AMRAS: A Visual Analysis System for Spatial Crowdsourcing

Qingshun Wu, Yafei Li*, Huiling Li, Di Zhang, Guanglei Zhu
School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou, China
{wqszzu, hlli, dzhangzzu, glzhu}@gs.zzu.edu.cn, ieyfli@zzu.edu.cn

ABSTRACT

The wide adoption of GPS-enabled smart devices has greatly promoted spatial crowdsourcing, where the core issue is how to assign tasks to workers efficiently and with high quality. In this paper, we build a novel visual analysis system for spatial crowdsourcing, namely AMRAS, which can not only intuitively present the task allocation for workers under different time window scales to users (e.g., data analysts and managers) in real-time, but also help users analyze task assignment decision model and its learning process. AMRAS has the following novel features. First, AMRAS provides two user-friendly interfaces that allow users to employ simple and easy-to-use console to perform statistical analysis. Secondly, AMRAS provides three powerful visualization tools, such as the visualization of assignment results, assignment process, and assignment decision model, which not only allow users to intuitively analyze the whole process of task assignment, but also help users discover the computational bottleneck of their task assignment solution. Finally, AMRAS enables online access to real-time data, providing users with instant assignment and instant analysis. We have implemented and deployed AMRAS on Alibaba Cloud and demonstrated its usability and efficiency in real-world datasets. The demonstration video of AMRAS has been uploaded to Google Drive.

PVLDB Reference Format:
Qingshun Wu, Yafei Li, Huiling Li, Di Zhang, Guanglei Zhu. AMRAS: A Visual Analysis System for Spatial Crowdsourcing. PVLDB, 15(12): 3690-3693, 2022.
doi:10.14778/3554821.3554876

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/wuqingshun/AMRAS.

1 INTRODUCTION

With the widespread diffusion of smart devices, Spatial Crowdsourcing (SC) [10] has attracted considerable attention from industry, such as Gigwalk, Uber, and Waze. According to the publishing model, SC applications can be classified into two types: *i) server assigned tasks (SAT)* [3, 5, 6, 8, 11], in which the SC server assigns location-dependent tasks to smart devices users, called workers; *ii) worker selected tasks (WST)* [1, 9], in which location-dependent tasks are selected by moving workers. Moreover, from assignment styles, the SAT model has two styles: instant task assignment [6, 8] and batch-based task assignment [3, 5]. Specifically, in the instant task assignment, the SC servers need to instantly assign available tasks to workers in a one-by-one style. It is easy to implement but may not generate high quality assignment results [10]. In reality, to further improve the quality of assignment results, most SC platforms adopt batch-based task assignment, where the servers accumulate a batch of tasks and then assign them to workers. The illustrative example of batch-based task assignment is shown in Figure 1, where the decision model monitors and captures the spatio-temporal distribution of the workers/tasks in the SC platform at each time slot, and judges whether the current time slot is suitable for dividing batch; then assigns tasks to workers within each batch and feeds back the assignment results to the SC platform. Existing researches on batch-based task assignment can be divided into three modes: fixed value-based [5], history-based [2], and learning-based [4], but there is currently no a visualization system to demonstrate the assignment process and results of these approaches. Motivated by above discussion, we intend to study a visual analysis system to manage and analyze the large assignment data generated by these approaches on the SC platform.

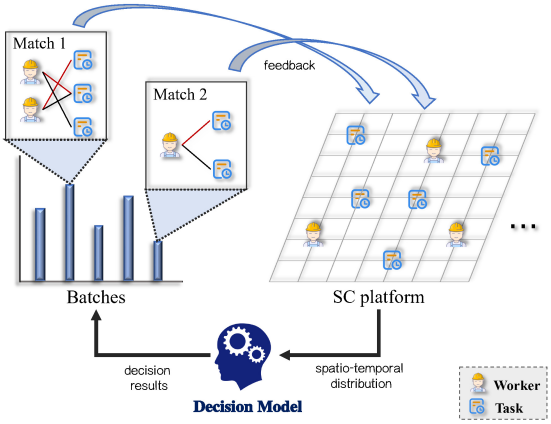


Figure 1: Illustrative example of batch-based task assignment

There are three major challenges to analyze batch-based task assignment data: (1) how to analyze the dynamics of assignment process? (2) how to help users identify batch size and quality easily? (3) how to help users judge the effectiveness of learning-based decision model and the rationality of its structure?

In this demonstration, we design and implement a visual analytical system called AMRAS, which performs efficient massive assignment data analysis for batch-based task assignment. It includes Web-based UI and computation engine modules, which aims to allow users (e.g., data analysts and managers in spatial crowdsourcing company) to intuitively analyze the assignment data from

* Corresponding author: Yafei Li

Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097.
doi:10.14778/3554821.3554876

SC platform and assist them to optimize their assignment algorithm. Additionally, AMRAS provides two user-friendly interfaces (i.e., Analysis Interface and Decision Model Interface) that allow users to customize the parameters required for the console, and visualize the instant results. In particular, users can analyze the learning process of the learning-based decision model in the Decision Model Interface. The demonstration video of AMRAS has been uploaded to Google Drive[1].

The main contributions of our work are summarized as follows:

- We design and implement the AMRAS, a visual analytical system that performs task assignment analysis efficiently. To the best of our knowledge, this is the first work on spatial task assignment results analysis.
- We provide three powerful visualization tools to allow users to intuitively analyze the task assignment and assist them to optimize their matching algorithm.
- We demonstrate how our AMRAS works through three demo scenarios on a real-world dataset.

2 AMRAS OVERVIEW

2.1 Problem Formulation

The task assignment problem can be formulated as: Given a task stream Γ and a set of workers W, the problem returns an assignment plan M of tasks to workers $M = \{(w, \tau) | w \in W, \tau \in \Gamma\}$ such that for given revenue function $\Phi(\cdot)$, $\Phi(M)$ is maximized:

$$\Phi(M) = \sum_{(w,\tau)\in M} \phi(w, \tau). \quad (1)$$

Next, we will define two basic concepts: worker and task.

Worker. A worker, denoted by $w = (l, k_c, d)$, has a location $w.l$, a service ability $w.k_c$, and a service distance $w.d$. In practice, the service ability of workers represents the number of seats in vehicles and the capacity of backpacks in logistics. The service range of worker w is a circle with $w.l$ as the center and $w.d$ as the radius. The ability $w.k_c$ and distance $w.d$ of worker w determine whether he/her can accept assignment.

Task. A task, denoted by $\tau = \{l, a_t, e_t, f\}$, has a location $\tau.l$, an appearance time $\tau.a_t$, an expiration time $\tau.e_t$, and a fee $\tau.f$. In spatial crowdsourcing, a task τ can be assigned only if $\tau.l$ is within a worker's service range after $\tau.a_t$ and the worker has sufficient service ability. Moreover, considering the expiration time, a task τ can be completed only if a worker can physically move to $\tau.l$ before $\tau.e_t$. Hence, there are six types of tasks: New (just appeared), Matched (already assigned to worker), Completed (already completed by worker), Expired I (unassigned expiration), and Expired II (assigned but not completed expiration). Note that a worker can gain revenue $\tau.f$ only if the task τ is in type "Completed".

Assignment Process. Initially, the customers send in their tasks to the server, while the workers report their current statuses (e.g., location and service ability) to the server. After receiving the tasks, the server divides the tasks into a sequence of batches to match based on some division strategies. According to batch's sequence, the server conducts the assignment for the tasks and available workers in a batch to maximize the revenue of the platform, while the available workers and unmatched tasks return to the streams for the next batch matching, and the expired tasks will be discarded from the server. Finally, the assignment results are notified to the workers and customers, respectively.

[1]https://drive.google.com/file/d/1ZNRNR-qG-au1A4wThBnMIS1LEqtfRBKk/view

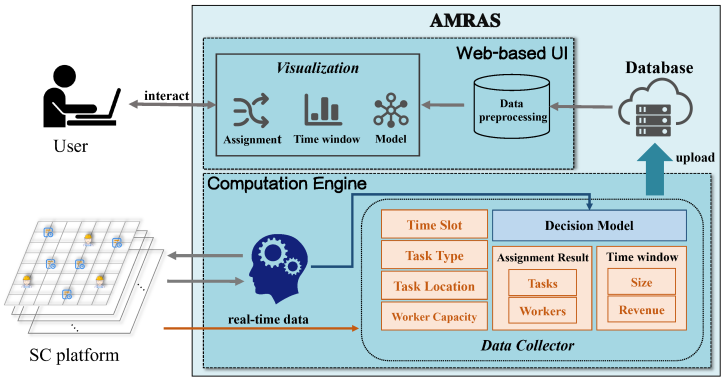


Figure 2: Overview of the AMRAS architecture

2.2 System Framework

The architecture of AMRAS is shown in Figure 2, which mainly includes two modules: Computation Engine and Web-based UI.

Computation Engine Module. The computation engine module is the core module that performs task assignment by utilizing decision model to divide batches. The data collector can collect assignment data, which is composed of real-time data from SC platform and learning data from learning-based decision model. Specifically, the real-time data includes: time slot, task type, task location, worker service ability, assignment result, time window size and revenue. Herein, a time window can be denoted by $\Psi = \{\psi_0, \psi_1, \ldots, \psi_n | n \in \mathbb{N}\}$, where ψ_n is a time slot. Then, we upload the above data to the database for use by the Web-based UI module.

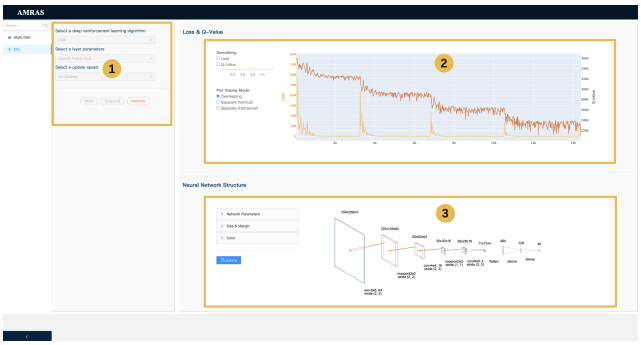
Web-based UI Module. The Web-based UI module is implemented upon the computation engine module, which bridges the gaps between the users and SC platform. This module allows users to conduct their analytical tasks by simple operations, which is vital for those users who are not familiar with SQL-like query language. First, we extract heterogeneous and massive assignment data from database, and preprocess the assignment data by exploiting the data cleansing techniques in previous work [7]. It builds spatial objects index to accelerate spatial query processing. Furthermore, we devise a suite of visualization tools (assignment visualization, time window visualization, and decision model visualization) to illustrate the analyzing results effectively.

3 DEMONSTRATION

The Web-based UI module of AMRAS contains the analysis and decision model interfaces as shown in Figure 3. It is developed using Dash (an efficient Python framework for building web applications based on Flask[2], Plotly.js[3] and React.js[4]). Its client-side is written in JavaScript under the ReactJS framework. Its server-side is written

[2]https://flask.pocoo.org/
[3]https://www.plotly.com/
[4]https://www.reactjs.org/

(a) Analysis Interface



(b) Decision Model Interface

Figure 3: Demonstration of AMRAS

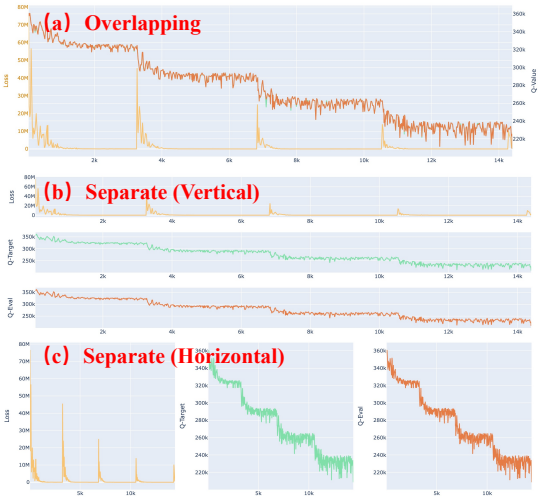


Figure 4: Display Mode

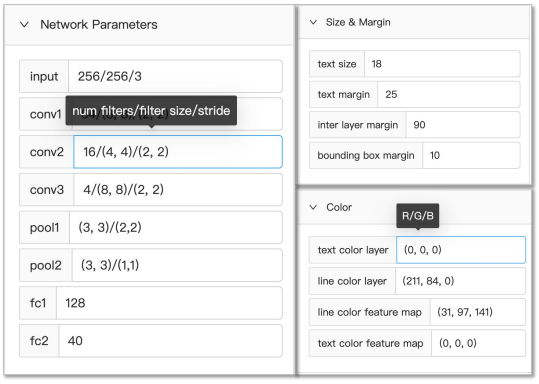


Figure 5: Network Structure Parameters

under the Flask framework. PlotlyJS is the graphics library adopted by Dash. We use RDS mysql, the Alibaba cloud version of MySQL, as the document database of the application. In addition, we use Gunicorn as the Web server and deploy the application to Aliyun[5].

3.1 Analysis and Decision Model Interfaces

As shown in Figure 3(a), the analysis interface contains 7 functional panels. Panel-① is the console, where we can set the time period and task type. When we slide the time period axis or select different task types, the contents displayed in the other six panels will change accordingly. Panel-② is a summary of the overall tasks, where we can view the proportion of different task types. Panel-③ is the task data statistics, where we can view the total number of tasks, total revenue, task assignment rate and completion rate. Panel-④ is a time window visualization, where we can view the size of each time window and the revenue generated by task assignment within this time window. Panel-⑤ is a satellite overview of the tasks, where we can view the geographic location, type of each task and the worker it belongs to. When we select a task in panel-⑤, we can view the service ability and revenue trend of the worker serving this task in panel-⑥. Meanwhile, we can view the task proportion of this worker in panel-⑦.

Although the analysis interface can already meet most of the functional needs of users, the learning-based batch task assignment mode has proved to be the best [4]. To allow users to further analyze the learning process, we implement the decision model interface. As shown in Figure 3(b), the decision model interface contains 3 functional panels. The most representative of learning-based batch task assignment is deep reinforcement learning, so we will take the deep reinforcement learning algorithm as an example to demonstrate. Panel-① is the console, where we can select the deep reinforcement learning algorithm, the network structure parameters of the algorithm and the update speed of the curve in panel-②. Similar to analysis interface, when we operate the console, the contents displayed on panel-② and panel-③ will be updated synchronously. Panel-② displays the trend of loss and Q-value during the learning of deep reinforcement learning algorithm, where we can adjust whether the curves are smooth, the degree of smoothness, and the display mode. AMRAS provides 3 display modes as shown in Figure 4. Panel-③ visualizes the network structure of the

[5]https://cn.aliyun.com/

deep reinforcement learning algorithm, where we can adjust the network parameters, size, margin and color of the drawn network structure. The specific adjustable parameters are shown in Figure 5. Note that when we hover the mouse over the parameter input box, it will pop up the comment of the corresponding parameter.

3.2 Demo Scenario

This section shows three demo scenarios where the users can interact with AMRAS to understand its operation from points of view of both: (a) a user who needs to use AMRAS, and (b) what can AMRAS do for users. We demonstrate these scenarios over New York City (NYC) dataset, which consists of road network data and order data. The road network data is extracted from OpenStreetMap[6] and it contains 264,346 nodes, and 366,923 edges. The order data is collected from 8:00 to 12:00 on 01/01/2016 by NYCTaxi[7] and it contains 20,089 orders, where each order can be represented as a quintuple (index, origin, destination, departure time, and arrive time).

Scenario 1: Assignment Analysis. The users can perform assignment analysis through the two main customizations (time period and task type) of the assignment visualization tool provided by AMRAS, which involves panels-①, ②, ③, ⑤, ⑥ and ⑦ of Figure 3(a). Specifically, the users can dynamically select the time period and task type to be analyzed on panel-①. Here, we preset three groups of task types: All (New, Matched, Completed, Expired I, Expired II), Online (New, Matched, Completed) and Offline (Expired I, Expired II). Note that users can also freely combine different task types for analysis. According to the selected time period and task type, the users can see the proportion and statistics (total number of tasks, total revenue, assignment rate and completion rate) of various tasks in panel-② and panel-③, respectively. In addition, AMRAS provides an intuitive map-based satellite overview in panel-⑤ that allows users to view the spatial distribution of various tasks. When the users click a task icon, the relevant information of the task will pop up, such as latitude and longitude, and the index of the worker serving it. Meanwhile, the users can see the service ability and revenue changes of the specified worker (worker serving the selected task in panel-⑤) in panel-⑥, and the proportion of different types of tasks served by this worker in panel-⑦.

Scenario 2: Time Window Analysis. AMRAS provides a time window visualization tool, which allows users to intuitively see the size and quality of the time window, and analyze the performance of decision model in dividing time windows. As shown in panel-④ in Figure 3(a), the visualization tool consists of a scatter chart (time window size) and a bar chart (assignment revenue within a time window), where the revenue can represent the quality of the time window. Here, the horizontal axis scale of these charts is the same time scale as in panel-①. The chart content of the time period selected by the users is drawn in dark color, and the rest is drawn in light color.

Scenario 3: Decision model Analysis. In this scenario, the users can utilize the decision model visualization tool (see panels-①, ②, ③ of Figure 3(b)) provided by AMRAS to intuitively view the learning quality (loss and Q-value convergence) and network structure of decision model. Specifically, users first need to select a decision model algorithm and a structure level (a decision model algorithm may have multiple structures) on the console of panel-①. Our AMRAS presets six learning-based decision model algorithms of three types, such as value-based (DQN, Dueling DQN), policy-based (PPO, DPPO) and actor-critic (A3C, DDPG). Then, the users can view the loss and Q-value convergence displayed in panel-② and the decision model network structure displayed in panel-③ by clicking the start button in the console. In panel-②, we also design two small tools: curve smoothing and display mode. The users can control whether the curve is smooth and degree of smoothness through the curve smoothing tool. And the users can switch three different curve display styles (i.e., overlapping, separate-vertical, and separate-horizontal) through the display mode tool (see Figure 4). In addition, AMRAS provides curve dynamic refresh function. The users can select the refresh speed in panel-② to view the real-time refresh effect. In panel-③, the users can customize their decision model network structure by adjusting three groups of parameters (see Figure 5).

[6]https://www.openstreetmap.org/
[7]http://www.nyc.gov/

4 CONCLUSION

We have demonstrated AMRAS, a visual analytical system that performs efficient massive assignment data analysis for batch-based task assignment. It is mainly composed of two modules Web-based UI and Computation Engine, which aims to allow users intuitively analyze the assignment data from SC platform and assist them to optimize their assignment algorithm. In addtion, we demonstrate three scenarios on a real-world dataset to illustrate the usability and efficiency of AMRAS.

ACKNOWLEDGMENTS

This work is supported by the grants: NSFC Grants 61972362 and 61602420; CPSF Grant 2018M630836; HNSF Grant 202300410378.

REFERENCES

[1] Dingxiong Deng, Cyrus Shahabi, and Ugur Demiryurek. 2013. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *SIGSPATIAL*. ACM, 314–323.
[2] Zhao-Hong Jia, Le-yang Gao, and Xing-yi Zhang. 2020. A new history-guided multi-objective evolutionary algorithm based on decomposition for batching scheduling. *Expert Syst. Appl.* 141 (2020).
[3] Maocheng Li, Jiachuan Wang, Libin Zheng, Han Wu, Peng Cheng, Lei Chen, and Xuemin Lin. 2021. Privacy-Preserving Batch-based Task Assignment in Spatial Crowdsourcing with Untrusted Server. In *CIKM*. 947–956.
[4] Yafei Li, Qingshun Wu, Xin Huang, Jianliang Xu, Wanru Gao, and Mingliang Xu. 2022. Efficient Adaptive Matching for Real-Time City Express Delivery. *TKDE* (2022), 1–1.
[5] Yexin Li, Yu Zheng, and Qiang Yang. 2020. Cooperative Multi-Agent Reinforcement Learning in Express System. In *CIKM*. 805–814.
[6] Tianshu Song, Yongxin Tong, Libin Wang, Jieying She, Bin Yao, Lei Chen, and Ke Xu. 2017. Trichromatic Online Matching in Real-Time Spatial Crowdsourcing. In *ICDE*. 1009–1020.
[7] Bo Tang, Chuan Yang, Long Xiang, and Jian Zeng. 2018. Deriving Real-time City Crowd Flows by Heterogeneous Big Urban Data. In *IEEE BigData 2018*. 3485–3494.
[8] Yongxin Tong, Jieying She, Bolin Ding, Libin Wang, and Lei Chen. 2016. Online mobile Micro-Task Allocation in spatial crowdsourcing. In *ICDE*. 49–60.
[9] Yongxin Tong, Yuxiang Zeng, Zimu Zhou, Lei Chen, Jieping Ye, and Ke Xu. 2018. A Unified Approach to Route Planning for Shared Mobility. *PVLDB* 11, 11 (2018), 1633–1646.
[10] Yongxin Tong, Zimu Zhou, Yuxiang Zeng, Lei Chen, and Cyrus Shahabi. 2020. Spatial crowdsourcing: a survey. *PVLDB* 29, 1 (2020), 217–250.
[11] Yan Zhao, Kai Zheng, Yang Li, Han Su, Jiajun Liu, and Xiaofang Zhou. 2020. Destination-Aware Task Assignment in Spatial Crowdsourcing: A Worker Decomposition Approach. *TKDE* 32, 12 (2020), 2336–2350.