# A Quantitative Comparison of Solo and Shared Ride

Sachin K Salim
under the guidance of Swaprava Nath

Department of Computer Science and Engineering
IIT Kanpur

February 2, 2020

# Introduction

- In urban India, it's observed that people prefer travelling alone rather than pooling their ride for day-to-day activities even though the latter offers cheaper fare.
- This could be attributed to a variety of reasons including comfort, waiting time and security.

# Introduction

- In urban India, it's observed that people prefer travelling alone rather than pooling their ride for day-to-day activities even though the latter offers cheaper fare.
- This could be attributed to a variety of reasons including comfort, waiting time and security.
- But if more people prefer to pool their rides in a locality, the total distance travelled by all the cars reduces, thereby saving fuel and making a lesser impact on environment.
- We make an attempt to quantify this improvement by modelling the locations in a city on a graph comparing the expected lengths of shortest routes when all passengers travel solo to that of shared ride.

# Literature Review

- In a paper by M. Furuhuta et. al, the authors discuss about the history of ridesharing and classify it based on the target market, service type, matching activity, pricing policy, etc.
- D. C. Parkes et. al, proposed a new pricing mechanism called *Spatio-Temporal Pricing* that is smooth in space and time. This ensures that drivers will not reject the trip dispatched to them and keeps them from flocking in a particular area for higher prices.

# Literature Review

- In a paper by M. Furuhuta et. al, the authors discuss about the history of ridesharing and classify it based on the target market, service type, matching activity, pricing policy, etc.
- D. C. Parkes et. al, proposed a new pricing mechanism called *Spatio-Temporal Pricing* that is smooth in space and time. This ensures that drivers will not reject the trip dispatched to them and keeps them from flocking in a particular area for higher prices.
- Almost the entire literature in ridesharing is focused on optimizing the ride for passengers and drivers. Here, our focus is on matching the passengers to a driver such that there is least impact on the environment.

# Ridesharing problem (RSP)

**Problem Statement**

Given a city with a list of passengers having a pickup point and a drop-off point, what is the shortest route of a driver to pick up all passengers and drop them off given the driver's current location?

# Notations

- $\mathcal{P} = 1, 2, \ldots, k$: Set of passengers in ridesharing.
- $\mathcal{L} \subset \mathbb{N} \times \mathbb{N}$, Set of locations.
- $p_i, d_i \in \mathcal{L}$: pickup and drop-off locations of passenger $i$
- $\delta : \mathcal{L} \times \mathcal{L} \to \mathbb{N}$ where $\delta(a, b)$ gives the distance between locations $a$ and $b$. For simplicity we take $\delta$ to be the manhattan distance.

# Notations

- $\mathcal{P} = 1, 2, \ldots, k$: Set of passengers in ridesharing.
- $\mathcal{L} \subset \mathbb{N} \times \mathbb{N}$, Set of locations.
- $p_i, d_i \in \mathcal{L}$: pickup and drop-off locations of passenger $i$
- $\delta : \mathcal{L} \times \mathcal{L} \to \mathbb{N}$ where $\delta(a, b)$ gives the distance between locations $a$ and $b$. For simplicity we take $\delta$ to be the manhattan distance.
- $\mathcal{D}$ : the driver.
- $o_{\mathcal{D}} \in \mathcal{L}$ : origin of the driver $\mathcal{D}$, i.e., the initial location.
- $\mathcal{C}$: Capacity of $\mathcal{D}$'s vehicle.

# Notations

- $\mathcal{P} = 1, 2, \ldots, k$: Set of passengers in ridesharing.
- $\mathcal{L} \subset \mathbb{N} \times \mathbb{N}$, Set of locations.
- $p_i, d_i \in \mathcal{L}$: pickup and drop-off locations of passenger $i$
- $\delta : \mathcal{L} \times \mathcal{L} \to \mathbb{N}$ where $\delta(a, b)$ gives the distance between locations $a$ and $b$. For simplicity we take $\delta$ to be the manhattan distance.
- $\mathcal{D}$ : the driver.
- $o_{\mathcal{D}} \in \mathcal{L}$ : origin of the driver $\mathcal{D}$, i.e., the initial location.
- $\mathcal{C}$: Capacity of $\mathcal{D}$'s vehicle.
- $\mathcal{R}$: Set of feasible routes of $\mathcal{D}$.
- $r \in \mathcal{R}$: A sequence of locations in $\mathcal{L}$

# Feasible route

A route $r$ is feasible iff it satisfies the following three conditions:

1. The route $r$ covers all the passengers' pickup and drop-off point. Thus $|r| = 2|\mathcal{P}|$.

# Feasible route

A route $r$ is feasible iff it satisfies the following three conditions:

1. The route $r$ covers all the passengers' pickup and drop-off point. Thus $|r| = 2|\mathcal{P}|$.
2. The pickup point of a passenger $i$ comes before the drop-off point for all $i \in \mathcal{P}$.

# Feasible route

A route $r$ is feasible iff it satisfies the following three conditions:

1. The route $r$ covers all the passengers' pickup and drop-off point. Thus $|r| = 2|\mathcal{P}|$.
2. The pickup point of a passenger $i$ comes before the drop-off point for all $i \in \mathcal{P}$.
3. At any location $a \in r$, number of pickups until $a$ must not be more than the sum of number of drop-offs until $a$ and the capacity $\mathcal{C}$.

# Objective

Our objective is to find the shortest length route $r^* \in \mathcal{R}$ such that

$$r^* = \operatorname*{argmin}_{r \in \mathcal{R}} \sum_{i=1}^{|r|-1} \delta(r_i, r_{i+1})$$

where $r_i$ is the $i^{th}$ location in the route $r$.

# Approach

## Solo riding problem

- An order $(\pi_1, \ldots, \pi_k)$ is decided over the $k$ passengers.
- The driver $\mathcal{D}$ picks up each of these passengers and then drops them off before picking another passenger in the order given by $\pi$.
- The algorithm would have to figure out the order of the passengers to be followed so that the total length of the travel route is minimized.

# Approach

## Shared riding problem

- All permutations of $(p_1, d_1, \ldots, p_k, d_k)$ are considered such that:
  - $p_i$ comes before $d_i$ for all $i \in \mathcal{P}$
  - the occupied passengers do not cross the capacity of cab

  when the passengers are served in the given order.
- The problem is to find out the order that minimizes the cost of the total travel.

# Sample routes

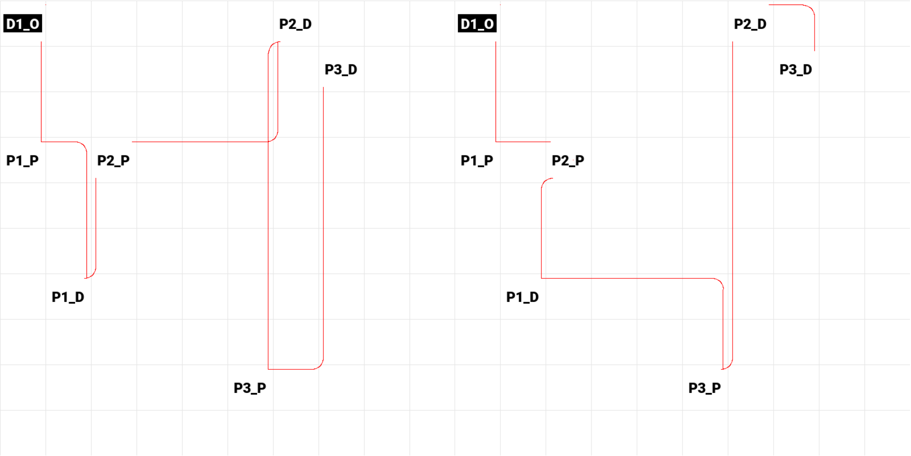


Figure: Left: A sample solo-ride route (Distance = 36)
Right: A sample shared-ride route (Distance = 26)

# Our objective

## Our objective

The problem in hand is to compare the following two values:

- minimum cost path in solo-riding
- minimum cost path in shared-riding

# Open Travelling Salesman Problem (OTSP)

## Open Travelling Salesman Problem

OTSP is a variation of TSP in which the salesman does not return to the starting point. The problem is to find out the shortest Hamiltonian path in a graph G starting from a given vertex.

## Formal definition of OTSP

Given a complete graph $G = (V, E)$ with positive weighted edges and a starting vertex $s \in V$, determine if there exists a Hamiltonian path whose sum of weights of edges in the path is not more than a given number $k$.

# NP-hardness of OTSP

Input: graph $G$; an instance of Hamiltonian path problem.
Output: graph $G'$; by the reduction of HP to OTSP.

# NP-hardness of OTSP

Input: graph $G$; an instance of Hamiltonian path problem.
Output: graph $G'$; by the reduction of HP to OTSP.

1. Create a complete graph $G'(V', E')$ with $V' = V$, $s' = s$ and $E' = \{(i,j) : i, j \in V,\ i \neq j\}$.
2. Define the weight function $w : E \rightarrow \mathbb{N}$ as:

$$w(e) = \begin{cases} 0 & \text{if } e \in E \\ 1 & \text{if } e \notin E \end{cases}$$

3. Pass the instance $(G', 0)$ as input to the OTSP problem.

# NP-hardness of OTSP

A Hamiltonian path exists in $G$ starting from $s$
$\implies$ the cost of each corresponding edge in $G'$ would be 0.
$\implies$ the total cost of that HP in $G'$ would be 0.

# NP-hardness of OTSP

A Hamiltonian path exists in $G$ starting from $s$
$\implies$ the cost of each corresponding edge in $G'$ would be 0.
$\implies$ the total cost of that HP in $G'$ would be 0.

No HP exists in $G$
$\implies$ The cost of any HP in $G'$ must be more than 0.

# NP-hardness of OTSP

A Hamiltonian path exists in $G$ starting from $s$
$\implies$ the cost of each corresponding edge in $G'$ would be 0.
$\implies$ the total cost of that HP in $G'$ would be 0.

No HP exists in $G$
$\implies$ The cost of any HP in $G'$ must be more than 0.

$G$ has a Hamiltonian path starting at $s \iff G'$ has a Hamiltonian path of cost at most 0 starting at $s'$ .
Hence, OTSP is NP-hard.

# NP-hardness of RSP

**solo-RSP**

We are given a graph $G(V, E)$, an instance of OTSP problem and we show here how to convert this to an instance of solo-RSP.

# NP-hardness of RSP

**solo-RSP**

We are given a graph $G(V, E)$, an instance of OTSP problem and we show here how to convert this to an instance of solo-RSP.

- Define a bijective map $\psi : V \to \mathcal{L}$, from vertex set $V$ to locations $L$.
- Construct a driver $\mathcal{D}$ with origin as $o_{\mathcal{D}} := \psi(s)$.

# NP-hardness of RSP

**solo-RSP**

We are given a graph $G(V, E)$, an instance of OTSP problem and we show here how to convert this to an instance of solo-RSP.

- Define a bijective map $\psi : V \rightarrow \mathcal{L}$, from vertex set $V$ to locations $L$.
- Construct a driver $\mathcal{D}$ with origin as $o_{\mathcal{D}} := \psi(s)$.
- Construct $|V| - 1$ passengers with each passenger's pickup and drop-off location, $p_i = d_i = \psi(v_i), v_i \in V/s$.
- Define $\delta(a, b) := w(\psi^{-1}(a), \psi^{-1}(b))$

# NP-hardness of RSP

**solo-RSP**

We are given a graph $G(V, E)$, an instance of OTSP problem and we show here how to convert this to an instance of solo-RSP.

- Define a bijective map $\psi : V \to \mathcal{L}$, from vertex set $V$ to locations $L$.
- Construct a driver $\mathcal{D}$ with origin as $o_{\mathcal{D}} := \psi(s)$.
- Construct $|V| - 1$ passengers with each passenger's pickup and drop-off location, $p_i = d_i = \psi(v_i), v_i \in V/s$.
- Define $\delta(a, b) := w(\psi^{-1}(a), \psi^{-1}(b))$

It is evident from the above definition that an HP with length $\leq$ k starting at $s$ exists in OTSP instance iff a route exists in the translated solo-RSP instance with length $\leq$ k and driver's origin at $o_{\mathcal{D}}$. Hence, solo-RSP is NP-hard.

# NP-hardness of RSP

**shared-RSP**

- Shared RSP is a superclass of solo-RSP problem as it deals with capacity of cab with any finite natural number.
- Since the instances over which the minimum is to be found out in shared-RSP is a superset of that of solo-RSP, it's intuitive that shared-RSP is also NP-hard.

# Dynamic Programming Approach

Let's discuss the DP approach [5] to solve OTSP problem. This algorithm runs in $O(k^2 * 2^k)$ time where $k$ is the number of passengers.

**Algorithm**

$\mathcal{V}$ = set of vertices;
$s$ = starting vertex;
$w(a, b)$ = weight of the edge $(a, b)$;
**for** $\mathcal{S} \subseteq \mathcal{V}$ *with* $|\mathcal{S}| = 2$ *and* $s \in S$ **do**
    $C(\mathcal{S}, i) = w(s, i)$;
**end**

**Algorithm contd.**

**for** $k \leftarrow 2$ **to** $n-1$ **do**
    **for** $\mathcal{S} \subseteq \mathcal{V}$ *with* $|\mathcal{S}| = k$ *and* $s \in S$ **do**
        **for** $i \in \mathcal{S} \setminus \{s\}$ **do**
            $C(\mathcal{S}, i) = \min\{C(\mathcal{S} \setminus \{i\}, j) + w(j, i)\} \quad \forall j \in \mathcal{S} \setminus \{i, s\};$
        **end**
    **end**
**end**

# Dynamic Programming Approach

**Algorithm contd.**

**for** $k \leftarrow 2$ **to** $n-1$ **do**
  **for** $\mathcal{S} \subseteq \mathcal{V}$ *with* $|\mathcal{S}| = k$ *and* $s \in S$ **do**
    **for** $i \in \mathcal{S} \setminus \{s\}$ **do**
      $C(\mathcal{S}, i) = \min\{C(\mathcal{S} \setminus \{i\}, j) + w(j, i)\} \quad \forall j \in \mathcal{S} \setminus \{i, s\};$
    **end**
  **end**
**end**
$C(\mathcal{V}) = \infty;$
**for** $i \in V \setminus \{s\}$ **do**
  $C(\mathcal{V}, i) = \min\{C(\mathcal{V} \setminus \{i\}, j) + w(j, i)\} \quad \forall j \in \mathcal{V} \setminus \{i, s\};$
  $C(\mathcal{V}) = \min(C(\mathcal{V}), C(\mathcal{V}, i));$
**end**
**Result:** $C(\mathcal{V})$

# Pricing

The fare of passenger $i \in \mathcal{P}$ is given by Shapley value [1] defined as follows:

$$Sh_i(\mathcal{P}, v) = \frac{\alpha}{n!} \sum_{S \subseteq \mathcal{P} \setminus \{i\}} |S|! \; (|\mathcal{P}| - |S| - 1)! \; \Big( v(S \cup \{i\}) - v(S) \Big)$$

where $\alpha$ is the fare per unit distance and $v(S)$ for any $S \subseteq \mathcal{P}$ is the length of the shortest route to serve all the passengers in $S$ and none of the passengers not in $S$.

# Pricing

The fare of passenger $i \in \mathcal{P}$ is given by Shapley value [1] defined as follows:

$$Sh_i(\mathcal{P}, v) = \frac{\alpha}{n!} \sum_{S \subseteq \mathcal{P} \setminus \{i\}} |S|! \ (|\mathcal{P}| - |S| - 1)! \ \Big( v(S \cup \{i\}) - v(S) \Big)$$

where $\alpha$ is the fare per unit distance and $v(S)$ for any $S \subseteq \mathcal{P}$ is the length of the shortest route to serve all the passengers in $S$ and none of the passengers not in $S$.
The Shapley value fare ensures individual rationality by distributing the total fare among the passengers in such a way that no passenger pays more for solo ride than shared ride.

# Approximate algorithm using MST

We'll show here an approximate algorithm to OTSP. Using the solution to this algorithm, we could compute the solution to the **solo RSP** problem by translating this solution.
We describe a method [4] to get a 2-approximate solution to solo-ride ridesharing problem.

# Approximate algorithm using MST

We'll show here an approximate algorithm to OTSP. Using the solution to this algorithm, we could compute the solution to the **solo RSP** problem by translating this solution.

We describe a method [4] to get a 2-approximate solution to solo-ride ridesharing problem.

**Algorithm**

1. Construct Minimum Spanning Tree $T$ of $G$ with $o_{\mathcal{S}}$ as root using Prim's Algorithm.

# Approximate algorithm using MST

We'll show here an approximate algorithm to OTSP. Using the solution to this algorithm, we could compute the solution to the **solo RSP** problem by translating this solution.
We describe a method [4] to get a 2-approximate solution to solo-ride ridesharing problem.
**Algorithm**

1. Construct Minimum Spanning Tree $T$ of $G$ with $o_{\mathcal{S}}$ as root using Prim's Algorithm.
2. Let $\mathcal{W}$ be the pre-order walk in $T$. A pre-order walk is a sequence of vertices visited in the order of DFS of $T$ along with its return. See fig: 2 for an example.

# Approximate algorithm using MST

We'll show here an approximate algorithm to OTSP. Using the solution to this algorithm, we could compute the solution to the **solo RSP** problem by translating this solution.
We describe a method [4] to get a 2-approximate solution to solo-ride ridesharing problem.
**Algorithm**

1. Construct Minimum Spanning Tree $T$ of $G$ with $o_{\mathcal{S}}$ as root using Prim's Algorithm.
2. Let $\mathcal{W}$ be the pre-order walk in $T$. A pre-order walk is a sequence of vertices visited in the order of DFS of $T$ along with its return. See fig: 2 for an example.
3. Return the Hamiltonian tour $\mathcal{R}_{apx}$ visited in the order of the above pre-order walk.

# Approximate algorithm using MST

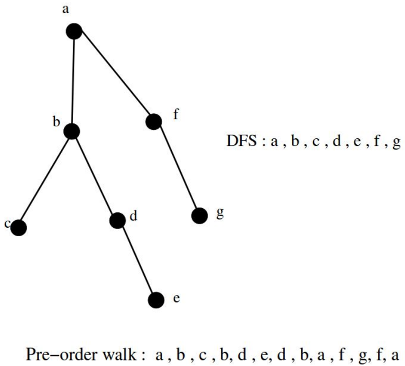


Figure: Example: Pre-order walk (credits: [4])

# Approximate algorithm using MST

**Proof**

The above algorithm runs in polynomial time since Prim's Algorithm has a polynomial complexity.

Let $\mathcal{R}_{opt}$ be the shortest Hamiltonian tour in $G$. Let $C(S_e)$ is the sum of the weights of the edges in $S_e$.

# Approximate algorithm using MST

**Proof**

The above algorithm runs in polynomial time since Prim's Algorithm has a polynomial complexity.

Let $\mathcal{R}_{opt}$ be the shortest Hamiltonian tour in $G$. Let $C(S_e)$ is the sum of the weights of the edges in $S_e$.

Since $\mathcal{W}$ is the MST,

$$C(T) \leq C(\mathcal{R}_{opt})$$

since $\mathcal{R}_{opt}$ is also a spanning tree.

# Approximate algorithm using MST

**Proof**

The above algorithm runs in polynomial time since Prim's Algorithm has a polynomial complexity.

Let $\mathcal{R}_{opt}$ be the shortest Hamiltonian tour in $G$. Let $C(S_e)$ is the sum of the weights of the edges in $S_e$.

Since $\mathcal{W}$ is the MST,

$$C(T) \leq C(\mathcal{R}_{opt})$$

since $\mathcal{R}_{opt}$ is also a spanning tree.

Since in the pre-order walk $\mathcal{W}$, every edge is visited twice except the final few edges since it doesn't come back to the starting point,

$$c(\mathcal{W}) \leq 2 * C(T)$$

. Combining both the inequalities above, we get,

$$c(\mathcal{W}) \leq 2 * C(\mathcal{R}_{opt})$$

# Approximate algorithm using MST

Now, observe that $\mathcal{R}$ is effectively a 'short-cut' of $\mathcal{W}$ since no vertices are visited are visited twice in $\mathcal{R}$ unlike $\mathcal{W}$ because the vertices are visited directly without tracing the route back. Under the assumption of triangle inequality, this would give

$$c(\mathcal{R}_{apx}) \leq C(\mathcal{W})$$

# Approximate algorithm using MST

Now, observe that $\mathcal{R}$ is effectively a 'short-cut' of $\mathcal{W}$ since no vertices are visited are visited twice in $\mathcal{R}$ unlike $\mathcal{W}$ because the vertices are visited directly without tracing the route back. Under the assumption of triangle inequality, this would give

$$c(\mathcal{R}_{apx}) \leq C(\mathcal{W})$$

and thus we have proved that

$$c(\mathcal{R}_{apx}) \leq C(\mathcal{R}_{opt})$$

Note: If the triangle inequality doesn't hold in $G$, we can return $\mathcal{W}$ instead of $\mathcal{R}_{apx}$ since our original problem doesn't prevent us from visiting the same vertex twice.

# Simulation results

The minimum total distance was calculated for a 1000X1000 grid for 1 driver and passengers varying from 1 to 5. The locations of driver and passengers were generated at random and the average of shortest distances over 1 lakh iterations were calculated. The following plots show these results.

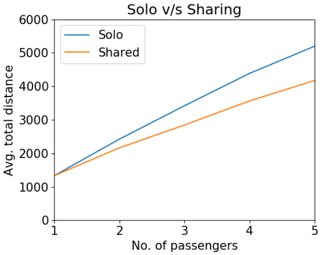


Figure: Avg. total distance for solo and shared ride

# Simulation results

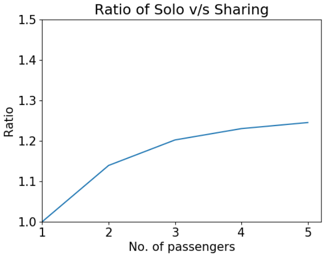


Figure: Ratio of avg. total distance for solo and shared ride

As we could see from the ratio plot, the ratio

$$r = \frac{\text{length of shortest solo-ride route}}{\text{length of shortest shared-ride route}} \simeq 1.25$$

for one driver and 5 passengers. We can see from the plot that this value is converging to a ratio $\simeq 1.25$

# Future directions

- The exponential time complexity of the optimal solution algorithm is a hindrance in implementing this algorithm in real case scenario.

# Future directions

- The exponential time complexity of the optimal solution algorithm is a hindrance in implementing this algorithm in real case scenario.
- The NP-hardness of shared-riding is to be proven formally.

# Future directions

- The exponential time complexity of the optimal solution algorithm is a hindrance in implementing this algorithm in real case scenario.
- The NP-hardness of shared-riding is to be proven formally.
- An approximation algorithm for shared riding is to be found out such that the deviation from this approximate solution from optimal solution is correlated with that of solo-ride. This will enable us to give a ratio of them for larger number of passengers.

# References


Lloyd S Shapley
*"The Shapley value"*, Cambridge University Press, 1988

Hongyao Ma, David C. Parkes and Fei Fang
*"Spatio-Temporal Pricing for Ridesharing Platforms"*

M Furuhata, M Dessouky, F Ordonez, ME Brunet, X Wang, S Koenig
*"Ridesharing: the State-of-the-art and Future Directions"*

Arash Rafie
*"Approximation Algorithms (Travelling Salesman Problem)"*
(URL: http://www.sfu.ca/~A14/lecture25.pdf)

Umesh Vazirani
*"Dynamic Programming"*
(URL: https://people.eecs.berkeley.edu/~vazirani/algorithms/chap6.pdf)