Fall 12-15-2020

Optimal Order Assignment with Minimum Wage Consideration (OOAMWC)

HAKEM ALAZMI

Optimal Order Assignment with Minimum Wage Consideration (OOAMWC)

A Thesis Presented to

The Faculty of the Computer Science Department

by

Hakem Alazmi

In Partial Fulfillment

of Requirements for the Degree

Master of Science, Computer Science

Kennesaw State University

December 2020

Optimal Order Assignment with Minimum Wage Consideration (OOAMWC)

Approved:

Dr. Ahyoung Lee – Advisor

Dr. Coskun Cetinkaya – Department Chair

Dr. Jeffrey Chastine – Dean

In presenting this thesis as a partial fulfillment of the requirements for an advanced degree from Kennesaw State University, I agree that the university library shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from, or to publish, this thesis may be granted by the professor under whose direction it was written, or, in his absence, by the dean of the appropriate school when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from or publication of, this thesis which involves potential financial gain will not be allowed without written permission.

_____

Hakem Alazmi

Notice To Borrowers

Unpublished theses deposited in the Library of Kennesaw State University must be used only in accordance with the stipulations prescribed by the author in the preceding statement.

The author of this thesis is:

Hakem Alazmi

526 E Lake Dr, Apt 526

Marietta, GA, 30062

The director of this thesis is:

Dr. Ahyoung Lee

680 Arntson Drive, J 306

Marietta, GA, 30060

Users of this thesis not regularly enrolled as students at Kennesaw State University are required to attest acceptance of the preceding stipulations by signing below. Libraries borrowing this thesis for the use of their patrons are required to see that each user records here the information requested.

Optimal Order Assignment with Minimum Wage Consideration (OOAMWC)

An Abstract of

A Thesis Presented to

The Faculty of the Computer Science Department

by

Hakem Alazmi

Previous degree (Bachelor of Science), Kennesaw State University, 2018

In Partial Fulfillment

of Requirements for the Degree

Master of Science, Computer Science

Kennesaw State University

December 2020

Abstract

While the application of crowdsourcing has increased over the years, the technology experiences various issues during implementation. Examples of some of the issues that affect crowdsourcing include task assignment, profit maximizations, as well as time window issues. In some instances addressing some of the issues results in the other issues being overlooked. An example is when assigning tasks to workers, the profits of the workers might not be considered and this ends up affecting the profit maximization aspect. Various algorithms have been proposed to address the task assignment, profit maximizations, and time window issues. However, these algorithms address the issues individually and this results in the occurrence of the other noted issues. Therefore, this calls for the definition of a solution to address the task assignment issue while taking into consideration the time window issue and the minimum wage constraint. Additionally, the solution should address the profit maximization of not only the workers but also the platform and the clients of the platform. To evaluate the efficiency of the proposed solution, a comparison with the different implemented solutions to address individual issues is recommended. Comparing such solutions can provide insight into the efficiency of the proposed approach when addressing multiple issues affecting crowdsourcing.

Optimal Order Assignment with Minimum Wage Consideration (OOAMWC)

A Thesis Presented to

The Faculty of the Computer Science Department

by

Hakem Alazmi

In Partial Fulfillment

of Requirements for the Degree

Master of Science, Computer Science

Advisor: Dr. Ahyoung Lee

Kennesaw State University

December 2020

Acknowledgment

It is a great chance to acknowledge everyone who played a significant role in this academic accomplishment. Firstly, my committee members, Dr. Ah young Lee, Dr. Donghyun Kim, and Dr. Yan Huang, each of whom has provided valuable advice and guidance during my thesis process. Second of all, my family, who supported me all the time. Finally, my words are powerless to express my gratitude for your efforts. Thank you so much for your consideration and cooperation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

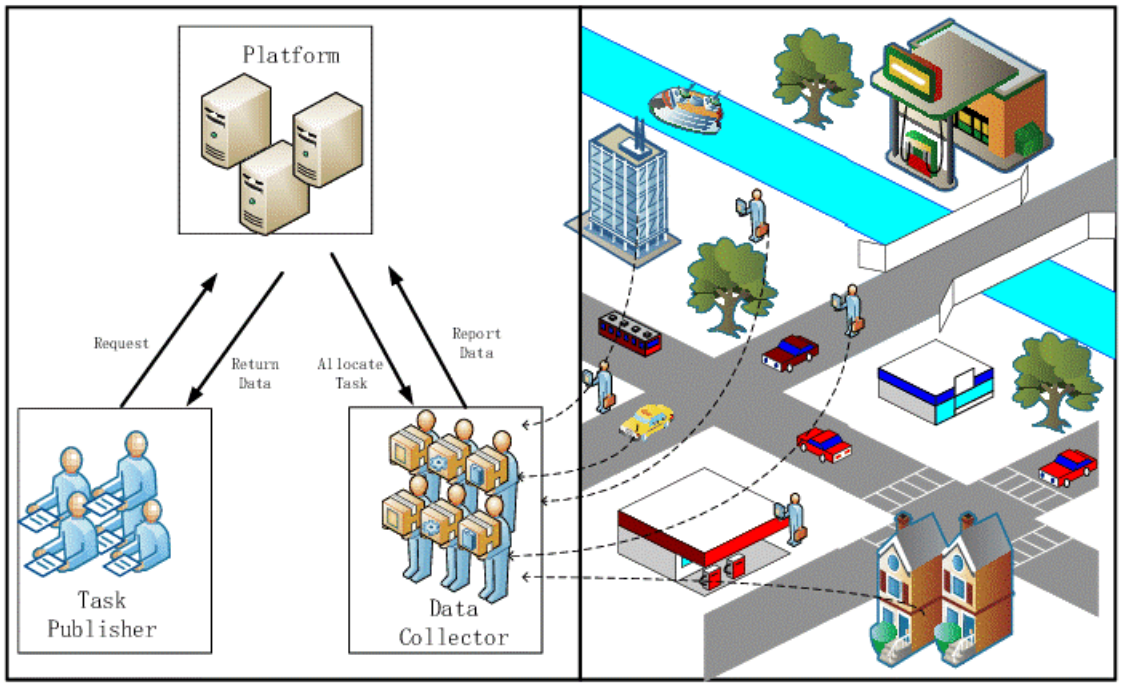

Figure 1.1: Crowdsensing Concept

Crowdsourcing, which can also be referred to as mobile crowdsensing, involves the use of mobile devices including smartphones, wearables, and even tablets to collect and share information concerning specific incidents. Mobile crowdsensing takes advantage of the current developments in mobile devices that include an increase in the sensing capabilities

of the mobile devices as well as the processing power and communication capabilities of the devices (Boubiche, Imran, Maqsood, & Shoaib, 2019). Crowdsourcing involves the recruitment of a group of individuals with mobile devices that have various sensing capabilities to perform the required sensing assignments at the specified locations. Taking into consideration the mobile application Uber, the aspect of crowdsourcing is noted in the collection of data from the workers, which is then used to perform tasks such as pairing clients with the available drivers. An example of this is the use of the Global Position System to collect information concerning the routes taken by the Uber drivers after which the most efficient route for the drivers is determined. The implementation of crowdsourcing experiences various challenges concerning the assignment of tasks, profit maximization, and even time window. Additionally, when addressing the different issues associated with crowdsourcing, the aspect of a minimum wage has to be considered. The aspect of minimum wage allows for the profit maximization of the platform as well as ensuring that the profit of each worker is maximized while taking into consideration the profit maximization of the platform.

On the issue of task assignment Li, Zhu, and Cui (2019) note that the task assignment problem arises when workers are assigned tasks based on their proximity with the tasks without taking into consideration aspects such as the possible changes in the locations of the workers or even the tasks. Considering the example of the mobile application Uber, assignment of tasks based on proximity prevents other drivers from being allocated tasks and this prevents both the platform and the workers from profiting from the aspect of a minimum wage. Some of the algorithms used to address the issue of task assignments include participatory-sensing-based mechanisms as well as opportunistic-sensing-based mechanisms (Gong, Zhang, & Li, 2018b). Considering the mobile application Uber, the participatory sensing mechanism involves the workers moving to the appropriate locations where the tasks are required while in the opportunistic sensing mechanism the requirements for the workers to change their trajectory behavior is not required, hence, they

can perform the tasks when they get to an area where the tasks are available (Gong, Zhang, & Li, 2018). The application of crowdsourcing in Uber can involve the determination of the various aspects that contribute to profit maximization. In the case of Uber, the issue of profit maximization involves the maximization of the profits for not only the drivers but also the Uber platform as well as the clients of the platform. This aspect of achieving optimal profit is prevented by various aspects such as the complicated allocation of requests from mobile devices (Han & Zhu, 2014). Profit maximization for both workers and the platform can be prevented by the lack of definition of the minimum wage. The definition of the minimum wage for the workers allows each worker to contribute to the platform's profits and in the process allow each worker to earn from the platform. An example of this is noted in a package delivery application, where while one worker is capable of delivering all the packages, this would maximize the profits of the platform, but not for the other workers on the platform. Hence, the requirement for the definition of a minimum wage that would allow other workers to deliver the packages.

Similar to the occurrence of task assignment issues, and profit maximization issues, crowdsourcing can also experience issues concerning time window management. According to Chen, Cheng, Zeng, and Chen (2019), that aspect of crowdsourcing applications such as Uber focusing on the task assignment issue result in the issue of task assignment delay occurring and this is an example of the issues associated with the time window issue. An illustration of the time window issue is noted in the delay that an Uber client has to wait until the Uber platform assigns a worker to pick up the client.

The concept of crowdsourcing involves the collection of data that can be used to perform different tasks. The processes involved in crowdsourcing can experience various challenges when assigning tasks, when maximizing the profits, and when managing time. Various algorithms such as the prediction based model, profit optimal online control algorithm, and the nearest neighbor heuristic algorithm can be used to address the issues involved in the various activities of crowdsensing. Tao and Song (2020) address the issue presented by

task assignment, as well as the profit maximization of the crowdsourcing platforms. However, Tao and Song (2020) fail to address the profit maximization of the workers as well as that of the clients. Similarly, Sun et al. (2019) address the issues of task allocation, and the profit maximization of the workers. However, the issue of profit maximization for the platform and the clients is not addressed.

Chapter 2

Literature Review

This chapter evaluates past literature of the used approaches to address the aforementioned issues in the crowdsourcing:

2.1 Task Assignment

A problem noted in the crowdsourcing technology that can reduce the efficiency of the activities performed during crowdsourcing includes the allocation of tasks to the workers. The aspect of task assignment in crowdsourcing involves the identification of the appropriate workers that can perform the available tasks in the crowdsourcing platform. During task assignment one of the aspects considered involves the minimum wage. The minimum wage involves a defined cost that would ensure that both the platform and the workers achieve maximum profits from the available tasks in the crowdsourcing platform. Similarly, other issues such as the maximization of profit and even the time window issue have to be considered when addressing the task allocation aspect of crowdsourcing. According to Tao and Song (2020), the issue of task allocation in mobile crowdsensing can be defined as being NP-hard. For an NP-hard problem, the implementation of a greedy algorithm such as the algorithm proposed by Wang, Wang, Wang, Zhang, & Kong (2018) can be used to address

the problem. Tao and Song (2020), on the other hand, note the presence of different constraints and propose the use of a double deep Q-network (DDQN). The application of deep reinforcement learning through DDQN results in better task allocation when compared to other algorithms such as greedy and ACS-based solutions (Tao & Song, 2020).

A shortcoming in the task allocation aspect of traditional algorithms used in mobile crowdsensing involved the algorithms not taking into consideration the changes in the location of the worker as well as location change for the tasks (Li, Zhu, & Cui, 2019). One of the approaches proposed to address the issue of task allocation includes the use of a prediction-based approach. In the prediction-based model proposed by Li, Zhu, and Cui (2019) the semi-Markov model is applied. This involves initially estimating the location of both workers and tasks before taking into consideration the time constraint. After addressing the time constraint, the appropriate tasks are then allocated to the the available workers and using of the semi-Markov model also results in the lowest traveling costs for the workers being considered (Li, Zhu, & Cui, 2019). Similarly, (Hu, Wang, Wu, & Helal, 2020) take into consideration the possible changes in the location of the workers, which can impact the task allocation aspect in mobile crowdsensing. The proposed novel framework by Hu, Wang, Wu, and Helal (2020) addresses task allocation issues through reinforcement learning while taking into consideration the noted challenges such as the location change of the worker.

Other algorithms that are used to ensure that the allocation of tasks is optimal include greedy algorithms and the divide-and-conquer algorithms. According to (J. Wang, Wang, Wang, Zhang, & Kong, 2018), the greedy algorithm involves the selection of the best choice until a stopping standard is reached. An example of a stopping standard is when the location of the Uber driver is not within the location of the tasks. Similarly, taking into consideration the profit maximization of the platform, the greedy algorithm can match workers with the tasks that are likely to result in the highest profits for the platform while ensuring that the worker will be paid the lowest price (Tao & Song, 2020). Another issue encoun-

tered when performing task assignment activities in mobile crowdsensing includes the time delay during the assignment process. To address the delay issue during the task allocation Chen, Cheng, Zeng, and Chen (2019) propose the threshold-based greedy algorithm, which matches a worker with the least travel time to a task, as well as the batch-based algorithm that buffers workers and tasks before matching workers to tasks from the previous batch. While addressing the aspect of time in task allocation, Cheng, Lian, Chen, and Shahabi (2017), note that some of the task allocation approaches do not address the aspect of future tasks and workers. To address this issue, the maximum quality task allocation (MQA) is proposed. The MQA involves two algorithms, which include the greedy algorithm as well as the algorithm that applies the divide-and-conquer approach. The divide-and-conquer algorithm involves the division of the issues into units that can be solved (Cheng, Lian, Chen, & Shahabi, 2017).

In some applications of mobile crowdsensing such as package delivery, the completion of tasks by a single worker is possible but not the most efficient method that can result in the profit maximization of the platform. Additionally, due to the requirement of a minimum wage, the allocation of all the activities to a single worker prevents other workers from profiting from the available tasks. This issue that exists due to the requirement for multiple workers to work collaboratively to perform different tasks in the crowdsourcing platform is addressed by Song et al. (2020). Apart from addressing the issue of allocating tasks to multiple workers in the platform, (Song, Xu, Li, Li, & Tong, 2020) note the aspect of task allocation to workers in real-time as both the workers and the tasks become available in the crowdsourcing platform.

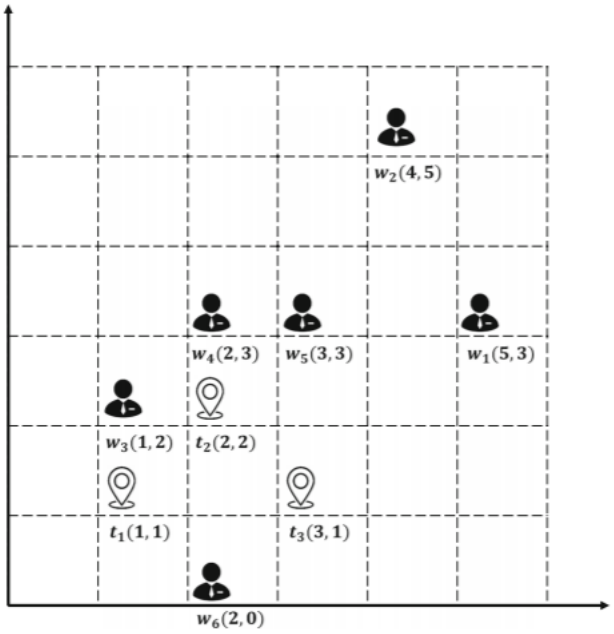

Figure 2.1: Locations of Workers and Tasks

To address the issue presented by the offline task allocation solutions, Song et al. (2020) provide an example involving different workers and different tasks. As indicated in the provided example, both the the available tasks and the available workers arrive and leave at different durations. During this duration, the task allocation process is noted to require updates as the workers and tasks arrive and leave. This addresses the issue of offline task allocation solutions that are static and do not take into consideration the aspect of tasks and workers being available during different durations. Since waiting for all the tasks and workers to be available on the crowdsourcing platform is not possible, then the task allocation process has to be conducted based on the available data concerning the workers and the tasks as they arrive on the crowdsourcing platform.

Furthermore: to address the issue of task allocation, Song et al. (2020) propose the use of the Online-Exact algorithm and the Online-Greedy algorithm and proceed to analyze the execution of both algorithms. In Online-Exact algorithm, the identification of the optimal allocation for the workers as well as the tasks that are available on the crowdsourcing plat-

form. The Online-Greedy algorithm considers the occupation of workers and the profits obtained from performing a task when determining the appropriate allocation for the workers and the tasks. One of the essential issues addressed by Song et al. (2020) is the issue of the proposed solutions not being practical in real-world applications. This issue is seen in the Online-Exact algorithm proposed by Song et al. (2020).

Another issue that is likely to arise in mobile crowdsensing involving the task allocation process is that the worker might choose not to accept to perform the allocated tasks. Most task allocation solutions use various algorithms to determine the most appropriate worker to work on the specific tasks while taking into consideration the minimum wage of the workers and the profit maximization of both workers and the platforms. While these solutions aim at ensuring that the most appropriate worker is assigned to a specific task, Hassan and Curry (2014) note that the worker might choose not to accept the task. An example of this is seen when some package delivery drivers opt not to accept the package delivery tasks for specific locations. To address this issue, Hassan and Curry (2014) propose the implementation of the Individualized Models for Intelligent Routing of Tasks (IMIRT) that aims at ensuring that the amount of the tasks that are assigned to the workers are maximized successfully. This involves identifying the appropriate workers that are more plausible to work on the allocated tasks. The proposed IMIRT framework, after the arrival of the tasks, information concerning the worker is used to identify the acceptance rate of the tasks by the workers and determines the worker that is likely to result in the success of the task allocation (Hassan & Curry, 2014).

2.2 Profit Maximization

When addressing the profit maximization of crowdsourcing, two of the parties to consider include the platform and the workers. This means that the allocation of the tasks

should ensure that the allocation of the tasks results in maximum profits for the platform as well as for the workers. Some of the algorithms used to address the issue of profit maximization include the random online control algorithm and the greedy online control algorithm as well as the profit optimal online control algorithm (POC), which uses the Lyapunov optimization technique, proposed by Han and Zhu (2014). In the random online control algorithm, the control decisions are randomized while the dispatching control makes use of the dispatch to the shortest queue strategy. The greedy online control algorithm also makes use of the dispatch to the shortest queue strategy while at the same time employing greedy control decisions. The proposed profit optimal online control algorithm makes independent control decisions and this results in better profit maximization when contrasted with the greedy online control algorithm as well as the random online control algorithm (Han & Zhu, 2014).

One of the ways to ensure the platform gains maximum profits includes the reduction of the storage costs paid by the mobile crowdsensing platform. To reduce the storage costs, Wang, Luan, Yang, and Wu (2019) propose the implementation of the edge-based mobile crowdsensing architecture that involves the execution of another layer between the cloud server and the diffeerent users. In traditional crowdsourcing architecture, the cloud server performs various data processing and storage activities. The introduction of another layer between the users and the cloud server means that the new layer would be responsible for addressing the data storage and processing (E. Wang, Luan, Yang, & Wu, 2019). In some applications of crowdsourcing, the workers are responsible for choosing the tasks they would like to perform. In some instances when the workers choose the tasks they would like to perform, they end up choosing tasks that do not maximize their profits. To address this shortcoming, Deng, Shahabi, and Demiryurek (2013) propose an approach that ensures that the worker achieves maximum profits from the tasks they choose. Some of the approaches included in the proposed approach include the branch-and-bound algorithm and the least expiration time heuristic (LEH) (Deng, Shahabi, & Demiryurek, 2013).

Taking into consideration the issue of profit maximization the application of the ACS-based solution can be applied to ensure maximum profit for the platform (Tao & Song, 2020). On the aspect of the profit maximization for the platform, Han and Zhu (2014) note the various challenges that face the profit maximization for the platform. These challenges involve the arrival of the tasks and the dynamic participation of the workers. Unlike the prediction-based approach that addresses future workers and tasks, the proposed online control framework based on stochastic Lyapunov optimization does not require any knowledge of future patterns to ensure maximum profits for the platform (Han & Zhu, 2014). According to (Silberman et al., 2018), the rewards offered to the participants that perform the crowdsourcing activities are the main motivator for the participants. Therefore, Silberman (2018) proposes that the different crowdsourcing platforms should pay the workers minimum incomes. In some cases, the workers who perform crowdsourcing activities rely on the tasks as the main source of income. Therefore, Silberman (2019) notes that they should be compensated similarly to other workers in different professions.

To get users that are willing to participate in the crowdsourcing activities, a platform has to take various aspects into consideration. One of these aspects involves the rewards that are offered to the workers who agree to participate in the process. An example of this is Uber drivers or package delivery drivers are more willing to participate in the crowdsourcing activities when there are rewards to be obtained following their participation. This aspect of workers requiring payment to participate in the crowdsourcing activities is noted by (Zhan, Xia, Zhang, & Wang, 2017) who state that the use of credits can be essential in motivating workers to perform the crowdsourcing activities. The aspect of paying the workers who agree to participate in the required activities is echoed by (W. Liu, Yang, Wang, & Wu, 2020). In their article Liu et al. (2020) introduce the aspect of the recruitment budget that limits the rewards that can be offered to those that participate in the crowdsourcing activities. To ensure that the platform achieves maximum profits, the workers that are recruited for the noted activities are required to provide maximum contribution while at the same

cost the least amount to the platform (Liu et al., 2020). An aspect to note while choosing the workers that cost the least to the platform while providing maximum contribution is the aspect of time delay and the issue of recruitment budget. Li et al. (2020) note that when addressing the issue of budget constraints, some of the proposed solutions choose to address the issue by dividing the budget into different stages and using the assigned sub-budgets to recruit the workers. This presents the issue of not addressing time constraints while addressing budget constraints. To address the noted issues, Li et al. (2020) propose a solution that takes into consideration both the recruitment budget and the time constraints. This is achieved by using historical data to identify the potential number of participating workers then identifying the workers that contributed the most to the profits of the platform while remaining under the budget constraint that is defined. This is conducted offline and the online process takes into consideration the worker information obtained from the offline process to conduct the recruitment process (Liu et al., 2020).

Furthermore, the issue of profit maximization in mobile crowdsensing can include achieving the maximum profits for the platform or even the users. Some of the noted solutions that are proposed to address the issues of profit maximization in a crowdsourcing platform make use of the historical data that is analyzed when offline to identify the best practices to implement. To address the issue of profit maximization, (H. Liu et al., 2013) propose the utilization of the 3G budget that is used during the crowdsourcing activities. This utilization of the budget is conducted through the implementation of an online rather than an offline solution that takes into consideration the historical data of the participants. While the offline solution is used by some of the proposed solutions to address the issue of profit maximization, other proposed solutions adopt a prediction-based approach to address the issue. An example of this is noted by Han and Zhu (2014) who state that the applicability of the prediction-based approaches depends on the accuracy of the proposed prediction-based algorithm.

2.3 Time Window

In the crowdsourcing, most of the proposed algorithms to allocate the different available tasks to the available workers only take into consideration the workers and tasks that are available within the given duration. An example of this is seen in the mobile application Uber, where different tasks are required at different durations and workers are available at different durations. Figure 2.2 illustrates this example:

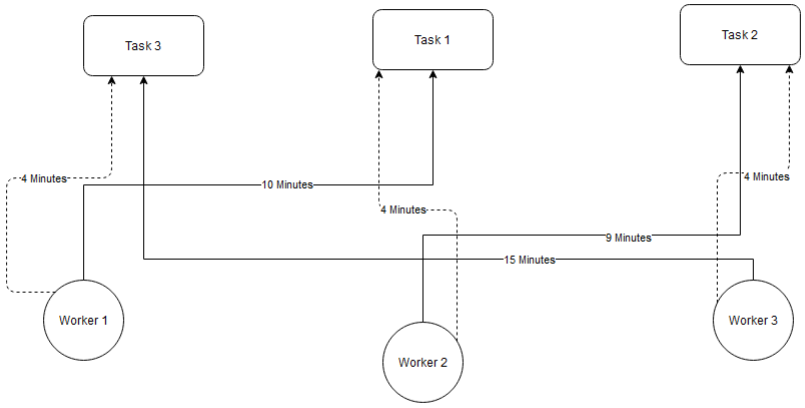

Figure 2.2: Time Window Example

In the provided example, task 1 is available at the same time as worker 1, task 2 is available at the same time as worker 2, and task 3 is available at the same time as worker 3. Therefore, the task allocation process includes assigning the first task to the first worker, the second task to the second worker, and the third task to the third worker. The allocation of tasks to the workers in the provided scenario results in different time durations for the workers. This causes a time delay issue. As noted in the scenario, allocation of the first task to the second worker, the second task to the third worker, and the third task to the first worker reduce the respective durations to only four minutes for each worker. Taking into

consideration the mobile application Uber, then the reduction in the duration between the workers and the tasks results in the reduction in the duration that the clients have to wait for the arrival of the Uber driver. One of the issues associated with the allocation process noted above is the availability of workers and tasks. As noted, the workers and the tasks are available at different duration, and this results in the available tasks being allocated to the available workers. According to Cheng, Lian, Chen, and Shahabi (2017), the traditional spatial crowdsourcing approaches only take into consideration the available tasks and workers at a given instance and when other workers and tasks are available, the result is increased waiting duration as well as the traveling costs. To address this time window shortcoming, Cheng, Lian, Chen, and Shahabi (2017) propose a Maximum Quality Task Assignment (MQA) solution that takes into consideration the movement of the workers to allocate tasks to workers across different time instances. In the proposed MQA solution, the aspect of workers moving around as well as the availability of activities that need to be performed before a specific duration expires are considered. The proposed MQA solution includes the MQA greedy algorithm and the MQA divide and conquer algorithm. In MQA greedy algorithm, both the current and the future tasks and workers are taken into consideration to create an efficient worker and task pair that addresses the time delay issue. In the MQA divide and conquer algorithm, the problem presented by the movement of workers and a difference in the availability of workers and tasks is divided into sections. These sections are solved separately and the results from the combined sections to provide a recommendation for the whole issue (Cheng, Lian, Chen, and Shahabi, 2017).

On the aspect of time delay, Chen, Cheng, Zeng, and Chen (2019) note that the time window issue includes the duration between when a task becomes available and when the worker is allocated to the task as well as the duration between when the task is assigned to the worker and when the worker reaches to perform the task. Taking the example of the ride hiring application Uber, the time window issue can be classified into when a rider requests a ride and before they are assigned a rider as well as when the duration that is taken

by the rider to arrive at the pickup point. To address the time delay issue, Chen, Cheng, Zeng, and Chen (2019) note the use of a threshold-based greedy algorithm as well as a batch-based model. In the threshold-based greedy algorithm, a time threshold is assigned to the travel time of a worker and any request for the worker is paired with the worker that has a travel time lower than the defined threshold. The threshold-based approach also takes into consideration the profit maximization for the client. This involves taking into consideration the maximum cost when defining the time threshold to prevent the occurrence of the maximum possible delay cost. In the batch-based model, the tasks and workers are buffered to create batches where the tasks and the workers are matched from the current batch or the previous batch. The batch-based model takes into consideration the time delay between when a task is available in a platform and the time when the platform assigns a worker to the task. To reduce this time delay, the batch-based model creates batches for the workers and the task. The buffering process allows a new task to be buffered to the defined duration and assigned to the appropriate worker either in the worker batch that arrived at the same time or the batch of workers from the next batch. (Z. Chen, Cheng, Zeng, & Chen, 2019). Chen, Cheng, Zeng, and Chen (2019) propose the use of the Hierarchically well-Separated Tree (HST) based solution that takes into consideration the time delay issue and buffers a task request to enable the identification of the best worker for a task as well as the duration that the buffering process should take. Another algorithm that can be adapted to address the time window issue is the Hamilton Energy-Efficient Routing (HEER) Protocol proposed by (Yi & Yang, 2016). In the HEER protocol, various clusters are formed and the components included in the clusters are linked to a Hamilton Path, which is generated using a greedy algorithm (Yi & Yang, 2016).

In one of the recommended solutions by Song et al. (2020), the lack of practical applicability of the solution is noted. Similarly, Yao, Xiong, Liu, and Liang (2016) note that apart from some of the recommended solutions to address the issue of task allocation in mobile crowdsensing platforms resulting in high communication costs, they also

have low applicability. Therefore, to address the various issues associated with task allocation in mobile crowdsensing, the recommended solutions should take into consideration the quality-of-service of the tasks. According to Yao et al. (2016), the process of measuring the quality-of-service of tasks involves addressing the completion time of the tasks. To address the issue of task completion duration, Yao et al. (2016) recommend the application of the offline activity allocation strategy and the online task allocation strategy. When compared, the online task allocation strategy proves to be more effective since it does not require a precalculated task allocation scheme to perform the task allocation that ensures that the duration taken to complete the tasks is minimized (Yao, Xiong, Liu, & Liang, 2017).

In mobile crowdsensing, the tasks allocation process is affected by various factors including the location of the workers and the tasks as well as the accessibility of the tasks. Additionally, time-sensitive tasks that involve a deadline increase the difficulty associated with task allocation, and this difficulty is also increased by the requirement for the definition of the trajectories of the workers. Akter and Yoon (2020) note that when addressing the task allocation process in mobile crowdsensing, apart from considering the deadline of the tasks, the response duration of the task has to be considered. This includes addressing the constant movement of the workers, which results in changes in the response time (Akter & Yoon, 2020). An example of this is seen in the ride hiring application, Uber, where the movement of a driver changes the duration within which they can reach the platform's client. Taking into consideration all the different aspects involved in the task allocation process, Akter and Yoon (2020) recommend the use of an approach that includes various algorithms including the greedy algorithm, the genetic algorithm, and the memetic algorithm to assigning the tasks. The proposed solution by Akter and Yoon (2020) not only addresses the time window issue but also addresses the total costs incurred by workers while performing different mobile crowdsensing activities.

According to Wu, Sun, Huang, Du, and Huang (2019), the application of various mobile crowdsensing activities involves addressing the different constraints that are associated

with the activities. In some cases, these tasks might have precedence constraints. Wu et al. (2019) note that the completion of these activities involves the minimization of the total execution time taken to perform the activities. An example provided on this includes the crowdsourcing system used in Unmanned Aerial Vehicles (UAVs), which include different tasks such as close air support (CAS), aerial refueling (AR), and even wide-area search and destroy (WASD) (Wu, Sun, Huang, Du, & Huang, 2019). Wu et al. (2019) also state that the different proposed solutions to address task allocation issues in mobile crowdsensing do not take into consideration the presence of precedence constraints in some of the crowdsourcing activities. Additionally, a sequence for completing the precedence activities is noted to increase the total execution time. Wu et al. (2019) propose an efficient task allocation algorithm that addresses the task allocation issue including the minimization of the execution time. The proposed algorithm involves the development of an allocation priority sequence that takes into consideration the expected completion time of the different tasks. Using the developed allocation priority sequence, the platform assigns the appropriate tasks to the workers (Wu et al., 2019).

In the various applications of mobile crowdsensing, one of the issues that affect the success of the crowdsourcing platform is task failure. An example of this can be seen in the application of mobile crowdsensing in the ride-hiring application Uber. In the current ride-hiring market various applications provide accessibility to the ride-hiring services. One of the competitors of Uber includes Lyft. In some instances, when clients that require ride-hiring services access the Uber platform to request a ride, they can complete the process, which results in them arriving at their destination using the rider or canceling the process and choosing to use another ride-hiring service such as the Lyft ride-hailing application. In these instances, the crowdsourcing activities are noted to have failed. According to Urbaczek, Saremi, Saremi, and Togelius (2020), the noted average failure ratio is 15.7

To address the issue of task failure, Urbaczek et al. (2020) propose a task scheduling model that increases the efficiency of the crowdsourcing activities, which results in the in-

crease of the success of the crowdsourcing tasks. The main aspects of mobile crowdsensing taken into consideration by Urbaczek et al. (2020) involve the rewards to the workers that complete the activities and the duration that is taken to perform the task. The proposed model involves predicting the probability of failure of the different tasks that arrive at the crowdsourcing platform and providing recommendations that would reduce the probability of the tasks failing (Urbaczek, Saremi, Saremi, & Togelius, 2020). (He, Shin, Zhang, & Chen, 2014) notes the importance of addressing the time budget issue as well as the difference in the locations of the tasks when addressing the issue of task allocation so as to provide an optimal approach of allocating tasks to the available workers. This involves taking into consideration the spatial movements of the clients that request the crowdsourcing activities, the geographical locations of the workers, and proposing the location ratio based algorithm (LRBA) that solves the task allocation issue in mobile crowdsensing. Deng, Shahabi, and Zhu (2015) also propose an approach that can address the issue of task matching and scheduling that can contribute to the failure of the tasks. In the study by Deng, Shahabi, and Zhu (2015), they note that the successful completion of spatial tasks within the defined duration depends on whether the assigned workers are able to get to the position of the tasks before the estimated duration of arrival expires. Therefore, addressing different noted issues of task assignment, time window, and profits maximization can reduce the probability of the tasks failing.

Chapter 3

Problem Description

3.1 System Model

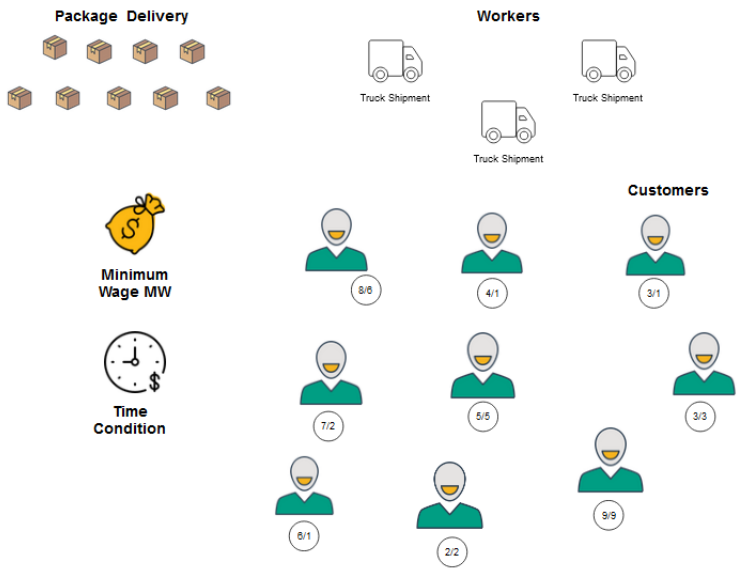

Figure 3.1: Problem Definition

The system addressed in this problem includes different workers that complete different tasks that are provided by a mobile crowdsensing platform. This system includes three

entities which include the mobile crowdsensing platform, the workers, and the available tasks that require to be completed. The workers who complete the tasks in the platform are denoted by $W = \{w_1, w_2, \cdots, w_n\}$ while the tasks that are competed by the workers are denoted by $O = \{o_1, o_2, \cdots, o_n\}$. The tasks are submitted to the platform then the mobile crowdsensing platform acts as the task manager and allocates the available tasks to the workers that are available. The workers are noted to originate from the same location. Apart from the workers and the tasks, the profit function is also defined as$p : O_j \rightarrow N^+$. Additionally, the system also includes a minimum wage which is denoted as MW.

3.2 Problem Formulation

In crowdsourcing, the main identified issues addressed in this study include the task allocation to the workers, the profit maximization of the workers and of the platform, as well as the minimum wage that is earned by the workers. In mobile crowdsensing, the crowdsensing platform acts as the in-between the requested tasks and the workers that perform the tasks. On one side of the mobile crowdsensing platform, there are clients that request the different tasks from the platform, and on the other side of the platform, there are workers that perform the requested tasks. Therefore, the problem addressed in this study involves allocating the requested tasks to the available workers. Additionally, after performing the tasks, there are different profits that are expected. These profits include those obtained by the workers as well as those obtained by the platform. After the performing the task, both the platform and the worker are able to obtain a fixed rate of profit p_j when a worker completes a task o_j. Therefore, when provided with a subset of orders $O' \subseteq Q$, the obtained profit from after the workers complete the tasks can be noted by:

$$\sum_{o_j \in Q} p(o_j) \tag{3.1}$$

One of the noted aspects of mobile crowdsensing includes the definition of the time taken by the workers to complete the tasks. This includes the duration from when the tasks are requested by the client on the platform as well as the time that the worker takes to complete the task after being assigned the specific tasks. The time taken by each worker w_i to complete a task o_j will be denoted by $t_{i,j}$. The total amount of time that each worker w_i can spend is bounded by T_i. Similar to the requirement profit that is required by the workers and the platform, each worker requires a specified minimum income that is denoted by I_i. The minimum income of each worker is defined to prevent some of the workers from not getting any profits from the platform. Additionally, the minimum income also prevents some of the workers from gaining more profits while some other workers only achieve little to no profits from the platform. One of the objectives of the problem is to ensure that the profit of the platform is maximized.

3.3 Proof of NP-Hardness

In some instances, there is no practical solution to the issue. An example of the this can be seen in the instances when the desired minimum income I_i is too high or the total time taken by the worker T_i is too low. In other instances when the task assignment for each worker is able to meet the defined constraints there is a feasible solution. This confirms that the issue is NP-Hard.

Chapter 4

Proposed approach

In the identified problem, the required solution involves ensuring that the defined minimum wage that is earned by the workers is maintained, assigning the tasks to the workers, and ensuring that both the workers and the platform attain maximum profits within the provided time window. The proposed solution includes the following algorithms:

4.1 Algorithm 1

In the first algorithm of the proposed solution, the initial step is to get the different required parameters from the user. The included parameters include the different tasks that are needed to be performed, the number of available workers that will be assigned the available tasks, the minimum wage that is required for the workers to ensure they profit, and the durations that are needed to complete each of the identified tasks. The proposed solution then conducts two calculations simultaneously. These calculations include identifying the duration necessary to finish the tasks assigned to workers and running the Knapsack algorithm to allocate the provided tasks to the available workers. Completion of the allocating of tasks to the workers and calculating the duration it will take to complete the assigned tasks, the solution proceeds to check whether the different identified solutions met the re-

quired minimum wage that is defined for the workers. When the identified solutions do not meet the required minimum wage conditions, then the algorithm will not continue to execute and this results in the end of the execution of the algorithm after the program prints out a message to the user to change the provided the minimum wage to a lower value to ensure that all the tasks have been allocated to the workers. If the assignment of the tasks to the available workers meets the required conditions for the minimum wage then the algorithm proceeds to execute. In the next phase of the algorithm, the program proceeds to check whether all available tasks have been assigned to the workers. If some of the tasks have not been assigned, the algorithm proceeds to allocate the tasks to the workers that have the least expected profits. In instances when all the workers have the same least expected profits, the algorithm proceeds to check the workers that have the least number of tasks to complete and assigns the tasks to them. In the instances when all the tasks have been assigned, the algorithm proceeds to update the time taken to complete the tasks. After updating the time required to complete the tasks, the algorithm prints the tasks for each worker, the profit that each worker will earn after completing the tasks, and the time that the worker will take to complete the tasks. The performance metrics, which include the average maximum profit that is attained by the platform as well as the average deficit are also included in the outputs. Printing these outputs results in the execution of the program ending.

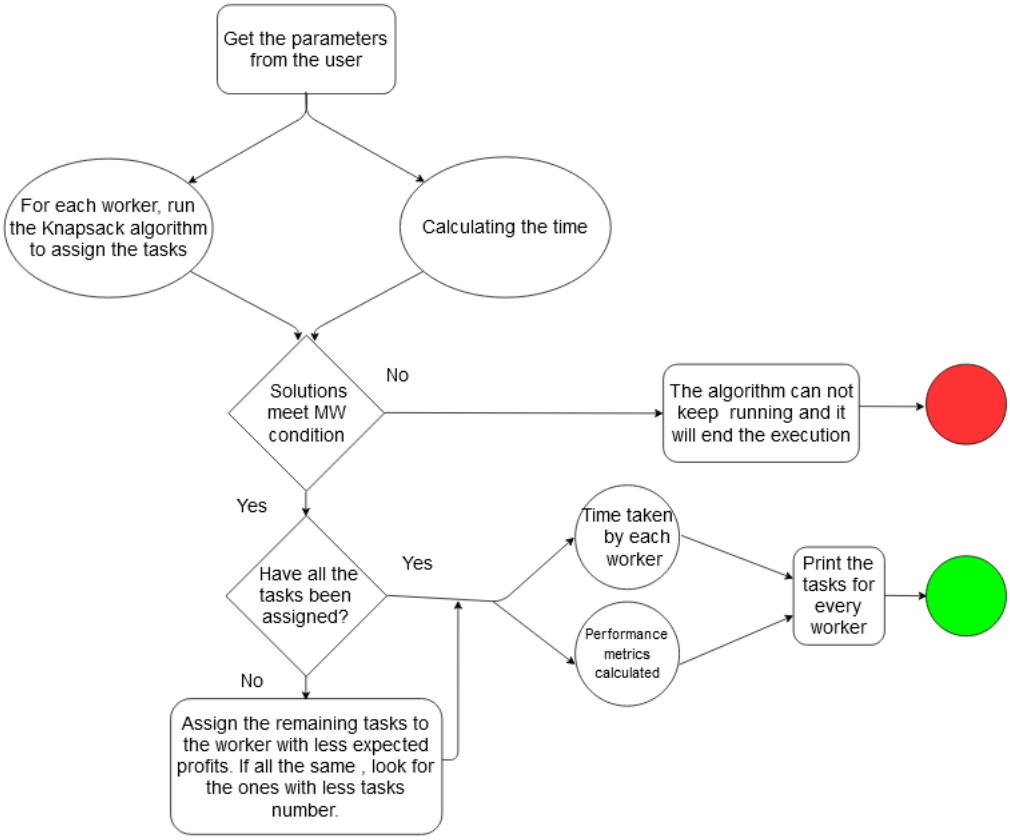

Figure 4.1: Flowchart for Algorithm 1

```
Input:  Tasks and Profits of the Tasks, Time to Complete the Tasks, Number of Workers, Minimum Wage.
Output: Tasks for Each Worker, Profits for Each Worker, Time to Complete Assigned Tasks, Platform Profit, Average Deficit.
1: Knapsack Algorithm (c, wt, v, n): //The knapsack algorithm used to assign tasks to workers
2: For i in range (n+1) do
3:      K[i][w] = K[i-1][w]
4: res = K[n][c] //store the results of Knapsack algorithm
End for
5: for i in range (n, 0, -1) do
6:      result_ksa.append(wt[i-1])
7: return result_ksa
8: Calculate the time taken by each worker to complete assigned tasks
9: find_time (task_list, time_list, sublist) do
10:     time_taken.append(time_list[indice])
11: return sum(time_taken)
12: Identify whether the solution meets the minimum wage conditions
13:     If solution does not meet minimum wage solution
14:             End program
15:     Else:
16:     If all(i > minimum_wage fro i in tasks_and_its_profits):
17:             Deliver_number = sum (tasks-and_its_profits)
18:             Check whether all tasks have been delivered
19:                     If some tasks have not been delivered
20:                             Assign them to the worker with less expected profits
21:                                     If all the expected profits are the same
22:                                             Assign tasks to workers with less numbers of tasks
23:                                     Else
24:                                             Continue
25:                                                     For i in tasks_and_its_profits do
26:                                                     result_array[index].append(i)
27:                                     Else
28:                             Update time taken by each worker
29:                             Calculate performance metrics
30: Print the tasks for each worker, profits for each worker, time to complete assigned tasks, platform profit, and average deficit.
```

Figure 4.2: Pseudocode for Algorithm 1

4.2 Algorithm 2

The second algorithm of the proposed solution involves assigning the available tasks to the workers, checking whether the minimum wage conditions for the workers have been met, as well as checking whether the time condition has been achieved. After the program is started, the initial process is to acquire the necessary parameters from the users. The essential parameters in the second algorithm include the different tasks that should be to be performed, the number of available workers that will be assigned the available tasks, and the minimum wage that is required for the workers to ensure they profit after completing the tasks. Additionally, the different durations that are required to complete the available tasks are also required inputs for the algorithm as well as the time condition. The algorithm then proceeds to run the Knapsack algorithm that allocates the available tasks to the available workers and calculates the needed duration to complete the required activity for the workers. After that, the program uses the time condition to check whether the available time is enough to allocate all the available tasks to the available workers. When time is not enough then this is the instance when the program continues to exit and print the alert to the user to modify the time condition. If there is enough time, then all the tasks will be assigned and the program progresses. If enough time has been determined to be available to have all the tasks assigned to the workers, the program proceeds to check whether the minimum wage conditions for the workers have been met. If the conditions for the minimum wage for the workers have not been met, then the program will proceed to allocate the tasks to the workers, but it will also prints an alert to the users to inform the user that at least one of the workers will earn less than the minimum condition and it will suggest to change the minimum wage. However, if the solution meets the minimum wage solutions, then the program prints the tasks assigned to each worker, the expected profits that the workers will earn after completing the tasks, the required time to complete the assigned tasks, as well as

the evaluation metrics.

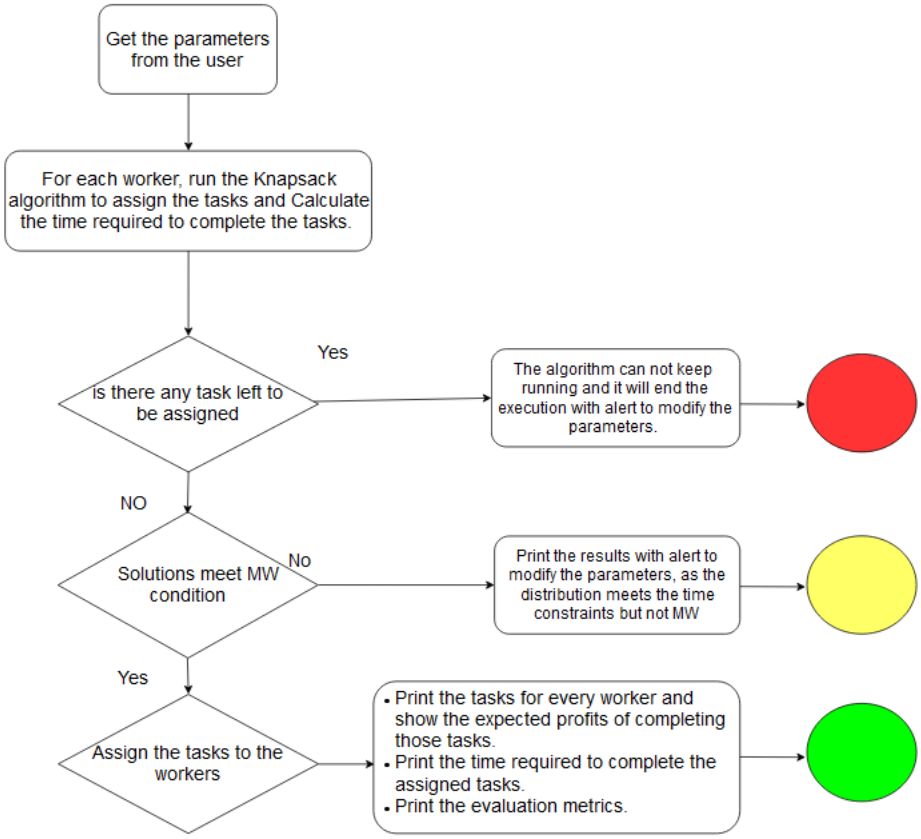

Figure 4.3: Flowchart for Algorithm 2

```
Algorithm 2: Proposed Solution
Input: Tasks and Profits of the Tasks, Time to Complete the Tasks, Number of Workers, Minimum Wage, Time Condition
Output: Tasks for Each Worker, Profits for Each Worker will Win, Time to Complete Assigned Tasks, Platform Profit, Average Deficit.
1: Knapsack Algorithm (W, wt, val, n): //The knapsack algorithm used to assign tasks to workers
2:      K = [[0 for w in range(W + 1)] for i in range(n + 1)]
3: Build table K[][] in bottom-up manner
4: for i in range(n + 1)
5:      K[i][w] = K[i-1][w]
6: res = K[n][W] //store the results of the Knapsack
7: Calculate the time
8: weights = time_of_tasks.copy()
9: tasks_for_time_calc = task_and_its_profits.copy()
10 number = time_condition
11:elements = len(tasks_and_its_profits)
12: //time_of_tasks = [number] * elements
13: Identify if there are any tasks left to assign
14:      If tasks left to assign
15:            If len(tasks_and_its_profits)> 0:
16:      Print the remaining tasks
17:      Print alert to modify parameters and end program
18:      Print alert to change the time contraint to a higher number
19:      Else
20:            Identify if solutions meet MW condition
21:                  If solution does not meet MW condition
22:                  If min(assert_mw_condition) < minimum_wage:
23:            Print alert to modify parameters and end program
24:            Print there is a feasible solution but MW condition not met
25:            Print recommendation to either increase profits of tasks or decrease MW condition
26:      Else
27:      Print the tasks for each worker, profits for each worker, time to complete assigned tasks, platform profit, and average deficit.
28: End program
```

Figure 4.4: Pseudocode for Algorithm 2

Chapter 5

Implementation and Simulation

In the proposed solution, the noted algorithms address the aspects of task assignment, the profit maximizations for both the workers and the platforms, as well as the required durations to complete the assigned tasks. Both algorithms require different parameters to perform the necessary tasks. These parameters include available tasks, the available workers, the required time to perform the assigned tasks, and the minimum wage required for the workers. Differences between the algorithms are noted in the calculations conducted by the algorithms. In the first algorithm, the Knapsack algorithm and the calculations for the required duration to complete the tasks are conducted simultaneously using different functions.

An example of the implementation process of the first algorithm includes: given the following set of parameters. ([8, 7, 6, 4, 5, 2, 3, 3, 9],[6, 2, 1, 1, 5, 2, 1, 3, 9], 3, 16) where [8, 7, 6, 4, 5, 2, 3, 3, 9] includes the list of the tasks to be assigned, [6, 2, 1, 1, 5, 2, 1, 3, 9] the list for the time required for the tasks, and (3,16) where 3 is the amount of available workers and 6 is the minimum wage. Running the program results in the following alert being printed out:

```
 ----jGRASP exec: python package_delivery.py
With Knapsack Algorithm Logic minimun expected profit is: 15
We recommend lower (or equal value) values than that for MW condition
----------------
Warning: the MW number is greater than the equitative expected profit every worker will make
No feaseable solution using Knapsack Algorithm
PLEASE TRY WITH A LOWER MW NUMBER

 ----jGRASP: operation complete.
```

The occurrence of the alert shows that the minimum wage needs to be reduced and in this case, it should be lowered to 15. After modifying the MW, the following output will be printed out:

```
 ----jGRASP exec: python package_delivery.py
Distributing of the tasks will be as follows:
----------------
Tasks for worker 1 are :[7, 8, 2] and the profit they will win is : 17 completed on time : 10
Tasks for worker 2 are :[5, 4, 6] and the profit they will win is : 15 completed on time : 7
Tasks for worker 3 are :[9, 3, 3] and the profit they will win is : 15 completed on time : 11
----------------
Metrics outputs:
Average maximun profit achieved by the platform is: 15.666666666666666
Average deficit is: 3.0
----------------
Delivery ended with no troubles and all the profits are maximized!

 ----jGRASP: operation complete.
```

The above result of the program includes the distribution that meets minimum wage conditions while ensuring that all the workers are allocated a similar amount of tasks to make the profits for the workers similar.

In the second algorithm, the Knapsack algorithm and the time calculations for the assigned tasks are conducted by the Knapsack algorithm at the same time. Additionally, in the second algorithm, the time calculations are also essential aspects in the determination of whether the minimum wage conditions for the workers have been achieved. In the second algorithm, the algorithm checks whether the time constraints have been met. This involves checking whether the defined time condition makes it possible to have enough time to have all available tasks be allocated to the available workers. For instance, same list used in the first algorithm is used again in the second algorithm with a time condition included and defined as 9 as follows: ([8, 7, 6, 4, 5, 2, 3, 3, 9],[6, 2, 1, 1, 5, 2, 1, 3, 9], 3, 15, 9). When the program runs the following output which includes an alert will be printed:

```
 ----jGRASP exec: python package_delivery_1.py
Remaining tasks to assign are
[5]
PLEASE CHANGE THE TIME CONSTRAINT TO A HIGHER NUMBER IN ORDER TO ASSIGN ALL TASKS

 ----jGRASP: operation complete.
```

The above output shows that due to the time constraints there are some tasks that will not be assigned to the workers and in this case, the remaining task is 5. Therefore, the user needs to increase the time condition. After increasing the time to 15, the following output will be printed which informs the user the modify the MW in order to meet the MW conditions.

```
 ----jGRASP exec: python package_delivery_1.py
Warning: THERE IS SOLUTION FEASIBLE BUT AT LEAST ONE WORKER WILL EARN LESS THAN THE MW CONDITION
We suggest two options: INCREASE the profits of each tasks OR DECREASE the MW condition
Tasks for worker 1 are :[3, 3, 4, 6, 7, 8] and the profit they will win is : 31 completed on time : 14
Tasks for worker 2 are :[9, 5] and the profit they will win is : 14 completed on time : 14
Tasks for worker 3 are :[2] and the profit they will win is : 2 completed on time : 2
Metrics Report
Average maximun profit achieved by the platform is: 15.666666666666666
Average deficit is: 3.0
----------------
Delivery ended with no troubles and all the profits are maximized

 ----jGRASP: operation complete.
```

After modifying the MW to 2, the following output will be printed:

```
 ----jGRASP exec: python package_delivery_1.py
Tasks for worker 1 are :[3, 3, 4, 6, 7, 8] and the profit they will win is : 31 completed on time : 14
Tasks for worker 2 are :[9, 5] and the profit they will win is : 14 completed on time : 14
Tasks for worker 3 are :[2] and the profit they will win is : 2 completed on time : 2
Metrics Report
Average maximun profit achieved by the platform is: 15.666666666666666
Average deficit is: 3.0
----------------
Delivery ended with no troubles and all the profits are maximized

 ----jGRASP: operation complete.
```

As seen in both algorithms, the average maximum profit that is achieved by the platform is the same, but the difference between the algorithms is noted in the distribution of the tasks to the workers. In the first algorithm the assignment is as follows:

$[7,8,2]$

$[5,4,6]$

$[9,3,3]$

However, in the second algorithm, the tasks allocation for the workers includes:

$[3,3,4,6,7,8]$

$[9,5]$

$[2]$

After both algorithms check whether the various minimum wage conditions have been achieved, they proceed to provide outputs for the calculations. The noted common outputs for both the algorithms include the different assigned tasks for each worker, the profit that they will win after the completion of the tasks, as well as the duration necessary to complete the tasks. Both the first and second algorithms also include the performance metrics in the outputs. Essential components of the performance metrics include the maximum profit that is earned by the platform and the average deficit. Additionally, in both algorithms, the overall profit for the platform is noted to be the same. A difference between the outputs of the two algorithms is noted in the distribution of the tasks. In the first algorithm, all the workers are assigned the same amount of tasks while in the second algorithm the use of the time condition prevents all the workers to have a similar amount of tasks that contribute to the differences in the profits. However, the second algorithm ensures that the overall profit of the platform is similar to that of the first algorithm. However, in the second algorithm, the minimum wage is changed to allow all the tasks to be assigned to the workers.

Algorithm 1	**Algorithm 2**
Required Parameters	**Required Parameters**
• Available Tasks • Number of Available Workers • Required Time to Complete the Tasks • Minimum Wage required for the Workers	• Available Tasks • Number of Available Workers • Required Time to Complete the Tasks • Minimum Wage required for the Workers • Time Condition
Calculations	**Calculations**
• Running the Knapsack Algorithm to assign the Tasks to the Workers simultaneously with the Calculations for the Time required to Complete the Tasks. • The calculations for the first algorithm are conducted using separate functions	• Running the Knapsack Algorithm to Assign the Tasks to the Workers. • Calculation to Identify the Required Time to Complete the Tasks is also conducted by the Knapsack Algorithm.
Outputs	**Outputs**
• Assigned tasks to each of the workers • Required Durations to complete the tasks • Expected Profits for the Workers • The performance metrics report	• Assigned tasks to each of the workers • Required Durations to complete the tasks • Expected Profits for the Workers • The performance metrics Report

Table 5.1: Algorithms Comparison

Chapter 6

Evaluation

A comparison has been conducted with an existing algorithm that involves the use of the multi-knapsack algorithm.
(see Ref. $https://developers.google.com/optimization/bin/multiple_knapsack$)
To test the multi-Knapsack algorithm the same list of variables that were used in the second algorithm was used again in the multi-Knapsack algorithm. The list of the variables includes: ([8, 7, 6, 4, 5, 2, 3, 3, 9],[6, 2, 1, 1, 5, 2, 1, 3, 9], 3, 15, 9). The obtained output of the multi-Knapsack algorithm using the provided input included:
$[7,8,6,4,5]$
$[2,3,3,9]$
$[0]$

The output includes the tasks for workers 1, 2, and 3 respectively. It can be noted that worker 3 has 0 assigned task and as a result, has 0 profit.

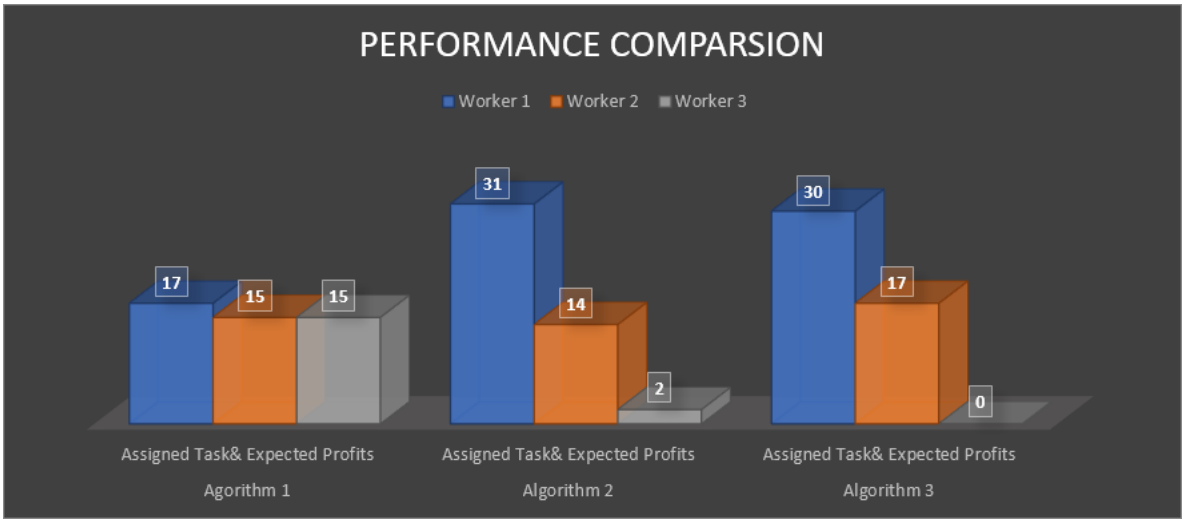

Figure 6.1: Performance Comparison

Figure 6.1 illustrates how the distribution of the tasks and the expected profits for the workers was performed in the three algorithms. In the first algorithm, the workers were able to obtain a similar number of profits and assigned tasks as well. In the second algorithm, the distribution did not follow a consistent pattern, and the profits varied for each worker. This was because of the presence of both time constraints and the minimum wage conditions. Although all the tasks were assigned, the minimum wage condition had to be lowered to ensure that the minimum wage requirement for the workers was achieved and that the workers were able to make a profit. Using the outputs data from the first two algorithms to compare the obtained outputs of the proposed solution to the multi-Knapsack algorithm, we noticed one of the workers was not able to make any profit. This proves the efficiency of the proposed solution to address the issue of profit maximization for the workers.

Chapter 7

Conclusion

Crowdsourcing involves various processes and these processes can be affected by different issues. These issues can include task assignment, profit maximization for the workers, clients, and the platform, as well as time window issues. While there are different algorithms proposed to address the issues, they do not address all the noted issues. To address this shortcoming, a proposed approach is noted to address the three noted issues at the same time rather than individually. A comparison between the proposed solution and other existing algorithms results in uneven profit distribution for the workers, where in some instances some workers are not assigned tasks which results in the workers not getting any profit or getting lower profit when compared to the other workers. The allocation of the tasks to the workers while maintaining the minimum wage requirement and ensuring profit maximization for the workers proves the efficiency of the proposed solution.

Bibliography

Akter, S., & Yoon, S. (2020). Datask: A decomposition-based deadline-aware task assignment and workers' path-planning in mobile crowd-sensing. *IEEE Access*, *8*, 49920-49932. doi: 10.1109/ACCESS.2020.2980143

Boubiche, D. E., Imran, M., Maqsood, A., & Shoaib, M. (2019). Mobile crowd sensing–taxonomy, applications, challenges, and solutions. *Computers in Human Behavior*, *101*, 352–370.

Chen, C., Zhang, D., Ma, X., Guo, B., Wang, L., Wang, Y., & Sha, E. (2016). Crowddeliver: Planning city-wide package delivery paths leveraging the crowd of taxis. *IEEE Transactions on Intelligent Transportation Systems*, *18*(6), 1478–1496.

Chen, Z., Cheng, P., Zeng, Y., & Chen, L. (2019). Minimizing maximum delay of task assignment in spatial crowdsourcing. In *2019 ieee 35th international conference on data engineering (icde)* (pp. 1454–1465).

Cheng, P., Lian, X., Chen, L., & Shahabi, C. (2017). Prediction-based task assignment in spatial crowdsourcing. In *2017 ieee 33rd international conference on data engineering (icde)* (pp. 997–1008).

Deng, D., Shahabi, C., & Demiryurek, U. (2013). Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *Proceedings of the 21st acm sigspatial international conference on advances in geographic information systems* (pp. 324–333).

Deng, D., Shahabi, C., Demiryurek, U., & Zhu, L. (2016). Task selection in spatial crowd-

sourcing from worker's perspective. *GeoInformatica*, *20*(3), 529–568.

Deng, D., Shahabi, C., & Zhu, L. (2015). Task matching and scheduling for multiple workers in spatial crowdsourcing. In *Proceedings of the 23rd sigspatial international conference on advances in geographic information systems* (pp. 1–10).

Gong, W., Zhang, B., & Li, C. (2018a). Location-based online task assignment and path planning for mobile crowdsensing. *IEEE Transactions on Vehicular Technology*, *68*(2), 1772–1783.

Gong, W., Zhang, B., & Li, C. (2018b). Task assignment in mobile crowdsensing: Present and future directions. *IEEE network*, *32*(4), 100–107.

Han, Y., & Zhu, Y. (2014). Profit-maximizing stochastic control for mobile crowd sensing platforms. In *2014 ieee 11th international conference on mobile ad hoc and sensor systems* (pp. 145–153).

Hassan, U. U., & Curry, E. (2014). A multi-armed bandit approach to online spatial task assignment. In *2014 ieee 11th intl conf on ubiquitous intelligence and computing and 2014 ieee 11th intl conf on autonomic and trusted computing and 2014 ieee 14th intl conf on scalable computing and communications and its associated workshops* (pp. 212–219).

He, S., Shin, D.-H., Zhang, J., & Chen, J. (2014). Toward optimal allocation of location dependent tasks in crowdsensing. In *Ieee infocom 2014-ieee conference on computer communications* (pp. 745–753).

Hu, Y., Wang, J., Wu, B., & Helal, S. (2020). Participants selection for from-scratch mobile crowdsensing via reinforcement learning.

Li, D., Zhu, J., & Cui, Y. (2019). Prediction-based task allocation in mobile crowdsensing. In *2019 15th international conference on mobile ad-hoc and sensor networks (msn)* (pp. 89–94).

Liu, H., Hu, S., Zheng, W., Xie, Z., Wang, S., Hui, P., & Abdelzaher, T. (2013). Efficient 3g budget utilization in mobile participatory sensing applications. In *2013 proceedings*

ieee infocom (pp. 1411–1419).

Liu, W., Yang, Y., Wang, E., & Wu, J. (2020). Dynamic user recruitment with truthful pricing for mobile crowdsensing. In *Proc. of the 39th ieee international conference on computer communications (infocom 2020).*

Ma, Z., Liu, L., & Sukhatme, G. S. (2016). An adaptive k-opt method for solving traveling salesman problem. In *2016 ieee 55th conference on decision and control (cdc)* (pp. 6537–6543).

Silberman, M. S., Tomlinson, B., LaPlante, R., Ross, J., Irani, L., & Zaldivar, A. (2018). Responsible research with crowds: pay crowdworkers at least minimum wage. *Communications of the ACM*, *61*(3), 39–41.

Song, T., Xu, K., Li, J., Li, Y., & Tong, Y. (2020). Multi-skill aware task assignment in real-time spatial crowdsourcing. *GeoInformatica*, *24*(1), 153–173.

Tao, X., & Song, W. (2018). Efficient path planning and truthful incentive mechanism design for mobile crowdsensing. *Sensors*, *18*(12), 4408.

Tao, X., & Song, W. (2020). Task allocation for mobile crowdsensing with deep reinforcement learning. In *2020 ieee wireless communications and networking conference (wcnc)* (pp. 1–7).

Urbaczek, J., Saremi, R., Saremi, M. L., & Togelius, J. (2020). Scheduling tasks for software crowdsourcing platforms to reduce task failure. *arXiv preprint arXiv:2006.01048*.

Wang, E., Luan, D., Yang, Y., & Wu, J. (2019). Facility location strategy for minimizing cost in edge-based mobile crowdsensing. In *2019 ieee 16th international conference on mobile ad hoc and sensor systems (mass)* (pp. 407–415).

Wang, J., Wang, L., Wang, Y., Zhang, D., & Kong, L. (2018). Task allocation in mobile crowd sensing: State-of-the-art and future opportunities. *IEEE Internet of Things Journal*, *5*(5), 3747–3757.

Wu, X., Sun, Y.-E., Huang, H., Du, Y., & Huang, D. (2019). Time-efficient allocation

mechanisms for crowdsensing tasks with precedence constraints. *Sensors*, *19*(11), 2456.

Xiaoyu, X., Huang, Z., & Lin, Z. (2018). Trajectory-based task allocation for crowd sensing in internet of vehicles. In *2018 international conference on robots & intelligent system (icris)* (pp. 226–231).

Yao, H., Xiong, M., Liu, C., & Liang, Q. (2017). Encounter probability aware task assignment in mobile crowdsensing. *Mobile Networks and Applications*, *22*(2), 275–286.

Yi, D., & Yang, H. (2016). Heer–a delay-aware and energy-efficient routing protocol for wireless sensor networks. *Computer Networks*, *104*, 155–173.

Zhan, Y., Xia, Y., Zhang, J., & Wang, Y. (2017). Incentive mechanism design in mobile opportunistic data collection with time sensitivity. *IEEE Internet of Things Journal*, *5*(1), 246–256.