One-to-One and One-to-Multiple User-Task Mapping in Task-Abundant Mobile Crowd Sensing

Yijun Liu, Ting Li

Oxford College of Emory University, Department of Computer Science

Introduction

Mobile crowd sensing (MCS) is a technique that relies on sensors in mobile devices that are commonly used among people, such as cameras and GPS in smartphones and wearables.[2] This paradigm uses humans who equip those devices to sense the world and act as sensors.[2]

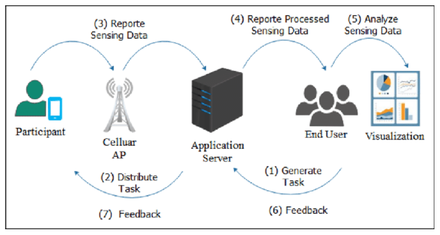

Fig. 1 Mobile Crowd Sensing[1]

With a large number of users and tasks, one of the fundamental problems is how to allocate these tasks to each user with the consideration of distance traveled by all users, maximum distance traveled by each user, the number of total users recruited, and time sensitivity for each task. Specifically, this problem is less focused in situations where the number of tasks is greater than the number of users from the total user pool. Thus, this study investigates the task allocation in task-abundant MCS. In particular, there are two scenarios studied: one-to-one mapping (each user can only fulfill one task) and one-to multiple mapping (each user can fulfill multiple tasks) between users and tasks.

Methods

STEP 1: User Data Cleaning - R

The data used in this research comes from the D4D dataset.[4] It contains 50,000 users' phone call records and cell phone tower locations. Each users' phone call recorded includes indicators to indicate whether it was an even or odd week. The data cleaning process includes 1. merging the data from biweekly to weekly and removing duplicates as needed; 2. deriving each user's available locations and schedules from their shown-up traces.

STEP 2: Task Data Generation - JAVA

A set of tasks with time and location is randomly generated, given the maximum and minimum boundaries for longitudes and altitudes. The boundaries are determined by users' traces from the D4D dataset.

STEP 3: Connection Boundary Determination - R

Connection boundary is the maximum allowed traveling distance between each user and task. The traveled distance distribution between 10,000 users and 10,000 tasks has been considered to set up the connection boundary, which is 1st quartile (6.06348km).

STEP 4: Algorithms - JAVA

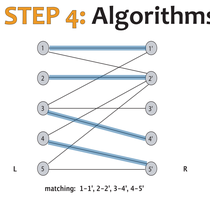

Fig. 2 Bipartite Matching Repeated Augmenting Path[5]

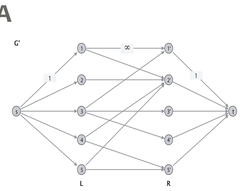

Fig. 3 Bipartite Matching Maximum Flow[5]

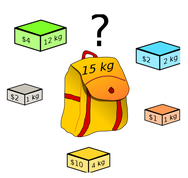

Fig. 4 Greedy Algorithm[3]

This research uses the three algorithms demonstrated in the figures above. Bipartite matching represents the one-to-one mapping between users and tasks. The repeated augmenting path includes using the Depth-First Search algorithm repeatedly until no augmenting path is found (Berge's Theorem)[5]. Maximum flow adopts the idea of adding virtual source and sink nodes; the edge between users and tasks is set to 1 for all, and the Ford-Fulkerson algorithm is used[5]. Greedy algorithms are used in the one-to-multiple mapping. The two goals in the greedy algorithms are selecting the minimum number of users and the minimum distance traveled.

STEP 5 & 6: Simulation - JAVA; Data Analysis

See results & conclusion/discussion.

Results

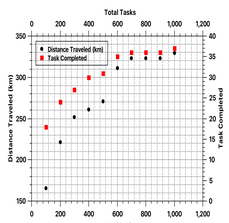

Fig. 5 Bipartite Matching & Bipartite Matching to Max Flow with Fixed Number of User at 80

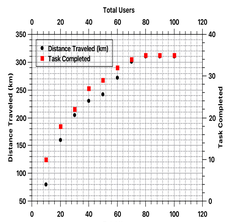

Fig. 6 Bipartite Matching & Bipartite Matching to Max Flow with Fixed Number of Task at 600

The two figures to the left demonstrate the bipartite matching algorithms for varying in numbers of users with fixed total tasks or varying in the number of tasks for fixed total users. Both bipartite matching algorithms (finding the augmenting path and maximum flow) yield the same result. Additionally, both the task completed and distance traveled increase as the total task/user increases, but the slope decreases.

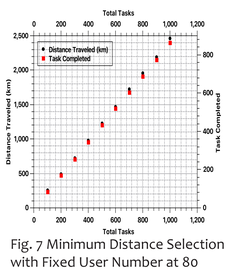

Fig. 7 Minimum Distance Selection with Fixed User Number at 80

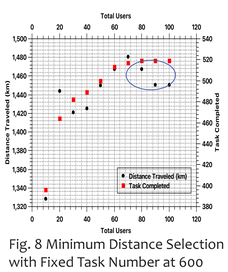

Fig. 8 Minimum Distance Selection with Fixed Task Number at 600

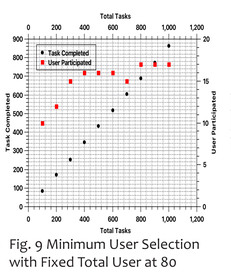

Fig. 9 Minimum User Selection with Fixed Total User at 80

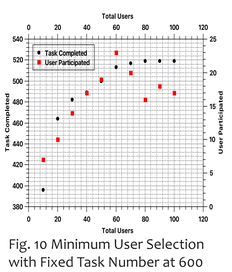

Fig. 10 Minimum User Selection with Fixed Task Number at 600

The four figures above demonstrate the greedy algorithms for varying in numbers of users with fixed total tasks or varying in the number of tasks for fixed total users. They present the changes in task completion and distance traveled/user participation for the minimum distance/minimum user selection algorithm. For figure 5, both the distance and task completeness increase as the total task increase; for figure 6, the distance traveled fluctuates. The distance for completing tasks decreases when there is a larger user pool. For figure 7, the user participation fluctuates because of the uncertainty of the tasks; and for figure 8, the user participation fluctuates because there are more users that can be selected from a larger user pool.

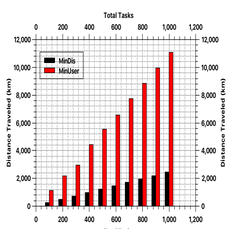

Fig. 11 Distance Traveled: Greedy Comparison with Fixed User Number at 80

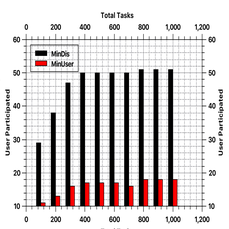

Fig. 12 User Participated: Greedy Comparison with Fixed User Number at 80

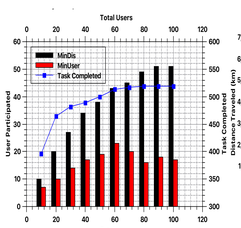

Fig. 13 User Participated: Greedy Comparison with Fixed Task Number at 600

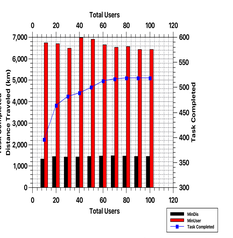

Fig. 14 Distance Traveled: Greedy Comparison with Fixed Task Number at 600

The four figures above compare the greedy algorithms for either the total user number or the total task number is fixed. They present that both greedy algorithms fulfill their purposes in having advantages in different criteria (distance traveled or user participation) over another.

The two figures to the right compare the bipartite matching algorithms to the greedy algorithms for both the percentage of task completeness and distance traveled by each user under the two scenarios when varying in numbers of users with fixed total tasks or varying in the number of tasks for fixed total users. The bipartite matching shows a much smaller task completeness percentage and distance traveled per user because each user can only be matched only once. Users have different abilities in those two settings.

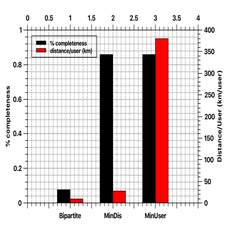

Fig. 15 Greedy and Bipartite Comparison with Fixed Number of User at 80

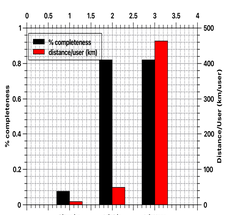

Fig. 16 Greedy and Bipartite Comparison with Fixed Task Number at 600

Conclusion & Discussion

In the one-to-one mapping scenario, bipartite matching presents lower task completeness rates (7.59%) in both user-fixed and task-fixed situations, while in one-to-multiple mapping, the rates yielded by the greedy algorithms are 85.8% and 82.0% for user-fixed and task-fixed algorithms respectively. Even though the greedy algorithms outperform the bipartite matching in task completeness, bipartite matching has the least total and per user distance traveled values.

In the one-to-one mapping scenario, although the task completeness for both greedy algorithms remains the same, there is a trade-off between user participation and distance traveled. With more users, more tasks can be completed with less distance. In the user-fixed situation, the average distance traveled to complete each task in MinUser (12.75 km/user) is 4.45 times more than MinDis (2.87 km/user); the average number of tasks each user completed in MinDis (9.64 tasks) is 3.09 times more than MinUser (29.77 tasks). In the task-fixed situation, the average distance traveled to complete each task in MinUser (13.60 km/user) is 4.63 times more than MinDis (2.94 km/user); the average number of tasks each user completed in MinDis (10.18 tasks) is 2.05 times more than MinUser (33.24 tasks). In short, the MinDis algorithm seems to have less loss than MinUser.

This study only demonstrates a basic comparison between the greedy algorithms and the bipartite matching algorithms. It fails to consider situations, where have a larger data size and the results are not significant in situations where there are more users than tasks. Additionally, there might be more complicated MCS tasks that include variables more than distance and time (e.g. sensors users have). Furthermore, this study does not take heterogenous time sensitivities for tasks into consideration. In the future, it is possible to test out more variables and mock more scenarios including the ones mentioned above.

Acknowledgement

Thanks to **Ting Li,** my mentor, for instructing and helping me throughout the program; thanks to **Summer Oxford Research Scholars Program** for providing this opportunity to me and its funding.

Bibliography

[1] Bayan Hashr Alamri & Muhammad Mostafa Monowar & Suhair Alshehri, 2018. "A privacy-preserving collaborative reputation system for mobile crowdsensing," International Journal of Distributed Sensor Networks, , vol. 14(9), pages 15501477188, September.

[2] Crowdsourcing to Smartphones: Incentive Mechanism Design for Mobile Phone Sensing, Dejun Yang, Guoliang Xue, Xi Fang, and Jian Tang, ACM International Conference on Mobile Computing and Networking (MobiCom), 2012.

[3]"Greedy Algorithms." Brilliant Math & Science Wiki, 2021, brilliant.org/wiki/greedy-algorithm/.

[4] V.D. Blondel, M. Esch, C. Chan, F. Clerot, P. Deville, E. Huens, F. Morlot, Z. Smoreda, and C. Ziemlicki. Data for development: the d4d challenge on mobile phone data. 2012.

[5] Wayne, K. (2013). 7. NETWORK FLOW II [Slides]. Princeton. https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/07NetworkFlowI.pdf