Supplemental Materials

Appendix A
Offline Algorithm

A.1 An optimal algorithm to Offline GOMA problem

In this subsection, we introduce an optimal algorithm to the offline version of the GOMA problem. Since the offline version of the GOMA problem can be reduced to the maximum weighted bipartite matching (MWBM) problem (detailed later), the main idea of the optimal algorithm is to first transform an offline GOMA problem instance to an MWBM instance and then adopts classical flow algorithms, *e.g.*, Hungarian algorithm [8], to calculate the optimal result.

Specifically, given an instance of the offline GOMA problem, which includes a set of tasks T, a set of workers W, and a utility function $U(.,.)$, we construct a flow network $G = (V, E)$ as follows. $V = W \cup T \cup \{s, d\}$, where s is a source node and d is a sink node. For each pair $t \in T, w \in W$, there is a weighted directed arc $e_{w,t} \in E$ with $e_{w,t}.weight = U(w,t)$ from w to t if the time interval $[a_w,\ d_w]$ of w overlaps with the time interval $[a_t,\ d_t]$ of t, namely $(a_w \leq d_t) \wedge (a_t \leq d_w)$. In addition, for each $w \in W$, there is a directed arc $e_{s,w} \in E$ from s to w with $e_{s,w}.weight = 0$. Similarly, for every $t \in T$, there is also a directed arc $e_{t,d} \in E$ from t to d with $e_{t,d}.weight = 0$. The total amount of flow in the network is $\min(|T|, |W|)$. And the flow network instance is constructed. Then, we use existing flow algorithms to obtain the optimal result of the MWBM instance. To obtain the allocation result of the offline GOMA problem, we simply assign a worker w to a task t if there is a non-zero flow between them in the MWBM result.

Appendix B
Proof for Extended Greedy-RT

B.1 Proof of Lemma 1:

Proof. *WLOG* we use G to denote the corresponding weighted bipartite graph of the GOMA input. Let $G_{[e^i,e^{i+1})}$ be a subgraph of G, which only contains the edges whose utilities lie in the interval $[e^i, e^{i+1})$, and $OPT_{[e^i,e^{i+1})}$ and $M_{[e^i,e^{i+1})}$ be the optimal match and the one returned by Extended Greedy-RT on $G_{[e^i,e^{i+1})}$, respectively. For any edge in $OPT_{[e^i,e^{i+1})}$, at least one of the two vertices of this edge must be matched in $M_{[e^i,e^{i+1})}$. So $|M_{[e^i,e^{i+1})}| \geq \frac{1}{2}|OPT_{[e^i,e^{i+1})}|$ and we have

$$\begin{aligned}
\mathbb{E}[\text{MaxSum}(M)] &= \frac{1}{\theta}\sum_{i=0}^{\theta-1}\text{MaxSum}(M_{[e^i,U_{max})}) \\
&\geq \frac{1}{\theta}\sum_{i=0}^{\theta-1} e^i \cdot |M_{[e^i,e^{i+1})}| \\
&\geq \frac{1}{\theta}\sum_{i=0}^{\theta-1} e^i \cdot \frac{|OPT_{[e^i,e^{i+1})}|}{2} \\
&\geq \frac{1}{2e\theta}\sum_{i=0}^{\theta-1}\text{MaxSum}(OPT_{[e^i,e^{i+1})}) \\
&\geq \frac{1}{2e\theta}\text{MaxSum}(OPT)
\end{aligned}$$

Thus, the expected competitive ratio of Extended Greedy-RT under the adversarial order model is

$$CR_{AO} = \frac{\mathbb{E}[\text{MaxSum}(M)]}{\text{MaxSum}(OPT)} = \frac{1}{2e\theta} = \frac{1}{2e\lceil\ln(U_{max}+1)\rceil}.$$

where U_{max} is the maximum utility of the GOMA input. $\square$

Appendix C
Proofs for TGOA-Greedy

C.1 Proof of Lemma 6

Proof. Let $M^{Gdy}_{(|T^\Delta|,n)}$ be the matching returned by Algo. 3 for the offline weighted bipartite graph. According to [10], $\text{MaxSum}(M^{Gdy}_{(|T^\Delta|,n)}) \geq \frac{1}{2}\text{MaxSum}(OPT_{(|T^\Delta|,n)})$, hence:

$$\mathbb{E}[\text{MaxSum}(M^{Gdy}_v)] \geq \frac{|T^\Delta||W^\Delta|}{2mn}\mathbb{E}[\text{MaxSum}(M^{Gdy}_{(|T^\Delta|,n)})]$$

$\square$

C.2 Proof of Theorem 2

Proof. According to Lemma 5 and Lemma 6, the final match returned by the TGOA-Greedy algorithm satisfies that

$$\begin{cases}\mathbb{E}[U(v,w) \in M] \geq \frac{\lfloor\frac{m+n}{4}\rfloor}{(|T^\Delta|-1)} \cdot \frac{|W^\Delta|}{2mn}\mathbb{E}[\text{MaxSum}(OPT)] & v \in T\\ \mathbb{E}[U(t,v) \in M] \geq \frac{\lfloor\frac{m+n}{4}\rfloor}{(|W^\Delta|-1)} \cdot \frac{|T^\Delta|}{2mn}\mathbb{E}[\text{MaxSum}(OPT)] & v \in W\end{cases}$$

By Theorem 1, $\mathbb{E}[\text{MaxSum}(M)] \geq \frac{1}{8}\mathbb{E}[\text{MaxSum}(OPT)]$. $\square$

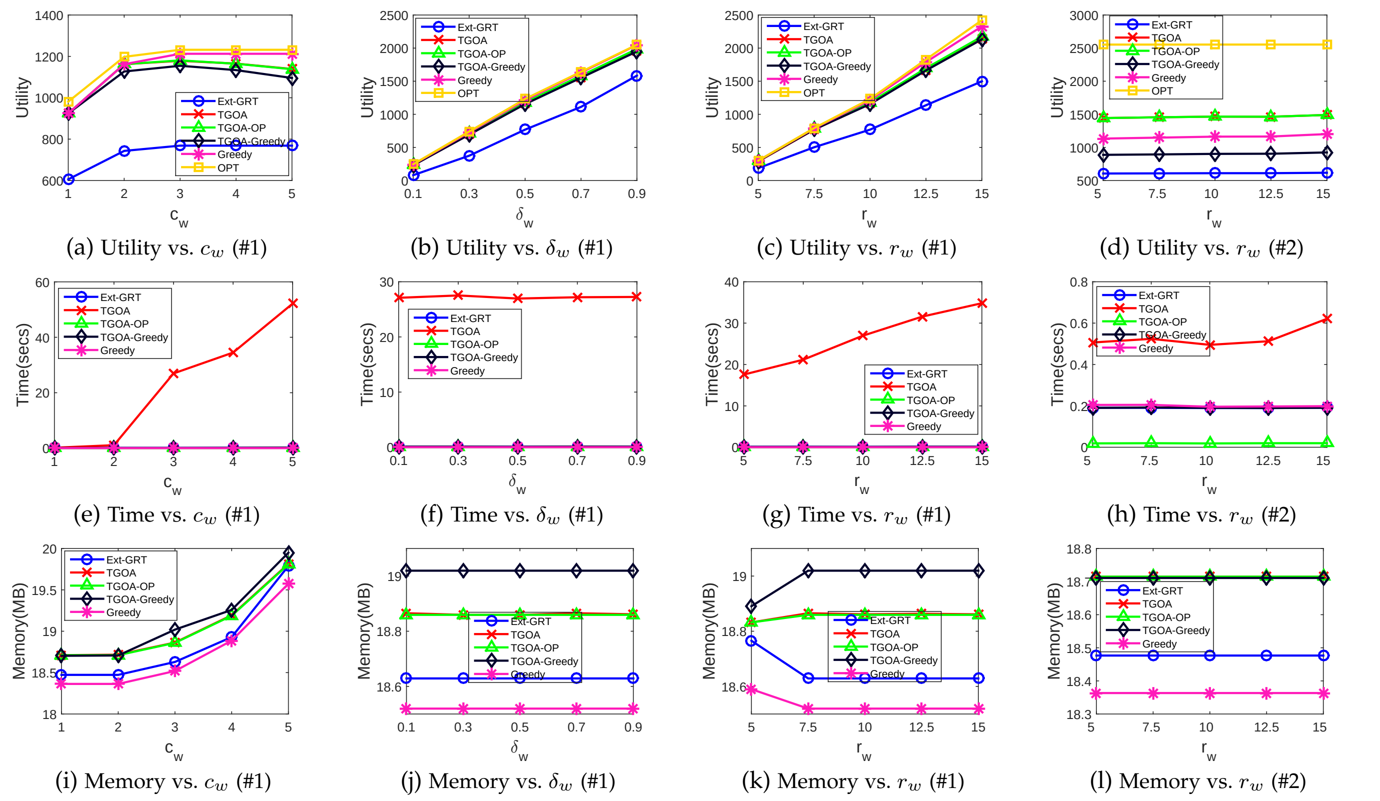
Appendix D
Experiment Results

D.1 Impact of capacity c_w on *Syn#1*

The results of varying c_w are presented in the first column of Fig. 6. The total utility increases at first when c_w increases. The TGOA-based algorithms and the Greedy algorithms performs better than the baseline Extended Greedy-RT. Greedy is the most effective and TGOA-OP is nearly as effective as it (*e.g.*, with no more than 5.14% lower utility). As for the running time, Greedy is the most efficient, and TGOA-Greedy and TGOA-OP have nearly the same efficiency as the baseline. As for the memory cost, the TGOA-based algorithms consume more space than Extended Greedy-RT while Greedy needs less space.

D.2 Impact of success ratio δ_w on *Syn#1*

The results of varying the success ratio of workers are presented in the second column of Fig. 6. The utility of all the algorithms increases with the increase of success ratio of workers. This is because the utility of each pair of task and worker becomes larger. Greedy is the most effective and the TGOA-based algorithms are more effective than Extended Greedy-RT. For the running time, TGOA is the most inefficient while Greedy is the most efficient. TGOA-OP is also efficient since each request is responded within 5ms in average. TGOA-based algorithms consume a little more memory, but the memory cost of all the algorithms seems insensitive to the success ratio of workers.

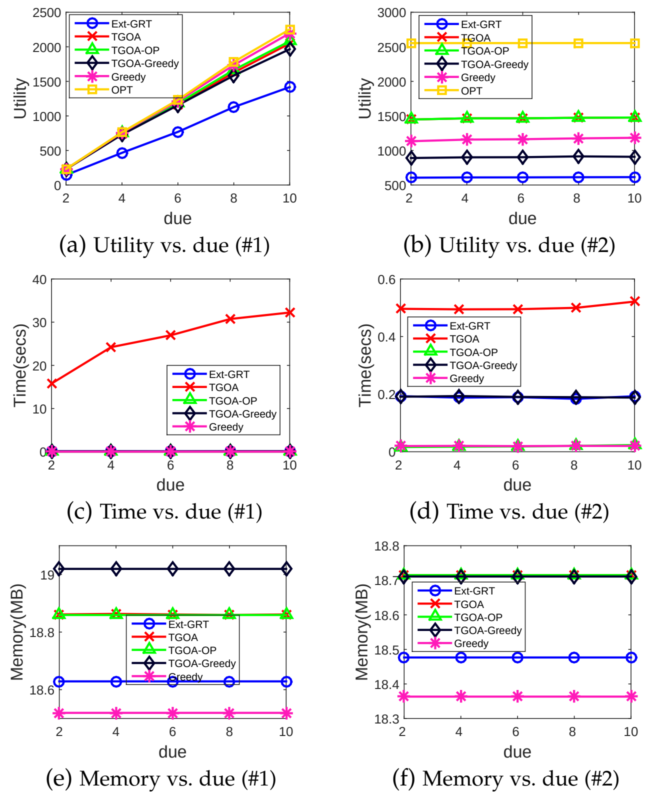


(a) Utility vs. c_w (#1) (b) Utility vs. δ_w (#1) (c) Utility vs. r_w (#1) (d) Utility vs. r_w (#2)

(e) Time vs. c_w (#1) (f) Time vs. δ_w (#1) (g) Time vs. r_w (#1) (h) Time vs. r_w (#2)

(i) Memory vs. c_w (#1) (j) Memory vs. δ_w (#1) (k) Memory vs. r_w (#1) (l) Memory vs. r_w (#2)

Fig. 6: Results on varying capacity c_w, success ratio δ_w and radius r_w in synthetic datasets.



(a) Utility vs. due (#1) (b) Utility vs. due (#2)

(c) Time vs. due (#1) (d) Time vs. due (#2)

(e) Memory vs. due (#1) (f) Memory vs. due (#2)

Fig. 7: Results on varying response deadline *due* on synthetic datasets.

D.3 Impact of radius r_w of restricted range

The results of varying the restricted range are shown in the last two columns of Fig. 6. The utility of all the algorithms increases when the restricted range r_w increases, since workers can conduct more tasks. The increase on *Syn#2* is less notable since the locations of tasks are dependently generated based on the locations of workers. Greedy achieves near-optimal results (up to 57.78% higher than Extended Greedy-RT) on *Syn#1*, but is less effective than TGOA and TGOA-OP with 25.95% lower utility on *Syn#2*. TGOA-Greedy is also more effective than Extended Greedy-RT, with 49.76% and 47.84% higher utility on *Syn#1* and *Syn#2* respectively. In terms of time and memory costs, Greedy is the most efficient and TGOA is the least efficient. Both TGOA-Greedy and TGOA-OP are efficient with 180ms running time and 19MB space.

D.4 Impact of due (deadline)

The results of varying deadline are presented in Fig. 7. The utility tends to increase when the deadline becomes longer, since more tasks and workers can be matched. For the similar reason, the increase is less clear on *Syn#2*. All the proposed algorithms are more effective than Extended Greedy-RT. Greedy is the most effective on *Syn#1*. TGOA and TGOA-OP are the most effective on *Syn#2*. As for the time and memory costs, the patterns are similar to previous experimental results.