Multi-Task Assignment for CrowdSensing in Mobile Social Networks

Mingjun Xiao\textsuperscript{a}, Jie Wu\textsuperscript{b}, Liusheng Huang\textsuperscript{a}
Yunsheng Wang\textsuperscript{c}, Cong Liu\textsuperscript{d}

\textsuperscript{a} University of Science and Technology of China,
\textsuperscript{b} Temple University,
\textsuperscript{c} Kettering University,
\textsuperscript{d} Sun Yat-sen University
Outline

• Background & Motivation
• Model & Problem
• Solution
• Simulation
• Conclusion
Background

• Mobile CrowdSensing
  – **Smart devices:**
    iPads, smart phones, portable devices, etc.
    (light sensor, GPS, camera, digital compass, etc.)
  – **A new paradigm:**
    mobile users exploit their carried smart devices to conduct complex computation and sensing tasks.
  – **Applications:**
    urban WiFi characterization, map labelling, traffic information mapping, etc.
CrowdSensing in Mobile Social Networks

Mobile Social Network (MSN):
- 3G/4G communication model
- Short-distance communication model (WiFi, Bluetooth)

Existing CrowdSensing Algorithms

- Task assignment
  - LRBA [Infocom’14]
  - Fair energy-efficient allocation algorithm [Infocom’14]

- Other issues
  - Incentive mechanisms, Privacy-preserving schemes,
Motivation

- Time-sensitive task assignment
Network Model

- **MSN users**: $V = \{v_0, v_1, \ldots, v_n\}$
- **Requester**: $v_0$
- **Communication model**: WiFi, Bluetooth
- **Contact ($v_0$, $v_i$)**: exponential distribution $\lambda_i$
Model & Problem

- **Problem**
  - **Tasks:** $J = \{ j_1, j_2, \ldots, j_m \}$
  - **Workloads:** $\tau_1, \tau_2, \ldots, \tau_m$
  - **Makespan $M(j)$:** the time that the requester finally receives the result of the task $j$.
  - **Task assignment strategy:**

$$\Pi = \{ J_1, J_2, \ldots, J_n \}$$

$J_i = \{ \ldots, j, \ldots, j', \ldots \}$ is an ordered subset of $J$

$j \leq j': j$ will be processed prior to $j'$

$$\sum_{i=1}^{n} J_i = J, \quad J_i \cap J_i' = \emptyset, \quad \forall J_i, J_i' \in \Pi$$
• **Problem**
  
  – **Average Makespan** $AM(\Pi)$:

  
  \[
  AM(\Pi) = \frac{1}{m} \sum_{j \in J} M(j) |_{\Pi}
  \]

  – **Objective**: Maximize $AM(\Pi)$

---

Model & Problem
Solution: offline Task Assignment

• FTA: offline Task Assignment

  the requester makes the task assignment decision before it encounters any other mobile user

• Basic Formula

  **Theorem 1:** The average makespan $AM(\Pi_F)$ for an arbitrary offline task assignment strategy $\Pi_F = \{J_1, J_2, \ldots, J_n\}$ satisfies:

  $$AM(\Pi_F) = \frac{1}{m} \sum_{i=1}^{n} \sum_{j \in J_i} \left( \frac{2}{\lambda_i} + \sum_{j' \in J_i \cap j' \leq j} \tau_{j'} \right)$$
Solution: oFfline Task Assignment

• Example: \( \Pi_F = \{J_1, J_2\} \), \( J_1 = \{ j_1, j_3 \} \), \( J_2 = \{ j_2, j_4 \} \)

\[
M(j_1) = \frac{2}{\lambda_1} + \tau_1, \quad M(j_2) = \frac{2}{\lambda_1} + \tau_1 + \tau_3, \quad M(j_3) = \frac{2}{\lambda_2} + \tau_2, \quad M(j_4) = \frac{2}{\lambda_2} + \tau_2 + \tau_4
\]

\[
AM(\Pi_F) = \frac{1}{4}(M(j_1) + M(j_2) + M(j_3) + M(j_4))
\]
Solution: offline Task Assignment

• Basic Property of Optimal FTA Solution: $\Pi^*_F$

**Theorem 2:** Suppose the optimal offline task assignment strategy is $\Pi^*_F = \{ J_1, J_2, \ldots, J_n \}$. Then, the tasks with small workloads will be processed first. More specifically, for $\forall j, j' \in J_i$ $(1 \leq i \leq n)$, if $\tau_j \leq \tau_{j'}$, then the order of tasks $j$ and $j'$ in $J_i$ satisfies $j \leq j'$.

– In the last example: $\tau_1 < \tau_2 < \tau_3 < \tau_4$

$\quad j_1 \leq j_3 \quad j_2 \leq j_4$
Solution: oFflne Task Assignment

• Basic Property of Optimal FTA Solution: $\Pi^*_F$

  – **Expected Processing Time ($EPT_i$):**

    the expected time for the user $v_i$ to meet the requester, process the tasks in hand, and return the result.

Example:

user $v_i$ has tasks $\dot{j}_{i_1}, \dot{j}_{i_2}, \ldots, \dot{j}_{i_k}$

$$EPT_i = \frac{2}{\lambda_i} + \tau_{i_1} + \tau_{i_2} + \cdots + \tau_{i_k}$$
Solution: oFfline Task Assignment

• Basic Property of Optimal FTA Solution: $\Pi^*_F$

**Theorem 3:** suppose that the workloads of the $j_1, j_2, \ldots, j_m$ satisfy $\tau_1 \leq \tau_2 \leq \ldots \leq \tau_m$, among which the tasks $j_1, j_2, \ldots, j_{k-1}$ ($1 \leq k \leq m$) have been assigned. Assume that the current expected processing time of user $v_i$ is $EPT_i$. Then, the optimal task assignment strategy $\Pi^*_F$ satisfies:

the task $j_k$ will be assigned to the user who currently has the minimum expected processing time, i.e.,

$$EPT_i = \min\{EPT_1, \ldots, EPT_n\} \Rightarrow j_k \in J_i$$
Solution: oFfline Task Assignment

• Basic Idea:
we always assign the minimum workload task among the tasks that have not been assigned to the user with the smallest expected processing time

- Example:

requester
\{ j_1, j_2, j_3, j_4 \}
\tau_1=4, \tau_2=6,
\tau_3=8, \tau_4=10

\lambda_1=1/4
\lambda_2=1/5
\lambda_3=1/11
Solution: offline Task Assignment

- Example:

The initial state

Assign task $j_1$

Assign task $j_2$

Assign task $j_3$

Assign task $j_4$
Solution: oFFline Task Assignment

• The Detailed FTA Algorithm:

Algorithm 1 The FTA Algorithm

Require: $J = \{j_1, j_2, \cdots, j_m : \tau_1 \leq \tau_2 \leq \cdots \leq \tau_m\}$,
$V = \{v_1, v_2, \cdots, v_n : \lambda_1, \lambda_2, \cdots, \lambda_n\}$.

Ensure: $\Pi_F = \{J_1, J_2, \cdots, J_n\}$

1: for each user $v_i$ do
2: \hspace{1em} $J_i = \emptyset$;
3: \hspace{1em} $EPT_i = \frac{2}{\lambda_i}$;
4: for task $j$ from $j_1$ to $j_m$ do
5: \hspace{2em} $i_{\min} = \arg\min\{EPT_1, EPT_2, \cdots, EPT_n\}$;
6: \hspace{2em} Assign task $j$ to $v_{i_{\min}}$: $J_{i_{\min}} = J_{i_{\min}} + \{j\}$;
7: \hspace{2em} $EPT_{i_{\min}} = EPT_{i_{\min}} + \tau_{i_{\min}}$;

Corollary 4: The FTA algorithm can achieve the optimal average makespan for the offline task assignment case.
Solution: oNline Task Assignment

*NTA: oNline Task Assignment*

the requester dynamically assigns tasks at each time when it encounters a mobile user

*Instant Processing Time (IPT)*

the time for a mobile user, who has just encountered and received some tasks from the requester, to process these tasks and return the results

**Example:** user $v_i$ has tasks $\hat{J}_{i_1}, \hat{J}_{i_2}, \cdots, \hat{J}_{i_k}$

\[
IPT_i = \frac{1}{\lambda_i} + \tau_{i_1} + \tau_{i_2} + \cdots + \tau_{i_k}
\]
Solution: oNline Task Assignment

• Basic idea
  – When the requester encounters a mobile user $v_i$, it first computes $IPT_i$ and $EPT$ values of other users who have not been met by itself.
  – The requester adopts the similar greedy strategy in FTA to assign tasks, while using $IPT_i$ to replace $EPT_i$ in FTA.
  – The requester will get a result $\{J_1, \ldots, J_i, \ldots, J_n\}$. Then, the requester assigns the tasks in $J_i$ to user $v_i$, while keeping the remaining tasks (i.e., the tasks in $J-J_i$) in hand.
Solution: oNline Task Assignment

- Example

\begin{itemize}
  \item \text{requester}
  \item \{j_1, j_2, j_3, j_4\}
  \item \tau_1 = 4, \tau_2 = 9, \tau_3 = 10, \tau_4 = 11
\end{itemize}

\begin{align*}
  \lambda_1 &= \frac{1}{4} \\
  \lambda_2 &= \frac{1}{6} \\
  \lambda_3 &= \frac{1}{7}
\end{align*}
Solution: oNline Task Assignment

• Example

the requester determines $J_2$ when it meets $v_2$: $J_2 = \{j_1, j_3\}$

the requester determines $J_1$ when it meets $v_1$: $J_1 = \{j_2, j_4\}$
The computation overhead is $O(mn^2)$
Solution: oNline Task Assignment

- **Performance analysis**
  - **Theorem 5:** The NTA algorithm can achieve a smaller average makespan than FTA, i.e.,
    \[
    AM(\Pi_N) \leq AM(\Pi^*_F)
    \]
  - **Theorem 6:** Assume that there is a god, who can foresee at what time the requester will meet which user. Based on this, it can give an ideal optimal task assignment strategy \(\Pi_{OPT}\). Then, we have:
    \[
    AM(\Pi_N) - AM(\Pi_{OPT}) \leq \sum_{i=1}^{n} \frac{2}{\lambda_i}
    \]
    \[
    \frac{AM(\Pi_N)}{AM(\Pi_{OPT})} \leq 1 + \frac{m \sum_{i=1}^{n} \frac{2}{\lambda_i}}{\sum_{j=1}^{m} \tau_j}
    \]
Simulation

- **Algorithms in comparison**
  - *FTA, NTA, OPT*
  - Water Filling (*WF*) [Mobihoc’12]
  - Largest-First (*LF*)

- **Metrics**
  - Average makespan
Simulation

• Real Traces
  – Statistics of traces

<table>
<thead>
<tr>
<th>Trace</th>
<th>Contacts</th>
<th>Length (hours)</th>
<th>Requester</th>
<th>Other users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel</td>
<td>2,766</td>
<td>99.8</td>
<td>9</td>
<td>128</td>
</tr>
<tr>
<td>Cambridge</td>
<td>6,732</td>
<td>145.6</td>
<td>12</td>
<td>223</td>
</tr>
<tr>
<td>Infocom</td>
<td>28,216</td>
<td>76.6</td>
<td>41</td>
<td>264</td>
</tr>
<tr>
<td>UMassDieselNet</td>
<td>227,657</td>
<td>95.3</td>
<td>4</td>
<td>36</td>
</tr>
</tbody>
</table>

– Other Settings

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tasks m</td>
<td>300</td>
<td>200-1000</td>
</tr>
<tr>
<td>Average workload τ hours)</td>
<td>20</td>
<td>10-50</td>
</tr>
</tbody>
</table>
Simulation

**Results:** average makespan vs. average workload

(a) Cambridge Haggle: Intel
(b) Cambridge Haggle: Cambridge
(c) Cambridge Haggle: Infocom
(d) UMassDieselNet
Simulation

Results: average makespan vs. number of tasks

(a) Cambridge Haggle: Intel
(b) Cambridge Haggle: Cambridge
(c) Cambridge Haggle: Infocom
(d) UMassDieselNet
## Simulation

- **Synthetic Traces**

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Default value</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of users $n$</td>
<td>400</td>
<td>100-1000</td>
</tr>
<tr>
<td>Number of requesters</td>
<td>5%</td>
<td>5%-10%</td>
</tr>
<tr>
<td>The average rate parameter $\lambda$</td>
<td>0.05</td>
<td>0.01-0.1</td>
</tr>
<tr>
<td>Number of tasks $m$</td>
<td>300</td>
<td>100-1000</td>
</tr>
<tr>
<td>The average workload</td>
<td>10</td>
<td>5 - 50</td>
</tr>
</tbody>
</table>
Simulation

Results:

(a) Change the number of users
(b) Change the average workload
(c) Change the number of tasks
(d) Change the average rate parameter
Conclusion

• FTA is the optimal offline task assignment algorithm

• NTA can achieve a smaller average makespan in the online decision case

• The absolute error of NTA mainly depends on the expected meeting time between the requester and other users. When the average expected meeting time is very small, our algorithm can even achieve the nearly optimal result.
Thanks!

Q&A