Exploring Computational Complexity of Ride-Pooling Problems

Computational complexity of ride-pooling in Amsterdam with ExMAS

Usman Akhtar, Rafal Kucharski

Jagiellonian University, Krakow, POLAND

JAGIELLONIAN UNIVERSITY IN KRAKÓW

Problem

We know that the size of the ride-pooling problem explodes. But: how?, when? and why?

Abstract

Ride-pooling is computationally challenging. The number of feasible rides grows with the number of travelers and the degree (capacity of the vehicle to perform a pooled ride) and quickly explodes to the sizes making the problem not solvable analytically.
Here, we explore it in more detail and provide an experimental underpinning to this open research problem. We trace how the size of the search space and computation time needed to solve the ride-pooling problem grows with the increasing demand and greater discounts offered for pooling.

Problem

We report the computational complexity of real-world ride-pooling problems (Amsterdam, The Netherlands) & trace the:

- search space sizes,
- computation times,
- ride-pooling performance,
- and properties of underlying *shareability graphs*.

Theoretical search space

The search space S of ride-pooling problem for Q travellers can be expressed as a number of possible subsets of size d in the set of all the travelers requesting pooled rides Q. This is further multiplied with the order in which these travelers are picked up ($d!$ combinations), and dropped-off ($d!$ again), which yields a theoretical formula of:

$$S = \binom{Q}{d} d!d! \quad (1)$$

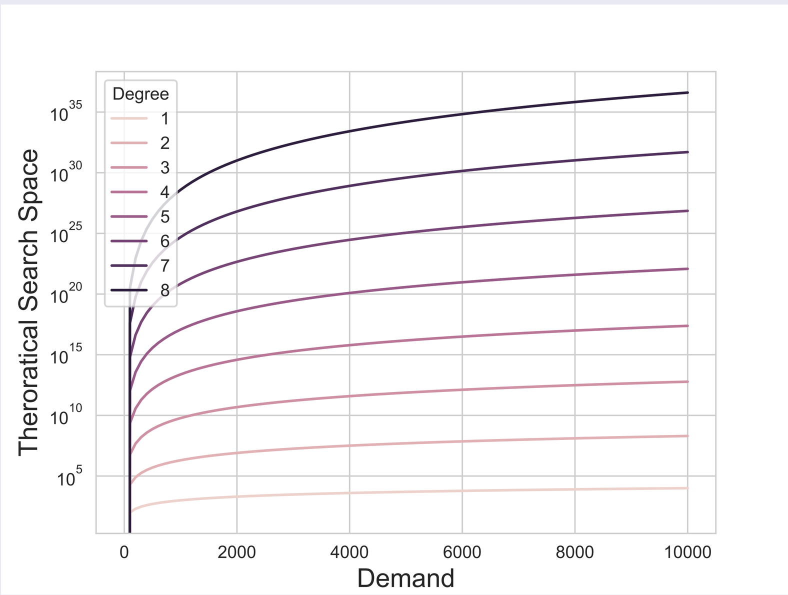


Figure: Theoretically computed search space of ride-pooling problems. The search space grows significantly in the log-scale and the growth is huge with respect to the number of travelers requesting pooled trips (x-axis), yet more importantly to the number of travelers riding together (degree)

Methodology - ExMAS algorithm

- We use our ExMAS algorithm (Kucharski and Cats, 2020, TR part B), an offline algorithm that addresses the complexity of the ride-sharing problem via the utility-driven approach.
- It explicitly restricts the search space to the attractive rides for which the utility of sharing exceeds the utility of travelling alone.
- The formula to filter for attractive rides only involves ride-pooling discount (λ), detour (t^s), delay (t^d) and behavioural willingness-to-sahare (β^s):

$$\Delta U = U^s - U^{ns} = \beta^c \lambda l + \beta^t \left(t - \beta^s \left(t^s + \beta^d t^d\right)\right) \quad (2)$$

- Thanks to those utility-based formulas the computations implode rather than explode:

	degree:	1	2	3	4	5	6	7
search space:	theoretical	3.00×10^3	3.60×10^7	6.47×10^{11}	1.55×10^{16}	4.65×10^{20}	1.67×10^{25}	7.01×10^{29}
	explored	3000	8997000	1807	226	123	24	0
	attractive	3000	5270	243	130	76	8	0
	assigned	1422	435	160	44	8	2	0

Table: Search space and its reduction for a sample of 3000 trips in Amsterdam.

- ExMAS is publicly available at `https://github.com/RafalKucharskiPK/ExMAS` along with reproducible examples. ExMAS can be used as a python library with `pip install exmas`

Results

Experimental setting

We run the ExMAS algorithm for the detailed network of Amsterdam.
We explored with the varying:

- shared discount range λ in 5, 10, 20, 25, 30, 35, 40 % lower than private ride.
- demand levels ranging from 300 to 3600 trips per hour (50-600 requests in 10 a minute batch).

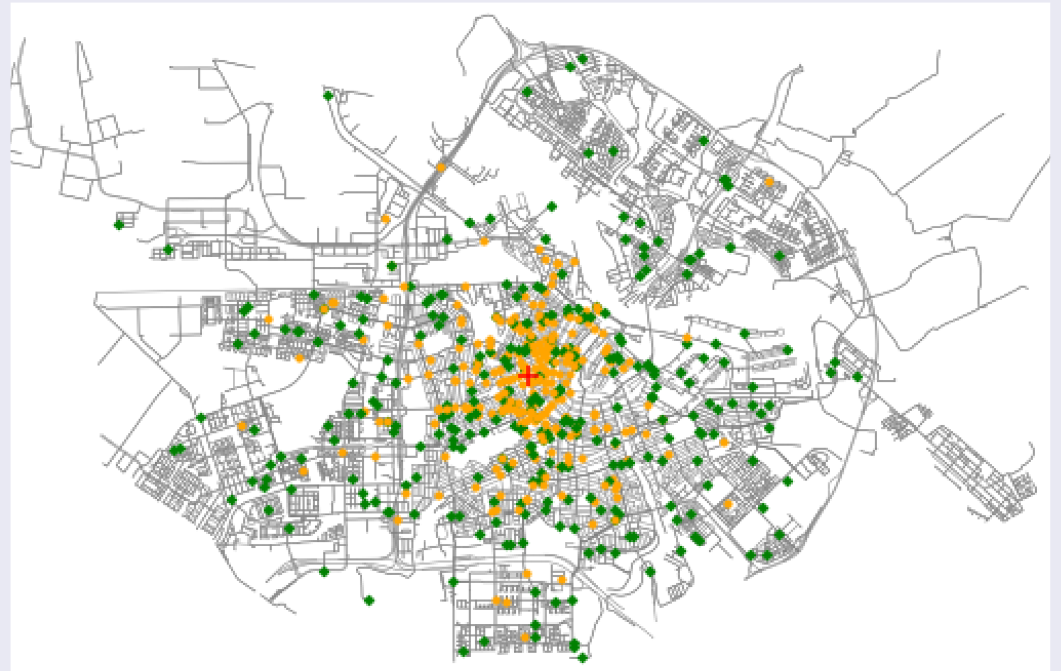
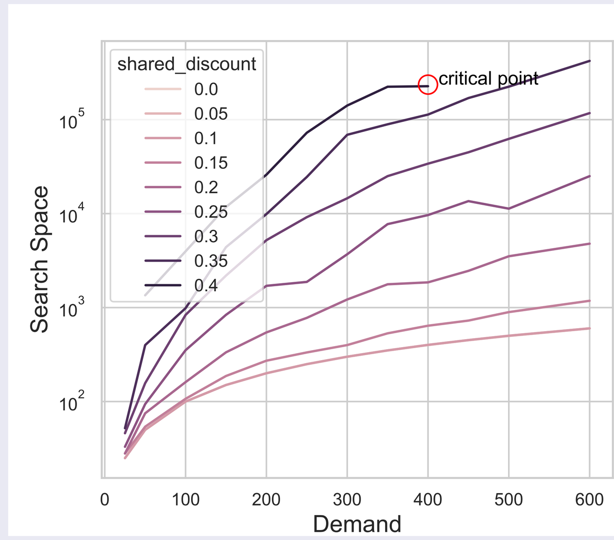


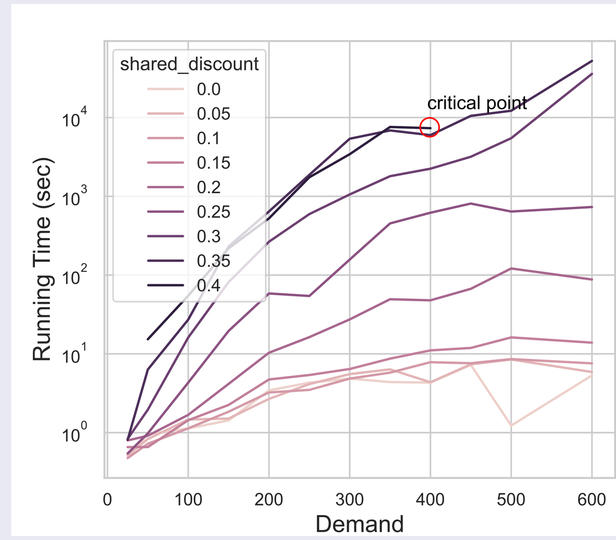
Figure: Synthetic demand for ride-pooling in Amsterdam, Netherlands used in our experiments. The origins marked green and destinations orange

Ride-pooling complexity and computation times

We can see that both the demand and discounts for shared rides λ have a strong impact on the size of the search space. Varying the demand and discount affects the overall performance, until it reaches a critical point; where computation become intractable.



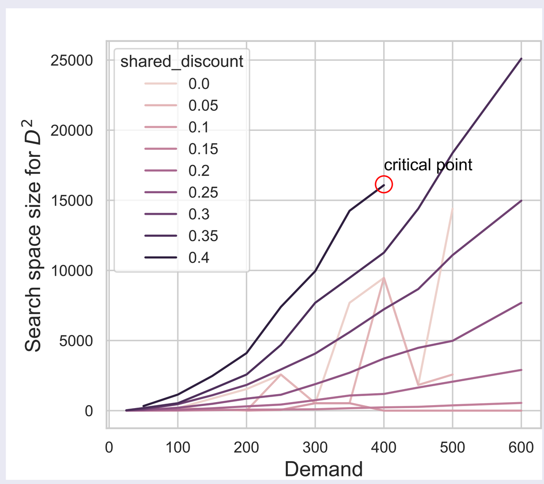
(a) Number of feasible rides explored with ExMAS algorithm



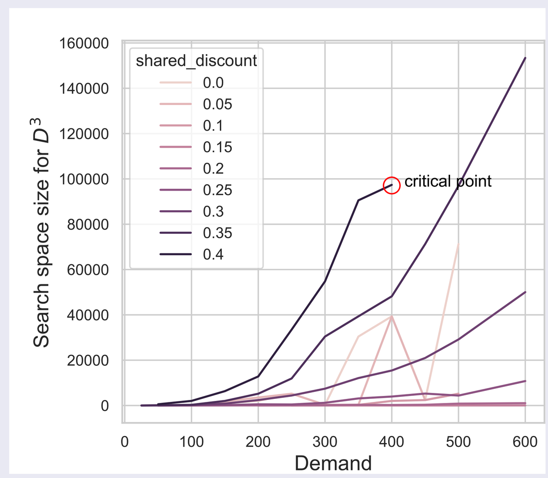
(b) Running time needed to solve ride-pooling problems with ExMAS

Trips of higher degree

For the lower discount levels the relation is linear, yet when greater discounts are offered, number of identified feasible pairs grows exponentially



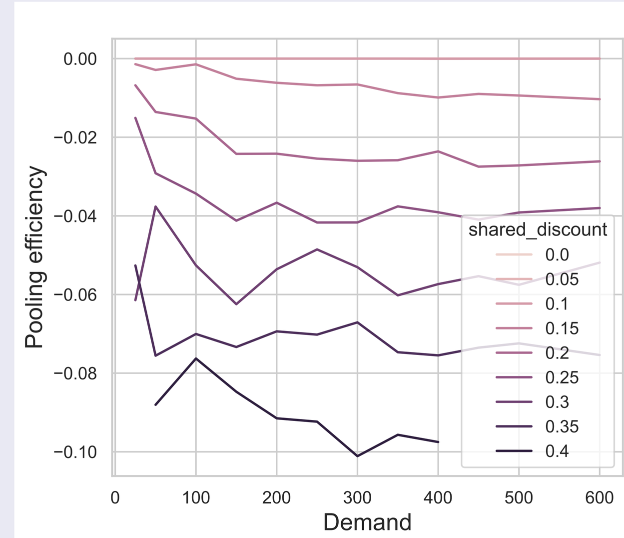
(c) Number of feasible rides of second degree identified for various demand levels (x-axis) and discounts (colours).



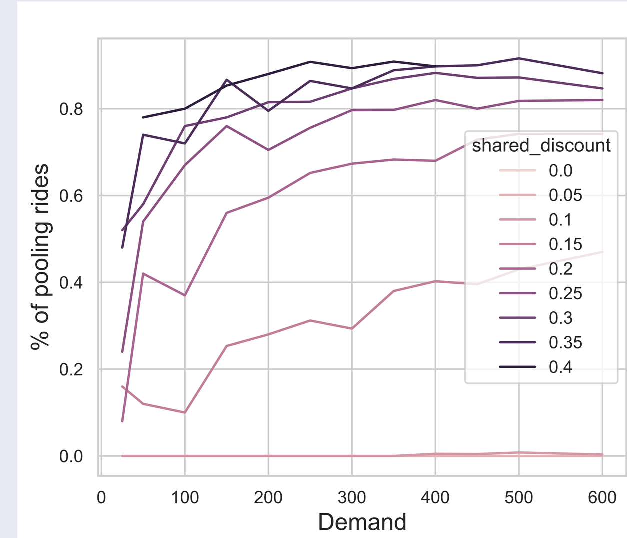
(d) Number of feasible rides of third degree (shared by three co-travellers).

Ride-pooling performance and Computational Complexity:

The greater complexity does not bring more efficiency - the performance hardly improves

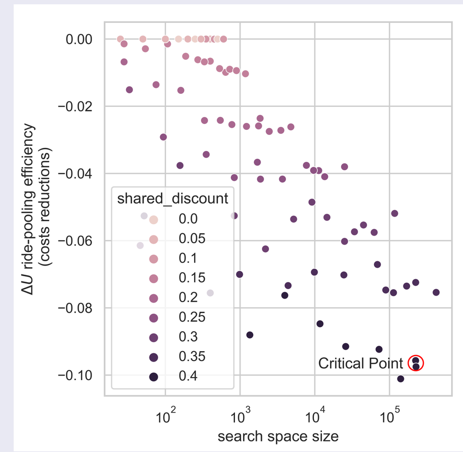


(e) Efficiency of pooling rides. The trend is most visible with the discounts offered, and (to lesser extent) with the demand levels

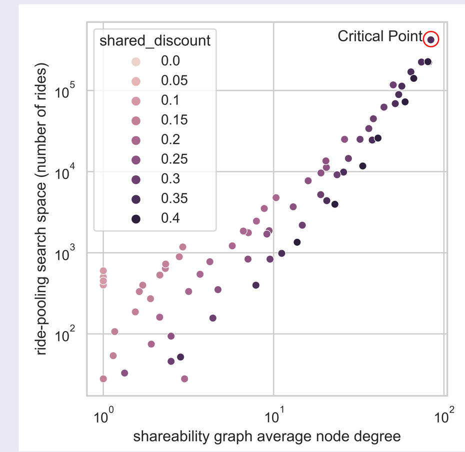


(f) It grows both with the demand level as well as the discounts offered. It exceeds 80% for discounts greater than 25% and demand levels above 300 - beyond which it stabilizes

Approach to harness the search space without the negative impact on the performance: a strong trend between the shareability graphs and search space.



(g) Relation between pooling efficiency (y-axis) and search space size. While there is a strong relation it is not always linear and evident. For instance the benefits of pooling are stable at the 7% for search space sizes varying from 10^3 to 10^6



(h) Strong trend (on log-log plot) between the properties of the so-called pair-wise shareability graph (x-axis) and search space of the problem. While for the lower demands and discounts there is still a significant dispersion, the trend becomes more profound with increasing search space.

Conclusion

This paper investigates the computational complexity of the ride-pooling problem.

- Our findings provide convincing evidence that shows the impact of λ on running time and search space complexity,
- While for 20% discount 200 requests takes ca 10s to compute (fig.1), for 500 trip requests it grows to 100 seconds. Yet for 600 trip requests this 100 seconds grows to 2.8 hours when discount is increased - which is hardly acceptable for real-time ride-pooling problems. For the mid-size demand of 500 trip request, the computation time increases to 100 seconds for 15% discount. The search space grows to 10^5 rides for the batch of 500 trips when the discount increases at the 40% discunt the computation becomes intractable.
- The main driver of the search space explosion is in the rides of thirds degree and more, which reach up to 150 000 feasible trips in our experiments, while number of pairs did not exceeded 25 000.
- The greater search space does not necessarily improve the ride-pooling performance

Acknowledgements

This research is funded by National Science Centre in Poland program OPUS 19 (Grant Number 2020/37/B/HS4/01847).

Join us

ERC Starting Grant on interactions between machines and humans in urban mobility.
PhD and PostDoc positions in one of top tourist destinations worldwide (Krakow) at the University est. 1364

European Research Council
Established by the European Commission

rafal.kucharski@uj.edu.pl