point(s), the best discrete location is chosen from the target locations. This, however, is not necessarily the location closest to an ideal point. First, when location potentials are taken into account, a location further from an ideal point but with a much larger potential may be superior. Additionally, the Euclidean proximity to an ideal point is not directly representative of the total distance deviation, shown in this example: Given an expected distance of 1 unit from S and 6 units from E, the ideal points are found as illustrated in Figure 2. Moving orthogonally to the illustrated ellipse from either ideal point has a stronger influence on the distances from S and E than moving tangentially to the ellipse by the same distance. The deviation gradients around the ideal points do not form circular funnels, but a more complex shape.
Thus, instead of choosing the closest location to an ideal point, we select a number of candidates around each ideal point, which are then evaluated (by the `Evaluation Function`) and the single best candidate selected (by the `Selection Function`). The difference between choosing the closest location and the "best" among several close locations is significant when optimizing for minimal deviation.
In the *Complex, recursive, Case*, the ideal anchor location is highly dependent on both previous and following trips, whose locations are yet undetermined. Using geometric constraints, the algorithm restricts the search area to overlapping feasible regions (rings, Figure 3) around the start (S) and end (E) points, defined by:

- Inner radius ($r_{\min}$): the minimum allowable distance.
- Outer radius ($r_{\max}$): the maximum allowable distance.

The maximum radii are the sum of all trip distances between S and the anchor (A_m), and A_m and E, respectively. Minimum radii are calculated combinatorially as the shortest distance reachable from S or E, considering trip lengths. For example, a very long trip followed by two short trips may prevent returning to the reference point, making $r_{\min}$ the difference between the long trip

Algorithm 1 SolveSegment

Require: Segment seg, target locations T, configuration C

Ensure: Optimally placed segment $\hat{\text{seg}}$, score

```
1:  if |seg| = 1 then                                          ▷ Base case: single trip
2:      return seg, 0
3:  else if |seg| = 2 then                                     ▷ Base case: two trips
4:      candidates ← CircleIntersections(seg[0].from, seg[1].to, $T$, $C$)
5:      scores ← EvaluationFunction(candidates)
6:      selected_candidate, selected_score ← SelectionFunction(candidates, scores, 1)
7:      Update seg with selected_candidate
8:      return seg, selected_score
9:  else                                                       ▷ Recursive case: multiple trips
10:     anchor ← ChooseAnchor(seg, $C$)
11:     $D_1min, D_1max, D_2min, D_2max$ ← GetFeasibleDistances(seg, anchor)
12:     candidates ← OverlappingRings($D_1min, D_1max, D_2min, D_2max, T, C$)
13:     scores ← EvaluationFunction(candidates)
14:     selected_candidates, selected_scores ←
15:       SelectionFunction(candidates, scores, $C$.number_of_branches)
16:     branch_segments, branch_scores ← []
17:     for all $c \in$ selected_candidates do
18:         Update seg with candidate $c$ at anchor
19:         $\hat{\text{seg}}_1$, $\text{score}_1$ ← SolveSegment(seg[0 : anchor], $T$, $C$)
20:         $\hat{\text{seg}}_2$, $\text{score}_2$ ← SolveSegment(seg[anchor + 1 :], $T$, $C$)
21:         Combine $\hat{\text{seg}}_1$ and $\hat{\text{seg}}_2$ into $\hat{\text{seg}}$
22:         $\hat{\text{seg}}$_score ← $\text{score}_1 + \text{score}_2$ + score of candidate $c$
23:         branch_segments.append($\hat{\text{seg}}$)
24:         branch_scores.append($\hat{\text{seg}}$_score)
25:     end for
26:     best_index ← arg max branch_scores
27:     return branch_segments[best_index], branch_scores[best_index]
28: end if
```

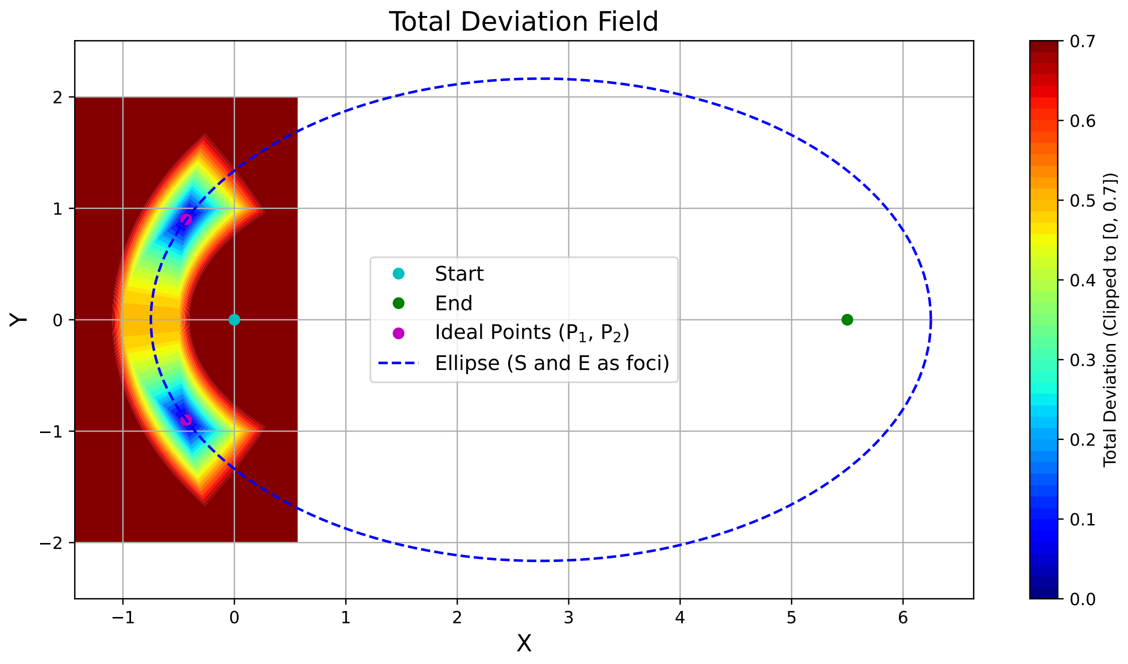

Figure 2: Total distance deviation of locations around the ideal points

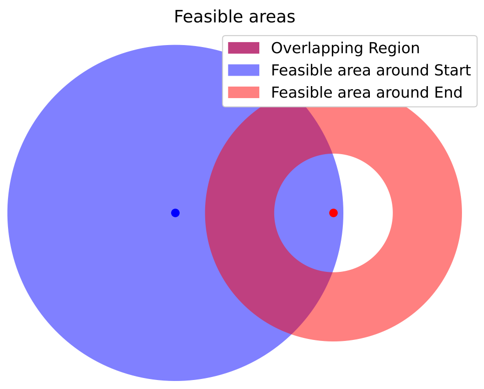

Figure 3: Feasible areas for the anchor location around Start and End

and the combined shorter trips. By constraining the search space to the overlap of these rings, the algorithm ensures that the placement respects travel distance constraints while significantly reducing computational effort.
Once the feasible region is determined, the function queries the spatial index for all locations within it which fit the given activity type (*candidates*). If there are no candidate locations found, this either implies that the rings do not overlap indicating an impossible trip chain (and may be expected given the nature of HTS), or simply indicates that there are no acceptable target locations in the overlap. In either case, the rings are expanded by shrinking the minimum and increasing the maximum radii until a specified minimum number of valid location is found. The identified candidates are then evaluated using the `Evaluation Function` and a subset of configurable size is selected using the `Selection Function`, ensuring a balance between location potential and distance deviations. The selected candidates then serve as inputs for the subsequent recursive step.

3. Scoring and Selection
The `Evaluation Function` assigns a score to each candidate:

$$\text{Score}(c) = \alpha \cdot P(c) - \beta \cdot D(c) \tag{2}$$

Here, α and β are weights that control the trade-off between location potential $P(c)$ and the distance deviation $D(c)$. This formulation can be extended to include nonlinear terms if needed. For the Two-trip Case, $D(c)$ is defined as:

$$D(c) \;=\; \big|\, d_1 \;-\; \|S - c\|_2 \big| \;+\; \big|\, d_2 \;-\; \|c - E\|_2 \big|. \tag{3}$$

Here, d_1 and d_2 are the expected distances, while $\|S-c\|_2$ and $\|c-E\|_2$ are the Euclidean distances between the points. In the one-trip case, this simplifies to a single term.
The final score of a whole chain is computed as the sum of scores for all selected locations in the chain. By summing up the individual contributions of spatial potential and distance accuracy, this resolves directly into the optimization function introduced earlier.
The `Selection Function` is responsible for selecting the best candidate locations based on their scores as computed by the Evaluation Function. While selecting more candidates (and thus creating more branches) improves the quality of the results, it also significantly increases processing times, particularly for long activity chains. To address this trade-off, the function supports several configurable heuristic strategies, allowing the algorithm to balance exploration, diversity, and computational efficiency. They are listed in Table 1.

3 Results and discussion

Comparing Hörl and CARLA: Trade-offs Between Performance and Accuracy

We evaluated the trade-off between runtime and accuracy for both Hörl and CARLA algorithms. For Hörl, we varied the maximum number of iterations, while for CARLA, we varied the number

Table 1: Selection Strategies Overview

Strategy	**Description**
Keep All	Selects all candidates, ensuring the optimal solution but with high computational cost.
Top-k	Retains the k highest-scoring candidates, effective but may overlook interdependent solutions.
Monte Carlo	Selects candidates probabilistically based on normalized scores, introducing exploration and stochasticity.
Top-k Monte Carlo	Combines deterministic (k top candidates) and probabilistic sampling, balancing quality and diversity.
Spatial Downsampling	Selects candidates evenly from a configurable grid of cells to ensure distribution across the feasible region. Much faster than the alternative of using k-means clustering.
Top-k Spatial Downsampling	A hybrid approach: selects k top candidates and applies spatial downsampling for diversity.

of branches. Since Hörl minimizes distance deviations only, we simplified CARLA's Evaluation Function to:

$$\mathrm{Score}(c) = -\,\mathrm{D}(c) \tag{4}$$

We profiled Hörl's algorithm and found no evident bottlenecks in its implementation. The evaluation used a sample of 1000 individuals from a German HTS, prefiltering persons with trips over 30 km as they exceeded the study area's bounds. Target locations were real-world data from Hanover, Germany, with activity types derived from OpenStreetMaps.
Hörl's implementation uses a relaxation solver and an assignment solver, with the maximum number of iterations set to 1000 and 20, respectively. Deviation thresholds determine when the algorithm returns. While they improve runtime, they limit the best achievable result. To evaluate this, we tested two configurations: the standard setup and a testing setup with increased limits and reduced thresholds, as shown in Table 2.

Table 2: Comparison of Hörl Standard and Hörl 1m1000 Configurations.

Parameter	**Hörl Standard**	**Hörl 1m1000**
Thresholds (Car driver/passenger, Public transport)	200 m	1 m
Thresholds (Walk, Bike)	100 m	1 m
AssignmentSolver Max Iterations	Limited to 20	Limited to 1000

Evaluating CARLA's perfomance, we applied a consistent base configuration (see Table 3).
Before using the relaxation solver, Hörl's algorithm polls distances from a distribution, as its input data lack direct distance values. In contrast, we directly input Euclidean distances, bypassing this step. However, Hörl's relaxation solver relies on feasible distances and performs poorly when

Table 3: Base CARLA Configuration.

Parameter	**Value**
number_of_branches	50
min_candidates_complex_case	10
candidates_two_trip_case	20
anchor_strategy	lower_middle
selection_strategy_complex_case	mixed (Top-k Monte Carlo Sampling)
selection_strategy_two_trip_case	top_k

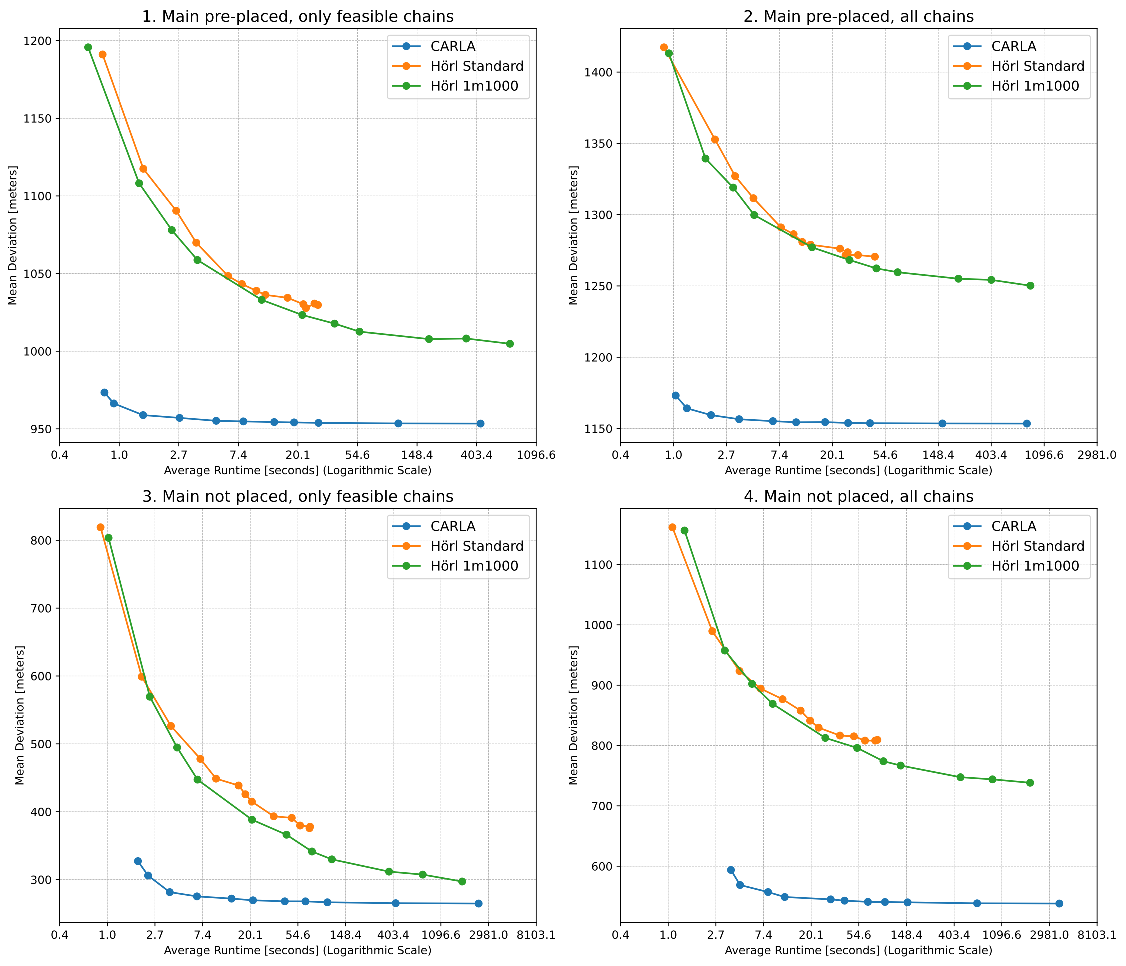

Figure 4: Mean deviation vs. runtime across scenarios, varying iterations for Hörl and branches for CARLA. Each point in the plots represents a result of a distinct parameter configuration of the respective algorithm.

handling infeasible ones. To address this, we tested two scenarios: one using the full HTS sample and another excluding persons with infeasible data, leaving 693 individuals for analysis.
Both algorithms can either be applied to locate all activities of a activity chain or used only between already assigned fixed main activities. Accordingly, we evaluated two additional scenarios: one where the main activity is pre-placed at a fixed location, requiring only the segments before and after the main activity to be placed, and another where the main activity is not pre-placed, requiring the entire activity chain to be assigned. In sum, the four scenarios emerged:

1. Main activity pre-placed, considering only feasible trip chains.
2. Main activity pre-placed, considering all trip chains.
3. Main activity not placed, considering only feasible trip chains.
4. Main activity not placed, considering all trip chains.

The results in Figure 4 showcase the performance of both algorithms. CARLA consistently outperforms Hörl in all scenarios, achieving solutions with a mean deviation close to the likely optimum. This is even the case at much shorter runtimes, with improvements observed as the number of branches increased. With a low number of branches (20), CARLA's results are not only better but also significantly faster.

Detailed comparison

A single run from scenario three, which excludes infeasible chains to isolate the algorithms' performance, highlights the differences. Using its base settings (Table 3), CARLA completed processing

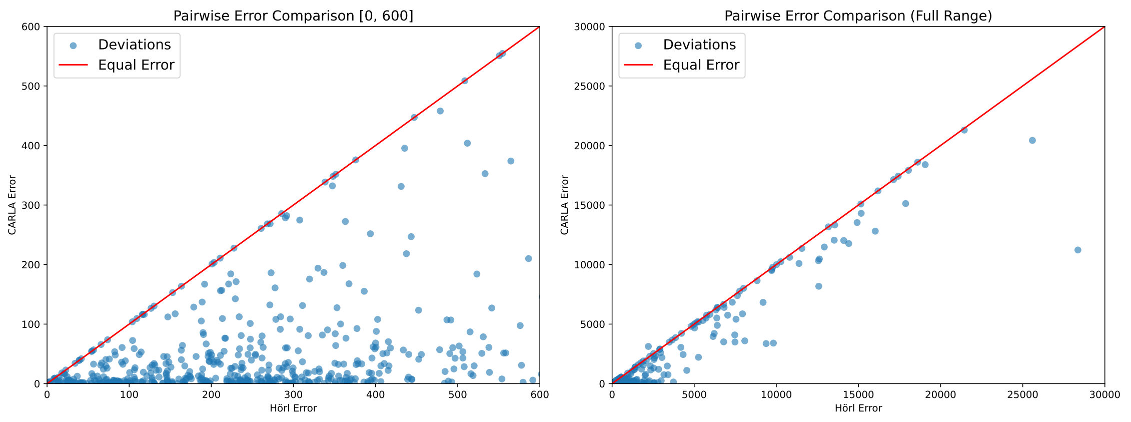

Figure 5: Pairwise comparison of total distance deviation per person.

in 45 seconds, while Hörl's standard configuration, with a maximum of 1000 iterations as per the published standard, required 89 seconds.
CARLA produced better results in 630 cases (90.9%). In 62 cases (8.9%), results were equal, indicating that both algorithms chose the same discrete locations. In a single case, Hörl found a location chain with a smaller total deviation, reflecting the heuristic nature of CARLA's settings in this test.
The results are visualized in Figure 5. As 92.5% of total deviations produced by the Hörl algorithm are below 600m, we focus on this area first. CARLA consistently manages to find total deviations well below Hörl's deviations, very often close to zero. The full range indicates that even for chains where a placement close to the given distances is not possible (e.g. due to sparse target locations), CARLA finds better results.

4 Conclusions

This paper presented CARLA, a novel recursive algorithm for assigning (secondary, or any) activity locations in activity-based travel models. CARLA optimizes for both location potential and adherence to observed distance distributions while handling real-world complexities such as infeasible distances or sparse target locations. Compared to Hörl's relaxation-discretization algorithm, CARLA consistently achieved lower mean deviations, even under constrained runtimes. Its recursive structure efficiently decomposes trip chains into manageable segments, leveraging geometric constraints to reduce the solution space. Configurable evaluation and selection functions allow CARLA to balance computational efficiency with solution quality and enable the integration of location potentials for improved realism. While CARLA demonstrates robustness and flexibility, its performance ultimately depends on accurate HTS inputs. Future work will focus on integrating it into a full framework, where it may replace several existing algorithms.

Acknowledgements

The results were created as part of the Urban Climate Future Lab UCFL, funded by zukunft.niedersachsen, a funding program of the Lower Saxony Ministry of Science and Culture and the Volkswagen Foundation. Responsibility for the content of this publication lies with the authors.

References

Bastarianto, F. F., Hancock, T. O., Choudhury, C. F., & Manley, E. (2023). Agent-based models in urban transportation: Review, challenges, and opportunities. *European Transport Research Review*, *15*(1), 19. doi: 10.1186/s12544-023-00590-5

Cui, Y., He, Q., & Bian, L. (2021). Generating a synthetic probabilistic daily activity-location schedule using large-scale, long-term and low-frequency smartphone GPS data with limited activity information. *Transportation Research Part C: Emerging Technologies*, *132*, 103408. doi: 10.1016/j.trc.2021.103408

Hörl, S., & Axhausen, K. W. (2023). Relaxation–discretization algorithm for spatially constrained secondary location assignment. *Transportmetrica A: Transport Science*, *19*(2), 1982068. doi: 10.1080/23249935.2021.1982068

Hörl, S., & Balac, M. (2021, September). Synthetic population and travel demand for Paris and Île-de-France based on open and publicly available data. *Transportation Research Part C: Emerging Technologies*, *130*, 103291. doi: 10.1016/j.trc.2021.103291

Hörl, S., Becker, F., & Axhausen, K. W. (2021). Simulation of price, customer behaviour and system impact for a cost-covering automated taxi system in Zurich. *Transportation Research Part C: Emerging Technologies*, *123*, 102974. doi: 10.1016/j.trc.2021.102974

Manout, O., Diallo, A. O., & Gloriot, T. (2024). Implications of pricing and fleet size strategies on shared bikes and e-scooters: A case study from Lyon, France. *Transportation*. doi: 10.1007/s11116-024-10559-5

Matet, B., Côme, E., Furno, A., Hörl, S., Oukhellou, L., & El Faouzi, N.-E. (2024). Improving the generation of synthetic travel demand using origin–destination matrices from mobile phone data. *Transportation*. doi: 10.1007/s11116-024-10524-2

Miller, E. (2023). The current state of activity-based travel demand modelling and some possible next steps. *Transport Reviews*, *43*(4), 565–570. doi: 10.1080/01441647.2023.2198458

Pereira, A. M., Dingil, A. E., Přibyl, O., Myška, V., Vorel, J., & Kříž, M. (2022). An Advanced Travel Demand Synthesis Process for Creating a MATSim Activity Model: The Case of Ústí nad Labem. *Applied Sciences*, *12*(19), 10032. doi: 10.3390/app121910032

Rasouli, S., & Timmermans, H. (2014). Activity-based models of travel demand: Promises, progress and prospects. *International Journal of Urban Sciences*, *18*(1), 31–60. doi: 10.1080/12265934.2013.835118

Rezvany, N., Kukic, M., & Bierlaire, M. (2024). A Review of Activity-based Disaggregate Travel Demand Models. *Findings*. doi: 10.32866/001c.125431

Sallard, A., Balać, M., & Hörl, S. (2021). An open data-driven approach for travel demand synthesis: An application to São Paulo. *Regional Studies, Regional Science*, *8*(1), 371–386. doi: 10.1080/21681376.2021.1968941

Salvadé, N., Hillel, T., Pougala, J., Haering, T., & Bierlaire, M. (2022). Representing location choice within activity-based models. *STRC conference paper 2022*.