Dynamic Car-Passenger Matching based on Tabu Search using Global Optimization with Time Windows

Marvin Erdmann
Fleet Management and Intelligence
BMW AG
Munich, Germany
marvin.erdmann@bmw.de

Florian Dandl
Intelligent Transportation Systems
Bundeswehr University Munich
Munich, Germany
florian.dandl@unibw.de

Prof. Dr. Klaus Bogenberger
Intelligent Transportation Systems
Bundeswehr University Munich
Munich, Germany
klaus.bogenberger@unibw.de

***Abstract*—On-Demand Mobility is an increasingly popular concept especially in urban areas, which has the potential to reduce congestion and the space needed by privately owned vehicles due to shared car fleets. To avoid a decline of flexibility and convenience for the customers and to minimize the costs for the service provider, a fleet management algorithm matches the requests and the vehicles in order to quickly find a reliable and time efficient solution for the whole system. The focus of this work is to introduce a new approach to find solutions periodically using a Tabu Search metaheuristic, called Global Optimization with Time Windows. It is shown that this method allows significantly better solutions compared to those found by the Nearest Neighbor Policy, without losing the ability to quickly inform customers about their pick-up time.**

***Index Terms*—Discrete Optimization, Metaheuristics, Tabu Search, Dial-a-Ride-Problem**

I. Introduction

Today's challenges of urban traffic could be met by On-Demand Mobility (ODM), a concept that would lead to an enhanced use of shared mobility services utilizing vehicles more efficiently. This trend becomes even more substantial with the development of autonomous vehicles. To operate with an increasing number of cars and customers, a fleet management algorithm is needed that matches service requests and vehicles of the fleet. The goal is to quickly find a reliable and time efficient solution for the whole system. This kind of problem is called Dial-a-Ride-Problem (DaRP) [1]. This work focuses on a use case referred to as ride-hailing in which at most one customer occupies a car at any time and customers are willing to be picked up as soon as possible.

This kind of optimization problem has received increasing attention in research. The approach of the Nearest Neighbor Policy (NNP) is supposed to be fast and robust [2]. NNP searches for the shortest distances between all available cars and the pick-up locations of all known requests. Hence, the computational effort for this approach is very low. On the other hand, it only provides the shortest waiting times for individual customers instead of an optimal solution for the whole system, which potentially increases the waiting times for later requests as well as the fleet costs.

There are studies using metaheuristics in a dynamic way by re-optimizing the current solution periodically, as it is done in this work. However, these approaches are considered as not well suited for the use case mentioned above for different reasons. In [3], all requests that had not been served before the optimization are taken into account, so customers would not know when they will be picked-up until shortly before the pick-up. Another approach was used in [4] as well as in four out of six assignment strategies proposed in [5], where after every optimization the matches are supposed to be permanent. Though, customers and vehicles of the ODM fleet have to wait for the optimization to finish until they receive information, which causes longer average waiting times and inconvenience.

The objective of this work is to present an approach which is considering only new requests coming in during a given time interval to find a new solution periodically for the ride-hailing use case while being able to inform the customer quickly about the time window he or she will be picked up in. Since the computational time to find such solutions rises exponentially with the problem size, a metaheuristic called Tabu Search (TS) is used. It is indicated that this method produces significantly better solutions than the NNP in a reasonable amount of computational time and that the difference in solution quality increases with rising problem size.

The remainder of this work is structured as follows. First, a short formulation of the DaRP (II) is given. Then, the used methods and the examined model parameters are explained (III). After that, results are presented and discussed (IV, V).

II. Problem Definition

This work adopts aspects of a well-established notation previously used in e.g. [6] to describe the DaRP. The number of all requests known at the moment of optimization is represented by n, the number of all vehicles is given by m. Each customer is associated with a pick-up location $i \in \{1, ..., n\}$ and a drop-off location $i + n$. The sets of all pick-up and drop-off locations are denoted by P and D, respectively. For all cars $v \in \{2n + 1, ..., 2n + m\}$ the set C represents the actual location (idle vehicles) or the drop-off location of the

last matched request of a vehicle at the time of optimization. The set of $P \cup D \cup C$ is represented by V, which is the set of all vertices of an undirected graph $G = (V, A)$. The set of all arcs A contains all possible pairs of different locations (a, b), where $a \in C$ and $b \in P$, $a \in P$ and $b \in D$ or $a \in D$ and $b \in P$). If in a solution a car v goes from a to b, the decision variable $x^v_{a,b} = 1$, otherwise it is zero. Every arc is weighted with a cost $c_{a,b}$, depending on the distance between the two locations a and b.

So, the costs of a solution are given by:

$$T_1 = \sum_{v \in C} \sum_{a \in V} \sum_{b \in V} c_{a,b}\, x^v_{a,b}. \tag{1}$$

The DaRP is to find a global minimum of an objective function f_{obj}. To define f_{obj} it is necessary to take into account the customer dissatisfaction as well. The unit of dissatisfaction is supposed to be the same as the unit of costs $c_{a,b}$. The overall dissatisfaction is:

$$T_2 = \sum_{i \in P} d_i. \tag{2}$$

In the proposed model, the dissatisfaction penalty d_i of a customer with a pick-up location i is only depending on the waiting time Δt between the time of request $t_{i,r}$ and the time of pick-up $t_{i,p}$. In the model used here, the dissatisfaction of a customer grows piece-wise linear as in [7], outlined in Fig. 1.

This function penalizes longer waiting times significantly stronger than shorter ones and therefore avoids very long individual waiting times without imposing hard time windows in which customers must be picked up in. That effect is used intentionally to give preference to solutions in which the rate of rejections of the matches by the customers in real-world ODM applications would be small, without decreasing the solution space by allowing rejections of requests in the algorithm used in this work.

The objective function is the sum of the costs for all cars to serve all requests and the resulting dissatisfaction for all customers:

$$f_{obj} = T_1 + T_2 \tag{3}$$

The solution found by minimizing f_{obj} must also satisfy the following constraints:

$$\sum_{v \in C} x^v_{i,i+n} = 1 \qquad \forall i \in P \tag{4}$$

$$\sum_{i \in P} x^v_{i,i+n} - \sum_{i,j \in P} x^v_{i+n,j} = 0 \quad \forall v \in C \tag{5}$$

$$t_{i,d} - t_{i,p} > 0 \qquad \forall i \in P \tag{6}$$

The first constraint in (4) is equivalent to the statement that every request must be matched exactly once. It also makes sure that no pooling is considered as every pick-up of a customer is followed by his or her drop-off. Equation (5) guarantees that every customer's pick-up and drop-off is performed by the same car and (6) states that each customer's pick-up at $t_{i,p}$ must take place before the respective drop-off time $t_{i,d}$.

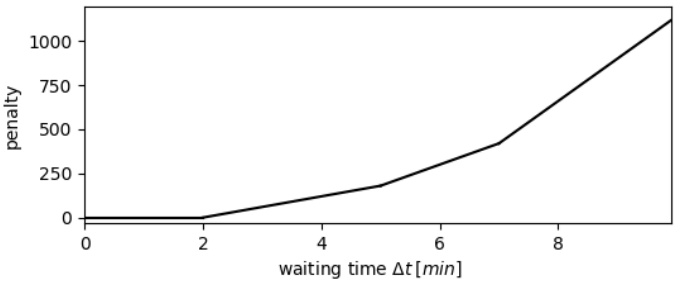

Fig. 1. Outline of the dissatisfaction penalty. In this work, if $\Delta t \leq 2$ min the dissatisfaction is zero. After that, until $\Delta t = 5$ min it rises with a rate of 1 per second. Then, the slope increases to 2 per second until $\Delta t = 7$ min after which it becomes 4 per second.

III. Methodology

In this section, a short introduction to the concept of Tabu Search is given as well as a deeper insight into the proposed method of Global Optimization with Time Windows (GOTW).

A. Search

To find solutions for the DaRP introduced in II, in this work a version of the metaheuristic approach called Tabu Search (TS) is used. The main idea of this well-established technique is to avoid getting stuck in local optima by allowing solutions which are worse with respect to the value of the objective function while forbidding solutions that were already found during the search.

The terminology to describe the TS metaheuristic in this work is used in e.g. [8]. The initial solution at the beginning of each optimization is produced by a NNP algorithm. Then, the first iteration of optimization starts. By applying local transformations (or local moves) to the current solution, a neighborhood is defined. This neighborhood is used to find new solutions. In this work, three different types of transformations are applied: *swaps*, *shifts* and *interchanges*. The *swap* switches requests within one car's route, while the *shift* moves the last request of one car's route to the end of another car's route. The *interchange* transforms the solution by exchanging requests of two different cars' routes.

All solutions in the neighborhood are ordered by their objective function value f_{obj}. A so called Tabu List is checked, which indicates whether a solution was recently visited and is therefore not allowed to be the new initial solution. If not already on it, the best neighboring solution (BNS) is accepted and added to the Tabu List. Accepted solutions are then compared to the best found solution (BFS), which is the initial solution in the first iteration. If the objective function value of the accepted solution $f_{obj,BNS}$ is smaller than the objective function value of the best found solution $f_{obj,BFS}$, this BNS becomes the new BFS. The Tabu List has a maximum number of entries, in this work this limit is set to 10. When this maximum number of entries is reached, the oldest entry is replaced by the latest added solution.

If the BNS is already on the Tabu List, it is not accepted and the next one on the ordered list of all neighbors is checked

until (a) a solution is found which is not on the Tabu List, (b) all neighboring solutions are checked or (c) a maximum number of already found solutions is checked, which in this work is equal to the maximum number of entries in the Tabu List. If a solution is accepted, a new iteration begins with this solution as initial solution. If either (b) or (c) is the case, the search terminates.

This procedure is executed iteratively until it either breaks up or until a maximum number of iterations is reached, here this number is set to 100. That limit of iterations (as well as the relatively small maximum number of entries in the Tabu List) is chosen to focus on a fast search procedure, which is reasonable since the initial solution is not produced randomly but is expected to be in a promising area of the solution space. When the search ends, the BFS is the resulting solution of the Tabu Search.

B. Global Optimization with Time Windows

The main focus of this work is to introduce the method of GOTW. This approach combines the advantages of global optimization and fast insertion algorithms.

An initial problem is solved with all requests known at that point in time. An empty list is set up—called *requests to optimize*—that will contain all requests subject to global optimization at the end of the GOTW period. As depicted in Fig. 2, the next time step is simulated after that. The algorithm checks whether new requests are submitted and added to the problem at that point in time. If that is the case, a simple algorithm checks every car's location and time when it has or will have delivered its last matched request. The algorithm searches for the car that could pick up a new customer the earliest, following the NNP. A tentative match is defined and a time window with a predefined length is set in which this customer must be picked up. This time window can be forwarded to the customer as soon as the NNP algorithm found the tentative match. After adding all new requests to the *requests to optimize*, the algorithm checks whether pick-up locations of tentatively matched customers are reached. If that is true, these requests are matched permanently by adding them to the current solution and they are deleted from the list of *requests to optimize*.

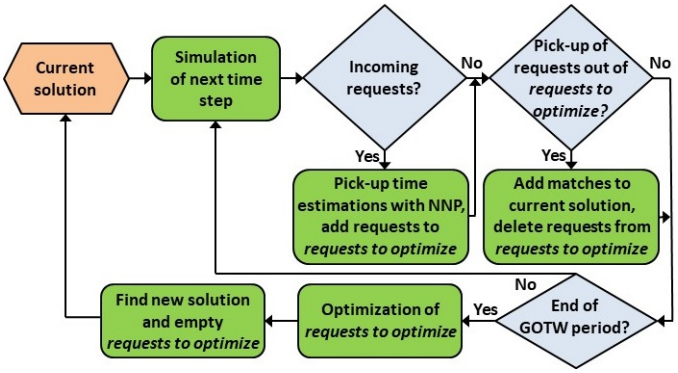

Fig. 2. Principle of the GOTW.

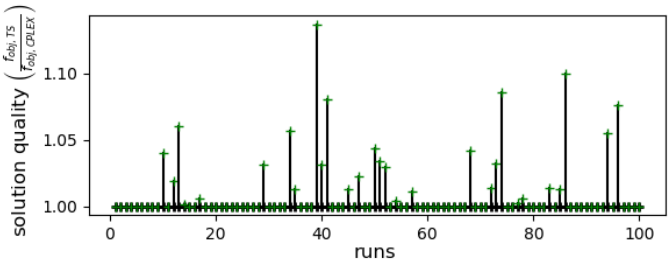

Fig. 3. Comparison of qualities of solutions found with TS and the global optimum for a static problem with 4 vehicles and 8 customers.

The simulation of consecutive time steps is executed until the end of the GOTW period. Then, all customers that have not been picked up yet and are therefore still on the list of *requests to optimize* are subject to a global optimization using the TS algorithm described in III-A. The found solution will be considered as unchangeable. Every subsequent optimization will consider only requests that are submitted later. Hence, the algorithm allows the customer to know exactly when he or she will be picked up at the end of a GOTW period. With this solution, a new optimization period is initiated. This procedure is performed periodically until the end of simulation.

IV. Results

In this section, the proposed algorithm is examined and results are presented. First, the Tabu Search metaheuristic is benchmarked by comparing the found results with the global optimum in static instances of the problem. After that, the GOTW is tested by comparing it to the NNP approach in three different parameter settings.

A. Comparison of the Tabu Search metaheuristic and an exact optimization algorithm

The implemented Tabu Search metaheuristic is compared to the optimization algorithm CPLEX from IBM [9] to validate the quality of the found solutions. Since finding the global optimum for the examined numbers of cars and requests per optimization period is very expensive from a computational time point of view, this validation is not done simultaneously to the dynamic search procedure. Instead, static instances are tested to illustrate the general ability of the TS algorithm to produce solutions close to the global optimum.

Fig. 3 shows the value of the objective function of the best solution found with the TS algorithm $f_{obj,TS}$ relative to the value of the objective function of the globally best solution of CPLEX $f_{obj,CPLEX}$ in a setting of four cars and eight customers randomly generated over a total number of 100 instances. The average difference in quality is $1.07\,\%$ (NNP/CPLEX: $17.66\,\%$) with a 95%-confidence interval of $[0.58\,\%; 1.56\,\%]$ ($[14.87\,\%; 20.45\,\%]$). In 71 instances the Tabu Search found the global optimum.

TABLE I
DEFINITION OF THE PARAMETER SETS I,II AND III

parameter	**set I**	**set II**	**set III**
number of cars	4	15	30
lenght of GOTW period	19.2 s (avg.)	30 s	45 s
customers per period	8	12.5 (avg.)	18.75 (avg.)
pick-up time windows	120 s	300 s	300 s
estimated problem size	2×10^2	5×10^3	2×10^4

B. Comparison of GOTW and NNP approaches with varying parameter settings

To test the algorithm, three different parameter sets are compared on a generic x-y-grid (Table I). Each set of parameters is tested in 100 instances. The only degree of freedom left is the initial distribution of the vehicles. New requests are generated with the New York Taxi Data Set [10] from June, 30th 2016 between 5:30 p.m. and 6:00 p.m (10 % of all requests are taken into account). This date is chosen because it is the most recent day at which GPS coordinates for pick-up and drop-off locations are available for this data set.

To produce comparable results, each simulation is run twice with the same instance of parameters and starting positions of the cars: one using GOTW, the other one following a NNP algorithm. To generate a meaningful number of instances, the qualities of the solutions produced by both methods are compared after 30 minutes of simulation with one second time steps on two Intel Core i5-6300U 2.4GHz processors.

The size of the solution spaces are estimated by the average numbers of generated neighboring solutions per GOTW period. The goal is to show an increasing difference in the quality of solutions found with the GOTW and the NNP approach with increasing problem size while being able to perform the global optimization fast enough to be executed live, i.e. the computational time needed to optimize the solution is shorter than the period in which new requests come in.

A parameter tested in this work is the number of available vehicles. Obviously, the solution space grows with an increasing number of cars which can pick up customers. Another parameter is the length of GOTW periods and the corresponding number of new requests in these periods. The longer such a time interval is, the more requests are subject to optimization at the end of it. Hence, the number of possible solutions is expected to be higher. Though, as in the case of more cars, longer GOTW periods will lead to longer computational times per optimization. The last examined parameter is the length of time windows customers are allowed to be picked up in, before and after the expected pick-up time. With longer time windows more solutions are feasible per customer.

Parameter set I is chosen to be examined with the same number of cars and customers (per period of optimization) that is used in the validation with the CPLEX algorithm for the static case in IV-A. That is the reason why each GOTW period ends after exactly eight new requests are submitted, in contrast to the other parameter sets in which the termination of each GOTW period is triggered by a given time as it is meant to work in real-world applications. The average quality of the TS solution compared to NNP after 30 minutes of simulation in 100 runs is $0.90\,\%$ (95%-confidence interval:$[0.69\,\%; 1.13\,\%]$) higher, which means that the objective function value using GOTW $f_{obj,GOTW}$ is lower by the same percentage on average. There are cases in which the optimized solution leads to longer waiting times for customers that are not known at the time of global optimization, compared to the solution found with NNP. Because of this effect of randomness, in some instances the solution found with the NNP is better than the one using the GOTW approach. Each optimization is performed in around one second on average and therefore can be performed live.

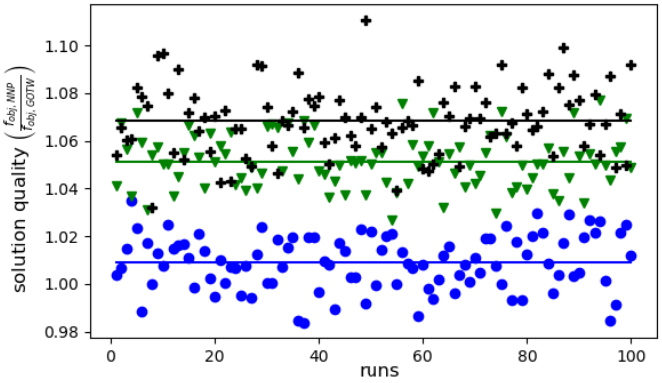

Fig. 4. Qualities of solutions found with TS relative to the simple insertion algorithm following NNP in 100 runs. Parameter sets I (blue dots), II (green triangles) and III (black crosses).

In parameter set II, the number of cars is increased to 15. Each GOTW period's length is set to 30 seconds, in which an average number of 12.5 requests are submitted. In 100 instances of simulation, the proposed method performs $5.15\,\%$ ($[4.91\,\%; 5.36\,\%]$) better than the insertion procedure following the NNP. Although the problem size increases, the average computational time is shorter than an optimization period with 6.8 seconds, allowing a live performance.

Finally, in parameter set III, the problem size is enlarged to 30 cars and the GOTW period's length is set to 45 seconds, which leads to 18.75 requests per period. The increase of performance quality by the global optimization process throughout 100 runs is $6.85\,\%$ ($[6.56\,\%; 7.16\,\%]$) on average. The average computational time per optimization period is 35.7 seconds, therefore optimization can also be executed within the given limitation of the GOTW period.

In Fig. 4, the solution qualities of all parameter sets are depicted. Resulting benefits of the GOTW method compared to the NNP approach are shown in relation to the respective problem sizes in Fig. 5. Both results are discussed in the following section.

V. DISCUSSION

The results produced with GOTW show a clear improvement compared to solutions found with NNP, which is an

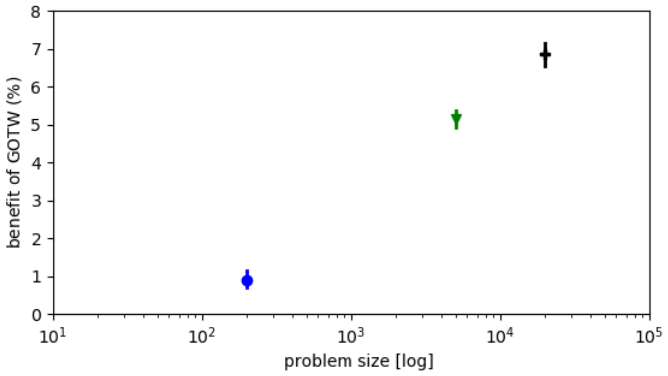

Fig. 5. Benefits of parameter sets I (blue dot), II (green triangle) and III (black line) in dependency to the respective problem size with their respective 95%-confidence intervals.

often used standard in the considered use case. Comparing the evaluated parameter sets, a trend stands out: with increasing problem size, the global optimization procedure becomes more beneficial relative to the simple insertion method. Furthermore, the Tabu Search Algorithm is able to push the solution quality as close as $1.07\,\%$ to the global optimum in a reasonable amount of time for small problem instances.

In general, that allows the usage of Global Optimization with Time Windows in dynamic use cases, such as the aforementioned ODM fleet management. Especially, this approach is beneficial if reservations of scheduled rides are allowed additionally. The requirements of these use cases match very well with the strength of the proposed algorithm, which is the ability to find near-to-optimal solutions in big solution spaces while being able to inform the customers about their pick-up times very fast.

There are many aspects that are not discussed in this work. Deeper examinations of the penalty function as well as the isolated effects of parameters such as the original car positions at the beginning of the simulations would clarify their impact on the quality of the search. A generic grid is used instead of a real-world map of Manhattan, which does not consider real-time traffic or varying average velocities. A comparison to the global optimum is performed only for small problem instances due to the exhausting computational effort needed for greater sizes.

To generate a meaningful number of tested instances and to improve the comprehensibility of the correlations in the proposed model, the problem sizes of the tested parameter sets are chosen to be small. Therefore, only every tenth request of the data set is taken into account and the number of cars is drastically decreased compared to the number of taxis needed to cover the ODM demand in New York. However, adjusting the numbers of cars and requests and the lengths of GOTW periods to the average time needed to serve a request will only lead to shorter individual waiting times but is not supposed to have any considerable negative impact on the effectiveness of the proposed algorithm. Instead, the results indicate that with growing problem size the benefit of GOTW increases.

VI. Conclusion

In this work, the objective was to introduce a dynamic approach for the Dial-a-Ride Problem in the context of the ride-hailing use case. Its performance was tested in different parameter settings to evaluate the usability in real-world applications. The concept of this approach is to gather new requests over a period of time and to find an optimized solution after each of these periods. It is based on the Tabu Search metaheuristic in order to find near-to-optimal solutions much faster than an exact optimization algorithm would be able to.

The methodology and results for three different parameter sets—each representing different areas in the parameter space defined by the number of cars, the number of requests per optimization period and the length of the pick-up time windows—were presented. The results illustrated a clear trend: with increasing problem size, the proposed algorithm was able to find increasingly better solutions compared to the simple insertion algorithm following a Nearest Neighbor Policy. The computational time needed for each optimization was short enough for all considered problem sizes to be performed within the respective periods.

Those results are very promising, since they show the capability of this approach to be used in real-world On-Demand Mobility fleet management systems. Future research should concentrate on an improved model that also considers reservations of rides and car pooling. The overall problem size should be increased to come closer to the numbers of the real-world applications with hundreds of requests per minute and thousands of vehicles in cities like New York. Moreover, the quality of the TS metaheuristic must be tested with bigger problem sizes by comparing it to exact optimization algorithms as well.

References

[1] H. N. Psaraftis, “A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem,” Transportation Science, vol. 14, no. 2, pp. 130–154, May 1980.

[2] A. Larsen, O. Madsen, and M. Salomon, “Partially dynamic vehicle routing—models and algorithms,” Journal of the Operational Research Society, vol. 53, pp. 637–646, June 2002.

[3] J. Alonso-Mora and E. Frazzoli, ‘On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment,” Proceedings of the National Academy of Sciences, vol. 114, no. 3, pp. 462–467, January 2017.

[4] F. Dandl and K. Bogenberger, “Booking Processes in Autonomous Car-sharing and Taxi Systems,” Proceedings of 7th Transportation Research Arena, April 2018.

[5] M. Hyland, H. S. Mahmassani, “Dynamic autonomous vehicle fleet operations: optimization-based strategies to assign AVs to immediate traveler demand requests,” Transportation Research Part C, vol. 92, pp. 278–297, July 2018.

[6] J.-F. Cordeau, “A branch-and-cut algorithm for the dial-a-ride problem,” Operations Research, vol. 54, no. 3, pp. 573–586, Jun 2006.

[7] K. B. Bergvinsdottir, J. Larsen, and R. M. Jorgensen, “Solving the dial-a-ride problem using genetic algorithms,” Journal of the Operational Research Society, vol. 58, no. 10, pp. 1321–1331, October 2007.

[8] M. Gendreau and J.-Y. Potvin, “Tabu Search,” in Handbook of Metaheuristics, 2nd ed., M. Gendreau, J.-Y. Potvin, Eds. New York: Springer US, 2010, pp. 41–59.

[9] IBM, “CPLEX Optimizer,” 2017, [Online]. Available: www.ibm.com/analytics/cplex-optimizer. [Accessed: 30- Nov- 2018].

[10] NYC Taxi and Limousine Commision, “TLC Trip Record Data,” 2016, [Online]. Available: www.nyc.gov/html/tlc/html/about/trip_record_data.shtml. [Accessed: 30- Nov- 2018].