Turbo-GTS: Scaling Mobile Crowdsourcing using Workload-Balancing Bisection Tree (Demo Paper)

Wei Li
Dept. of Computer Science and Software Engineering
Auburn University
Auburn, AL, USA
wei.li@auburn.edu

Haiquan Chen
Department of Computer Science
California State University, Sacramento
Sacramento, CA, USA
haiquan.chen@csus.edu

Wei-Shinn Ku
Dept. of Computer Science and Software Engineering
Auburn University
Auburn, AL, USA
weishinn@auburn.edu

Xiao Qin
Dept. of Computer Science and Software Engineering
Auburn University
Auburn, AL, USA
xqin@auburn.edu

ABSTRACT

In mobile crowdsourcing, workers are financially motivated to perform self-selected tasks to maximize their revenue. Unfortunately, the existing task scheduling approaches in mobile crowdsourcing fail to scale for massive tasks and large geographic areas. We present Turbo-GTS, a system that assigns tasks to each worker to maximize the total number of the tasks that can be completed for an entire worker group while taking into account various spatial and temporal constraints, such as task execution duration, task expiration time, and worker/task geographic locations. The core of Turbo-GTS is WBT-NNH and WBT-NUD, our two newly developed scheduling algorithms, which build on the algorithms, QT-NNH and QT-NUD, proposed in our prior work [5]. The key idea is that Turbo-GTS performs dynamic workload balancing among all workers using the proposed Workload-balancing Bisection Tree (WBT) in support of large-scale Geo-Task Scheduling (GTS). Turbo-GTS includes an interactive interface for users to load the current task/worker distributions and compare the task assignment of each worker returned by different algorithms in a real-time fashion. Using the Foursquare mobile user check-in data in New York City and Tokyo, we show the superiority of Turbo-GTS over the state of the art in terms of the total number of the tasks that can be accomplished by the entire worker group and the corresponding running time. We also demonstrate the front-end interface of Turbo-GTS with two exploratory use cases in New York City.

CCS CONCEPTS

• **Information systems → Location based services**.

KEYWORDS

Mobile Crowdsourcing, Workload balancing, Quadtree

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGSPATIAL '20, November 3–6, 2020, Seattle, WA, USA
© 2020 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8019-5/20/11.
https://doi.org/10.1145/3397536.3422335

ACM Reference Format:
Wei Li, Haiquan Chen, Wei-Shinn Ku, and Xiao Qin. 2020. Turbo-GTS: Scaling Mobile Crowdsourcing using Workload-Balancing Bisection Tree (Demo Paper). In *28th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '20), November 3–6, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3397536.3422335

1 INTRODUCTION

The ubiquity of mobile platforms and smart phones breeds a large number of mobile crowdsourcing applications where businesses utilize a crowd of people to do jobs which would have been done by their employees, contractors or outsourcing suppliers in the past [3]. A common feature of all these applications is that workers are required to go to the exact spot of each task in person and perform the task by means of a mobile device. A few task scheduling solutions [1] [4] have been proposed to maximize the number of the accomplished tasks for a single worker. On the other hand, BLALS-T [2] was proposed to solve the multi-worker scheduling problem. However, most of the existing approaches suffer from poor performance when coping with massive tasks distributed in large geographic areas. In our prior work [5], we presented two algorithms in support of large-scale multi-worker scheduling, QT-NNH and QT-NUD, which were able to return a higher number of the completed tasks while reducing the computation cost by up to three orders of magnitude, compared to BLALS-T [2].

In this demo, we present Turbo-GTS, a workload-balancing system that extends our prior work [5] in support of large-scale Geo-Task Scheduling (GTS). Specifically, Turbo-GTS optimally assigns tasks to each worker to maximize the total number of the tasks that can be completed for an entire worker group while taking into account various spatial and temporal constraints, such as task execution duration, task expiration time, and worker/task geographic locations. The core of Turbo-GTS is WBT-NNH and WBT-NUD, our two newly developed scheduling algorithms, where the dynamic workload balancing is performed among all workers using the proposed Workload-balancing Bisection Tree (WBT). Also, Turbo-GTS is able to visualize and compare the task assignment of each worker returned by different algorithms in a real-time fashion. Using the Foursquare mobile user check-in data in New York City and Tokyo,

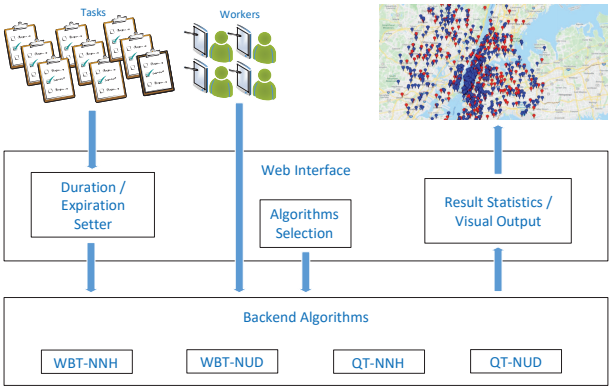


Figure 1: An illustration of the Turbo-GTS framework.

we show the superiority of Turbo-GTS over the state of the art in terms of the total number of the tasks that can be accomplished by the entire worker group and the corresponding running time. The interactive front-end interface of Turbo-GTS can be accessed via a web-based application.

2 TURBO-GTS OVERVIEW

2.1 System Architecture

As depicted in Figure 1, the core of Turbo-GTS is WBT-NNH and WBT-NUD, our two newly developed scheduling algorithms, which build on the algorithms QT-NNH and QT-NUD proposed in our prior work [5]. Turbo-GTS takes various spatio-temporal constraints regarding tasks and workers as input, e.g., the current locations of workers, the locations of tasks, the execution duration of each task, and the expiration time of each task. There are four different scheduling algorithms for users to choose from: WBT-NNH, WBT-NUD, QT-NNH, and QT-NUD. Depending on the selected scheduling algorithm(s), the task execution schedule for each worker will be returned to the front end for visualization. Turbo-GTS is able to visualize and compare the task assignment of each worker returned by different algorithms for comparison purpose in a real-time fashion.

2.2 Datasets

Real datasets were collected from Foursquare mobile user check-ins [10] [9] [8], featuring New York City (42981 tasks in total) and Tokyo (67124 tasks in total). The task execution duration of each task was uniformly sampled from dur_min (5 seconds) and dur_max (5 minutes). The expiration time of each task was randomly generated from the following three ranges starting from the task requested time: 30 to 90 minutes, 60 to 120 minutes, 90 to 150 minutes.

3 SCHEDULING ALGORITHMS

State of the Art: BLALS-T [2]. This method partitions the worker taskflow network from the top down recursively and then merges all the partitions from the bottom up. However, BLALS-T suffers from the following limitations which compromises its performance when coping with large-scale mobile crowdscourcing. First, BLALS-T assumed that each worker does not leave her original location too far away and she can only choose tasks within her neighborhood. Second, BLALS-T assumed that each worker can not be assigned to more tasks than a predefined threshold. Last but not the least, the insertion scheduling method used in BLALS-T was extremely inefficient when the ratio of the number of tasks over the number of workers was high.

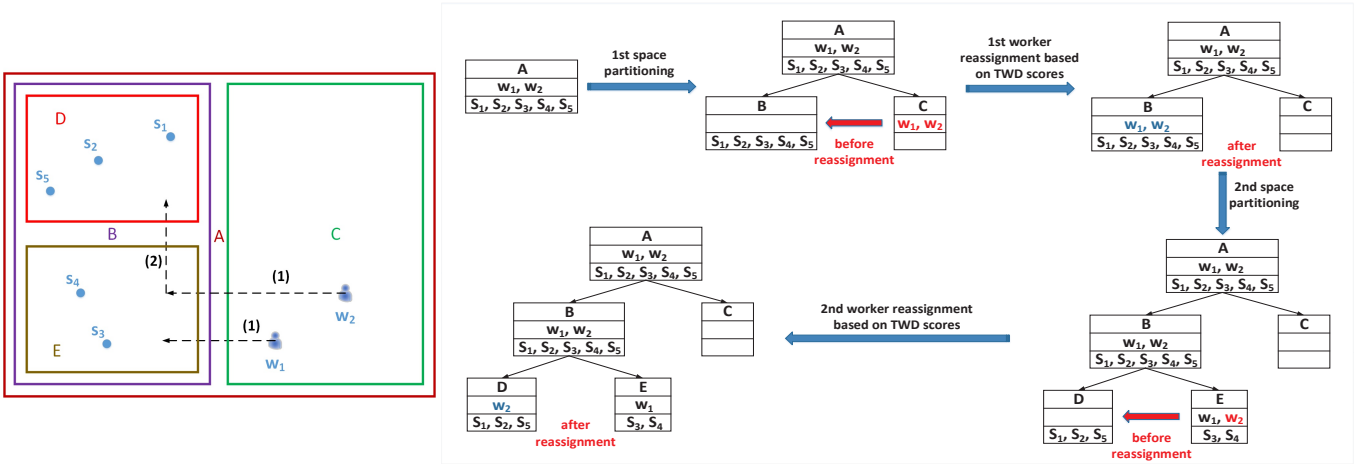
QT-NNH [5]. In our prior work, QT-NNH was proposed to employ Point-Region (PR) quadtree [7] to recursively divide a rectangular region into four equal quadrants for space partitioning among multiple workers. Once the PR quadtree is created, QT-NNH employs NNH [1] to assign tasks to each worker in each leaf node.

QT-NUD [5]. Similar to QT-NNH, QT-NUD employs PR quadtree to perform space partitioning while utilizing NUDPF [4] [5] to assign tasks to each worker in each leaf node.

WBT-NNH. Instead of using the point-region quadtree, in WBT-NNH, we propose to utilize the new tree structure, the Workload-balancing Bisection Tree (WBT), to divide a rectangular region recursively into two equal parts. The workload balancing in WBT is performed based on the calculation of the task-worker-density of each tree node, which can be estimated using Equation 1, where N represents a WBT node, w and s represent a worker and a task in N respectively, e_s and u_s represent the expiration time and execution duration of s respectively, and t_w represents the current time of w. The workload balancing in WBT is performed based on the calculation of the Task-Worker-Density (TWD) of each tree node, which can be estimated using Equation 1. The TWD of each tree node is to estimate the likelihood that the workers currently in that node are able to finish all the tasks currently in the same node, assuming the locations of all the tasks are uniformly distributed. In Equation 1, $RPD(N)$ is the Random Point Distance (RPD) [6], which is the average distance of the two uniformly distributed random points in a rectangular region.

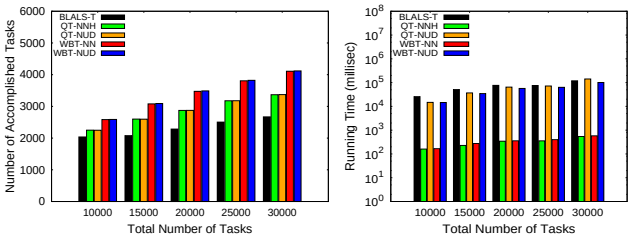
$$task_worker_density(N) = \frac{\sum_{s_i \in N} \frac{1}{u_{s_i}}}{\sum_{w_i \in N} \frac{max\{e_{s_i} | s_i \in N\} - t_{w_i}}{RPD(N)}} \quad (1)$$

In WBT, node splitting is based on the bisection tree while worker reassignment is conducted to ensure all the leaf nodes reach a similar TWD score for achieving load-balancing. Figure 2 illustrates how a spatial region with two workers and five tasks can be partitioned using WBT. As shown in Figure 2 (b), each WBT tree node consists of three parts: the region which the tree node represents, the workers belonging to that region, and the tasks belonging to that region. First the whole region A, as illustrated in Figure 2 (a), is divided into two sub-regions B and C, with all workers located in region C and all tasks located in B. By calculating the task-worker-density scores of B and C, the workers will be reassigned to B from C. Next, the region B is divided into D and E. Similarly, by checking the task-worker-density scores of D and E, w_2 in E is reassigned to D. As a result, the final WBT tree will have three leaf nodes, C, D, and E, with C having no workers or tasks, D having one worker and three tasks, and E having one worker and two tasks. According to the created WBT tree, WBT-NNH then employs NNH [1] to assign tasks to each worker in each leaf node. Note that the workers can still be reassigned to a sibling node or a parent node if needed during the task assignment process once the WBT tree is created, e.g., when a



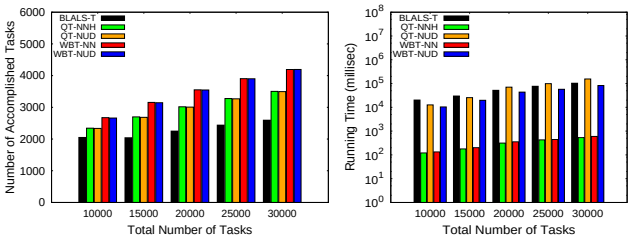
(a) Space partitioning using WBT (b) WBT expansion and worker reassignment

Figure 2: An illustration of the space partitioning using the proposed workload-balancing bisection tree, which results in two occurrences of worker reassignments (highlighted as red arrows) based on the estimated task-worker-density scores for tree nodes.



(a) Number of Accomplished Tasks (b) CPU Cost

Figure 3: Impact of the total number of tasks (New York City)

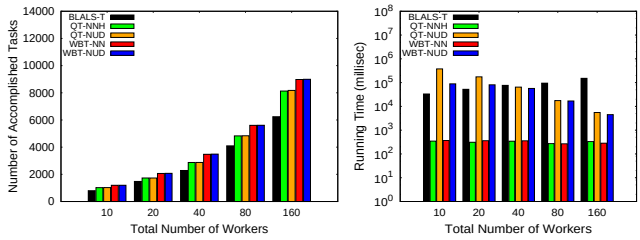


(a) Number of Accomplished Tasks (b) CPU Cost

Figure 4: Impact of the total number of tasks (Tokyo)

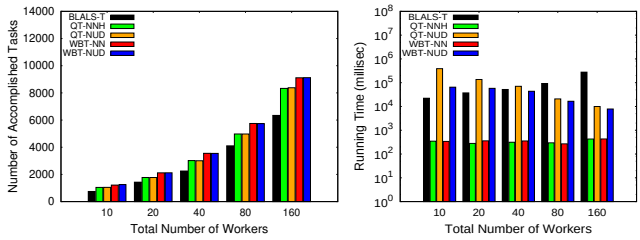
worker finishes all the tasks originally residing in the same region as that worker.

WBT-NUD. Similar to WBT-NNH, WBT-NUD employs the WBT for spatial partitioning and uses the task-worker-density to determine the tree nodes for which the workload needs to be balanced. Instead of using NNH, WBT-NUD builds on NUD-IC [4] [5] for the assignment of tasks to workers within each tree node once the WBT is created.



(a) Number of Accomplished Tasks (b) CPU Cost

Figure 5: Impact of the total no. of workers (New York City)



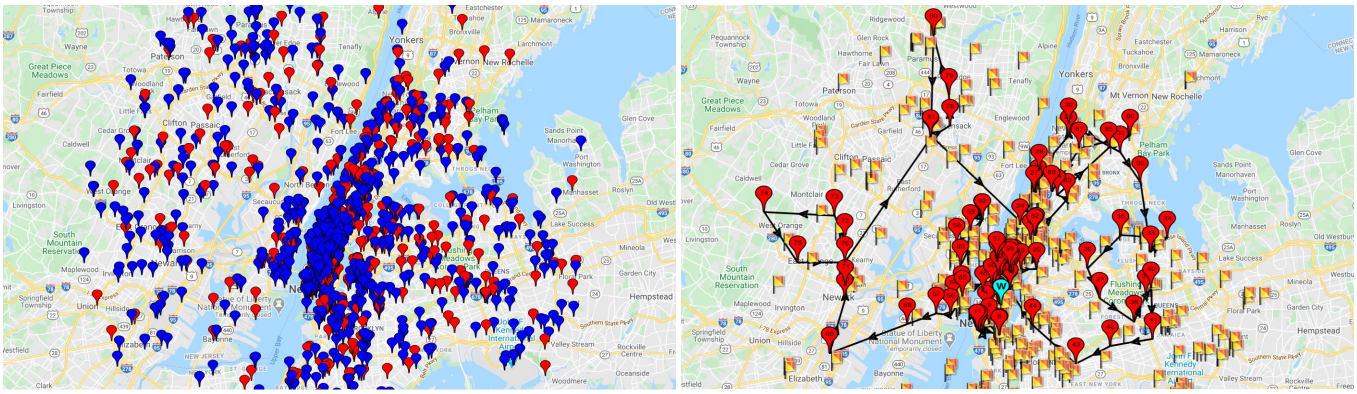
(a) Number of Accomplished Tasks (b) CPU Cost

Figure 6: Impact of the total number of workers (Tokyo)

4 PERFORMANCE

In this section, we compare the performance of all the scheduling algorithms supported in Turbo-GTS.

Impact of the total number of tasks. Figures 3 and 4 show the impact of the total number of tasks on the performance of all the algorithms in terms of the number of tasks accomplished and the running time in New York City and Tokyo respectively. As shown in Figures 3 and 4, WBT-NNH and WBT-NUD consistently outperformed all other methods in terms of the number of tasks accomplished without compromising their efficiency.



(a) Comparing and visualizing different scheduling algorithms

(b) Assigning optimal sequence of tasks to a specific worker

Figure 7: Demonstration of two interactive use cases supported by Turbo-GTS.

Impact of the Total Number of Workers. Figures 5 and 6 show the impact of the total number of workers on the performance of all the algorithms in terms of the number of tasks accomplished and CPU cost in New York City and Tokyo. As depicted in Figures 5 and 6, WBT-NNH and WBT-NUD consistently outperformed all other methods in terms of the number of tasks accomplished without compromising their efficiency.

5 DEMONSTRATION

We built Turbo-GTS using Java on Google Maps API and a back-end MySQL database under PHP framework. Turbo-GTS incorporates WBT-NNH and WBT-NUD, our two newly developed scheduling algorithms, as well as BLALS-T, QT-NNH, and QT-NUD. The project source code is available at https://github.com/WeiTerryLi/Turbo-GTS-Demo.

Turbo-GTS shows task statistics/pattern for users to compare different methods in terms of total task count, task assignment distribution over all workers, and task distribution for any worker or for any spatial region specified by users. Here we demonstrate two essential interactive use cases: (1) comparing task assignments for a worker group returned by different algorithms and (2) identifying the optimal task assignment for a specific worker. Figure 7 shows the task assignments using different scheduling algorithms in Turbo-GTS given 20,000 randomly distributed tasks and 20 workers randomly located in New York City. Specifically, Figure 7(a) compares the tasks that can be completed by the entire worker group when using QT-NNH and WBT-NNH, where QT-NNH returned 1,473 tasks (red dots) while WBT-NNH returned 1,957 tasks (blue dots). Notice that the tasks identified by both QT-NNH and WBT-NNH are omitted in Figure 7. Figure 7(b) shows the task assignment identified by WBT-NNH for a specific worker, where w represents the starting location of the worker, red markers represent the tasks assigned to that worker (with the number in each marker indicating their execution order), and the yellow flags represent tasks not assigned to that worker.

6 CONCLUSION

In this demo, we presented Turbo-GTS - a task assignment system for large-scale mobile crowdsourcing. Turbo-GTS is adroit at assigning tasks to each worker to maximize the total number of tasks to be completed for an entire worker group while taking into account various spatial and temporal constraints (e.g., task execution duration, task expiration time, and worker/task geographic locations). Turbo-GTS performs dynamic workload balancing among all workers using the proposed Workload-balancing Bisection Tree (WBT) in support of large-scale mobile crowdsourcing. Testing Turbo-GTS driven by the Foursquare mobile user check-in data in New York City and Tokyo, we shed bright light on the superiority of Turbo-GTS over the state-of-the-art techniques. We also demonstrate the front-end interface of Turbo-GTS with the two exploratory use cases in New York City.

ACKNOWLEDGMENTS

This research has been funded in part by the National Science Foundation grants IIS-1618669 (III) and ACI-1642133 (CICI).

REFERENCES

[1] D. Deng, C. Shahabi, U. Demiryurek, and L. Zhu. Task selection in spatial crowdsourcing from worker's perspective. *GeoInformatica*, 20(3):529–568, 2016.
[2] D. Deng, C. Shahabi, and L. Zhu. Task matching and scheduling for multiple workers in spatial crowdsourcing. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 21. ACM, 2015.
[3] J. Howe. The Rise of Crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.
[4] W. Li, H. Chen, W.-S. Ku, and X. Qin. Scalable spatiotemporal crowdsourcing for smart cities based on particle filtering. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 63. ACM, 2017.
[5] W. Li, H. Chen, W.-S. Ku, and X. Qin. Turbo-gts: A fast framework of optimizing task throughput for large-scale mobile crowdsourcing. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 6(1):1–29, 2020.
[6] A. Mathai, P. Moschopoulos, and G. Pederzoli. Random points associated with rectangles. *Rendiconti del Circolo Matematico di Palermo*, 48(1):163–190, 1999.
[7] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, 16(2):187–260, 1984.
[8] D. Yang, D. Zhang, L. Chen, and B. Qu. NationTelescope: Monitoring and Visualizing Large-scale Collective Behavior in LBSNs. *J. Network and Computer Applications*, 55:170–180, 2015.
[9] D. Yang, D. Zhang, and B. Qu. Participatory Cultural Mapping Based on Collective Behavior Data in Location-Based Social Networks. *ACM TIST*, 7(3):30:1–30:23, 2016.
[10] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu. Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs. *IEEE Trans. Systems, Man, and Cybernetics: Systems*, 45(1):129–142, 2015.