Online Mobile Micro-Task Allocation in Spatial Crowdsourcing

Yongxin Tong [1], Jieying She [2], Bolin Ding [3], Libin Wang [1], Lei Chen [2]

[1] SKLSDE Lab, Beihang University, China
[2] The Hong Kong University of Science and Technology, Hong Kong, China
[3] Microsoft Research, Redmond, WA, USA
[1] {yxtong, lbwang}@cse.ust.hk, [2] {jshe, leichen}@cse.ust.hk, [3] bolind@microsoft.com

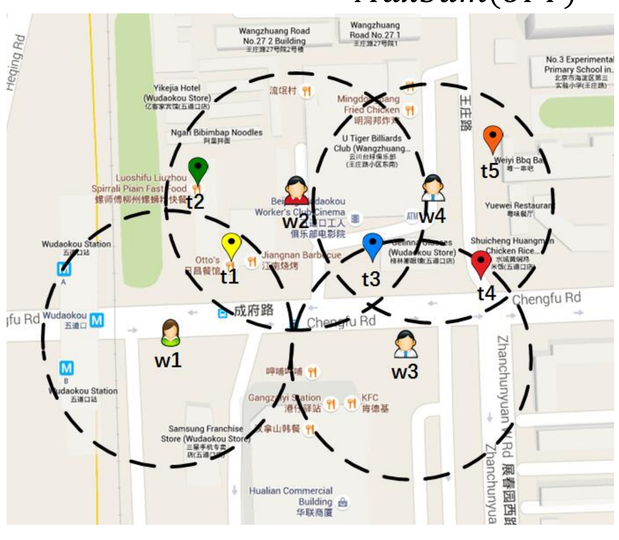
Introduction

- Spatial Crowdsourcing (a.k.a Mobile Crowdsourcing)
  - Online platforms that facilitate spatial tasks to be assigned and performed by crowd workers, e.g. O2O applications.

- Motivation
  - Dynamic micro-task assignment is absent.
  - Most O2O applications need to be addressed in real-time:
    - Fast Food Delivery.
    - Real-Time Taxi-Calling Service.
    - Product Placement Checking of Supermarkets.

The GOMA Problem

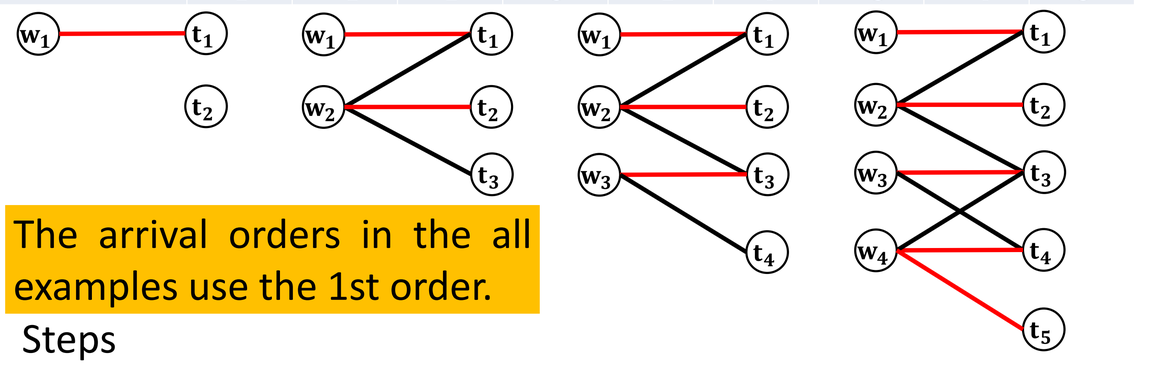
- Given
  - A set of spatial tasks T
    - Each $t \in T$: location $\boldsymbol{l}_t$, arriving time a_t, deadline d_t and payoff p_t.
  - A set of crowd workers W
    - Each $w \in W$: location $\boldsymbol{l}_w$, arriving time a_w, deadline d_w, range radius r_w, capacity c_w and success ratio δ_w.
  - Utility Function: $U(t,w) = p_t \times \delta_w$.
- Find an online allocation M to maximize the total utility *MaxSum(M)*=$\sum_{t\in T, w\in W} U(t,w)$ s.t.
  - Deadline Constraint.
  - Capacity Constraint.
  - Range Constraint.
  - Invariable Constraint: Once a task t is assigned to a worker w, the allocation of (t, w) cannot be changed.
- Online Algorithm Evaluation: Competitive Ratio (CR)
  - Adversarial Model: Worst-Case Analysis
    - $CR_A = min_{\forall G(T,W,U) and \forall v \in V} \frac{MaxSum(M)}{MaxSum(OPT)}$
  - Radom Order Model: Average-Case Analysis
    - $CR_{RO} = min_{\forall G(T,W,U)} \frac{\mathbb{E}[MaxSum(M)]}{MaxSum(OPT)}$



	t_1	t_2	t_3	t_4	t_5
$\boldsymbol{w_1}(1)$	**4**	-	-	-	-
$\boldsymbol{w_2}(1)$	3	2	**7**	-	-
$\boldsymbol{w_3}(1)$	-	-	2	**11**	-
$\boldsymbol{w_4}(2)$	-	-	6	3	**5**

Extended Greedy-RT Algorithm

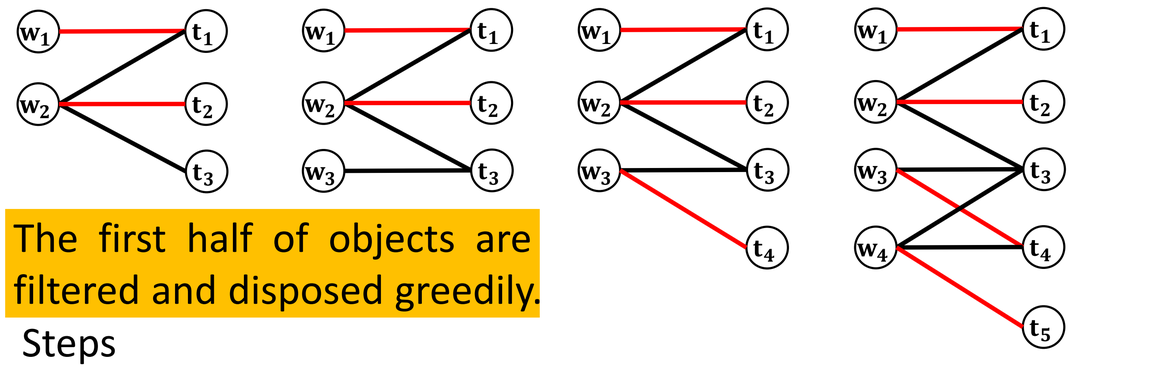
Arrival Time	8:00	8:01	8:02	8:07	8:08	8:09	8:09	8:15	8:18
1st Order	w_1	t_1	t_2	w_2	t_3	w_3	t_4	w_4	t_5
2nd Order	t_1	w_1	t_2	t_3	w_2	t_4	w_3	w_4	t_5



The arrival orders in the all examples use the 1st order.

- Steps
  - 1. Choose an integer k from 1 to $\lceil \ln(U_{max}+1) \rceil$ randomly.
  - 2. filter the edges with weights greater than e^k.
  - 3. Use a greedy strategy on the remaining edges.
- Competitive Ratio (Adversarial Model): $CR_A = \frac{1}{2e\lceil \ln(U_{max}+1) \rceil}$

Two-Phase-based Framework (TGOA Algorithm)



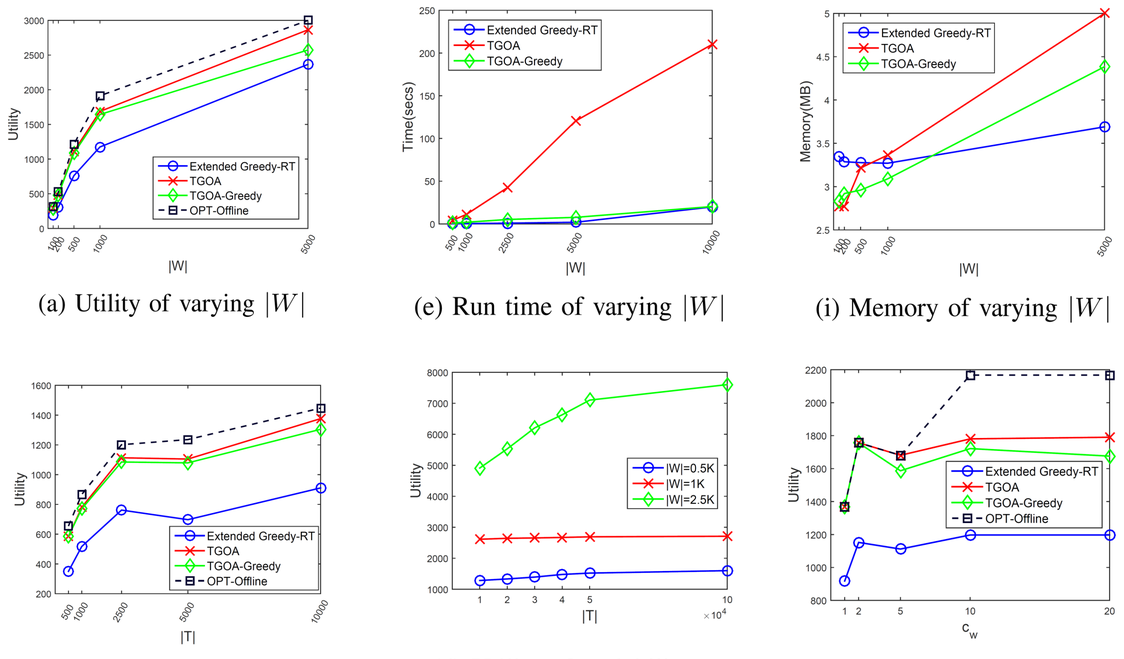
The first half of objects are filtered and disposed greedily.

- Steps
  - 1. Take a fixed fraction of arriving objects as samples and dispose the samples in a greedy way.
  - 2. When a new object arrives, compute the optimal matching on the revealed part of the graph.
  - 3. Match the new object to its adjacent node in the optimal matching if possible.
- Competitive Ratio (Random order Model):$CR_{RO} = \frac{1}{4}$

TGOA-Greedy Algorithm

- Optimize the efficiency using a greedy solution to get the matching instead of the optimal matching in the second phase.
- Competitive Ratio (Random order Model):$CR_{RO} = \frac{1}{8}$

Experimental Evaluation



(a) Utility of varying $|W|$ (e) Run time of varying $|W|$ (i) Memory of varying $|W|$

(b) Utility of varying $|T|$ (d) Utility of scalability test (a) Uility of EverySender