Coverage and Workload Cost Balancing in Spatial Crowdsourcing

Ning Wang, Jie Wu and Pouya Ostovari
Temple University
Outline

• Introduction
• **Model and problem**
• **Solution in 1-D scenario**
• **Solution in 2-D scenario**
• Experiments
• **Conclusion and future work**
Background

From crowdsourcing to Spatial Crowdsourcing

• **Crowdsourcing**
  - Outsourcing a set of tasks to a set of workers
    ∷ Human Intelligence Tasks (hard for computer, easy for human)

• **Spatial Crowdsourcing**
  - Crowdsourcing a set of spatial task to a set of workers
    ∷ Traffic monitoring
    ∷ climate measurement
    ∷ Interesting point review
Real applications:

- **Spatial Crowdsourcing Service**
  - TaskRabbit (Home repair and refresh)
  - Uber (Passenger/food delivery)
  - WeGoLook (Inspection)
  - FiELD Agent/ Gigwalk

- **Information sharing**
  - Waze/Trapster (Traffic update)
  - WeatherSignal/OpenSignal
  - Local review (Google Local guide)
Related works

- **Worker trajectory planning**
  - Plan worker’s trajectory in crowdsourcing service
    - Maximize number of a worker’s task
    - Maximize multiple workers’ tasks (competition)
    - Crowdsourcing task can be time conflicted

- **Worker recruitment problem**
  - Ensure crowdsourcing quality with worker’s trajectory
    - Maximize the coverage area
    - Minimize the overall recruitment cost
Network Model

Our model

- Information sharing*
- Worker recruitment problem*

Sharing economy!
- You will not be bothered by the crowdsourcing platform, but you and others can benefit from this.
Network Model

• **Multiple Workers**, \( \{w_1, w_2, \ldots, w_n\} \)
  - known trajectory, \( t_i \), and recruiting cost, \( c_i \), for visiting a crowdsourcing location.

• **Many crowdsourcing locations**, \( \{l_1, l_2, \ldots, l_m\} \)
  - Pay worker \( c_i \) when \( w_i \) passes this location

• **Grid network**
  - Fit real road networks
Coverage and Balanced Crowdsourcing Recruiting (CBCR) problem

- **Coverage requirement**
  - All the crowdsourcing locations should be covered/visited

- **Balancing crowdsourcing location cost**
  - The maximum cost of crowdsourcing location should be minimized

\[
\min \quad \max_i \sum_{l_i \in t_j} c_j x_j \\
\text{s.t.} \quad \sum_{l_i \in t_j} x_j \geq 1, \quad \forall l_i \quad x_j \in \{0, 1\},
\]

- **NP-hard in general scenario**
1-D scenario

Application scenario
People/vehicles in highway, main street

• illustration

<table>
<thead>
<tr>
<th>worker</th>
<th>Covered locations</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>l1, l2</td>
<td>1</td>
</tr>
<tr>
<td>w2</td>
<td>l2, l3, l4</td>
<td>1.5</td>
</tr>
<tr>
<td>w3</td>
<td>l3, l4</td>
<td>3</td>
</tr>
<tr>
<td>w4</td>
<td>l4</td>
<td>2</td>
</tr>
</tbody>
</table>
1-D scenario

- Min-max Greedy algorithm (MG)
  - While the network is not covered, we select the worker who can minimize the maximum cost among all the crowdsourcing locations in the network.

<table>
<thead>
<tr>
<th>worker</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>1</td>
</tr>
<tr>
<td>w2</td>
<td>1.5</td>
</tr>
<tr>
<td>w3</td>
<td>3</td>
</tr>
<tr>
<td>w4</td>
<td>2</td>
</tr>
</tbody>
</table>

- Analysis: the error can be accumulated/ nonsubmoduar
1-D scenario

Coverage-Only Greedy algorithm (CO)
- While the network is not covered, we select the worker who can increase coverage most from one side to the other side (e.g., from left to right).

```
<table>
<thead>
<tr>
<th>worker</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>1</td>
</tr>
<tr>
<td>w2</td>
<td>1.5</td>
</tr>
<tr>
<td>w3</td>
<td>3</td>
</tr>
<tr>
<td>w4</td>
<td>2</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>round</th>
<th>Selected worker</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>w1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>w3</td>
<td>3</td>
</tr>
</tbody>
</table>
```

Theorem: The CO algorithm has a $2\max|c_i/c_j|$, for all $i,j$, approximation ratio in the 1-D scenario.
1-D scenario

- PTAS in CO algorithm
  - Analysis: worker with high cost can be selected
  - Idea: Set work with high cost a low priority

- Algorithm implementation
  - Set a threshold, $\epsilon$, separate workers into two sets in terms of cost
  - costly workers and cheap workers
  - Apply CO algorithm in cheap workers
    - If it successes, reduce the threshold
    - If it fails, increase the threshold
  - Binary search to find the smallest threshold

Theorem: The CO algorithm has a $2+\epsilon$ approximation ratio in the 1-D scenario.
**1-D scenario**

- Dynamic programming (Optimal sub-structure)
  - Sort all trajectories based on end points from left to right
  - The optimal solution for crowdsourcing location $i$ with worker $w_j$ as the last worker.

$$d[i, j] = \begin{cases} 
0 & i = 0 \\
\min_{i' < i, j' \leq j} \max\{d[i', j'], c_{j'} + c_j\} & \text{Otherwise}
\end{cases}$$

<table>
<thead>
<tr>
<th>worker</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>1</td>
</tr>
<tr>
<td>w2</td>
<td>1.5</td>
</tr>
<tr>
<td>w3</td>
<td>3</td>
</tr>
<tr>
<td>w4</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>location</th>
<th>worker</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1</td>
<td>2.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2.5</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>2.5</td>
<td></td>
<td>3</td>
<td>3.5</td>
</tr>
</tbody>
</table>
**Challenge**

- The overlapping relationship becomes complex
  - 1-D continuous overlap
  - 2-D discrete overlap

- Optimal substructure does not exist and dynamic programming does not work
2-D scenario

Idea
- Extend the proposed algorithms in 1-D scenario
  - Min-max algorithm is still the same.
  - Coverage-only algorithm can be used line-by-line.

Randomized Rounding Algorithm
- Relax the original problem into the linear problem
  \[
  \begin{align*}
  \text{min} & \quad \theta \\
  \text{s.t.} & \quad \sum_{l_i \in t_j} c_j x_j \leq \theta, \quad \sum_{l_i \in t_j} x_i \geq 1, \quad \forall i, j \quad x_i \in [0, 1]
  \end{align*}
  \]
- Use the expected value as the selection probability and randomly select workers.

Theorem: The randomized rounding algorithm has a \(O(\log(n)/\log\log(n))\) expected approximation ratio
Experiment Setting

- **Trajectory Trace Information**
  - EPFL: 500 taxies in San Francisco, USA
  - Seattle: 236 buses in Seattle, USA
- **Trajectory Trace Information**
  - Uniform/exponential distribution with 5 cost
- **Experimental area**: downtown
Algorithm comparison

• Four algorithms in 1-D:
  - Min-Max greedy (MG)
  - Coverage-Only (CO)
  - PTAS (PT)
  - Dynamic programming (DP)

• Four algorithms in 2-D:
  - Min-Max greedy (MG): the same
  - Coverage-Only (CO): row-by-row /
  - PTAS (PT)
  - Dynamic programming (DP): do not apply
  - Randomized Rounding (RD)
1D

- Different number of crowdsourcing locations

San Francisco

Seattle
1D

- Different cost distribution
  - Uniform/Exponential
2D
• Different number of crowdsourcing locations

San Francisco

Seattle
Conclusions

• We investigate a worker recruitment problem in spatial crowdsourcing scenario, where coverage and balance location cost are jointly considered.

• A series of algorithm is proposed in 1-D scenario to trade-off the performance and computation complexity.
  - Coverage-Only algorithm
  - PTAS algorithm
  - Dynamic programming algorithm

• A randomized rounding scheme is proposed in a general scenario.
Future works

- Efficient deterministic algorithm in 2-D scenario
- Weighted coverage and heterogeneous cost
- Trade-off between detour and benefit
- System implementation (if possible)
Thank you and Question

Ning Wang
ning.wang@temple.edu