A Parcel Delivery Scheduling Scheme in Road Networks

1st Junhee Kwon
Computer Science & Engineering
Sungkyunkwan University
Suwon, Republic of Korea
juun9714@skku.edu

2nd Jaehoon (Paul) Jeong
Computer Science & Engineering
Sungkyunkwan University
Suwon, Republic of Korea
pauljeong@skku.edu

***Abstract*—Due to COVID-19, ordering food through online shopping increased. Accordingly, the use of logistics and delivery services is also increasing. As the number of parcels to be delivered gets bigger, the efficiency of the delivery mechanism and battery efficiency becomes important. The problem of finding the route traveling several destinations at once is called as Traveling Salesman Problem (TSP). There are several algorithms suggested to solve it in polynomial time. Among them, this paper experimented to compare the performance of two algorithms, the greedy algorithm, and the branch-and-bound algorithm. We used the Simulation of Urban Mobility (SUMO) program to test the vehicle running based on the calculated route by two algorithms. The average running time and charging time are recorded to evaluate the performance. Through this experiment, we found out that the branch-and-bound algorithm provides in a faster route selection and consumes less battery than the greedy algorithm.**

***Index Terms*—parcel delivery, shortest path, branch-and-bound algorithm, SUMO, TraCI, road network**

I. Introduction

As the amount of time spent indoors has increased recently due to COVID-19, the scale of online shopping transactions has increased significantly. In Korea, online shopping transactions of foods such as grocery, beverage, agricultural, and fishery products increased by 37.8% compared to 2 years ago. In addition, the number of delivery food orders also increased by 97.7% in two years. In February 2022, the total online shopping transaction amount was 15.4314 trillion won, up to 30.9% from two years ago [1].

This increase in online shopping naturally led to heavy courier logistics volume. Although the trend of the domestic delivery market has been steadily increasing since 2012, the growth rate of the delivery volume in 2020 compared to the previous year was 20.9%. Considering that the number of parcels has been increasing by around 10% every year since 2016, it is considered a radical increase. The total number of parcels in 2021 was 3.62 billion.

In this situation, it is natural that the efficiency of delivery becomes important. The factor that determines the efficiency of parcel delivery is cost, and the cost consists of time and resources. Resources are petroleum or electric batteries for courier transportation. International crude oil prices for the year were at a low price of 62.410 USD and a high price of 115.68 USD. The price of resources is very unstable. When cost fluctuates, it is not easy for companies that operate courier services to maintain their business stably. Therefore in transportation services, time is considered the same as cost. The faster and shorter the delivery is, the more resources we can save. At the same time, environmental pollution caused by petroleum is also a serious problem. Thus, it is desirable to maximize the efficiency of petroleum and further transfer existing vehicles to electric vehicles.

There are several methods to improve the efficiency of courier delivery. Those methods enhance a courier distribution network and collect logistics for a nearby location to deliver them at once. However, methods such as clustering and networking are already difficult to improve. Currently, the most needed enhancement in the delivery industry can be an efficient route selection for the visit of several delivery destinations.

Now, when courier drivers are assigned the parcels to be delivered, they first check their assigned parcels' destinations and efficient routes by themselves. Furthermore, they load the courier boxes into the vehicle by the order they decide to visit. They load the parcels that will be delivered at last at the innermost side and load the parcels that will be delivered first at the outermost side. When loaded in this way, it is possible to take out items in visit order and deliver them to customers when they arrive at each delivery destination without finding the parcels to deliver.

Experienced courier drivers can use their skills to find an efficient route. However, this way cannot guarantee full optimization from the overall perspective of the courier service company. Applying an efficient algorithm to the delivery service would ensure better optimization from the overall perspective, rather than determining the route by the driver himself.

With this goal, the authors of this paper decided to research which algorithm can find the faster route to visit many delivery destinations. Therefore, this paper evaluates the running time and the charging time of the fastest route based on the different algorithms.

In this paper, the performance of the two algorithms (i.e., Greedy and Branch-and-Bound algorithms) was compared, and a simulation program was used to check the execution time of the two algorithms. The Simulation of Urban Mo-

bility program, called SUMO, is a program that implements activities of various entities on the road network such as a pedestrian, vehicle, and public transportation [2], [3]. Using the SUMO program, users can define specific vehicles with various characteristics and construct roads. In this paper's simulation, the vehicle is defined as an electric vehicle, so it is charged by a charging station while driving depending on the remaining battery capacity and heads back to its original destination after the charging. The road network in which such an electric vehicle drives consists of a round trip 8-lane road in a 12x12 grid. Traffic lights and several electric charging stations are located on the road, and vehicles start from the same start point and visit their delivery destinations one by one.

Using this simulation, this paper calculates and compares the performance of the two heuristic algorithms. The author of this paper selected the Greedy algorithm and Branch-and-Bound algorithm as performance evaluation targets. This is because the Greedy algorithm and Branch-and-Bound algorithm have different approaches to finding the shortest route.

In a general problem, the most efficient one is selected after calculating all cases. However, the problem like "finding the fastest route to visit all destinations" that we currently have to solve is not the case. If we calculate all cases in this problem, it becomes impossible to find answers in polynomial time, i.e., reasonable time. It is because there are too many cases to calculate, and the problems like this are called NP-Hard problems. The optimal solution for NP-Hard problems cannot be obtained without calculating all cases. As a result, the Heuristic algorithm is a way to escape from this dilemma. There are several heuristic algorithms. Greedy algorithm and Branch-and-Bound algorithm belong to it.

The Greedy algorithm makes decisions that look best at each step. As it is the simplest and easiest way to implement and understand, it is popular and used a lot in implementation [4]. However, it does not guarantee the most efficient decisions throughout the system. Although it may seem inefficient at this step, the choice may be a better decision from the overall point of view [5].

The Branch-and-Bound algorithm improves the method that considers all cases which consumes a lot of time to find the answer [6]. It tries to calculate all cases, but the algorithm skips the cases that are unlikely to be the most efficient and starts calculating other remained cases. By skipping the meaningless cases, this algorithm can find the answer faster. By comparing the running time consumed to deliver all the parcels based on two different algorithms, this paper suggests the direction for the logistics industry to develop further.

When the more efficient algorithm is determined, that algorithm can be applied to the entire courier service system. Recently, as the number of parcels has increased, the burden of work has increased significantly, making courier drivers overwork [7], [8]. Applying the effective algorithm to the entire system will reduce the burden on courier drivers. Once courier drivers spend less time deciding the delivery order, they can focus on the delivery itself. In addition, if the order of the visited delivery destinations is systematically determined, the loading order of the delivery volume can also be systematically determined accordingly. That is why the burden on delivery drivers can be reduced consecutively. Lastly, if the order of delivery visit and loading can be systematically implemented in the future, autonomous delivery can also be started. When the system of the delivery industry gets better, the workload and burden of drivers may be reduced. Furthermore, the cost of the delivery service may be significantly reduced through autonomous delivery.

The remainder of this paper is organized as followed. Section II summarizes the research about the algorithms for the Traveling Salesman Problem (TSP). Section III describes the architecture of our simulation for the performance evaluation. Section IV shows the implementation and Section V performance evaluation of our simulation. Lastly, Section VI concludes our paper along with the future work.

II. Related Work

The algorithms to find the shortest route that visits all the destinations at once have been an important aspect of driving efficiency research. This section reviews the required knowledge to understand this paper and the existing research of algorithms proposed by the recent literature.

The traveling salesman problem is called TSP in short [9]. TSP is a popular problem in the software area because a lot of academic and commercial areas make use of the solutions to this problem. This problem is about finding the shortest way to visit all the destinations. The salesman starts the trip to visit all the destinations and should return to the start point. To reduce the cost of a trip, finding the shortest path is the most important part. As there are extremely many possible ways to tour all those destinations, thoroughly calculating them all is impossible in polynomial time. A problem like this is called an NP-Hard problem. Although the NP-Hard problem needs to calculate all the cases to get the shortest route, it is impossible to calculate all in polynomial time. To solve this NP-Hard problem, a heuristic algorithm is used. The heuristic algorithm can make the approximate solution for the problem in reasonable time [10].

The greedy algorithm and branch-and-bound algorithm belong to the heuristic algorithm. The greedy algorithm chooses the best option at each step without considering the overall optimization. As a result, it does not ensure the shortest route for the problem. However, it is a fast and simple way to implement the algorithm that finds a reasonable answer. Another algorithm is the branch-and-bound algorithm. The branch-and-bound algorithm is devised to overcome the time-consuming classic algorithms [6]. This algorithm reduces the calculation time. It keeps the minimum value while calculating the cases and compares the minimum value with the calculating value. If the middle value of a specific case already exceeds the minimum value, it skips the remained calculation of the case and starts the next case's calculation. By this mechanism, it can reduce the meaningless calculation [11].

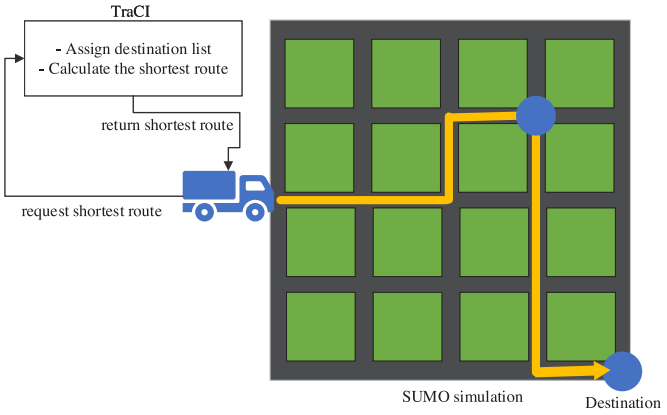

Fig. 1. SUMO and TraCI architecture

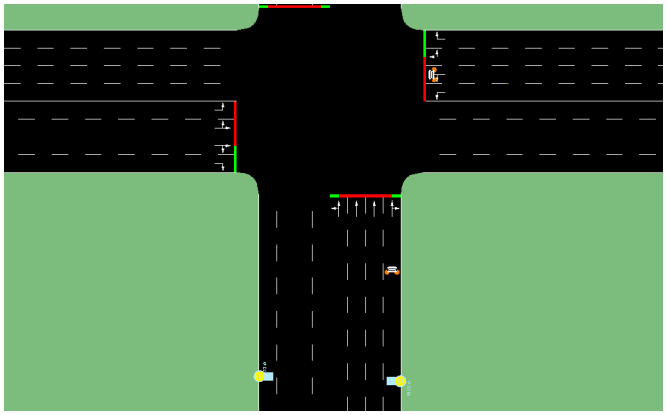

Fig. 2. Road, Vehicles and Charging station

The genetic algorithm which belongs to the heuristic algorithm is suggested to solve the TSP in [12]. The genetic algorithm tries to figure out the balance between exploration and exploitation to obtain the global optimum. [12] applies the genetic algorithm in TSP, and examines the performance of the algorithm when using different selection strategies.

[13] adopted the TSP heuristic algorithm for routing order pickers in warehouses. In warehouses, order picking is the main work and the most important work. The cost of order picking occupies from 55% to 65%. Therefore, it adopted the heuristic algorithms in multi-block configuration.

[14] suggested the benefits of the branch-and-bound algorithm in the NP-Hard problem. It proposes the combinatorial branch-and-bound algorithm based on partial schedules. As this algorithm can construct the lower bound.

III. Architecture

This section discusses the architecture and working flow of our simulation to evaluate the performance of two algorithms. The simulation of this paper manages the electric vehicles that visit randomly assigned destinations. Each electric vehicle sets the order of visiting according to the algorithm and visits the destinations based on the decided order. Each vehicle ends up running after it visits all the destinations, and it records the total running time.

A. SUMO and TraCI

Our simulation utilizes the open-source program Simulation of Urban Mobility (SUMO) [2]. This simulation allows programmers to make road networks with custom features such as the number of lanes, traffic lights, and charging stations. Furthermore, SUMO supports the objects moving around the road network like a pedestrian, vehicles, and public transportation such as trains, buses, and trams. Through this support, SUMO enables the simulation of urban traffic.

TraCI is the python library that enables the SUMO simulation to operate dynamically. Through TraCI, the user can define the object for the simulation and run it in the simulation. For example, a user can write python code that creates the vehicle with dynamic characteristics or according to the condition through TraCI. Users can also record the logs generated while the simulation is running.

B. Simulation Network Architecture

Our simulation's road network consists of edges, nodes, and lanes. Each edge is comprised of lanes and connected through nodes. The network is a 12x12 grid shape and each edge is 8 round-trip lanes. The vehicle can go straight, turn left and turn right according to the traffic light. All nodes have a traffic light and traffic lights continue to repeat patterns including red, green, and yellow.

Vehicles have the characteristics such as id, state, battery capacity, destination, and next charging station id. According to the value of these characteristics, the behavior of the vehicle change. Through vehicle id, simulation can assign the destinations that the vehicle should visit. There are states such as running, charging, and nothing. Each vehicle is an electric vehicle and keeps the battery capacity charged above a specific level. When the battery capacity of the vehicle gets low, the vehicle finds the nearest charging station and forwards to it. The destination is the target location that vehicle needs to forward now. There are several charging stations on the road network. As each charging station can charge one vehicle at once, each charging station manages the vehicles waiting in order.

C. Working Flow

When TraCI code is executed, simulation starts, and python code retrieves the index of the road and charging station. Based on the id of the road, the TraCI code randomly creates vehicles and the destination list for the created vehicles. After that, each destination list is assigned to the vehicle. Each vehicle proceeds based on the determined route. At this point, greedy and branch-and-bound methods operate differently.

For the greedy algorithm, full paths of visiting orders are not determined before the initial departure. Since the greedy algorithm makes the best choice at each step without considering overall cost, it starts running after only calculating the closest destination from the starting point. Then, after arriving

at the destination the vehicle was heading to, it searches for the next destination among the remaining destinations based on the current location. In this way, the vehicle can tour all destinations one by one.

For the branch-and-bound algorithm, full paths are determined before initial departure. Unlike the greedy algorithm, which searches for the next destination when it arrives at each destination, it initially sets all visiting orders and then starts running. Therefore, when delivery at each destination is completed, no additional search time is required for the next destination. After the destination is assigned to each vehicle, all of the required time for a trip between the destinations is calculated. Based on the information, the fastest path can be explored. In this process, DFS and branch-and-bound algorithms are used. Although the DFS algorithm explores all possible cases, the cases that are unlikely to be the fastest route are excluded by stopping the calculation in the middle.

When there is no more destination left to visit, the vehicle will end its operation at that time in both algorithms. Each vehicle records the time consumed from the start of running to the end of delivery. This record is used to compare the time consumed based on the path obtained by two algorithms. Additionally, each vehicle charges its battery when the battery is low at the near charging station. When the determined route is inefficient, the vehicle will travel a longer distance, making the vehicle charge during operation. For this reason, the charging time of the vehicles may also be an indicator of the efficiency of the route. For this purpose, if the vehicle charges, the charging time is also recorded.

Algorithm 1 A Greedy Algorithm

```
function FINDNEXT(from)
    for i to length(D) do
        if distance(from, D[i]) < min then
            min ← distance(from, D[i])
            T ← i
        end if
    end for
end function

T ← 0
D ← DestinationList
S ← StartingPoint
while length(D) > 0 do
    min ← sys.maxsize
    if Not Running then
        FINDNEXT(S)
        RUN()
    else if Running & Arrived then
        REMOVEFROMD(T)
        FINDNEXT(T)
        RUN()
    end if
end while
```

Algorithm 2 A Branch-and-Bound Algorithm

```
function DFS(visited, value, min, final, destNum)
    if min < value then
        return                          ▷ Branch-and-Bound part
    end if
    if length(visited) == length(destNum) then
        final ← visited
        return
    end if
    for i to length(destNum) do
        if i not in visited then
            VISITED.APPEND(i)
            value ← value + distanceTo(i)
            DFS(visited, value, min, final, destNum)
            VISITED.POP()
        end if
    end for
end function
```

IV. Implementation

In this section, detailed implementations of the greedy algorithm and branch-and-bound algorithm are described.

A. Greedy Algorithm

The greedy algorithm does not determine full paths in the first step at departure because it makes the best-looking choices in each step, as mentioned in the section III. Instead, whenever each vehicle arrives at every destination, it searches for the next destination. As shown in Algorithm 1, T is the road ID of the destination where the vehicle is heading right now. D is the list of destinations assigned to each vehicle. S is the starting point where the vehicle starts delivery. The algorithm updates the min initialized with the system's max-size value and searches for the nearest destination through the $FindNext()$. The algorithm is repeated until the vehicle has visited all the assigned destinations. The algorithm is divided into two cases.

The first case is before the vehicle leaves its starting point. Before departure, the vehicle calculates which destination is closest to the current location based on the starting point and destination list. And then, without searching for the second destination to head to after arriving at the first target, the driving begins immediately to the first destination. Therefore, the greedy algorithm does not require much computational time before departure.

When the vehicle arrives at the target destination, the algorithm operates slightly differently from the first case. As the vehicle arrived destination it targeted, it needs to search for the next destination to forward. Through the $RemoveFromD()$ the vehicle deletes the destination that has already arrived from the list. After that, it searches for the closest destination from its current location among the remaining delivery targets by $FindNext()$. This iteration ends when there is no more destination to go to, that is when the destination list is empty.

B. Branch-and-Bound Algorithm

Unlike the greedy algorithm, the branch-and-bound algorithm does not make good-looking choices at each step. However, it includes a mechanism that can reduce computational time while considering the number of all cases. The branch-and-bound algorithm utilizes the Depth-First-Search (DFS) algorithm [15]. The algorithm chooses the method of visiting the last end when calculating the cost of each case. Like the Greedy algorithm, the branch-and-bound algorithm utilizes parameters such as min variables and D lists. Additionally, it uses variables such as $visited$, $final$ lists and $value$, $destNum$. A $visited$ list is for sequentially recording destinations already visited. $final$ contains the path that the $DFS()$ function will eventually return. The $destNum$ means the number of destinations to visit. The $value$ records the amount of time accumulated while searching the route.

Unlike the Greedy algorithm, the branch-and-bound algorithm predetermines all paths to visit in the pre-start phase. Therefore, the calculation process before departure is longer than the greedy algorithm. However, instead, when it arrives at each destination, it can leave for the next destination immediately without any additional calculation steps.

$DFS()$ is a recursive structure. When recursively calling the function, the $value$ is updated to the accumulated amount of time plus the additional amount of time to reach the next destination. The $visited$ adds the next destination and is conveyed as the parameter. If the length of the $visited$ list equals $destNum$, it means that the vehicle considered all destinations. Then it saves the path in $final$ and returns $final$.

Algorithm 2 includes the branch-and-bound mechanism for this process. In $DFS()$ of Algorithm 2, the min value is continuously updated. If the middle-cost value of a particular path is already greater than the min value, that case no longer proceeds. By the conditional statement in the algorithm, the unnecessary calculation may be reduced. The disadvantages of the DFS algorithm, which has a large amount of computation, can be mitigated using these means.

V. Performance Evaluation

This section describes the experiment environment and the performance results of the two algorithms.

A. Experiment

As shown in Figs. 3 and 4, we experimented by evaluating the average running time and average charging time of the vehicles which conducted the delivery based on the route calculated by the two algorithms. The experiment was operated with the same starting point and the same destination list for the vehicles. The only difference is the way that vehicles use to get the shortest route. We analyzed the performance difference according to the number of destinations.

Experiments were carried out for 8 cases, from 3 to 10 destinations. The number of vehicles for each case is fixed as 10 vehicles. The experiment was repeated 10 times for each case, and the average value was compared. In both

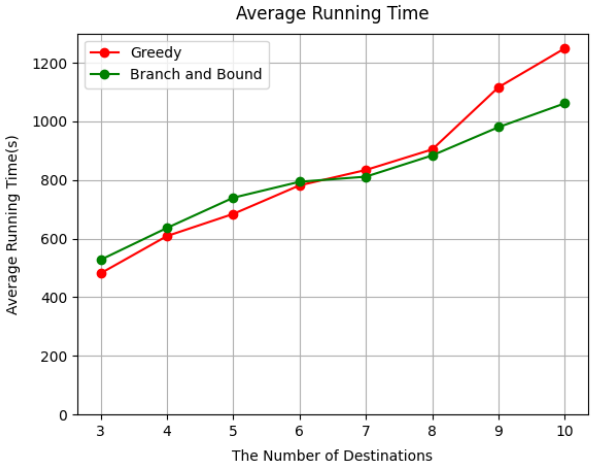

Fig. 3. Average Running Time according to the Number of Destinations

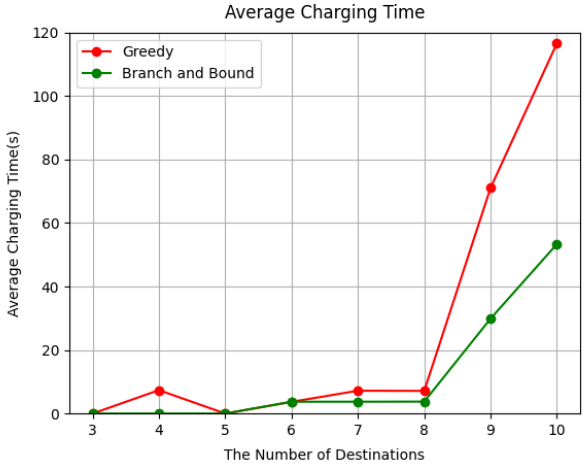

Fig. 4. Average Charging Time according to the Number of Destinations

algorithms, the duration time recorded by the simulation itself was used. At the same time, the time required for charging was also recorded with driving time. The time unit is the simulation's time second. As mentioned in Section III, if the determined path is inefficient, unnecessary driving and battery consumption increase. Then the charging time also increases accordingly. Recording and analyzing the charging time will help compare the efficiency. As for the charging time, the time consumed while the vehicle state was "charging" was recorded.

B. Testing Results

Fig. 3 shows the average running time according to the number of destinations. The horizontal axis represents the number of destinations, and the vertical axis represents the average running time. As shown in Fig 3, as the number of destinations increases, the running time for both algorithms increases. The running time of the greedy algorithm was a little smaller than the branch-and-bound algorithm when the

number of destinations is smaller than 7. However, the running time of the greedy algorithm increased drastically than that of the branch-and-bound algorithm. Then, the greedy algorithm overtook the branch-and-bound algorithm. As the number of destinations increases, the branch-and-bound algorithm's performance got better.

Fig. 4 shows the average charging time according to the number of destinations. The horizontal axis means the number of destinations, as shown in Fig. 3, and the vertical axis represents the average charging time. Both algorithms recorded similar times when the number of destinations is smaller than 7. Due to the small number of destinations, vehicles did not have to charge. As the number of destinations increases, the driving time naturally increases too. Therefore, the charging time increases regardless of the efficiency of the route. However, when the number of destinations was over 7, the charging time of the greedy algorithm got much bigger than the branch-and-bound algorithm. As a result, the difference in charging time was 41.1 and 63.2 seconds for each 7 and 9 case. The difference in route efficiency between the two algorithms confirmed in the running time analysis had also an effect on the charging time analysis.

VI. CONCLUSION

This paper proposed an algorithm that calculates the most efficient route when delivery vehicles have to visit multiple destinations. The performance was measured through the average running time and charging time according to the number of parcel delivery destinations. As a result of the experiment, the greedy algorithm resulted in bigger values than the branch-and-bound algorithm in both average running time and charging time. We found out that as the number of destinations increased, the shortest path by the branch-and-bound algorithm was faster than the one by the greedy algorithm.

While performing experiments in this paper, it was difficult to set the route for the branch-and-bound algorithm. As the number of destinations increased, the number of cases to be calculated also increased. As a result, it took considerable time to calculate it, even though the branch-and-bound mechanism excluded cases that cannot be a correct answer. Whereas the greedy algorithm only needs to take into account the small number of cases because it searches for the next target when it arrives at each destination.

If branch-and-bound algorithms get applied to real industries, larger computational power will be needed. In future work, we will research how to further reduce the amount of calculation. Based on the result of this paper, we can find out a more efficient way to reduce the calculation time. In addition, we are going to research the Graph Neural Network (GNN) to enhance the parcel delivery service by utilizing collected travel-time data in navigation systems.

ACKNOWLEDGMENTS

This work was supported by the Ministry of Science and ICT (MSIT), Korea, under the Information Technology Research Center (ITRC) support program (IITP-2022-2017-0-01633) supervised by the Institute for Information & Communications Technology Planning & Evaluation (IITP). This work was supported in part by the IITP grant funded by the Korea MSIT (No. 2022-0-01199, Regional strategic industry convergence security core talent training business). Note that Jaehoon (Paul) Jeong is the corresponding author.

REFERENCES

[1] J. Cho, B. Ahn, K. Hong, and I. Cheong, "Issues of digital trade rules and implications for korea in the post covid-19 world," *Journal of International Logistics and Trade*, vol. 18, no. 3, pp. 137–147, 2020.

[2] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of sumo-simulation of urban mobility," *International journal on advances in systems and measurements*, vol. 5, no. 3&4, 2012.

[3] D. Krajzewicz, "Traffic simulation with sumo–simulation of urban mobility," in *Fundamentals of traffic simulation*. Springer, 2010, pp. 269–293.

[4] C. Imielińska, B. Kalantari, and L. Khachiyan, "A greedy heuristic for a minimum-weight forest problem," *Operations Research Letters*, vol. 14, no. 2, pp. 65–71, 1993. [Online]. Available: https://www.sciencedirect.com/science/article/pii/016763779390097Z

[5] B. Alidaee, G. A. Kochenberger, and M. M. Amini, "Greedy solutions of selection and ordering problems," *European Journal of Operational Research*, vol. 134, no. 1, pp. 203–215, 2001.

[6] M. E. Pfetsch, "Branch-and-cut for the maximum feasible subsystem problem," *SIAM Journal on Optimization*, vol. 19, no. 1, pp. 21–38, 2008.

[7] Y. Asahina and J. Yang, "Death from overwork in a time of pandemic: How delivery work became a locus of public debate in south korea," *Journal of Contemporary Asia*, pp. 1–18, 2022.

[8] M. Anshari, Q. Sakalayen, M. Fithriyah, and S. A. Lim, "Decision aid in logistics during covid-19 induced disruption and significance of a digital ecosystem," in *2022 International Conference on Decision Aid Sciences and Applications (DASA)*, 2022, pp. 1371–1377.

[9] G. A. Croes, "A method for solving traveling-salesman problems," *Operations research*, vol. 6, no. 6, pp. 791–812, 1958.

[10] D. S. Hochba, "Approximation algorithms for np-hard problems," *ACM Sigact News*, vol. 28, no. 2, pp. 40–52, 1997.

[11] W. Zhang, "Truncated branch-and-bound: A case study on the asymmetric tsp," in *Proc. Of AAAI 1993 Spring Symposium on AI and NP-hard problems*, vol. 160166, 1993.

[12] N. M. Razali, J. Geraghty *et al.*, "Genetic algorithm performance with different selection strategies in solving tsp," in *Proceedings of the world congress on engineering*, vol. 2, no. 1. International Association of Engineers Hong Kong, China, 2011, pp. 1–6.

[13] C. Theys, O. Bräysy, W. Dullaert, and B. Raa, "Using a tsp heuristic for routing order pickers in warehouses," *European Journal of Operational Research*, vol. 200, no. 3, pp. 755–763, 2010.

[14] J. Rambau and C. Schwarz, "On the benefits of using np-hard problems in branch & bound," in *Operations research proceedings 2008*. Springer, 2009, pp. 463–468.

[15] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.