Submitted to *Transportation Science*
manuscript

Parking search in the physical world: Calculating the search time by leveraging physical and graph theoretical methods

Nilankur DUTTA
Institut Lumière Matière, CNRS and Université Claude Bernard Lyon 1,
F-69622 Villeurbanne, France, nilankur.dutta@gmail.com

Thibault CHARLOTTIN
Institut Lumière Matière, CNRS and Université Claude Bernard Lyon 1,
F-69622 Villeurbanne, France, and
École nationale des travaux publics de l'État (ENTPE), Université de Lyon,
F-69518 Vaulx-en-Velin, France

Alexandre NICOLAS
Institut Lumière Matière, CNRS and Université Claude Bernard Lyon 1,
F-69622 Villeurbanne, France, alexandre.nicolas@polytechnique.edu, https://www.alexandrenicolas.net

---

A. How other agent-based models are encompassed by the proposed framework

There already exist a number of parking microsimulation models. In the main text, we have claimed that the proposed modelling framework, albeit mathematically concise, can encompass the gist of many of these models, as far as the parking search process is concerned (i.e., without considering the feedback loop of parking search on the parking demand, which pertains to the higher-level transport model into which our parking model can be integrated). This section supports this statement.

Let us start with the recent PARKGRID model described in (Fulman and Benenson 2021), in which agents are injected into a grid-like city with a Poisson rate of injection (as in our case). They start searching a parking space within 500 m of their destination, driving along street links and parking in the first available space, regardless of the parking rate. In our framework, this means that all spots within 500 m of the destination have the same attractiveness $A_i = 0$ and the other ones have an attractiveness $A_i \to -\infty$. If drivers reach the end of the street link before finding a vacant spot, they will turn to an adjacent link with probabilities set according to the results of earlier virtual experiments similar to those in (Fulman, Benenson, and Elia 2020). These probabilities can straightforwardly be encoded in our matrix of turning probabilities $\underline{\underline{T}}$. Finally, if they fail to park after 20 minutes, they stop searching for on-street parking, which is equivalent to setting our upper bound for the parking search time to 20 minutes. Accordingly, the foregoing rules can naturally be described within our framework. Note, however, that this paper assumes a uniform distribution of parking times, whereas we considered a constant departure rate here.

Turning to PARKAGENT (Benenson, Martens, and Birfir 2008), the route choice is simpler in this model, insofar as drivers choose the street segment that brings them closest to their destination, which can of course also be encoded in our turn-choice matrix. They start searching for parking when they are within a distance of 100 m (rather than 500 m) of the destination and reduce their driving speed to do so, which can be implemented in our model (because driving speeds on street links may vary with the category of drivers), although we have not done it in our test studies. But interestingly they do not necessarily park in the first vacant spot. Instead, they assess the odds of finding a vacant spot before the destination, on the basis of the percentage p_{unocc} of vacant spots among those that they have seen so far and the estimated number N_s of spots to be encountered before reaching the destination. Taking this into account, they will accept a spot with probability $\max[0, \min(1, 1.5 - 0.5 * p_{\text{unocc}} \cdot N_s)]$. In our model, this can approximately be transcribed by setting $\beta^{(\alpha)} = 1 - \phi \approx p_{\text{unocc}}$ (where ϕ is the occupancy) and A_i equal to 1.5 times the negative of the distance from spot i to the destination, measured in number of spots, if one accepts that $e^{-1.5x}$ is of order 1 for $x < 1$ and much smaller than 1 for $x > 3$. Nevertheless, note that in this formula the percentage of vacant spots is assessed based on the actual occupancy ϕ, and not the user's estimate p_{unocc}, which we cannot transcribe in our model. If drivers have failed to park before reaching their destination, they will park in the first vacant spot within a disk whose radius grows with time. The new strategy is tantamount to setting A_i to 0 or $-\infty$ depending on the distance

from spot i to the destination, as in PARKGRID, but so far we have not implemented the possibility of changes in parking strategies over time. Still, the change in strategy should be of little importance in the simulations, because the specified route choices appear to maintain the driver close to the destination, where the probability to accept a vacant spot is very high. Finally, drivers consider the search unsuccessful after 10 minutes, rather than the aforementioned 20 minutes in PARKGRID.

The SUSTAPARK model Dieussaert et al. (2009) hinges on a discrete-choice model (with logit probabilities) for the choice of a parking mode (on-street, private, or garage) and also for the probability p_i to accept a vacant on-street parking space. The utility U_i of the latter is a linear combination intrinsic characteristics of the spot, to which a term describing the occupancy ϕ_{street} in the street is added. The resulting parking acceptance probability $p_i = \frac{1}{1+e^{-\beta_\phi \phi_{\text{street}}} e^{-U_i}}$ can be recast into the general form of Eq. 2, but not into the Boltzmann-like form of Eq. 3. However, the methodology exposed in Sec. 3.3 does not rely on this functional form and is therefore still applicable. On the other hand, our model was only applied to on-street parking; it does not describe the changes in parking mode implemented in SUSTAPARK. Finally, drivers follow random routes close to their destination, which can be encoded in our turn-choice matrix, but drivers change their speed when they start searching for parking.

Axhausen et al.'s microsimulation introduced in Horni et al. (2013) in view of an integration in MATSim also considers different parking strategies, but it presents specific features for the search process. First, agents may either drive directly to the destination and start searching from there, or start their search a given distance away from it. An easy way to take this into account in our analytical derivation is to consider that the agents are injected at the position where they start searching for parking. Drivers then accept a parking space depending on its distance to destination (as in our model), but also on the elapsed search time. This specific feature, which introduces a dependence on the driver's history, cannot be accurately replicated by our approach. Still, on a more *qualitative* level, our parking tension parameter β has the same effect as the elapsed search time: parking acceptance will be enhanced by both of these variables. Regarding route choices, they are determined probabilistically once the search has begun, which is compatible with our approach.

Finally, the approach most closely related to our model is probability the network of finite-capacity queues introduced by Ratliff et al. (2016), Dowling, Ratliff, and Zhang (2019), and

Tavafoghi, Poolla, and Varaiya (2019), in that it also involves analytical developments; this elegant approach was employed to address the problem of estimating the cruising traffic on the basis of empirical occupancy data (Dowling, Ratliff, and Zhang 2019). In the model, street segments are block-faces on a Manhattan-city grid and each one of them is handled as a finite-capacity queue that serves the driver if it contains a vacant space, or rejects it to adjacent block-faces otherwise. We note that, compared to ours, this model rests on the following additional assumptions:

,

(i) drivers at junction select the next street segment purely randomly and the nodes of the street network only differ by their degree,

(ii) every agent will accept the first vacant space they encounter (which translates into a uniform attractiveness field in our model),

(iii) adjacent block-faces have similar levels of occupancy.

B. Details pertaining to the Analytical Calculations

B.1. Search time in real time units

In the main text, we have exposed the calculation of the search time expressed in arbitrary units (Eq. 7). To do so, drivers were split into distinct categories α, each endowed with their own matrix of turning probabilities $\underline{\underline{T}}^{(\alpha)}$ and generalised transition matrix $M_{ij}^{(\alpha)} = (1 - p_i^{(\alpha)}\,\hat{n}_i)\cdot T_{ij}^{(\alpha)}$. Since ${M^{(\alpha)}}^K$ determines how the density of cars evolves on the graph in K steps (i.e., hops from node to node), we were able to calculate the average number of steps before parking.

In order to derive a search time in minutes, a slightly more elaborate method is needed. For that purpose, we insert the transition matrix into a 'generating' function $z \to \underline{\underline{N}}^{(\alpha)}(z)$, where z is a real variable, $N_{ij}^{(\alpha)}(z) = z^{\tau_{ij}}\, M_{ij}^{(\alpha)}$, and τ_{ij} is the travel time from node i to node j. At this stage, one can notice that the exponentiation of this matrix to the power K yields

$$\left[\underline{\underline{N}}^{(\alpha)^K}\right]_{ij}(z) = \sum_{\pi \text{ s.t. } \pi_0=0, \pi_K=j} z^{\tau_{\pi_0\pi_1}+\ldots+\tau_{\pi_{K-1}\pi_K}} \prod_{k=0}^{K-1} M_{\pi_k\pi_{k+1}}, \tag{S1}$$

where the sum runs over permutations π of indices (or 'paths') such that $\pi_0 = i$ and $\pi_K = j$. It immediately follows that

$$\frac{d}{dz}\left[\underline{\underline{N}}^{(\alpha)^K}\right]_{ij}(z=1) = \sum_{\pi \text{ s.t. } \pi_0=0, \pi_K=j} \left(\tau_{\pi_0\pi_1}+\ldots+\tau_{\pi_{K-1}\pi_K}\right) \prod_{k=0}^{K-1} M_{\pi_k\pi_{k+1}} \tag{S2}$$

is the probability to reach spot j a number K of steps after injection of the car at node i, multiplied by the total travel time from i to j. The (unbound) travel time before parking thus reads

$$\begin{aligned} T_s^{(\alpha)} &= H_i^{(\alpha)}(0) \cdot \sum_{K=0}^{\infty} \frac{d}{dz} \left[\underline{\underline{N}}^{(\alpha)K} \right]_{ij} (z=1) \cdot \tilde{p}_j^{(\alpha)} \\ &= H_i^{(\alpha)}(0) \cdot \frac{d}{dz} \left[\left(\underline{\underline{\mathbb{I}}} - \underline{\underline{N}}^{(\alpha)}(z) \right)^{-1} \right]_{ij} (z=1) \cdot \tilde{p}_j^{(\alpha)} \end{aligned}$$

But, since $\frac{d\underline{\underline{A}}^{-1}}{dz} = -\underline{\underline{A}}^{-1}(z) \frac{d\underline{\underline{A}}}{dz} \underline{\underline{A}}^{-1}(z)$ for any differentiable function $z \to \underline{\underline{A}}(z)$ of invertible matrices $\underline{\underline{A}}(z)$ and $\underline{\underline{N}}^{(\alpha)}(z=1) = \underline{\underline{M}}^{(\alpha)}$,

$$T_s^{(\alpha)} = H_i^{(\alpha)}(0) \cdot \left[\left(\underline{\underline{\mathbb{I}}} - \underline{\underline{M}}^{(\alpha)} \right)^{-1} \cdot \underline{\underline{N}}^{(\alpha)\prime}(z=1) \cdot \left(\underline{\underline{\mathbb{I}}} - \underline{\underline{M}}^{(\alpha)} \right)^{-1} \right]_{ij} \cdot \tilde{p}_j^{(\alpha)}, \tag{S3}$$

where we recall that $N_{ij}^{(\alpha)\prime}(z=1) = \tau_{ij} M_{ij}^{(\alpha)}$ and $\tilde{p}_j^{(\alpha)} = \hat{n}_j p_j^{(\alpha)}$.

Equation S3 expresses the mean search time in minutes (or, more generally, in the same units as τ_{ij}) as a function of the occupancy field (n_j).

B.2. Complexity of the calculations

How does the complexity of the calculations required to solve Eq. S3, in terms of number of floating-point operations, compare with that of direct numerical simulations? First, note that the efficient algorithm that we detail in Appendix D to solve the agent-based model involves

$$\mathcal{O}\Big[N_t \cdot \Big(N_{\text{active-cars}} \cdot \max(v \cdot dt \cdot \rho_{\text{spots}}, n_{\text{out-streets}}) + N_{\text{parked-cars}} \Big) \Big]$$

operations, where N_t is the number of time steps (of duration dt each), $N_{\text{active-cars}}$ and $N_{\text{parked-cars}}$ are the average numbers of active and parked cars simultaneously present in the network, v is the car speed, and ρ_{spots} is the average linear density of parking spaces along a street, and $n_{\text{out-streets}}$ is the number of outgoing street-links at a typical intersection.

By comparison, once the stationary occupancy field (n_j) is known, solving the analytical equations giving the search time in real time units, Eq. S3, requires only

$$\mathcal{O}\Big[N_{\text{car-types}} \cdot N_{\text{inv}} \Big]$$

operations, where $N_{\text{car-types}}$ is the number of categories of drivers and N_{inv} is the complexity of solving a linear problem with a sparse matrix of linear size N_{nodes} (the number of parking

spaces), but including only around $N_{\text{nodes}} + N_{\text{street-links}} \cdot n_{\text{out-streets}}$ non-zero values, where $N_{\text{street-links}}$ is the number of street-links. Unfortunately, while solving such a linear problem is much faster than inverting the corresponding dense matrix, the time it takes will depend on the specific form of the matrix and we do not know any general upper bound on the complexity. In practice, however, the associated computational time will be almost negligible compared to the time needed to build the different matrices and derive the stationary occupancy via Eq. 15 (which has approximately the same complexity as Eq. S3, but must be performed iteratively until convergence, typically after ~ 100 iterations).

B.3. Capped search time in arbitrary and real time units

Equations 6 and 7 of the main text give the mean search time of drivers that will hypothetically keep cruising forever. In reality, one expects them to quit searching after a given time.

Let us first assume that this upper time bound is given as a number $K_{\max}$ of steps from node to node. Then, capping the search simply implies restricting the sum on K in Eqs 6 and 7 to $0 \leqslant K \leqslant K_{\max}$, which yields Eq. 8 for the probability $\bar{P}_j^{(\alpha)}$ to reach spot j and park there. The stationary occupancy field (n_j) can then be derived iteratively by inserting the capped reaching probabilities $\bar{R}_j^{(\alpha)} \equiv \bar{P}_j^{(\alpha)}/\tilde{p}_j$ into Eq. 15.

Once the occupancy field is known, one can turn to the capped search time expressed as a number of steps,

$$\begin{aligned}
\bar{\mathrm{T}}_s^{(\alpha)} &= H_i^{(\alpha)}(0) \cdot \left[\sum_{K=0}^{K_{\max}} K \, \underline{\underline{M}}^{(\alpha)K} + \sum_{K=K_{\max}+1}^{\infty} K_{\max} \, \underline{\underline{M}}^{(\alpha)K} \right]_{ij} \tilde{p}_j^{(\alpha)} \\
&= H_i^{(\alpha)}(0) \cdot \left[\sum_{K=0}^{K_{\max}} K \, \underline{\underline{M}}^{(\alpha)K} + K_{\max} \left(\underline{\underline{\mathbb{I}}} - \underline{\underline{M}}^{(\alpha)} \right)^{-1} \cdot \underline{\underline{M}}^{(\alpha)K_{\max}+1} \right]_{ij} \tilde{p}_j^{(\alpha)} \\
&= H_i^{(\alpha)}(0) \cdot \left[\sum_{K=0}^{K_{\max}} \sum_{l=0}^{K-1} \underline{\underline{M}}^{(\alpha)K} + K_{\max} \left(\underline{\underline{\mathbb{I}}} - \underline{\underline{M}}^{(\alpha)} \right)^{-1} \cdot \underline{\underline{M}}^{(\alpha)K_{\max}+1} \right]_{ij} \tilde{p}_j^{(\alpha)},
\end{aligned}$$

where one has arbitrarily defined as $K_{\max}$ the search time of cars that quit searching.

Using the following identity,

$$\sum_{K=0}^{K_{\max}} \sum_{l=0}^{K-1} \underline{\underline{M}}^{(\alpha)K} = \sum_{l=0}^{K_{\max}-1} \sum_{K=l+1}^{K_{\max}} M^K$$

$$= (\underline{\underline{\mathbb{I}}} - \underline{\underline{M}}^{(\alpha)})^{-1} \sum_{l=0}^{K-1} \underline{\underline{M}}^{(\alpha)^{l+1}} \left(\underline{\underline{\mathbb{I}}} - \underline{\underline{M}}^{(\alpha)^{K_{\max}-l}} \right)$$
$$= \underline{\underline{M}}^{(\alpha)} \cdot (\underline{\underline{\mathbb{I}}} - \underline{\underline{M}}^{(\alpha)})^{-2} \cdot (\underline{\underline{\mathbb{I}}} - \underline{\underline{M}}^{(\alpha)^{K_{\max}}}) - K_{\max} (\underline{\underline{\mathbb{I}}} - \underline{\underline{M}}^{(\alpha)})^{-1} \cdot \underline{\underline{M}}^{(\alpha)^{K_{\max}+1}},$$

we arrive at

$$\begin{aligned} \bar{\mathrm{T}}_s^{(\alpha)} &= H_i^{(\alpha)}(0) \cdot \left[\underline{\underline{M}}^{(\alpha)} \cdot (\underline{\underline{\mathbb{I}}} - \underline{\underline{M}}^{(\alpha)})^{-2} \cdot (\underline{\underline{\mathbb{I}}} - \underline{\underline{M}}^{(\alpha)^{K_{\max}}}) \right]_{ij} \tilde{p}_j^{(\alpha)} \\ &= \mathrm{T}_s^{(\alpha)} - H_i^{(\alpha)}(0) \cdot \left[(\underline{\underline{\mathbb{I}}} - \underline{\underline{M}}^{(\alpha)})^{-2} \cdot \underline{\underline{M}}^{(\alpha)^{K_{\max}+1}} \right]_{ij} \tilde{p}_j^{(\alpha)}. \end{aligned} \tag{S4}$$

As in the main text, Einstein's summation convention on repeated indices is implied.

To recover real time units, we calculate an average conversion factor between steps K and seconds using the case of unbound searches, by equating the (unbound) search time given by Eq. 7 and that given by Eq. 10. The maximum number of steps $K_{\max}$ is then first estimated from the maximum allowed time and the capped search time $\bar{\mathrm{T}}_s^{(\alpha)}$ is eventually converted into seconds on the same basis.

Unfortunately, evaluating the capped search time via Eq. S4 involves the computation of $\underline{\underline{M}}^{(\alpha)^{K_{\max}+1}}$ for $K_{\max} \gg 1$, which will not necessarily be a sparse matrix even if $\underline{\underline{M}}^{(\alpha)}$ is. This computation turns out to be numerically very demanding if the number of nodes in the graph is huge, as in the Lyon test case. However, the method is operational and quick on smaller networks. Consider for example the 'toy' network introduced in Fig. 2a. Capping search times to 180 s reduces the simulated mean search time all the more as the injected rate is high, as expected and shown on Fig. S1. These capped times are very well captured by the theoretical method outlined above, culminating in Eq. S4, as can be seen on Fig. S1.

C. Detailed Input for the Simulations of Lyon

Section 4 presented a large-scale application of the proposed framework in the case of the city of Lyon, France. In this application, drivers are classified into 36 categories according to their final destination (the list of which is given in Tab. S1). Their cars are injected into the networks at one of the 49 entry points enumerated in Tab. S3, with a relative probability inferred either from the population of the surrounding neighbourhood or on a rough estimation of the inflow from the periphery of the city, if the entry point is located at its boundary.

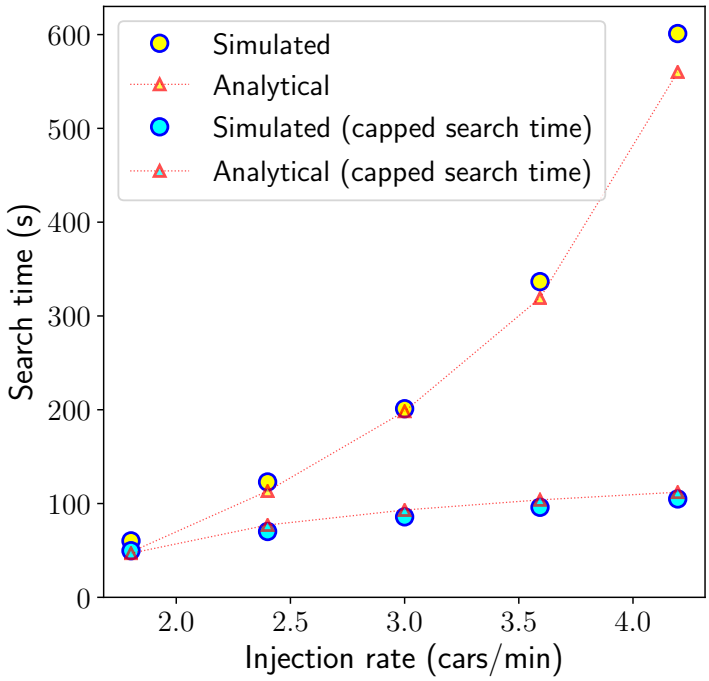

Figure S1 **Variations of the driving and cruising time in the 'toy' network of Fig. 2a (with free spots) with the car injection rate. The outcome of the simulation (circles) is compared with the analytical predictions (triangles), both in the case of unbound search times and when search times are capped to** $180\,\mathrm{s}$**.**

Label	Name	x	y	Probability
0	Saint-Rambert - industrie	841367.9	6523437.7	0.013
1	La Duchère	839095.5	6522460.5	0.009
2	Vaise	840269.9	6521523.7	0.047
3	Champvert – Point-du-jour	840501.8	6520877.9	0.016
4	Ménival - La Plaine	839660	6519395.7	0.012
5	Saint-Just	840845.5	6519201.4	0.012
6	Vieux-Lyon	841609.7	6519231.4	0.012
7	Chazière -Flammarion	841413.3	6520955.4	0.028
8	Cœur Croix-Rousse	842199.3	6521377.5	0.028
9	Les Chartreux	841921.4	6520617	0.018
10	Pentes	842580	6520652.1	0.013
11	Terreaux – Cordeliers	842555.7	6520004.7	0.019
12	Ainay	842195	6519007.4	0.026
13	Sud Perrache	841731.4	6517736.3	0.031
14	Tête d'or – Foch	843412.4	6520782.8	0.04
15	Brotteaux – Europe	844411.3	6520546.7	0.028
16	Bellecombe – Thiers	844989.5	6519947.4	0.041
17	Mutualité-Préfecture	843080.8	6519272	0.032
18	Part-Dieu - Bir Hakeim	843782.1	6519416.5	0.032
19	Paul Bert – Villette	845042.8	6519497.8	0.036
20	Dauphiné - Sans Souci	845321.7	6518826.4	0.032
21	Montchat	846350.6	6518432.2	0.032
22	Jean Macé	842992.7	6518754.3	0.037
23	Guillotière	843092.8	6518310.3	0.047
24	Blandan	843950.4	6518243.7	0.047
25	Gerland nord	842799.7	6517560.4	0.047
26	Gerland sud	842756.2	6516727.5	0.047
27	Grand Trou – Moulin à vent	844554.8	6516452.7	0.027
28	Monplaisir	844588.2	6517391.2	0.027
29	Le Bachut	844966.5	6517711.6	0.036
30	Etats-Unis	845574.1	6517087.7	0.023
31	Mermoz – Laennec	846128.7	6516429	0.022
32	Général André - Santy	846210.7	6515824.3	0.022
33	Pinel	845979	6518623	0.014
34	6e arrondissement Sud	843836.9	6520125.1	0.040
35	5e arrondissement Sud	838846.3	6518189.5	0.012

Table S1 List of the 36 destinations implemented in our study of Lyon. The 'probability' column specifies the fraction of cars bound to a given destination. Coordinates are given in the RGF-93/Lambert-93 reference system.

Label	x	y	Proba of injection $I_i(0)$
0	843661	6520685	0.017
1	841446	6516539	0.161
2	844237	6520113	0.017
3	840046	6520488	0.024
4	842098	6515492	0.011
5	846715	6516345	0.048
6	846268	6515772	0.012
7	843023	6521479	0.017
8	841709	6517901	0.014
9	844843	6520013	0.017
10	846215	6518226	0.020
11	843218	6515591	0.011
12	844862	6519680	0.020
13	842529	6520514	0.007
14	844989	6517334	0.012
15	839541	6521161	0.064
17	842670	6517036	0.011
18	844226	6518050	0.011
20	841862	6516427	0.011
21	841071	6522242	0.017
22	847245	6517797	0.039
23	841429	6520673	0.007
24	844289	6516742	0.012
25	842837	6518111	0.011
26	842041	6520903	0.007
27	839305	6519151	0.012
28	839305	6519151	0.012
29	841647	6520269	0.012
30	845762	6518982	0.023
31	839240	6521852	0.024
32	846693	6515848	0.012
33	844198	6515464	0.048
34	843204	6518705	0.011
35	844645	6516244	0.048
36	840767	6519147	0.012
38	845656	6516640	0.012
39	843257	6519411	0.020
40	844439	6520869	0.042
41	844554	6517528	0.012
42	844889	6516878	0.012
43	842446	6521030	0.007
44	844431	6519483	0.020
45	845021	6519236	0.020
47	842325	6519031	0.015
48	839654	6522705	0.023

Table S3 List of the 49 entry points considered for the injection of cars into the street network, with their relative probabilities. Coordinates are given in the RGF-93/Lambert-93 reference system.

D. Numerical Implementation of the Agent-based Model

The numerical implementation of the agent-based model relies on the C++ classes:

City
Attributes: name, number of car categories, list of streets, list of nodes, matrices of turn-choices for all car categories, list of active cars, list of parked cars, various storage vectors

Street
Attributes: name, identifier, start and end nodes, number of spots, length, effective speed, various storage vectors

Node
Attributes: identifier, GPS coordinates, lists of incoming and outgoing streets

Spot
Attributes: (inherits from Node), identifier, vacancy status, attractiveness, various storage vectors

Car
Attributes: car number, car type, car status (active, searching for parking, parked, frozen, exited), destination, beta, position (on street or spot), matrix of turn-choices, various timestamps.

The main loop of the script runs as explained in Alg. 1 in pseudo-code:

References

Benenson I, Martens K, Birfir S, 2008 *Parkagent: An agent-based model of parking in the city. Computers, Environment and Urban Systems* 32(6):431–439.

Dieussaert K, Aerts K, Steenberghen T, Maerivoet S, Spitaels K, 2009 *Sustapark: an agent-based model for simulating parking search. AGILE International Conference on Geographic Information Science, Hannover*, 1–11.

Dowling CP, Ratliff LJ, Zhang B, 2019 *Modeling curbside parking as a network of finite capacity queues. IEEE Transactions on Intelligent Transportation Systems* 21(3):1011–1022.

Fulman N, Benenson I, 2021 *Approximation method for estimating search times for on-street parking. Transportation Science* .

Fulman N, Benenson I, Elia EB, 2020 *Modeling parking search behavior in the city center: A game-based approach. Transportation Research Part C: Emerging Technologies* 120:102800.

Horni A, Montini L, Waraich RA, Axhausen KW, 2013 *An agent-based cellular automaton cruising-for-parking simulation. Transportation Letters* 5(4):167–175.

Ratliff LJ, Dowling C, Mazumdar E, Zhang B, 2016 *To observe or not to observe: Queuing game framework for urban parking. 2016 IEEE 55th Conference on Decision and Control (CDC)*, 5286–5291 (IEEE).

Tavafoghi H, Poolla K, Varaiya P, 2019 *A queuing approach to parking: Modeling, verification, and prediction. arXiv preprint arXiv:1908.11479* .

Algorithm 1 Main loop

for t in all time steps ($dt = 1\,\mathrm{s}$) **do**
- draw number of new cars to be injected from a Poisson distribution
- **for** car in cars to be injected **do**
  - compute the normalized cumulative distribution function (cdf) of all probabilities at all possible injection nodes
  - select an injection node among the possibilities by mapping a random number onto the interval of the computed cdf
  - inject the car at the chosen node
- **end for**
- **for** car in active cars **do**
  - update the perceived tension β
  - **while** current timestep is not over **do**
    - move the car so far as possible along current street
    - store in a list L_{spots} the spots by which the car has driven
    - compute cumulative distribution function (cdf) of probabilities of all possible parking choices in L_{spots} based on attractiveness
    - **for** spot in L_{spots} **do**
      - decide if car parks here by mapping a random number on the interval of the computed cdf
      - **if** car parks here **then**
        - end 'WHILE' loop
      - **end if**
    - **end for**
    - **if** car has reached end of street **then**
      - compute the cdf of probabilities of all possible transitions available from the node at the end of street
      - select the outgoing street in which to move the car in the usual way
      - move the car into next street
    - **end if**
    - record car position
    - update leftover time in current timestep
  - **end while**
- **end for**
- **for** car in parked cars **do**
  - compute departure rate
  - randomly decide if car is removed (departs)
- **end for**
- keep track of the global balance of injected, active, parked and departed cars

end for