An Efficient Insertion Operator in Dynamic Ridesharing Services

Yi Xu, Yongxin Tong, Yexuan Shi,
Qian Tao, Ke Xu, Wei Li

Key Laboratory of Software Development Environment,
School of Computer Science and Engineering,
Beihang University, China
Outline

- Background
- Problem Statement
- Partition-based Framework
- Segment-based DP Algorithm
- Experiments
- Conclusion
Dynamic Ridesharing Services

- Dynamic ridesharing: services that arrange one-time shared rides on short notice
Dynamic Ridesharing Services

- Dynamic ridesharing: services that arrange one-time shared rides on short notice.
- Car-pooling,
Dynamic Ridesharing Services

- Dynamic ridesharing: services that arrange one-time shared rides on short notice.
- Car-pooling, Food Delivery,
Dynamic Ridesharing Services

- Dynamic ridesharing: services that arrange one-time shared rides on short notice.
- Car-pooling, Food Delivery, Last-mile Logistics

How to plan a route for the worker is central for DR
Dynamic Ridesharing Services

- The problem of route planning in dynamic ridesharing is very difficult and most existing solutions are heuristic.

**Insertion Operator:** insert a new request into the current route.
**Insertion Operator**

*Given:*  
: old route  
\( l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_4 \)  

: new request  
\( o_r \rightarrow d_r \)

**Insertion:**  
finding the appropriate location to insert the new request

*Goal:*  
: old route  
\( l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_4 \)  

: new route  
\( o_r \rightarrow d_r \)
Outline

- Background
- Problem Statement
- Partition-based Framework
- Segment-based DP Algorithm
- Experiments
- Conclusion
Worker

- **Worker**: \( w = \langle o_w, c_w \rangle \)
  - \( o_w \): current location
  - \( c_w \): capacity
  - Capacity constraint: the number of passengers he takes at the same time is less than his capacity.

- **Request**: \( r = \langle o_r, d_r, t_r, e_r, c_r \rangle \)
  - \( o_r, d_r \): origin and destination
  - \( t_r, e_r \): release time and deadline
  - \( c_r \): occupation
  - Deadline constraint: the request is delivered before the deadline.
Route

- Route: $S_R = \langle l_0, l_1, \ldots, l_n \rangle$
  - $R$: a set of requests
  - $l_0$: the current location of worker
  - $l_i$: either origin or destination of $r$ in $R$

Feasible: (1) Capacity constraint; (2) Deadline constraint
Problem Formulation

● Insertion Operator

  ● Given:
    ○ Worker \( w \), feasible original route \( S_R \), new request \( r' \)

  ● Goal:
    ○ Inserts \( r' \) into \( S_R \) to obtain a new feasible route with:
      - Minimizing maximum flow time of all requests: i.e., minimizing maximum waiting time of all requests (waiting time = delivery time - release time)
      - Minimizing total travel time of the worker, i.e., the delivery time of the last request for the worker

Focus of This talk.
Example

<table>
<thead>
<tr>
<th></th>
<th>$t_r$</th>
<th>$e_r$</th>
<th>$o_r$</th>
<th>$d_r$</th>
<th>$c_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1$</td>
<td>0</td>
<td>25</td>
<td>(4,4)</td>
<td>(10,4)</td>
<td>1</td>
</tr>
<tr>
<td>$r_2$</td>
<td>0</td>
<td>37</td>
<td>(8,8)</td>
<td>(4,0)</td>
<td>2</td>
</tr>
<tr>
<td>$r_3$</td>
<td>0</td>
<td>33</td>
<td>(10,2)</td>
<td>(10,0)</td>
<td>1</td>
</tr>
<tr>
<td>$r'$</td>
<td>0</td>
<td>26</td>
<td>(4,6)</td>
<td>(6,2)</td>
<td>1</td>
</tr>
</tbody>
</table>
### Example

![Diagram showing a network of routes with distances and times]

**Table of Routes**

<table>
<thead>
<tr>
<th></th>
<th>$t_r$</th>
<th>$e_r$</th>
<th>$o_r$</th>
<th>$d_r$</th>
<th>$c_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1$</td>
<td>0</td>
<td>25</td>
<td>(4,4)</td>
<td>(10,4)</td>
<td>1</td>
</tr>
<tr>
<td>$r_2$</td>
<td>0</td>
<td>37</td>
<td>(8,8)</td>
<td>(4,0)</td>
<td>2</td>
</tr>
<tr>
<td>$r_3$</td>
<td>0</td>
<td>33</td>
<td>(10,2)</td>
<td>(10,0)</td>
<td>1</td>
</tr>
<tr>
<td>$r'$</td>
<td>0</td>
<td>26</td>
<td>(4,6)</td>
<td>(6,2)</td>
<td>1</td>
</tr>
</tbody>
</table>
Problem

Time complexity: $O(n^3)$

Calculate objective and check constraints in $O(n)$

Enumerate all possible insertion pairs in $O(n^2)$

Basic Algorithm
Goal

Time consuming

Basic Algorithm

How to reduce the time complexity?
Outline

- Background
- Problem Statement
- Partition-based Framework
- Segment-based DP Algorithm
- Experiments
- Conclusion
Partition-based Framework

Time complexity: $O(n^3)$

Calculate objective and check constraints in $O(n)$

Enumerate possible insertion pairs in $O(n^2)$

Basic Algorithm

Time complexity: $O(n^2)$

Calculate objective and check constraints in $O(1)$

Enumerate all possible insertion pairs in $O(n^2)$

Pre-processing in $O(n^2)$

Partition-based Framework
Insertion\((i, j)\)

insertion\((i, j)\)

\[ \text{route } S_R \]

\[ \text{detour} \]

\[ l_i \quad l_{i+1} \quad \cdots \quad l_j \quad l_{j+1} \quad \cdots \]

insert \(o_{r'}\) after \(l_i\)

insert \(d_{r'}\) after \(l_j\)
**Partition-based Framework**

- **Basic idea:**
  - Partition the set of requests $R^+$ into four parts
    - $R_1$: requests delivered before $i$-th location
    - $R_2$: requests delivered between $i$-th and $j$-th location
    - $R_3$: requests delivered after $j$-th location
    - $R_4$: new request $r'$
  - **Objective calculation:**
    - $\text{OBJ}(S_{R^+}) = \max\{mf_1, mf_2, mf_3, mf_4\}$

- $mf_1$: the maximum flow time of all requests in $R_1$
- $mf_2$
- $mf_3$
- $mf_4$
Partition-based Framework

Time complexity: $O(n^2)$

Calculate objective and check constraints in $O(1)$

Enumerate possible insertion pairs in $O(n^2)$

Pre-processing in $O(n^2)$
Pre-processing

- A matrix $mobj$
  - $mobj(i, j)$: the maximum flow time for requests between $i$-th and $j$-th location

$$mobj(i, j) = \max\{mobj(i, j - 1), \text{flow time of request with destination } l_j\}$$
Pre-processing

- A matrix \( mobj \)
  - \( mobj(i, j) \): the maximum flow time for requests between \( i-th \) and \( j-th \) location

\[
\text{mobj}(i, j) = \max \{ \text{mobj}(i, j - 1), \text{flow time of request with destination } l_j \} \quad O(n^2)
\]

\[
\text{mobj}(1, 4) = \max \{ \text{mobj}(1, 3), 0 \} = \max \{ 12.2, 0 \} = 12.2
\]
Partition-based Framework

Time complexity: $O(n^2)$

Calculate objective and check constraints in $O(1)$

Enumerate possible insertion pairs in $O(n^2)$

Pre-processing in $O(n^2)$
Objective Calculation

- The formulas of $mf_1, mf_2, mf_3, mf_4$:
  - $mf_1 = mobj(0, i)$
  - $mf_2 = det(i, o_{r'}) + mobj(i + 1, j)$
  - $mf_3 = det(i, o_{r'}) + det(j, d_{r'}) + mobj(j, n)$
  - $mf_4 = arr(j) + det(i, o_{r'}) + dis(l_j, d_{r'}) + (\alpha - 1)t_{r'}$

We can calculate the objective in $O(1)$ time with formulas.
Objective Calculation

- Insertion (1,5)
- \( mf_1 = mobj(0,1) = 0 \)

\[ R_1 = \emptyset \]
Objective Calculation

- **Insertion (1,5)**
  - \( mf_2 = det(1, o_{r'}) + mobj(2,5) = 0.8 + 16.2 = 17 \)

\( det(k, p) \): the detour of inserting \( p \) after \( k \)-th location

\[ mf_2 \]

\[ R_2 = \{ r_1, r_3 \} \]

\[
\begin{array}{cccccccc}
 i & j & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
 \hline
 0 & 0 & 0 & 0 & 12.2 & 12.2 & 16.2 & 22.2 \\
 1 & - & 0 & 0 & 12.2 & 12.2 & 16.2 & 22.2 \\
 2 & - & - & 0 & 12.2 & 12.2 & 16.2 & 22.2 \\
 3 & - & - & - & 12.2 & 12.2 & 16.2 & 22.2 \\
 4 & - & - & - & - & 0 & 16.2 & 22.2 \\
 5 & - & - & - & - & - & 16.2 & 22.2 \\
 6 & - & - & - & - & - & - & 22.2 \\
\end{array}
\]
Objective Calculation

- Insertion (1,5)
  \[ mf_3 = det(1, o_{r'}) + det(5, d_{r'}) + mobj(6,6) = 0.8 + 1.3 + 22.2 = 24.3 \]

\[ R_3 = \{ r_2 \} \]
Objective Calculation

- Insertion (1,5)
  - \( mf_4 \) is the flow time of \( r' \): \( 21.5 - 0 = 21.5 \)

### Table

<table>
<thead>
<tr>
<th>( i )</th>
<th>( j )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12.2</td>
<td>12.2</td>
<td>16.2</td>
<td>22.2</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>12.2</td>
<td>12.2</td>
<td>16.2</td>
<td>22.2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>12.2</td>
<td>12.2</td>
<td>16.2</td>
<td>22.2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>12.2</td>
<td>12.2</td>
<td>16.2</td>
<td>22.2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>16.2</td>
<td>22.2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>16.2</td>
<td>22.2</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>22.2</td>
</tr>
</tbody>
</table>

\( mf_4 \)
Objective Calculation

- Insertion (1,5)
- The answer is $\max(0, 17, 24.3, 21.5) = 24.3$
Partition-based Framework

- Constraint checking:

  **Capacity constraint:**
  
  \[
  pck(i) \leq c_w - c_{r'}, \\
  pck(j) \leq c_w - c_{r'},
  \]

  **Deadline constraint:**
  
  \[
  det(i, o_{r'}) \leq slk(i) \\
  det(i, o_{r'}) + det(j, d_{r'}) \leq slk(j) \\
  arr(j) + det(i, o_{r'}) + dis(l_j, d_{r'}) \leq e_{r'}
  \]

Constraint checking takes \( O(1) \) time with formulas
Partition-based Framework

- Time complexity: $O(n^2)$
- Calculate objective and check constraints in $O(1)$
- Enumerate possible insertion pairs in $O(n^2)$
- Pre-processing in $O(n^2)$
Outline

- Background
- Problem Statement
- Partition-based Framework
- Segment-based DP Algorithm
- Experiments
- Conclusion
Segment-based DP Algorithm

- **Time complexity:** $O(n^2)$
  - Calculate objective and check constraints in $O(1)$
  - Enumerate possible insertion pairs in $O(n^2)$
  - Pre-processing in $O(n^2)$

Partition-based Framework

- **Time complexity:** $O(n)$
  - Find optimal location to insert destination in $O(1)$
  - Enumerate locations to insert origin in $O(n)$
  - Pre-processing in $O(n)$

Segment-based DP Algorithm
Segment-based DP Algorithm

Enumerate $i$ and $j$

$O(n^2)$

Partition-based Framework
Segment-based DP Algorithm

Enumerate $i$ and $j$

Enumeration $i$, find optimal $j^*$

$O(n^2)$ → $O(n)$

Partition-based Framework → Segment-based DP Algorithm
**Segment-based DP Algorithm**

- **Objective Calculation**

  - $m_f_1 = \text{mobj}(0, i)$
  - $m_f_2 = \text{det}(i, o_{r'}) + \text{mobj}(i + 1, j)$
  - $m_f_3 = \text{det}(i, o_{r'}) + \text{det}(j, d_{r'}) + \text{mobj}(j, n)$
  - $m_f_4 = \text{det}(i, o_{r'}) + \text{arr}(j) + \text{dis}(l_j, d_{r'}) + (\alpha - 1)t_{r'}$

  $\text{OBJ}(S_{R^+}) = \max\{m_f_1, m_f_2, m_f_3, m_f_4\}$

- **Partition-based Framework**

- **Segment-based DP Algorithm**

\[
\begin{align*}
A(i) &= \text{det}(i, o_{r'}) \\
B(j) &= \max \left\{ \text{det}(j, d_{r'}) + \text{mobj}(j, n), \right. \\
&\quad \left. \text{arr}(j) + \text{dis}(l_j, d_{r'}) + (\alpha - 1)t_{r'} \right\}
\]

- **Objective**

  \[\text{OBJ}(S_{R^+}) = A(i) + B(j)\]

- **Can**

\[
\min_{i,j} \text{OBJ}(S_{R^+}) = \min_i A(i) + \min_j B(j) ?
\]

- **It may violate constraints**
Segment-based DP Algorithm

- How to get feasible $\min_{i,j} A(i) + B(j)$?
Segment-based DP Algorithm

- How to get feasible $\min_{i,j} A(i) + B(j)$?

- Method 1:
  - Enumerate $i$, calculate $A(i)$
    - Enumerate $j$, calculate $B(j)$ and check feasibility
    - Maintain the minimum and feasible $A(i) + B(j)$  $O(n^2)$
Segment-based DP Algorithm

- How to get feasible \( \min_{i,j} A(i) + B(j) \)?

- Method 1:
  - Enumerate \( i \), calculate \( A(i) \)
    - Enumerate \( j \), calculate \( B(j) \) and check feasibility
    - Maintain the minimum and feasible \( A(i) + B(j) \) \( O(n^2) \)

- Method 2:
  - Enumerate \( i \), calculate \( A(i) \)
    - Find feasible and minimum \( B(j^*) \) quickly \( O(?) \)
Segment-based DP Algorithm

How to get feasible $\min_{i,j} A(i) + B(j)$?

Method 1:
- Enumerate $i$, calculate $A(i)$
  - Enumerate $j$, calculate $B(j)$ and check feasibility
  - Maintain the minimum and feasible $A(i) + B(j)$ $O(n^2)$

Method 2:
- Enumerate $i$, calculate $A(i)$
  - Find feasible minimum $B(j^*)$ directly $O (?)$

$B(j^*) = \min_{i<j<\text{brk}(i)} B(j)$
$\det(i, o_r, t) \leq \text{thr}(j)$

Segment tree $O(\log n)$ $O(n \log n)$
Fenwick tree (dynamic) $O(1)$ $O(n)$
Example

- Initialization
  - Initialize $thr(\cdot)$, $B(\cdot)$

<table>
<thead>
<tr>
<th>index</th>
<th>$thr(\cdot)$</th>
<th>$B(\cdot)$</th>
<th>$3(d_{r_1})$</th>
<th>$4(o_{r_3})$</th>
<th>$5(d_{r_3})$</th>
<th>$6(d_{r_2})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>thr(\cdot)</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
</tr>
<tr>
<td>B(\cdot)</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
</tr>
</tbody>
</table>

Objective relevant to $j$

$$\min_{i<j<\text{brk}(i)} B(j)$$

$$\text{det}(i,o_{r'},d_{r'}) \leq thr(j)$$
Example

- Enumerate $i = 5$ ($A(i) = 8.5$)
- Update $B(6) = 25$ at point $thr(6) = -1$

<table>
<thead>
<tr>
<th>index</th>
<th>0($o_w$)</th>
<th>1($o_{r_1}$)</th>
<th>2($o_{r_2}$)</th>
<th>3($d_{r_1}$)</th>
<th>4($o_{r_3}$)</th>
<th>5($d_{r_3}$)</th>
<th>6($d_{r_2}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$thr(\cdot)$</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
<td>-1</td>
</tr>
<tr>
<td>$B(\cdot)$</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
<td>25</td>
</tr>
</tbody>
</table>

Optimal solution: $\infty$

Update value of this point as 25
Example

- Enumerate $i = 5$ ($A(i) = 8.5$)
- Query the segment $[\text{det}(5, o_{r'}), \infty)$

<table>
<thead>
<tr>
<th>index</th>
<th>$0(o_w)$</th>
<th>$1(o_{r_1})$</th>
<th>$2(o_{r_2})$</th>
<th>$3(d_{r_1})$</th>
<th>$4(o_{r_3})$</th>
<th>$5(d_{r_3})$</th>
<th>$6(d_{r_2})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{thr}(\cdot)$</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
<td>-1</td>
</tr>
<tr>
<td>$B(\cdot)$</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
<td>25</td>
</tr>
</tbody>
</table>

**Optimal solution:** $\infty$

$\text{det}(5, o_{r'}) = 8.5$
The answer is $\infty$
Example

- Enumerate $i = 4$ ($A(i) = 13.7$)
- Update $B(5) = 23.5$ at point $thr(5) = 3.3$

<table>
<thead>
<tr>
<th>index</th>
<th>$0(o_w)$</th>
<th>$1(o_{r1})$</th>
<th>$2(o_{r2})$</th>
<th>$3(d_{r1})$</th>
<th>$4(o_{r3})$</th>
<th>$5(d_{r3})$</th>
<th>$6(d_{r2})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$thr(\cdot)$</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
<td>-1</td>
</tr>
<tr>
<td>$B(\cdot)$</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
<td>25</td>
</tr>
</tbody>
</table>

Optimal solution: $\infty$

Update value of this point as 23.5
Example

- Enumerate $i = 4$ ($A(i) = 13.7$)
- Query the segment $[det(4, o_{r'}), \infty)$

<table>
<thead>
<tr>
<th>index</th>
<th>$0(o_w)$</th>
<th>$1(o_{r_1})$</th>
<th>$2(o_{r_2})$</th>
<th>$3(d_{r_1})$</th>
<th>$4(o_{r_3})$</th>
<th>$5(d_{r_3})$</th>
<th>$6(d_{r_2})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$thr(\cdot)$</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
<td>−1</td>
</tr>
<tr>
<td>$B(\cdot)$</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
<td>25</td>
</tr>
</tbody>
</table>

$$\min_{i < j < brk(j)} \quad B(j) \quad \text{det}(i, o_{r'}) \leq thr(j)$$

Optimal solution: $\infty$

$det(4, o_{r'}) = 13.7$

The answer is $\infty$
Example

- Enumerate $i = 3$ ($A(i) = 11.5$)
- Update $B(4) = 28.7$ at point $thr(4) = 5.8$

<table>
<thead>
<tr>
<th>index</th>
<th>$0(o_w)$</th>
<th>$1(o_{r1})$</th>
<th>$2(o_{r2})$</th>
<th>$3(d_{r1})$</th>
<th>$4(o_{r3})$</th>
<th>$5(d_{r3})$</th>
<th>$6(d_{r2})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$thr(·)$</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
<td>-1</td>
</tr>
<tr>
<td>$B(·)$</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
<td>25</td>
</tr>
</tbody>
</table>

Optimal solution: $\infty$

Update value of this point as 28.7
Example

- Enumerate $i = 3$ ($A(i) = 11.5$)
- Query the segment $[\text{det}(3, o_{r'}), \infty)$

<table>
<thead>
<tr>
<th>index</th>
<th>$0(o_w)$</th>
<th>$1(o_{r_1})$</th>
<th>$2(o_{r_2})$</th>
<th>$3(d_{r_1})$</th>
<th>$4(o_{r_3})$</th>
<th>$5(d_{r_3})$</th>
<th>$6(d_{r_2})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{thr}(\cdot)$</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
<td>-1</td>
</tr>
<tr>
<td>$B(\cdot)$</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
<td>25</td>
</tr>
</tbody>
</table>

Optimal solution: $\infty$

$\text{det}(3, o_{r'}) = 11.5$

The answer is $\infty$
Example

- Enumerate $i = 2$ ($A(i) = 6.3$)
- Update $B(3) = 28.7$ at point $thr(3) = 6.3$

<table>
<thead>
<tr>
<th>index</th>
<th>$0(o_w)$</th>
<th>$1(o_{r1})$</th>
<th>$2(o_{r2})$</th>
<th>$3(d_{r1})$</th>
<th>$4(o_{r3})$</th>
<th>$5(d_{r3})$</th>
<th>$6(d_{r2})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$thr(\cdot)$</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
<td>−1</td>
</tr>
<tr>
<td>$B(\cdot)$</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
<td>25</td>
</tr>
</tbody>
</table>

Optimal solution: $\infty$

Update value of this point as 28.7
Example

- Enumerate $i = 2$ ($A(i) = 6.3$)
- Query the segment $[det(2, o_{r'}), \infty)$

<table>
<thead>
<tr>
<th>index</th>
<th>$0(o_w)$</th>
<th>$1(o_{r_1})$</th>
<th>$2(o_{r_2})$</th>
<th>$3(d_{r_1})$</th>
<th>$4(o_{r_3})$</th>
<th>$5(d_{r_3})$</th>
<th>$6(d_{r_2})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$thr(\cdot)$</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
<td>-1</td>
</tr>
<tr>
<td>$B(\cdot)$</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
<td>25</td>
</tr>
</tbody>
</table>

Optimal solution: $\infty$

$det(2, o_{r'}) = 6.3$

The answer is 28.7
Example

- Enumerate $i = 2$ ($A(i) = 6.3$)
- Query the segment $[det(2, o_{r'})$, $\infty)$

<table>
<thead>
<tr>
<th>index</th>
<th>0($o_w$)</th>
<th>1($o_{r_1}$)</th>
<th>2($o_{r_2}$)</th>
<th>3($d_{r_1}$)</th>
<th>4($o_{r_3}$)</th>
<th>5($d_{r_3}$)</th>
<th>6($d_{r_2}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$thr(\cdot)$</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
<td>-1</td>
</tr>
<tr>
<td>$B(\cdot)$</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
<td>25</td>
</tr>
</tbody>
</table>

Optimal solution: 35

We find a feasible solution: insertion(2,3)
The maximum flow time is $6.3 + 28.7 = 35$
Example

- Enumerate $i = 1$ ($A(i) = 0.8$)
- Update $B(2) = 28.5$ at point $thr(2) = 4.5$

<table>
<thead>
<tr>
<th>index</th>
<th>0(o_w)</th>
<th>1(o_r₁)</th>
<th>2(o_r₂)</th>
<th>3(d_r₁)</th>
<th>4(o_r₃)</th>
<th>5(d_r₃)</th>
<th>6(d_r₂)</th>
</tr>
</thead>
<tbody>
<tr>
<td>thr(·)</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
<td>-1</td>
</tr>
<tr>
<td>B(·)</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
<td>25</td>
</tr>
</tbody>
</table>

Optimal solution: 35

Update value of this point as 28.5
Example

- Enumerate $i = 1$ ($A(i) = 0.8$)
- Query the segment $[det(1, o_{r'}), \infty)$

<table>
<thead>
<tr>
<th>index</th>
<th>0$(o_w)$</th>
<th>1$(o_{r_1})$</th>
<th>2$(o_{r_2})$</th>
<th>3$(d_{r_1})$</th>
<th>4$(o_{r_3})$</th>
<th>5$(d_{r_3})$</th>
<th>6$(d_{r_2})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{thr}(:)$</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
<td>$-1$</td>
</tr>
<tr>
<td>$B(\cdot)$</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
<td>25</td>
</tr>
</tbody>
</table>

**Optimal solution:** 35

$\min_{i<j<\text{brk}(i)} B(j)$

$\text{det}(i, o_{r'}) \leq \text{thr}(j)$

$\text{det}(1, o_{r'}) = 0.8$

The answer is 25.5
Example

- Enumerate $i = 1$ ($A(i) = 0.8$)
- Query the segment $[det(1, o_{r'}), \infty)$

<table>
<thead>
<tr>
<th>index</th>
<th>0($o_w$)</th>
<th>1($o_{r_1}$)</th>
<th>2($o_{r_2}$)</th>
<th>3($d_{r_1}$)</th>
<th>4($o_{r_3}$)</th>
<th>5($d_{r_3}$)</th>
<th>6($d_{r_2}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$thr(\cdot)$</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
<td>-1</td>
</tr>
<tr>
<td>$B(\cdot)$</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
<td>25</td>
</tr>
</tbody>
</table>

Optimal solution: 24.3

Another feasible solution: insertion(1,5)

The maximum flow time is $0.8 + 23.5 = 24.3$
Example

- Enumerate \( i = 0 \) (\( A(i) = 2.8 \))
- Update \( B(1) = 25.6 \) at point \( thr(1) = 7.4 \)

<table>
<thead>
<tr>
<th>Index</th>
<th>( 0(o_w) )</th>
<th>( 1(o_{r1}) )</th>
<th>( 2(o_{r2}) )</th>
<th>( 3(d_{r1}) )</th>
<th>( 4(o_{r3}) )</th>
<th>( 5(d_{r3}) )</th>
<th>( 6(d_{r2}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( thr(\cdot) )</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
<td>-1</td>
</tr>
<tr>
<td>( B(\cdot) )</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
<td>25</td>
</tr>
</tbody>
</table>

\[
\min_{i < j < \text{brk}(i)} B(j) \quad \text{det}(i, o_{r'i}) \leq thr(j)
\]

Optimal solution: 24.3

Update value of this point as 25.6
Example

- Enumerate $i = 0$ ($A(i) = 2.8$)
- Query the segment $[det(0, o_{r'}), \infty)$

<table>
<thead>
<tr>
<th>index</th>
<th>0($o_w$)</th>
<th>1($o_{r_1}$)</th>
<th>2($o_{r_2}$)</th>
<th>3($d_{r_1}$)</th>
<th>4($o_{r_3}$)</th>
<th>5($d_{r_3}$)</th>
<th>6($d_{r_2}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$thr(\cdot)$</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
<td>-1</td>
</tr>
<tr>
<td>$B(\cdot)$</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
<td>25</td>
</tr>
</tbody>
</table>

Optimal solution: 24.3

$det(0, o_{r'}) = 2.8$

The answer is 23.5
Example

- Enumerate $i = 0$ ($A(i) = 2.8$)
- Query the segment $[\text{det}(0, o_{r'}), \infty)$

<table>
<thead>
<tr>
<th>index</th>
<th>$0(o_w)$</th>
<th>$1(o_{r_1})$</th>
<th>$2(o_{r_2})$</th>
<th>$3(d_{r_1})$</th>
<th>$4(o_{r_3})$</th>
<th>$5(d_{r_3})$</th>
<th>$6(d_{r_2})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{thr}()$</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
<td>-1</td>
</tr>
<tr>
<td>$B()$</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
<td>25</td>
</tr>
</tbody>
</table>

Optimal solution: 24.3

Another feasible solution: insertion(0,5)
The maximum flow time is $2.8 + 23.5 = 26.3$
Finally, the optimal solution:

- Insertion(1,5) with maximum flow time 24.3

```
<table>
<thead>
<tr>
<th>index</th>
<th>0(o_w)</th>
<th>1(o_r_1)</th>
<th>2(o_r_2)</th>
<th>3(d_r_1)</th>
<th>4(o_r_3)</th>
<th>5(d_r_3)</th>
<th>6(d_r_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>thr(·)</td>
<td>5.5</td>
<td>7.4</td>
<td>4.5</td>
<td>6.3</td>
<td>5.8</td>
<td>3.3</td>
<td>-1</td>
</tr>
<tr>
<td>B(·)</td>
<td>27.5</td>
<td>25.6</td>
<td>28.5</td>
<td>28.7</td>
<td>28.7</td>
<td>23.5</td>
<td>25</td>
</tr>
</tbody>
</table>
```

\[
\begin{align*}
\min_{i<j<\text{brk}(i)} B(j) \\
\text{det}(i,o_r,r) &\leq \text{thr}(j)
\end{align*}
\]
Outline

- Background
- Problem Statement
- Partition-based Framework
- Segment-based DP Algorithm
- Experimental Evaluations
- Conclusion
Experiments: Setup

- Two real datasets:
  - Taxi: collected in New York City, public dataset.
    - 517,100 requests
    - Worker’s capacity is small (Default: 4)
  - Logistics: collected in Shanghai by Cainiao, an urban logistics platform in China.
    - 345,849 requests
    - Worker’s capacity is large (Default: 120)
## Experiments: Setup

### Compared Algorithms:

<table>
<thead>
<tr>
<th>Method</th>
<th>Complexity</th>
<th>Goal</th>
<th>Minimize maximum flow time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BF [ICDE’13]  [EJOR’11]</td>
<td>$O(n^3)$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Kinetic [VLDB’14]</td>
<td>$O(n^2)$</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>LDP [VLDB’18]</td>
<td>$O(n)$</td>
<td>✓</td>
<td>×</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Complexity</th>
<th>Goal</th>
<th>Minimize maximum flow time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NDP</td>
<td>$O(n^2)$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ST</td>
<td>$O(n \log n)$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FT</td>
<td>$O(n)$</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Experiments: Result

- Minimizing total travel time
  - FT is faster than LDP on Taxi and as fast as LDP on Logistic

As for minimizing total travel time, FT is 878.4 times faster than BF on Logistics

Dataset: Taxi  
Dataset: logistics
Experiments: Result

- Minimizing maximum flow time
- FT outperforms the other algorithms

FT is 2.2 times faster than BF on Taxi and 998.1 times faster on Logistics

Dataset: Taxi

Dataset: logistics
Experiments: Result

- Memory
  - The gap of memory cost among algorithms (except NDP) is marginal.
  - The gap of memory cost between FT to other algorithms is less than 0.1MB

Dataset: Taxi

Dataset: logistics
Experiments: Result

- **Scalability**
  - Our algorithms are fit for hundred thousands of requests.
Outline

- Background
- Problem Statement
- Partition-based Framework
- Segment-based DP Algorithm
- Experiments
- Conclusion
Conclusion

- We propose a partition-based framework to reduce the time complexity of the generic insertion operator from $O(n^3)$ to $O(n^2)$.

- By utilizing some efficient index structures, we further propose a linear insertion operator.

- Experiments on real datasets show that the insertion operator can be accelerated by up to 998.1 times on urban-scale datasets.
Q & A

Thank You