SCIENCE CHINA
Information Sciences

• Supplementary File •

Profit-aware online crowdsensing task assignment for intelligent transportation services

Guanglei Zhu[1], Yafei Li[1,2,3*], Jiawen Zhang[1], Ke Wang[1,2,3], Lei Chen[4] & Mingliang Xu[1,2,3*]

[1]*School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou 450001, China*
[2]*National Supercomputing Center In Zhengzhou, Zhengzhou 450001, China*
[3]*Engineering Research Center of Intelligent Swarm Systems, Ministry of Education, Zhengzhou 450001, China*
[4]*Huawei Noah's Ark Lab, Hong Kong SAR, China*

Appendix A Proof of Theorem 1

Proof. We establish the hardness of the POCTA problem by transforming it into an instance of a classical NP-hard problem, known as the 0-1 knapsack problem, which can be described as follows. Given a set of items $\mathcal{I}$ and a backpack B with maximum capacity c_B, where each item has two properties, i.e., weight w_i and value v_i, the 0-1 knapsack problem aims to select a subset of items that maximizes the total value in the backpack. It is subject to two conditions: (i) the total weight of the items in B is no more than c_B, and (ii) each item can be selected only once.

We consider an instance of the POCTA problem as follows. Given a set of tasks $\mathcal{T}$ and a set of workers W, the POCTA problem aims to find a subset $\mathcal{T}_{sub}$ of $\mathcal{T}$ that maximizes the total platform profit. It satisfies two conditions: (i) the size of $\mathcal{T}_{sub}$ does not exceed $c_w \cdot |W|$, and (ii) each task τ in $\mathcal{T}$ is assigned to at most one worker w in W. It is obvious that the instance of the POCTA problem can be mapped to the 0-1 knapsack problem. Therefore, the proof is completed.

Algorithm A1 Multi-round bipartite graph matching

Input: A set of workers W_t, a set of tasks $\mathcal{T}_t$;
Output: A matching plan M_t;
1: $\mathcal{B}_t \leftarrow \emptyset$ and $M_t \leftarrow \emptyset$;;
2: **for** each pair $(w, \tau) \in W_t \times \mathcal{T}_t$ **do**
3: **if** the matching instance (w, τ) is valid **then**
4: $u(w, \tau) \leftarrow$ Calculate matching reward;
5: $\mathcal{B}_t \leftarrow \mathcal{B}_t \bigcup \{(w, \tau, u(w, \tau))\}$;
6: **end if**
7: **end for**
/* *Multiple-round KM Matching* */
8: Build the weighted bipartite graph $\mathcal{G}_b$ based on $\mathcal{B}_t$;
9: **while** $\mathcal{G}_b \neq \emptyset$ **do**
10: Find the optimal matching M_t' in $\mathcal{G}_b$ using KM algorithm;
11: $M_t \leftarrow M_t \cup M_t'$;
12: Update $\mathcal{G}_b$ based on M_t';
13: **end while**
Return: M_t

Appendix B Task assignment for the POCTA problem

Appendix B.1 Multi-round bipartite graph matching

For the MBiGM algorithm, given a set of workers W_t and tasks $\mathcal{T}_t$ in the t-th matching batch b_t, we construct a weighted bipartite graph $\mathcal{G}_b = (W_t, \mathcal{T}_t, \mathcal{E}_t)$, where $\mathcal{E}_t$ denotes the set of valid matching edges between workers and tasks. Each edge $e \in \mathcal{E}_t$ links a worker $w \in W_t$ and a task $\tau \in \mathcal{T}_t$ if the matching of w and τ is valid. The edge is associated with a weight that indicates the matching revenue $u(w, \tau)$ for w serving τ. The Kuhn-Munkres (KM) algorithm [1] is widely adopted for efficiently solving one-to-one assignment problems in bipartite graphs, where each left node is matched to at most one right node. However, in the POCTA problem, each worker may be assigned multiple tasks. Therefore, we design MBiGM to derive the optimal matching plan M_t based on $\mathcal{G}_b$.

Algorithm A1 presents the pseudo-code of MBiGM, which first computes edge weights in $\mathcal{G}_b$ (Lines 2–7), followed by task assignment using the KM algorithm (Lines 8–13). In each round, we update $\mathcal{G}_b$ based on M_t' by removing matched pairs and updating the statuses of the corresponding tasks and workers. We finally return the matching plan M_t.

Complexity analysis. The time complexity of MBiGM is $O(k \cdot max\{n^3, m^3\})$, where k, m, n are the number of matching rounds, workers, and tasks, respectively.

* Corresponding author (email: ieyfli@zzu.edu.cn, iexumingliang@zzu.edu.cn)

Algorithm B1 Breaking-based task packing assignment

Input: A set of workers W_t, a set of tasks $\mathcal{T}_t$;
Output: A matching plan M_t;
1: $\mathcal{G}_b \leftarrow \emptyset$, $M_t \leftarrow \emptyset$;
/* *Step1: Task packing* */
2: $M_D \leftarrow$ Compute distances between tasks in $\mathcal{T}_t$;
3: $Tree_Z \leftarrow$ Perform hierarchical clustering on M_D;
4: $\Gamma^{\mathcal{T}} \leftarrow fcluster(Tree_Z, \delta)$;
/* *Step2: Package breaking* */
5: **for** Each package $\Gamma^\tau \in \Gamma^{\mathcal{T}}$ **do**
6: $W_1^{\Gamma^\tau} \leftarrow$ Find the candidate workers by using the index strategy TBIL and d^τ_{max};
7: **if** $W_1^{\Gamma^\tau} \neq \emptyset$ **then**
8: $W_2^{\Gamma^\tau} \leftarrow$ Filter workers with valid schedule constraint;
9: **if** $W_2^{\Gamma^\tau} \neq \emptyset$ **then**
10: $\mathcal{G}_b \leftarrow \mathcal{G}_b \cup \{(w_i, \Gamma^\tau, [u(w_i, \Gamma^\tau), \Delta g(w_i, \Gamma^\tau)]) | w_i \in W_2^{\Gamma^\tau}\}$;
11: **else**
12: $\Gamma_1^\tau, \Gamma_2^\tau \leftarrow SplitPackage(\Gamma^\tau, c'_{max})$;
13: $\mathcal{G}_b \leftarrow \mathcal{G}_b \cup \{(w_i, \Gamma_1^\tau, [u(w_i, \Gamma_1^\tau), \Delta g(w_i, \Gamma_1^\tau)]) | w_i \in W_1^{\Gamma^\tau},\ c_{w_i} \geqslant c'_{max}\}$;
14: $\Gamma^{\mathcal{T}} \leftarrow \Gamma^{\mathcal{T}} \setminus \{\Gamma^\tau\} \cup \{\Gamma_2^\tau\}$;
15: **end if**
16: **else**
17: $\Gamma_s^\tau \leftarrow$ Remove the tasks with $\bar{t}_{min}$ from Γ^τ;
18: **if** $\Gamma_s^\tau \neq \emptyset$ **then**
19: $\Gamma^{\mathcal{T}} \leftarrow \Gamma^{\mathcal{T}} \cup \{\Gamma_s^\tau\}$;
20: **end if**
21: **end if**
22: **end for**
/* *Step3: Multi-round matching* */
23: **while** $\mathcal{G}_b \neq \emptyset$ **do**
24: $\mathcal{E} \leftarrow$ Sort pairs of $\mathcal{G}_b$ in descending order of $\Delta V(w, \Gamma^\tau)$;
25: **while** $|\mathcal{E}| \neq \emptyset$ **do**
26: $(\overline{w}, \overline{\Gamma^\tau}) \leftarrow$ Obtain the first pair in $\mathcal{E}$;
27: **if** $\Delta V(\overline{w}, \overline{\Gamma^\tau})$ is the maximum of $(w_i, \overline{\Gamma^\tau})$ in $\mathcal{G}_b$ **then**
28: $M_t \leftarrow M_t \cup \{(\overline{w}, \overline{\Gamma^\tau})\}$;
29: Remove the pairs of $(w_i, \overline{\Gamma^\tau})$ in $\mathcal{E}$;
30: **end if**
31: Remove the pairs of $(\overline{w}, \Gamma_j^\tau)$ in $\mathcal{E}$;
32: **end while**
33: Update $\mathcal{G}_b$ based on M_t;
34: **end while**
Return: M_t

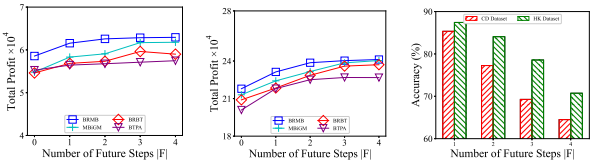
Appendix B.2 Packing-aware matching operator

Algorithm B1 outlines three steps of the BTPA algorithm, i.e., task packing (lines 2-4) and package breaking (lines 5-22), and a multi-round matching mechanism (lines 23-34). We elaborate on each step below.

Task packing. To reduce computational overhead, we group spatially proximate tasks (e.g., in the same neighborhood or on the same street) into a single package Γ^τ. The pairwise distance between any two tasks in Γ^τ satisfies $\pi(l_{\tau_i}, l_{\tau_j}) \leqslant \delta$, where δ is a predefined threshold indicating the maximum allowable separation. We employ a δ-distance truncated dendrogram-based task packing technique to generate task packages (Lines 2–4). We adopt the δ-distance truncating dendrogram task packing technique to pack tasks into packages (lines 2-4). Specifically, we first compute the spatial distance matrix M_D for tasks in $\mathcal{T}_t$, followed by hierarchical clustering on M_D to group tasks by proximity. We then apply the function $fcluster(\cdot)$ to truncate clusters where intra-cluster distances do not exceed δ, thereby obtaining the package set $\Gamma^{\mathcal{T}}$ in accordance with the packing policy.

Package breaking. After task packing, we identify workers who can feasibly serve each package Γ^τ. Due to task deadline constraints, the reachable distance for Γ^τ is determined by the minimum remaining time $\bar{t}_{\min}$ across all tasks in the package, computed as $d^\tau_{\max} = \bar{t}_{\min} \times V_{\max}$, where $V_{\max}$ denotes the maximum allowable traffic speed. We also adopt the Time-Bounded Inverted Lists (TBIL) method [2] to index workers and update their status, thereby accelerating candidate worker retrieval. Note that $d^\tau_{\max}$ serves as the initial search radius, but in some cases, no reachable worker can be found within this range. To address this, we design a package-breaking strategy (Lines 7–19), where tasks with the smallest remaining time $\bar{t}_{\min}$ are iteratively removed to form a sub-package Γ_s^τ until a candidate worker set $W_1^{\Gamma^\tau}$ is found or the package becomes empty. We then refine the candidate set to $W_2^{\Gamma^\tau}$, which includes only those workers who satisfy the valid scheduling constraints (as defined in Definition 3) for the entire package. If no such workers exist, we invoke the function $SplitPackage(\Gamma^\tau, c'_{\max})$, which greedily selects workers capable of executing the largest number of tasks in the package. Here, $c'_{\max}$ denotes the maximum remaining c_w of worker $w \in W_1^{\Gamma^\tau}$. Finally, we add the candidate matching pair (w, Γ^τ), along with its dual weights $[u(w, \Gamma^\tau), \Delta g(w, \Gamma^\tau)]$, to the bipartite graph $\mathcal{G}_b$ (Lines 10 and 13).

Multi-round matching mechanism. We sort all edges $\mathcal{G}_b$ in descending order according to the matching value $\Delta V(w, \Gamma^\tau) = u(w, \Gamma^\tau) - \Delta g(w, \Gamma^\tau)$ (line 24). Note that we do not leverage $\Delta V(w, \Gamma^\tau)$ to replace the dual weights in $\mathcal{G}_b$. Because, in the breaking operator, we need to examine their effects separately. In each round (lines 25-31), we choose the matching pair



(a) F VS HK (b) F VS CD (c) F VS ACC

Figure B1 Matching results on CD and HK datasets with different F.

$(\overline{w}, \overline{\Gamma^\tau})$ with the highest matching value, and update related matching pairs in $\mathcal{E}_t$ that associated with $\overline{w}$ and $\overline{\Gamma^\tau}$. At the end of each round, we update $\mathcal{G}_b$ according to the current matching plan M_t (line 33). Finally, BTPA outputs the final optimal matching plan M_t.

Complexity analysis. The time complexity of BTPA is $O(m^2 \log m + pn + pn \log pn)$, where m, n, and p denote the number of tasks, workers, and packages, respectively. Specifically, $O(m^2 \log m)$ is incurred by the task packing step, $O(pn)$ arises from the package breaking process, and $O(pn \log pn)$ accounts for the multi-round matching procedure.

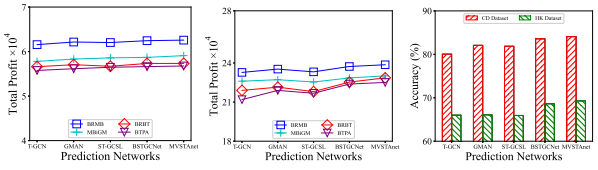
Table B1 Matching results on HK dataset by varying δ values.

δ	Metrics	10	25	50	100	200
BTPA	TP	60883.73	56761.86	53867.23	53464.38	50993.01
	BPT	658.13	535.13	423.14	327.13	234.13
PBO	TP	57384.72	55127.77	53031.55	52143.44	47562.24
	BTR	435.16	371.14	314.16	226.61	134.16

Appendix C Additional experiments

Detail of experimental parameters. The detailed experimental settings for the proposed methods in task prediction and task assignment are summarized as follows:

- *Task prediction.* For MVSTAnet training, we adopt the Adam optimizer with a mini-batch size of 48, a hidden size of 64, a learning rate of 0.005, and 1000 training epochs. The dataset is divided into training, testing, and validation sets in the ratio of 70%, 20%, and 10%, respectively. The entire road network is partitioned into 100 non-overlapping regions of equal size (i.e., $|\mathcal{R}| = 100$), and each day is discretized into $\frac{24\times 60 \text{ min}}{\Delta t}$ time steps, where we set $\Delta t = 15$ minutes in default for task prediction. The lengths of the three temporal input sequences (i.e., closeness X_C, period X_P, and trend X_Q) are set to be equal, i.e., $l_c = l_p = l_q = P$.
- *Task Assignment.* We use data from the HK dataset on May 1st and the CD dataset on November 1st, spanning the period from 6:00 to 13:00, which includes both idle (6:00–8:00) and peak (9:00–13:00) hours, to evaluate the performance of task matching algorithms. A total of 15,237 and 57,961 order instances are observed in the HK and CD datasets, respectively, during the selected period. We set the matching batch size to 20 seconds and the prediction time step to 2 minutes. Besides, we set coefficients $\alpha = \beta = \epsilon = 0.5$, and $\lambda = 0.8$ by default. To more clearly investigate the effect of supply and demand on prices, we set the fare of the task as $f_\tau = 10$.



(a) Error VS HK (b) Error VS CD (c) Error VS ACC

Figure C1 Matching results on CD and HK datasets with different prediction error values.

Exp-1: Effect of δ. We examine the effect of package size δ on the performance of packing algorithms PBO [2] and BTPA. As shown in Table B1, increasing the value of δ leads to a decline in TP for both PBO and BTPA. This is attributed to the inherent trade-off in packing strategies, which sacrifice task quality to improve computational efficiency. As the package size enlarges, the corresponding loss in task quality also increases. Moreover, BTPA suffers less performance degradation compared to PBO, as it incorporates a package-breaking mechanism that mitigates quality loss. Therefore, in practical scenarios, both BTPA and PBO offer viable trade-offs between computational efficiency and task quality.

Exp-2: Effect of F. Figure B1 (a)(b) report the effects of varying future time steps F on the performance of matching algorithms, and Figure B1 (c) presents the accuracy (ACC) of our proposed prediction model MVSTAnet under different F values. As F increases, the TP of matching algorithms generally improves. However, we observe that beyond a certain threshold (e.g., $F = 2$ in the HK dataset), further increases in F do not yield significant gains in TP. We attribute this to two main factors: (i) forecasting a greater number of future time steps increases the model's prediction difficulty, as evidenced by the drop in ACC in Figure B1 (c), leading to higher prediction errors; and (ii) the marginal benefit of distant future information for current task assignment diminishes over time. Consequently, although TP increases with longer forecasting horizons, the marginal improvement gradually approaches zero.

Exp-3: Effect of prediction error. Figure C1 (a)(b) illustrate how prediction errors affect task assignment optimization. As shown in Figure C1 (c), under default settings, different prediction models exhibit varying levels of prediction error. It is evident that smaller prediction errors, which indicate higher model precision, are associated with improved TP. Furthermore, MVSTAnet consistently outperforms other models by achieving the highest prediction accuracy and, consequently, delivering the best task assignment outcomes.

References

1 Kuhn H W. The Hungarian method for the assignment problem. Naval research logistics quarterly, 1955, 2: 83–97
2 Li Y F, Li Y F, Peng Y, et al. Auction-Based Crowdsourced First and Last Mile Logistics. IEEE Trans Mob Comput, 2024, 23: 180–193