SIMULATED ANNEALING BASED OPTIMIZED DRIVER SCHEDULING FOR VEHICLE DELIVERY

Shashika Ranga Muramudalige

(158041K)

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

May 2018

SIMULATED ANNEALING BASED OPTIMIZED DRIVER SCHEDULING FOR VEHICLE DELIVERY

Shashika Ranga Muramudalige

(158041K)

Thesis submitted in partial fulfillment of the requirements for the degree Master of Science in Computer Science and Engineering

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

May 2018

Declaration, copyright statement and the statement of the supervisor

"I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books)."

Signature:	Date:

The above candidate has carried out research for the Masters thesis under our supervision.

Name of the supervisor:	Dr. HMN Dilum Bandara
Signature of the supervisor:	Date:

Name of the supervisor:	Eng. Nishal Samarasekara
Signature of the supervisor:	Date:

Abstract

Simulated annealing based optimized driver scheduling for vehicle delivery

Vehicle delivery is a major business where third-party drivers are hired to deliver vehicles when they are relocated, sold, or while returning rental cars. This is motivated due to the busy schedule of individuals and companies, convince, and cost saving. A vehicle delivery company typically operates in a chosen geography varying from a region of a country to a set of countries that are nearby. Hence, the drivers are also geographically dispersed. This is a complicated process due to the wide variation in collection/delivery locations, driver availability, time bounds, types of vehicles, special skills required by drivers, and impact due to traffic and weather. Currently the process is manipulated manually by a scheduling manager who creates next day's schedule at the end of the working day based on the jobs received. However, as the number of jobs and drivers increase, it is hard to decide on the most appropriate driver for the job such that both the customer and company goals are optimally satisfied. We propose an automated driver scheduling solution to maximize the number of vehicle deliveries and customer satisfaction while minimizing the delivery cost and distributing driver income based on their availability. Proposed solution consists of a rule checker and a scheduler. Rule checker enforces constraints such as deadlines, vehicle types, license types, skills, and working hours. Scheduler uses simulated annealing to assign as many jobs as possible while minimizing the overall cost. Using a workload derived from an actual vehicle delivery company, we demonstrate that the proposed solution has good coverage of jobs while minimizing the cost and equitably distributing the income among drivers based on their availability. Moreover, the proposed solution has the flexibility to tolerate exceptions due to breakdowns, excessive traffic, and bad weather without a considerable impact on the majority of the already scheduled jobs.

Keywords: Optimization, Scheduling, Simulated Annealing, Vehicle Delivery

Dedication

I dedicate my thesis work to my family and many friends. A special feeling of gratitude to my loving parents, Sarathchandra and Aruna Muramudalige whose words of encouragement and push for tenacity ring in my ears. My sister Dulari Muramudalige and my wife Sonali Muthukumarana have never left my side and are very special.

I also dedicate this thesis to my many friends and colleagues who have supported me throughout the process. I will always appreciate all they have done, especially for helping me develop my technology skills.

Acknowledgment

I wish to thank my evaluation panel members who were more than generous with their expertise and precious time. A special thanks to Dr. Dilum Bandara, my research supervisor for his countless hours of reflecting, reading, encouraging, and most of all patients throughout the entire process. Thank you Dr. Shehan Perera, and Mr. Afkam Azeez for agreeing to serve on my evaluation panel and Eng. Nishal Samarasekara for agreeing to serve as my external supervisor.

I would like to acknowledge and thank Department of Computer Science and Engineering, University of Moratuwa, for allowing me to conduct my research and providing any assistance requested. Special thanks go to both academic and non-academic staff of the department for their continued support. I also gratitude to the University of Moratuwa for the financial support as the research was supported in part by the Senate Research Grant of the University of Moratuwa under award number SRC/LT/2016/14.

Finally, I would like to thank the teachers, evaluators and colleagues that assisted me with this project. Their excitement and willingness to provide feedback made the completion of this research an enjoyable experience.

Table of Content

Declaration, copyright statement and the statement of the supervisor iii
Abstract iv
Dedication v
Acknowledgement vi
List of Figures ix
List of Tables x
List of Abbreviations xi
1. INTRODUCTION 1
1.1 Motivation 1
1.2 Problem Statement 2
1.3 Objectives 3
1.4 Outline 3
2. LITERATURE REVIEW 4
2.1 Driver and vehicle scheduling in Limousine rental 4
2.2 Column generation based hyper-heuristic solution for bus-driver scheduling 7
2.3 Artificial neural systems for delivery truck scheduling 10
2.4 RMC truck dispatching using machine-learning techniques 11
2.5 Truck driver scheduling problem 13
2.6 Summary 14
3. PROBLEM FORMULATION 16
3.1 Constraints 16
3.2 Objectives 19
4. PROPOSED SOLUTION 21
4.1 Rule checker 21
4.2 Job scheduler 23
5. PERFORMANCE ANALYSIS 25
5.1 Workload Creation 25
5.2 Results 28
5.2.1 Comparison with other algorithms 32
5.2.2 Effects of the unavoidable delays and issues 37
5.2.3 Effects of public transportation use of drivers 40
5.2.4 Income distribution of drivers 42
6. SUMMARY AND FURURE WORK 44

6.1 Conclusion 44
6.2 Future work 45
References 47

List of Figures

Figure 4.1. Solution model for rule checker. 22
Figure 5.1. (a) Job distribution of dataset 1 and (b) Job distribution of dataset 2. 26
Figure 5.2. Driver distribution. 27
Figure 5.3. Job coverage against total driver available hours in each day. 31
Figure 5.4. Job coverage against different cooling rates. 32
Figure 5.5. Pseudo code of initial solution. 33
Figure 5.6. Pseudo code of enhanced hill climbing algorithm. 33
Figure 5.7. Job coverage against various algorithms. 36
Figure 5.8. Job coverage against various algorithms min-max range. 37
Figure 5.9. Impact of delayed jobs against different delays. 39
Figure 5.10. Impact of delayed jobs against different delays by regenerating the solution. 39
Figure 5.11. Job coverage against different travel time factors. 41
Figure 5.12. Profit against different travel cost factors. 41
Figure 5.13. Weekly average of driver availability and income with ±1H time window for dataset 42
Figure 5.14. Weekly income/availability ratio with ±1H time window for dataset 1. 43
Figure 5.15. Weekly average of driver availability and income with ±1H time window for dataset 43
Figure 5.16. Weekly income/availability ratio with ±1H time window for dataset 43

List of Tables

Table 2.1. Driver related symbols. 17

Table 2.2. Job related symbols. 17

Table 2.3. Solution related symbols. 17

Table 5.1. Driver availability by day. 25

Table 5.2. SA Acceptance rates against initial temperatures with 0.003 cooling rate. 28

Table 5.3. Results against different time windows on Monday for dataset 1 and 2 with 0.003 cooling rate. 29

Table 5.4. Results against different time windows across a week. 30

Table 5.5. Results against different time windows on Monday for dataset 1 and 2 with 0.03 cooling rate. 31

Table 5.6. Results against different time windows on Monday for dataset 1 and 2 with hill climbing algorithm. 34

Table 5.7. Results against different time windows on Monday for dataset 1 and 2 with initial solution. 34

Table 5.8. Results against different time windows across a week with hill climbing algorithm. 35

Table 5.9. Min, max, average results against different algorithms on Monday for dataset 1. 36

Table 5.10. Impact of 5% of delayed jobs for dataset 1 and 2. 38

Table 5.11. Impact of 5% of delayed jobs for dataset 1 and 2 with regenerating the solution. 38

Table 5.12. Impact of 10% of delayed jobs for dataset 1 and 2 with regenerating the solution. 38

Table 5.13. Results against different travel time factors on Monday for dataset 1 and 2 with ±1H time window. 40

Table 5.14. Results against different public transportation cost factors on Monday for dataset 1 and 2 with ±1H time window. 40

List of Abbreviations

ANN	Artificial neural network
ANS	Artificial Neural Systems
BDSP	Bus Driver Scheduling Problem
DA	Dispatch Area
GPS	Global Positioning System
HC	Hill Climbing
IBK	Naive Bayes classifier
ILP	Integer Linear Programming
J48	Decision tree
ML	Machine Learning
NB	K nearest neighbor
PART	Rule based algorithm
RMC	Ready Mix Concrete
RMP	Restricted Master Problem
SA	Simulated Annealing
SMO	Support vector machine
TDSP	Truck Driver Scheduling Problem
VD	Vehicle Delivery

1. INTRODUCTION

While taxis and rental cars are popular services all around the world, there are other forms of business-to-business services such as Vehicle Delivery (VD) using third-party drivers. When a customer wants to move a vehicle from one place to another, e.g., due to a sale or to return a rental car, he/she inquires a VD service to identify a suitable driver and a schedule. This is motivated due to the busy schedule of individuals and companies, convince, and cost saving. A VD company typically operates in a chosen geography varying from a region of a country to a set of nearby countries. Hence, the drivers are also geographically dispersed. The VD company needs to allocate an inquiry to the most suitable driver based on a set of parameters such as vehicle collection and delivery location and time, type of vehicle, driver's location and availability, labor laws, traffic, and weather constraints. Moreover, driver allocation should focus on increasing customer satisfaction, operational efficiency, balanced driver income, and company profit. This is a dynamic environment where the schedules may change or even canceled due to reasons such as customer changing the pickup/delivery time, canceling a job, the arrival of a last-minute job from a high-priority customer, vehicle breakdown or accident, excessive traffic, or unexpected driver unavailability due to sickness. Therefore, driver scheduling is also a complex scheduling problem, though the volumes are not as high as taxies or rental cars. Therefore, the VD industry requires a robust and scalable solution to maximize the customer satisfaction, efficiency, driver earnings, profit, and driver satisfaction. The solution should be capable of covering as many jobs as possible while minimizing overall cost and equitably distribute the income among drivers.

1.1 Motivation

Currently, driver scheduling is mostly manipulated manually by an experienced scheduling manager, who creates the next day's schedule at the end of the previous working day based on the orders received. The scheduling manager also needs to keep track of the progress of jobs (usually by calling drivers and customers) and make necessary adjustments due to dynamism as the day progresses. However, as the number of jobs and drivers increase, it becomes difficult to decide on the most appropriate driver for a job such that both the customer and company goals are optimally satisfied. Moreover, last-minute schedule changes could trigger a chain reaction to subsequent jobs. Therefore, the industry is in need of scalable and automated scheduling solutions that increase customer satisfaction, efficiency, company profits, and driver satisfaction. The company needs to maintain driver satisfaction as drivers are the primary asset, which is hard to replace and find in VD industry. Recruiting and training drivers are also costly processes which add to the bottom line of the company. Therefore, it is essential that drivers should not be idle, and jobs should be assigned relatively giving them an opportunity to earn a fair income based on their availability and willingness to contribute.

However, as the driver, route, and vehicle scheduling problems are known to be NP-hard, we cannot get the optimal solution within polynomial time [1], [2], [3]. Researchers have worked on driver, route, and vehicle scheduling problems and proposed various techniques. Therefore, it requires identifying a suitable heuristic-based solution that can still maximize the customer satisfaction, efficiency, driver earnings, and company profit.

1.2 Problem Statement

We consider a vehicle delivery company with a set of drivers that are spread around given geography. We assume the set of jobs to be assigned is known a priory. These jobs are typically assigned by an experienced scheduling manager at the end of the previous business day. Handling last minutes jobs is left as future work. Therefore, in this context the problem to be addressed can be formulated as follows:

Given a set of drivers D and jobs J, how to automatically schedule drivers while maximizing customer satisfaction, efficiency, profit, and driver satisfaction?

Problem is more formally expressed in Chapter 3.

1.3 Objectives

Following set of objectives are to be used to address the above problem statement:

- Identify parameters related to the drivers and customers and then formulate the driver scheduling problem as a constrained optimization problem with multiple objectives
- To conduct a comprehensive literature study to identify suitable appropriate approaches to solve the given constrained optimization problem
- Solve the constrained optimization problem using a suitable technique
- To evaluate the performance of the proposed solution using a dataset from a real vehicle delivery company. Moreover, we plan to compare the results with other well-known solutions to similar problems

1.4 Outline

The rest of the thesis is organized as follows. Literature review is presented in Chapter 2. Problem formulation including driver and job constraints and research objectives are presented in Chapter 3. Solution approach with the proposed rule checker and job scheduler are presented in Chapter 4. Performance analysis is presented in Chapter 5 while summary and future work are presented in Chapter 6.

2. LITERATURE REVIEW

Researchers have worked on driver, route, and vehicle scheduling problems for years and have proposed various methodologies and algorithms to solve the problem in a particular setting. We present related work on the driver and vehicle scheduling that is more relevant to driver scheduling problem in the vehicle delivery industry. Under each approach, we explore both the problem formulation and solution approach.

2.1 Driver and vehicle scheduling in Limousine rental

An automated driver and vehicle scheduling solution for a limousine renting company is proposed in [4]. The rental company has drivers and single depot where all vehicles are parked. Various trips are requested by customers daily. Therefore, the company's goal is to schedule resources to cover as many trips as possible. The quality of service is a crucial issue, a schedule must comply with a set of important constraints, while optimizing some economic objectives. The rental company covers the Paris city and its suburbs, representing a surface of approximately 12,000 km^2. Based on planners experience in allocating past jobs, this extensive area has been partitioned into 26 zones. Additionally, major traffic generators, such as large hotels or airports have been precisely identified. More than 95% of the places involved in the problem match these specific locations. If it is not the case, the place is approximated to the zone center it belongs to. Travel times between all the identified locations have been pre-computed and stored in the database. To avoid having a null value within a zone, a threshold value has been set. Because travel times fluctuate a lot in big cities, authors have introduced possible travel time variations according to the type of day (working/holiday period) and the time range within the day.

Every evening, according to the already booked trips (on average 70% of the total), a schedule is manually determined for the day after. This problem is highly combinatorial because instances can involve hundreds of trips, drivers, and vehicles per day. Moreover, the problem has to deal with various constraints and available resources. Real-Time management is then achieved throughout the day, including change requests. Much uncertainty surrounds the trips as they can be booked, canceled, or modified at the last minute according to the customers' wishes. Furthermore, this problem takes place in a dynamic environment because the company provides services in the Paris area where daily congestion and delays are high.

Authors proposed a sequential, two-phase heuristic algorithm to solve the scheduling problem, where a constrained model is first used to get an initial solution, which is then optimized using SA. It is a random search technique inspired by annealing process where a solid is slowly cooled until its structure reaches a minimum energy configuration. Constraints and objectives are formulated considering a set of trips, drivers, and vehicles.

The primary objective of the solution is to meet trip demands of customers. Therefore, the first goal is to find a schedule that maximizes the number of trips covered. The secondary objective is to reduce the number of working drivers and used vehicles. To reduce costs, it is also useful to minimize the number of upgrades, the time drivers spend waiting, and driving between trips. Authors mathematically formulate a set of constraints for capacity, category, features, skills, maximum spread time, possible sequences, and pairing constraints. Based on these constraints initial schedule is constructed. The initial solution is motivated by the Best Fit Decreasing Strategy introduced in [5] to tackle the bin-packing problem. The trips are sorted in decreasing order of duration, subsequently labeled one by one by a driver-vehicle pair that can handle it. After each assignment, a forward checking procedure is applied to prevent future conflicts.

The initial schedule is then improved using an SA algorithm, which can find the global optimum in a large search space. It is well-known that the neighborhood is one of the most essential components of any SA algorithm [4]. For this reason, authors define and experiment with different neighborhood operators ranging from simple to complex compound moves, as follows:

- *1_change* – Affects one variable (driver or vehicles) at a time
- *swap* – Affects two variables

- *1_change_&_re-assigns* – Picks a variable at random and assigns to a new value within its domain, if the variable does not create any conflict the value is stored, and then loop through all unassigned variables to assign values
- *Ejection_chain* [6] – Is a mechanism which loops through all possible variables within its domain without creating any conflicts. It reduces the weakness of not maintaining the full employment of the resources like in *1_change*
- *Ejection_chain_&_re-assigns* – Merges two former neighborhood operators. It overcomes their weaknesses and takes advantage of their respective strengths

Computational experiments were conducted ten datasets which were taken from the limousine rental company and representing different workloads. Due to the incomplete and non-deterministic nature of the methods, twenty independent runs were carried out on each instance with different random seeds. For neighborhood comparison, authors used a pure Hill Climbing algorithm with one instance, despite its tendency to fall into local optima and its incapacity to escape from them, Hill Climbing is a neutral algorithm, requiring almost no tuning of parameters, and thus, particularly adequate for comparing different neighborhoods [4].

Authors observed that the *swap* mechanism behaves poorly and quickly trapped in a local minimum. The reason for this is that the evaluation function is dominated by the weighted number of assigned variables. The *swap* mechanism leaves this number unchanged and thus, cannot improve the objective function for long. *1_change* modifies the value of a variable without creating conflicts. Its results outperform those of the swap neighborhood but remain unsatisfactory because the algorithm rapidly falls to a local minimum. Furthermore, the decrease is too slow to be employed in a real-time context. *Ejection_chain* overcomes the first difficulty encountered by *1_change* with its aggressive search strategy. The curve of *1_change_&_re-assigns* delineates a rapid decrease because this neighborhood continually seeks a maximal consistent assignment. Eventually, *Ejection_chain_&_re-assigns* combines the respective strengths of the two previous neighborhoods and provides the best results.

Moreover, authors compared results between manual scheduling and SA. Results based on real data sets show a significant improvement compared with the actual practice.

- at least 80% of the workload is automatically assigned
- the constraints are all satisfied
- the operational costs are reduced
- the solution is displayed within a short amount of time (10 min)

Within a short time, the solution supplies good quality schedules in which the major part of the trips is assigned. The constraints are all satisfied whereas the operational costs, including the number of resources, the number of upgrades, and the total idle time are reduced. However, the results show that total idle time of drivers has increased in SA-based solution compared to manual scheduling.

Authors proposed a simultaneous approach for a driver and vehicle scheduling problem in a Limousine rental company. This research work forms a sound basis for our problem though the context is somewhat different.

2.2 Column generation based hyper-heuristic solution for bus-driver scheduling

Column generation based hyper-heuristic solution in [7] addressed the bus-driver scheduling problem (BDSP). Authors presented a solution to public transit providers who are facing continuous pressure to assign bus drivers to particular duties while improving service quality and reduce operating costs. Before scheduling the driver tasks, the vehicle routes must be constructed. A *trip* is a movement of a vehicle on a given path. It is the basic unit of service in the sense that each trip must be operated by a single vehicle. A *vehicle block* is a sequence of trips to be done by one vehicle from the time that it leaves the depot until it returns to the depot. From the viewpoint of driver scheduling, drivers can only be relieved at some designated places called *relief points*. The time slots when the vehicles are at the relief points are known as *relief opportunities*. The work between two consecutive relief opportunities on the same vehicle is called a *piece of work* (or *task*) for the driver. The work of a driver in a day is known as a *duty* (or *shift*). Note that not all relief opportunities will be used to relieve drivers and therefore a driver may be covering several consecutive pieces of work, called a *spell*. According to which period the duties cover, the legal duties can be classified into five types as an early duty, late duty, night duty, day duty, and middle duty.

Column generation approach is used to solve linear programming problems with many columns, and it assumes that there exists a sub-problem to optimize [9]. Column generation has been widely used in vehicle routing problems. It is well-known that one of the drawbacks of column generation is the so-called “tailing-off” effect, where many iterations that do not significantly modify the optimal value of the Restricted Master Problem (RMP). Hyper-heuristic is an emerging technique in search and optimization [8], which reduces drawbacks in column generation.

Authors then build a mathematical formulation and implement an objective function with the set of legal duties and the set of pieces of work to be covered. The column generation algorithm for the BDSP is initialed by giving a small set of duties (columns). The pricing subproblem of BDSP is typically modeled as a constrained shortest-path problem solved over a directed acyclic graph and then solved using dynamic programming techniques [9]. Nevertheless, because of the drawback that the subproblems for generating the columns would be computationally expensive in real problem instances [10], column generation did not make much progress toward solving large instances. Thus, authors, main work relies on the subproblem to speed up the whole process of column generation. Hence, authors propose the column generation based hyper-heuristic methodology to solve the BDSP. This approach provides two significant advantages. First, because column generation is known for its poor convergence, it is not necessary to select only one column with lowest reduced cost; in fact, any column with a negative reduced cost will do, which improves the overall efficiency. Thus, more than one duty with negative reduced costs may even be possible to be brought into the restricted subset per iteration. Second, the diverse columns with a negative reduced cost that are selected (or generated) can be performed efficiently.

The outline of the algorithm consists of following seven steps:

- *Step 0 (preprocessing)* – Generate all valid duties. Then these duties form a column pool P.
- *Step 1 (construct an initial solution)* – Take a small subset of the duties $p \subset P$ as inthe itial set of columns.
- *Step 2 (solving the RMP)* – Solve the Linear Programming (LP) over the current duty subset and compute the shadow prices of the set of columns p.
- *Step 3 (duty management)* – Control the column pool P and remove the columns in the RMP if necessary.
- *Step 4 (selection of new duties)* – Select duties with a negative reduced cost which will improve the solution and add them to the RMP.
- *Step 5 (stopping criterion)* – If the stopping criterion is met, then go to Step 6; else go to Step 2.
- *Step 6 (finding integer solutions)* – Solve ILP and obtain an integer solution.

Authors demonstrated that the proposed solution is more scalable and outperforms all other well-known scheduling algorithms such as linear programming and SA. They use five instances which respectively increase the data size (number of pieces of work). In most cases of datasets, the proposed method could find solutions which similar to best-known solutions. The method does not perform well in some small-size datasets compared to linear programming, even to SA. However, as the size of instance increases, their method performs better. This is because that, for a small dataset, the algorithm is too complex, and its computing time is long. However, for large datasets, the classic methods meet the problem of the combinational explosion which is the typical feature of an NP-hard problem. The proposed algorithm shows its advantage of saving computing time when increasing the size of the dataset.

Column generation algorithms are best used when there is a more extensive dataset with a limited number of constraints. In our case, we have to deal with the relatively small size of the dataset with a large number of constraints where column generation approach may provide inefficient results.

2.3 Artificial neural systems for delivery truck scheduling

Feasibility of using artificial neural systems for delivery truck scheduling using a small scale, dynamic routing, and scheduling problem is presented in [11]. This research has addressed to a specific problem domain related to scheduling of delivery trucks from a regional distribution center.

Authors proposed to demonstrate the applicability of Artificial Neural Systems (ANS) in the more general business environment. In this environment, data is often noisy and solutions to specific problems are in most instances non-optimal and in many cases unknown. As such, the learning capabilities of neural systems can be utilized as a source of improved solutions. Because these problems and NP-hard, there is no neural or heuristic approach exists that can optimally solve large-scale or small-scale dynamic driver vehicle scheduling problems. ANS could function in the domain of unstructured and pattern recognition problems. However, it is important to limit the search space initially because ideal problem domains have an infinite search space. With experiential knowledge, which could be captured in a ANS, significantly reduce the search space. There are four general phases in the development of an operational, artificial neural system, namely:

1. Conceptualization and development of initial ANS framework
2. Operational development and training of network
3. Testing and retraining of operational network
4. Implementation and tracking

The ANS is formulated using an organization that operates a large regional package distribution center in a medium-size commercial/urban area. From this center, three delivery trucks are loaded and dispatched to separate delivery areas (DA). The delivery areas are divided geographically into six zones. A DA can be serviced by multiple trucks, or one truck can service multiple areas. Within a DA there normally will be multiple deliveries along with possible pickups, all performed by the same truck. Not all DA’s will have scheduled pickup points, one pickup point can serve multiple delivery areas.

The example problem was selected for expositional purposes only and was restricted to three trucks and six locations for simplicity of explanation. Even with this small problem size, over 262,000 possible combinations of trucks and delivery areas are generated. All of those combinations are not feasible from a real-world point of view, and in fact, an organization may utilize no more than ten or twenty of the combinations. Authors use multi-layer perceptron, feed-forward, continuous valued input, and supervised learning environment to find a solution to dispatch delivery vehicles optimally.

Authors discuss some of the drawbacks of the solution too. Standard optimization routines must wait until all data is known and then solve the problem. If the input dataset or other parameters change, optimization techniques require recalculation or modification of the model, while an adaptive neural system can modify itself. Authors show that further research is needed to determine if indeed neural systems can solve the complex and dynamic problems encountered in the business environment. Therefore, this approach would not provide acceptable results in dynamic environments where frequent changes in the datasets and parameters could occur.

2.4 RMC truck dispatching using machine-learning techniques

In [12], authors demonstrated that ready-mix concrete truck dispatching can be automated through machine learning techniques. A construction project consists of a wide range of complex tasks. Due to this complexity, most construction tasks are performed by humans. Additionally, it is very difficult to accurately predict performance and unavoidable errors. The managers or senior engineers play a key role in construction and rarely are their positions replaced by an automated process. This is because the tasks associated with most construction jobs are complex, highly dependent on specific project constraints, environmental conditions, and must adapt quickly based on incomplete as well as rapidly changing information. Feasibility of automation in RMC dispatching was studied using six ML techniques, namely:

1. J48 - Decision tree
2. PART - Rule-based algorithm

3. ANN - Artificial neural network
4. SMO - Support vector machine
5. NB - K nearest neighbor
6. IBK - Naive Bayes classifier

Those techniques were selected and tested on data that was extracted from a developed simulation model. The results were compared by a human expert to ensure the accuracy of solutions. The simulation model consists of a single batch plant and three projects in a day. The effective parameters on the performance of RMC were extensively discussed in many kinds of literature such as [13], [14], [15] and based on the conducted research the following attributes were selected to build the training sets:

Training set = {DD, AOC_1, TT_1, ST_1, LP_1, AOC_2, TT_2, ST_2, LP_2, AOC_3, TT_3, ST_3, LP_3, OS}

- DD - day of delivery in the week
- AOCi - amount of ordered concrete for project i
- TTi - travel time for project *i* include loading, hauling, pouring and return time
- STi - spacing time for project *i* (time between each pour)
- LPi - location of project *i*
- OS - order of supply

For constructing a real situation in the simulation model, a metropolitan area consisting of seven suburbs was selected. In this area, there is a batch plant that supplies concrete for all seven selected suburbs. The generated data by simulation is sent to the dispatch manager of that batch plant for him to prioritize the projects in each day. The 200 instances are prioritized by the dispatcher in two stages with each time involving 100 instances. The same dataset was used in the six selected algorithms, and the ten-folds cross-validation was selected for evaluating the selected algorithms. This means that the data set was divided into ten-folds with around 90% of each fold used for training and the remaining 10% of data being used for testing.

Regarding comparison, the essential feature is accuracy. This reflects the ability of each algorithm to identify the correct decisions that are the primary task of a classifier. Regarding accuracy, ANN achieved the best performance and IBK was the worst. The accuracy rates of J48, PART, ANN, SMO, and NB are similar. In instances where algorithms are slightly different due to randomness concerns, which algorithms outperform others based only on accuracy cannot be identified [16]. Moreover, the best algorithm in one feature may not necessarily be the best in other performance metrics. The results show that, except for IBK which was outperformed by most of the other techniques, there is no significant difference between SMO, ANN, J48, NB, and PART. Thus, according to the ML approach, the performance of all selected algorithms, except for IBK, were the same. This demonstrates the strength of ML techniques in predicting human minds. Regarding solution building time, J48 and NB give better performance than PART, SMO, and ANN. In conclusion, authors demonstrated that the decision trees and k-nearest neighbor techniques give better results with lesser time than neural network and support vector machine based solutions.

While the problem applies to a dynamic environment, it is more specific to the RMC trucks and drivers. However, our problem is highly combinative and complicated with many constraints and parameters and drivers compared to the RMC trucks scheduling problem. As authors demonstrated, decision trees and k-nearest neighbor techniques may provide acceptable results for our problem, but the computational time may increase due to the complexity.

2.5 Truck driver scheduling problem

A detailed discussion of the truck driver scheduling problem (TDSP) is presented in [17]. They present a literature review of the truck drivers scheduling problem since the year 2000 and to classify them to four criteria in which the primary objective is to offer a complete reference frame grouping the works dedicated to the truck driver scheduling. They suggest classifying the papers which related to truck drivers scheduling problem to the following four criteria.

1. The types of problems: Appropriation (driver, crew, vehicle) and Routing
2. The nature of the goods transported (ordinary, hazardous)
3. Types of work hours regulation (United States, European Union, Canada, etc.)
4. The various constraints required (duty, rest duration, meal break, safety, etc.)

According to the presented classification, authors note that: the interest of many researchers is focused on the work hour's regulations and the constraints related to duty, rest, and meal break durations on the one hand and there are a lack of works treating the constraints related to the safety, psycho-physical, and driver's experience.

Above research papers implies that researchers have solved the driver, crew, and vehicle scheduling problems by using various standard and innovative algorithms for many years. There is no ideal way to solve these problems because these problems are NP-hard. Moreover, each solution thoroughly depends on the problem context, problem size, nature of the related constraints. However, there are several notable differences between the VD and other delivery problems. For example, in VD problem both the vehicles and drivers are geographically dispersed, whereas in limousine renting and bus scheduling problems vehicles and drivers are dispatched from a specific depot. Moreover, in VD problem most jobs are one way, types of vehicles and driver skills required to operate them drastically vary, some vehicles are not delivered on the same day as pickup (depending on distance and receiver availability), more flexible pickup/delivery times, and drivers are willing to work only on chosen geographies and times. Therefore, it is essential to formulate the optimization problem for the specific domain and devise a more fitting solution.

2.6 Summary

Several related work proposes to address the driver and vehicle scheduling problem using different optimization and machine-learning techniques. Because this is an NP-hard problem, there is no ideal solution for this problem. Therefore, it is crucial to find out an appropriate approach in a given context that provides an acceptable result. Moreover, the chosen approach largely depends on the problem context such as the size of the dataset (e.g., drivers and vehicles), number of constraints involved, and accepted output time. The problem we focus on contains a relatively higher number of constraints including multiples set of objectives to be achieved in priority order. In such cases, neural networks and machine-learning techniques generate enormous numbers of combinations that cannot be solved within a reasonable time. Therefore, a more scalable and computationally efficient approach is needed to solve driver scheduling in vehicle delivery problem. Based on the related work simulated annealing seems to be a more robust technique that can deal with highly nonlinear models, chaotic and noisy data, and with many constraints [23]. Moreover, it is flexible and able to approach global optimality than other local search methods.

3. PROBLEM FORMULATION

In this chapter, we identify all parameters and constraints that affect the driver scheduling process. Then we used a constrained-based approach to filter out the possible search space as reducing the search space is important to achieve an acceptable solution in NP-hard problems.

Let **J** be the set of Vehicle Delivery (VD) jobs, where each job $j \in \mathbf{J}$ has a pickup and delivery location and time, type of vehicle, and a preferred set of drivers (typically for recurrent jobs). These jobs are to be processed by a set of drivers **D**, where each driver $d \in \mathbf{D}$ has a set of required skills and is licensed to drive a set of vehicle categories. A driver has a home location, preference to work only on a set of geographies, preferred working days and times (e.g., part-time drivers), and is willing to accept only a given number of jobs per day/week. Moreover, regulatory requirements such as a maximum number of driving/working hours a day and days per week/month need to be met. Let f_j be the fee for job *j*, which depends on the distance between pickup and delivery locations. Whereas cost per driver *d* for job *j* (c_d^j) depends on the distance drove, driver's skill level, and driver's personal expenses to reach the pickup location, and return to the next job's pickup location or return to the home. Our objective is to cover all jobs **J** with drivers **D**, such that $f_j - c_d^j$ is maximized across all the jobs. Next, we describe each of the parameters and constraints in details and then formulate the optimization problem. Table 2.1 lists the driver related symbols. Table 2.2 and 2.3 lists the job and solution related symbols, respectively.

3.1 Constraints

To be eligible for a job, drivers need to satisfy the following set of constraints:

Vehicle Type Constraint – Driver *d* must have a valid driving license to drive the vehicle type defined in job *j*, e.g., car, van, or truck. Therefore,

$$j_{type} \in d_{type} \rightarrow d \qquad (1)$$

Table 2.1. Driver-related symbols.

Symbol	**Description**
d_{id}	Driver ID
d_{type}	Set of vehicle types driver *d* can drive with a valid driving license
categories	Set of vehicle categories *d* can drive based on special training
$d_{time\_max\_day}$ / $d_{time\_max\_week}$	Maximum allowed driving time per day/week for driver *d*
d_{work_time}	Working days and hours of driver *d*
d_{job_areas}	Set of preferred geographies driver *d* is willing to accept jobs
d_{job_count}	No of jobs assigned so far to driver *d* on a given day
$d_{max\_jobs\_day}$	Maximum number of jobs driver *d* prefers to handle on a day
$d_{drive\_time\_day}$ / $d_{drive\_time\_week}$	Total driving time of driver *d* for a given day/week

Table 2.2. Job-related symbols.

Symbol	**Description**
j_{id}	Job ID
j_{type}	Vehicle type demanded by job *j*
$j_{category}$	Vehicle category demanded by job *j*
j_{pickup_loc} / $j_{delivery\_loc}$	Vehicle pickup/delivery location of job *j*
j_{pickup_time} / $j_{delivery\_time}$	Vehicle pickup/delivery day/time of job *j*
$j_{preferred\_drivers}$ / $j_{excluded\_drivers}$	Customers may specify a set of drivers that they prefer/don't prefer to work with based on the past experiences

Table 2.3. Solution-related symbols.

Symbol	**Description**
f_j	The fee charged from a customer for job *j* (depends on job distance)
j_{travel_time}	Estimated travel time for job *j*
d_{jobs}^{count}	Number of assigned jobs to driver *d* for on a given day
λ	Cost margin to assign a preferred driver

Vehicle Category Constraint – Some jobs demand specific driver skills and training, e.g., specific training may be required to operate certain luxury and heavy vehicles. Moreover, while delivering a high-end car or special-purpose vehicle to a new buyer, the manufacturer may train the driver on customer service and vehicle maintenance tips. Therefore, *d* must have the relevant training and skills to take the job:

$$j_{category} = \phi \cup j_{category} \in d_{categories} \rightarrow d \tag{2}$$

Number of Jobs Constraint – As the VD company may also rely on part-time drivers, drivers have the flexibility to mention how many jobs they prefer to complete per day, without considering the complexity of jobs or income. Therefore, for a given day

$$d_{job\_count} + 1 \leq d_{max\_jobs\_day} \rightarrow d \tag{3}$$

Job Area Constraint – Drivers may mention their preferred area of work. Therefore, the driver should be assigned to jobs only within his/her preferred area:

$$j_{pickup\_loc} \subseteq d_{job\_areas} \cap j_{delivery\_loc} \subseteq d_{job\_areas} \rightarrow d \tag{4}$$

Driver Availability Constraint – Drivers have the flexibility to work on any day any time. Hence, their availability should match the job's timeline as follows:

$$j_{pickup\_time} \in d_{work\_time} \cap j_{deliver\_time} \in d_{work\_time} \rightarrow d \tag{5}$$

Travel Time Constraint – For regulatory purposes, usually, there are limits on how many hours a driver can drive per day and a week. Therefore, for a given day and a week, total driving time of *d* and estimated driving time of the new job *j* should be within the maximum allowed:

$$d_{drive\_time\_day} + j_{travel\_time} \leq d_{time\_max\_day} \cap d_{drive\_time\_week} + j_{travel\_time} \leq d_{time\_max\_week} \rightarrow d \tag{6}$$

where estimated travel time can be more accurately calculated today using services such as the Google Maps API [18] based on pickup/delivery location and time, traffic, and weather. Therefore, we define a job's travel time as follows:

$$j_{travel\_time} = ttf(j_{pickup\_loc}, j_{delivery\_loc}, j_{pickup\_time}) \tag{7}$$

Feasible Sequences Constraint – Once a job is completed, it may not be possible to start the next job immediately. Therefore, a driver needs sufficient time to travel to the pickup location of the next job. This depends on many parameters such as delivery time of the previous job, availability of public transportation, and traffic. Therefore,

$$j_{next\_pickup\_time} = j_{delivery\_time} + ttf(j_{delivery\_loc}, j_{delivery\_time}, j_{next\_\_pickup\_loc}) \tag{8}$$

Pairing Constraint – Frequent customers may specify a list of preferred or excluded drivers based on the past experiences. While such a list is typically specified per customer, from the modeling point of view, we can inherit job's preference from customer's list, without modeling the customer separately. Therefore, a preferred driver should be assigned when possible, while a driver in the excluded list of a job should not be assigned at all. Thus, the following constraints need to be satisfied:

$$d_{id} \notin j_{excluded\_drivers} \rightarrow d \tag{9}$$

$$d_{id} \in j_{preferred\_drivers} \rightarrow d \tag{10}$$

However, a preferred driver cannot be assigned under all costs, as the objective is also to minimize the cost. Therefore, constraint (10) is enforced only when the cost of assigning a preferred driver is within a reasonable difference (say λ) from the other driver(s) with the least cost.

3.2 Objectives

Given a set of jobs **J** and drivers **D**, our main objective is to cover as many jobs as possible. This is required to improve customer satisfaction as some of the customers are engaged in a long-term business relationship. We further satisfy a secondary objective, namely to maximize VD company’s overall profit. To minimize the cost, it is imperative to find the most appropriate driver who can do the job within the customer requested time frame with minimum cost. Drivers should not be idle, as they do not make any income when they are not driving. Hence, driver earning should be good, else they are likely to leave the company. Therefore, the objective function can be formulated as follows:

$$\forall\, j \in \mathbf{J},\ \forall\, d \in \mathbf{D}\ \text{Max}(|j \text{ with assigned } d|) \tag{11}$$

$$\text{Max} \sum_{\mathrm{j \in J, d \in D}} f_j - c_d^j \tag{12}$$

$$d_{income}\ \alpha\ d_{work\_time} \tag{13}$$

Constraint (11) maximizes the number of jobs with an assigned driver. Profit is given as $f_j - c_d^j$.

4. PROPOSED SOLUTION

As in many scheduling problems [4], we assume that every evening, the next day's schedule is determined based on the already confirmed jobs and available drivers. It is difficult to estimate the job end time because it depends on operational inefficiencies, traffic, and weather which are hard to predict on a given day. While we investigate the flexibility of the automatically generated schedule in Chapter 5, dynamic scheduling to overcome last-minute changes is left as future work. Our solution consists of a *rule checker* that enforces the constraints in Section 3.1 and an optimization phase that attempts to cover as many jobs as possible while minimizing the overall cost. For optimization, we choose Simulated Annealing (SA) algorithm because it is used in global optimization problems in a wide range of areas including limousine renting in [4]. SA provides a reasonably optimized solution within a reasonable time and can be optimized according to the context [19].

4.1 Rule checker

Given a set of job and drivers, the rule checker first evaluates the constraints in Section 3.1 as shown in Figure 4.1. This reduces the search space of Simulated Annealing, as the number of potential drivers to be assigned to a job depends on the vehicle type, driver training, availability. When a new job comes, all drivers are evaluated. Rule checker proceeds with each constraint sequentially. When it goes through a constraint, it outputs the possible set of drivers who are eligible under the particular constraint. For the next constraint, only those filtered drivers would be fed. Likewise, the set of eligible drivers gradually decrease as we proceed through the constraints. Initially, rule checker proceeds with static constraints like vehicle type constraint, vehicle category constraint, job area constraint, and excluded driver check under pairing constraint, which can reduce the search space without depending on any external factors.

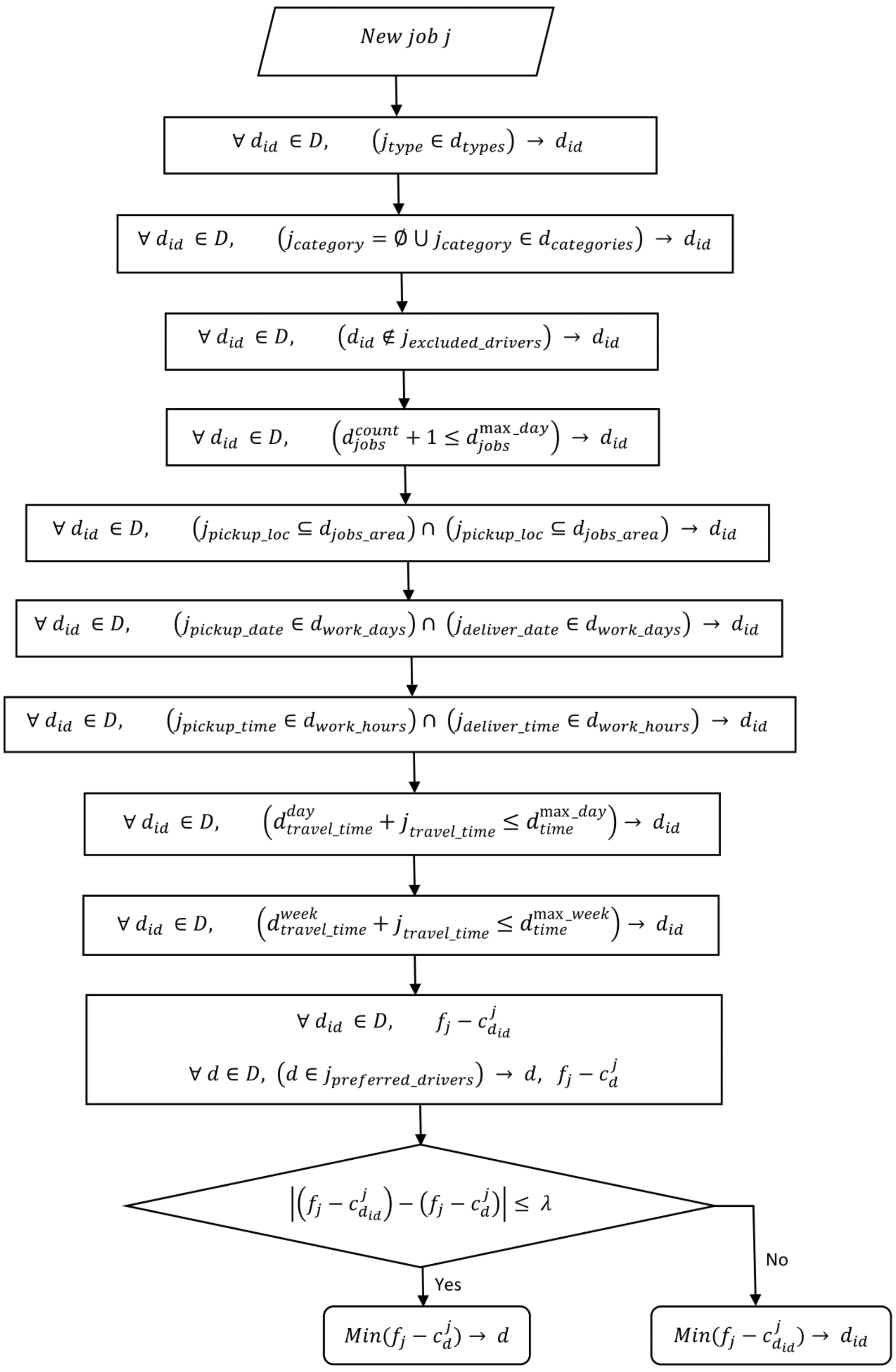

$New\ job\ j$
$\forall d_{id} \in D, \quad (j_{type} \in d_{types}) \rightarrow d_{id}$
$\forall d_{id} \in D, \quad (j_{category} = \emptyset \cup j_{category} \in d_{categories}) \rightarrow d_{id}$
$\forall d_{id} \in D, \quad (d_{id} \notin j_{excluded\_drivers}) \rightarrow d_{id}$
$\forall d_{id} \in D, \quad (d_{jobs}^{count} + 1 \leq d_{jobs}^{\max\_day}) \rightarrow d_{id}$
$\forall d_{id} \in D, \quad (j_{pickup\_loc} \subseteq d_{jobs\_area}) \cap (j_{pickup\_loc} \subseteq d_{jobs\_area}) \rightarrow d_{id}$
$\forall d_{id} \in D, \quad (j_{pickup\_date} \in d_{work\_days}) \cap (j_{deliver\_date} \in d_{work\_days}) \rightarrow d_{id}$
$\forall d_{id} \in D, \quad (j_{pickup\_time} \in d_{work\_hours}) \cap (j_{deliver\_time} \in d_{work\_hours}) \rightarrow d_{id}$
$\forall d_{id} \in D, \quad (d_{travel\_time}^{day} + j_{travel\_time} \leq d_{time}^{\max\_day}) \rightarrow d_{id}$
$\forall d_{id} \in D, \quad (d_{travel\_time}^{week} + j_{travel\_time} \leq d_{time}^{\max\_week}) \rightarrow d_{id}$
$\forall d_{id} \in D, \quad f_j - c_{d_{id}}^{j}$
$\forall d \in D, \ (d \in j_{preferred\_drivers}) \rightarrow d, \ f_j - c_d^j$
$\left|(f_j - c_{d_{id}}^{j}) - (f_j - c_d^j)\right| \leq \lambda$
Yes
No
$Min(f_j - c_d^j) \rightarrow d$
$Min(f_j - c_{d_{id}}^{j}) \rightarrow d_{id}$

Figure 4.1. Solution model for rule checker.

Then the dynamic constraints such as travel time constraint and preferred driver check under pairing constraint are checked. Rule checker tolerates λ cost on selection between a any driver and a preferred driver. This enables assigning preferred drivers to a particular job to improve the customer satisfaction, as far as the additional expense is reasonable.

4.2 Job scheduler

Reduction in search space is essential to achieve an acceptable solution in NP-hard problems. This is achieved through the rule checker. Then the job driver (j_{id}, d_{id}) pairs, which are identified as eligible for further processing, run through Simulated Annealing algorithm to find an optimal solution while maximizing the job coverage and minimizing the overall cost. Once, a driver is assigned to a job, his/her availability for other jobs will change. For example, once a job is assigned, a driver is not eligible for another job with pickup time that is less than the delivery time of the current job. Moreover, the cost of reaching the next pickup location can either reduce or increase. Therefore, after each job assignment, respective driver's availability for all other unassigned jobs are rechecked using the rule checker, before considering for further job assignment by the Simulated Annealing algorithm.

All distances and estimated times for VDs are taken from Google Distance API [20] to achieve more reliable and accurate estimates. Fee for a job f_j is calculated based on the estimated distance between pickup and delivery locations of the job. We also define several parameters to make the solution more accurate. For example, a driver may need to use public transportation to reach the point of collection or to return to home or next job after a delivery. That journey may take more time than the driving time to/from delivery/pickup location, hence needs to be accurately captured while assigning subsequent jobs, as well as to estimate the departure/return time to/from a job. While the cost of public transportation is usually lower than taking a taxi or rental car, it still needs to be considered while calculating the cost of a job c_d^j. We capture such delays and fees due to the use of public transportation by introducing two factors to multiply time and distance. A few countries or cities provide APIs to access bus and train schedules, as well as associated fees. In such cases, the solution can be further improved by substituting values derived from such APIs instead of the proposed factors.

SA is a probabilistic, single-solution-based search method inspired by annealing process where a solid is slowly cooled until its structure reaches a minimum energy configuration [21]. Optimization algorithms may converge to local optimums, and SA has its technique to avoid those disadvantages. SA accepts worst solutions at higher temperatures to avoid local optimums by setting its acceptance probability to a higher value. The main parameters of SA algorithm are an initial temperature, a rule for accepting a damaging move (i.e., higher cost solution), the rate at which the temperature decrements, the maximum number of neighboring solutions that can be generated at each temperature, and a stop criterion [22].

Optimization search algorithms mostly optimize a cost or distance matrix. In our solution, our primary objective is to cover all possible jobs and then our secondary objective is to optimize cost matrix to minimize the overall job cost. In SA implementation, we prioritize to cover all possible jobs and then optimize the overall cost.

It is essential to check out the reasons to select SA algorithm in driver scheduling over other optimization algorithms [23]. SA is a robust technique that can deal with highly nonlinear models, chaotic and noisy data, and with many constraints. Our solution is also built on many constraints and may contain chaotic and noisy data. SA is flexible and able to approach global optimality than other local search methods. SA is quite versatile as it does not rely on any restrictive properties of the model. SA is easily customizable and tuned. In our solution, the primary objective is not the cost optimization. We prioritize the order of objectives to maximize the job coverage and then minimize the job cost. We also able to get a stable solution within acceptable time and results will be discussed in next chapter.

5. PERFORMANCE ANALYSIS

5.1 Workload Creation

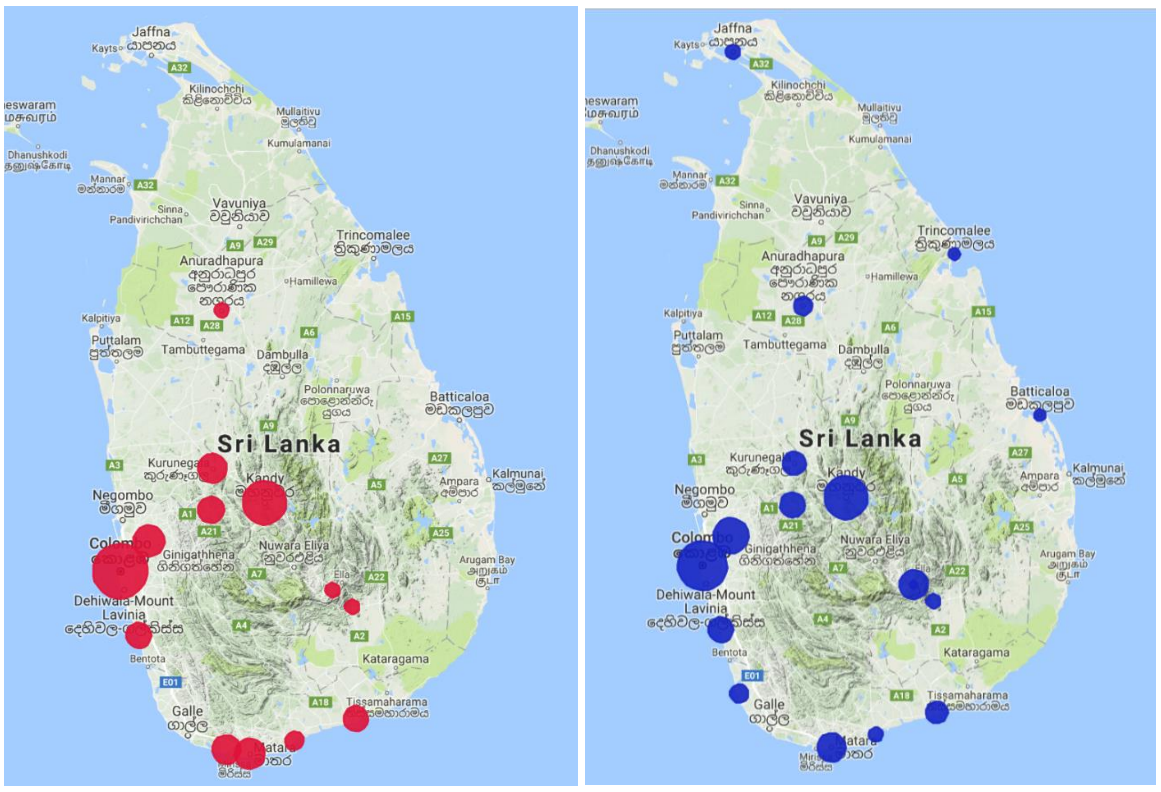
We used two different job datasets each with 80 jobs, against a set of 60 drivers. These datasets were created based on properties extracted from a dataset of a real VD company. This includes the distribution of job locations, pickup and delivery times (some jobs span across two days), driver skill distribution, and other constraints. Table 5.1 shows a summary of driver availability across a week. Figure 5.1 illustrates the job locations for both datasets and Figure 5.2 illustrates driver locations. The size of circles represents the number of jobs and drivers of a city. As most of the jobs are from urban areas, recruited driver population also reflect somewhat matching behavior because VD companies are likely to recruit drivers from areas where they get many job traffic frequently. Jobs in the second dataset are more diverse than the first dataset; hence, excepted to take more time to complete jobs. For each day’s workload, we plan the schedule in the previous evening.

Table 5.1. Driver availability by day.

Day	No of Drivers	Average Available Time (H)
Sunday	13	7.23
Monday	58	9.56
Tuesday	48	8.48
Wednesday	49	7.51
Thursday	50	8.34
Friday	39	8.00
Saturday	33	9.59

Several parameters need to be considered as the problem runs in a dynamic environment. Instead of assigning a currency value for job fee and costs, we calculate it regarding the distance to be driven or distance on public transportation. We assume that the cost of taking a unit distance on public transportation is 0.7 of the cost of driving a vehicle. Moreover, the time required to take public transportation is calculated by multiplying the estimated time from Google Distance API by a factor of

1.2 (for the region considered in the analysis, Google Distance API does not give the time for public transportation). We later perform a parameter sweep to measure the impact of these values on the effectiveness of the solution. Moreover, a job will be assigned to a preferred driver, if the cost of assigning a preferred driver λ is within 50 form the lowest cost driver not in the preferred list.



(a) (b)

Figure 5.1. (a) Job distribution of dataset 1 and (b) Job distribution of dataset 2.

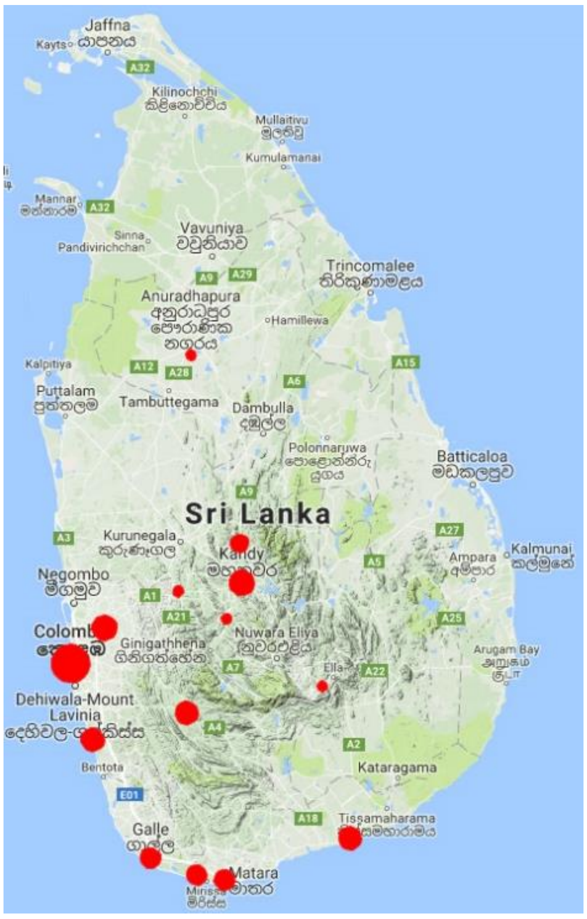


Figure 5.2. Driver distribution.

The tuning process of the SA algorithm is a delicate issue. It depends on the cooling strategy such as linear or exponential, cooling rate, and the energy of the system [24]. We ran the simulation five times with same job and driver dataset, and same configurations while varying the random seed. The simulation ran on a machine which has Intel Core i5-5200U CPU, 2.20 GHz processor, 8 GB RAM, and 3 MB Cache. We have implemented a Java-based simulator which initiates from a Java code for travel salesman problem [25]. Different combinations were tested on datasets resulting in the following set of parameters:

- initial temperature of 10^4,
- cooling rate of 0.003,
- terminating condition of temperature ≤ 1.

In tuning process, we tried various initial temperatures w.r.t acceptance rate, which defines the percentage of accepted transitions overall generated transitions. The results are listed in Table 5.2 Initial temperature with more than 40% of acceptance rate will give reasonable results in SA [26]. That value is for a default SA algorithm which takes the primary objective as the cost optimization. Because our primary objective is to maximize the job coverage, it is essential to traverse through many possible transitions. We used an initial temperature of 10^4 which has a higher acceptance rate and can be processed within acceptable running time, and traverse through a relatively higher number of transitions with cooling rate of 0.003.

Table 5.2. SA Acceptance rates against initial temperatures with 0.003 cooling rate.

Initial Temperature	**Acceptance Rate (%)**
500	29
750	33
1,000	36
2,000	41
5,000	48
10,000	52
20,000	54

5.2 Results

In the VD business, pickup and delivery times on most jobs are not strict and can be advanced or delayed by a couple of hours, as far as the client is kept informed. This enables flexibility in optimally assigning jobs while minimizing costs, as well as dealing with unexpected delays due to breakdowns, traffic, and weather. The solution enables flexibility by using time windows.

Table 5.3 shows a summary of the resulting solution under varying levels of flexibility in job pickup and delivery times. The number of jobs covered (aka., job coverage) depends on constraints among jobs and drivers, driver availability, job collection/delivery time, and time gap allowed to complete jobs. The job coverage is highest when the time window is ±1 hour on Monday when the highest number of drivers are available. When the time window is set to ±1 hour on Monday, 92.5% and 87.5% of the jobs are covered in dataset 1 and dataset 2, respectively. While an increased time window provides more flexibility, it also reduces job coverage and increases the driver idle time. Cost is a measurement of the distance traveled by the driver to deliver the vehicle. This is the reason that the cost reduces as the number of assigned jobs reduces with increasing time window. Execution time increases as the time window increases, because of the increased search space in SA. Table 5.4 shows the summary of results over a week in different time windows.

Table 5.3. Results against different time windows on Monday for dataset 1 and 2 with 0.003 as cooling rate.

Time Window (h)	0		±1		±2		±3		±4	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	4	7	74	70	68	65	63	59	55	52
Execution Time (s)	392	406	205	244	234	254	243	269	254	297
Cost (×100)	6	9	170	166	157	154	146	149	138	133
Profit (×100)	11	22	205	206	171	185	144	146	126	144

Figure 5.3 illustrates the total available hours against jobs coverage for both datasets in each day. It shows that the number of assigned jobs is proportional to the available number of drivers and their available times.

Table 5.4. Results against different time windows across a week.

	Tuesday									
Time Window (h)	0		±1		±2		±3		±4	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	3	6	62	58	55	53	51	49	47	42
Execution Time (s)	372	381	227	290	238	293	261	304	268	316
Cost (×100)	5	11	136	132	130	135	119	121	109	92
	Wednesday									
Time Window (h)	0		±1		±2		±3		±4	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	3	5	58	55	48	47	42	40	38	37
Execution Time (s)	362	373	223	285	240	297	274	344	289	389
Cost (×100)	5	11	119	133	99	135	97	121	78	92
	Thursday									
Time Window (h)	0		±1		±2		±3		±4	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	3	6	67	62	59	56	53	50	49	42
Execution Time (s)	371	373	217	264	225	299	242	324	262	329
Cost (×100)	5	12	154	140	132	132	118	132	120	95
	Friday									
Time Window (h)	0		±1		±2		±3		±4	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	2	5	51	48	44	42	40	38	34	32
Execution Time (s)	348	359	224	265	236	299	252	317	271	326
Cost (×100)	3	9	109	112	95	86	85	86	64	73
	Saturday									
Time Window (h)	0		±1		±2		±3		±4	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	3	6	61	55	52	49	45	42	40	39
Execution Time (s)	350	339	211	258	219	269	222	289	228	307
Cost (×100)	5	12	129	122	121	123	105	100	106	96
	Sunday									
Time Window (h)	0		±1		±2		±3		±4	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	2	4	18	17	16	14	14	13	13	13
Execution Time (s)	287	323	255	295	252	294	245	306	254	302
Cost (×100)	3	6	32	32	33	28	21	20	16	19

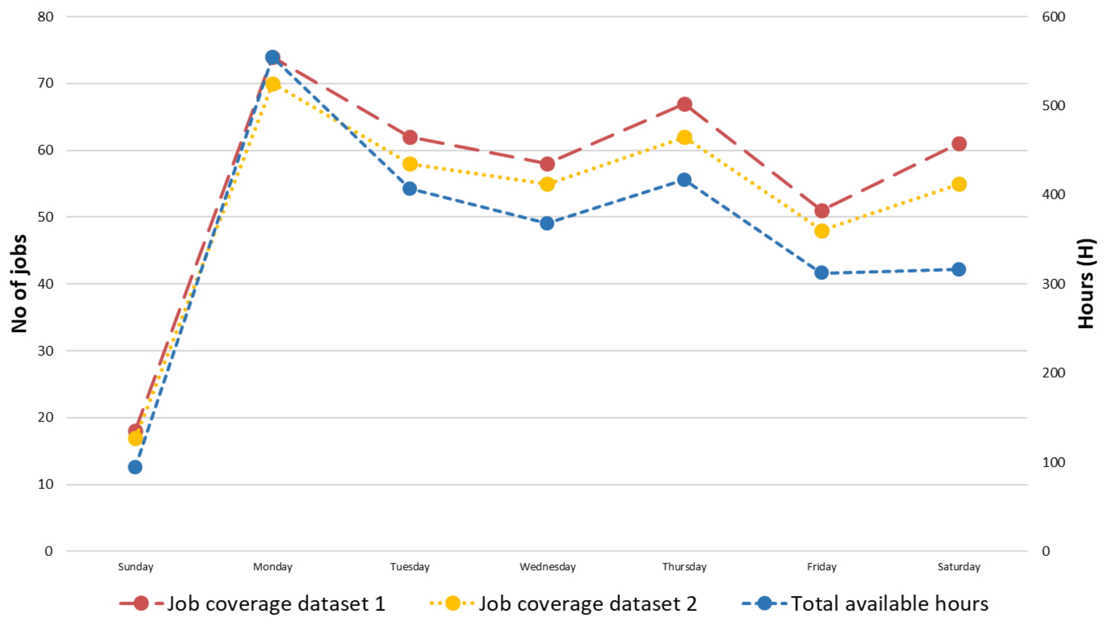


Figure 5.3. Job coverage against total driver available hours in each day.

Change of the cooling rate affects the number of transitions in SA under same initial temperature. We increased the cooling rate to 0.03 without changing the initial temperature. Table 5.5 shows the results against different time windows on Monday for both datasets. Figure 5.4 shows the jobs coverage against different cooling rates with different time windows in dataset 1. We still got same results or nearly same results in each case with less execution time. However, the resulting solution was less stable across different simulations, compared to when the cooling rate is 0.003.

Table 5.5. Results against different time windows on Monday for dataset 1 and 2 with 0.03 cooling rate.

Time Window (h)	0		±1		±2		±3		±4	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	4	6	73	69	69	64	62	59	54	52
Execution Time (s)	43	34	22	27	24	31	24	31	26	31
Cost (×100)	9	13	163	178	166	159	148	157	129	138
Profit (×100)	22	37	204	200	169	175	150	143	113	118

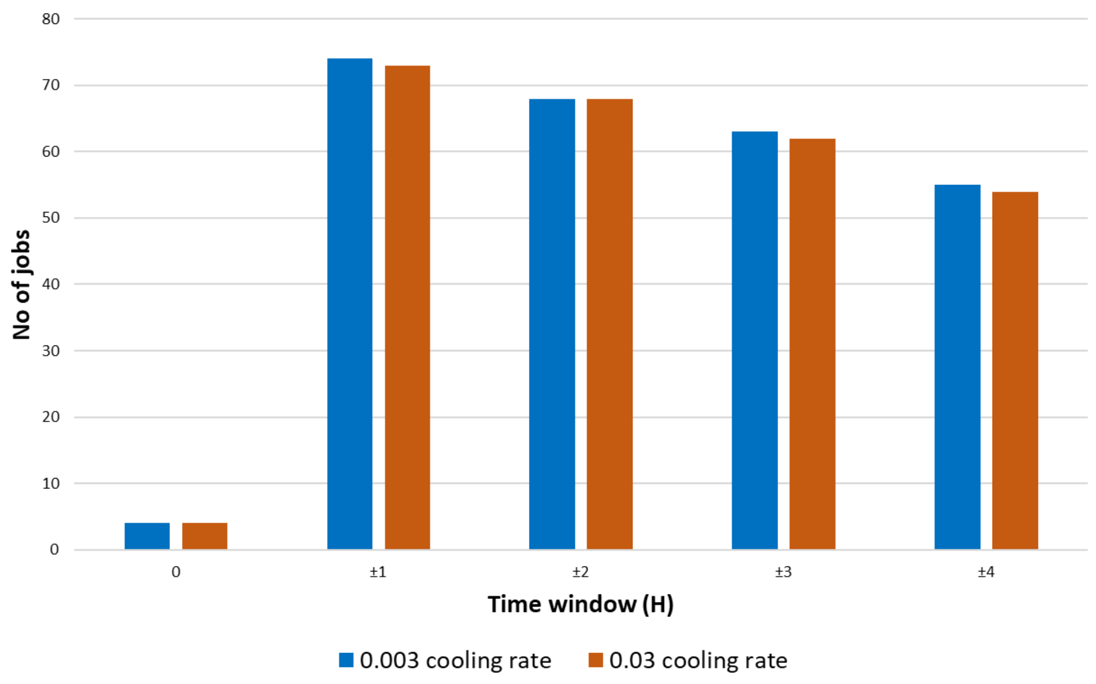


Figure 5.4. Job coverage against different cooling rates.

5.2.1 Comparison with other algorithms

Performance of the SA solution was also compared to two other algorithms. The results were compared with the Hill Climbing algorithm which is another optimization algorithm. The initial solution randomizes all available drivers and assigns to jobs sequentially after checking the eligibility with rule checker. Initial solution propagates through whole search space because all drivers and jobs are randomized. We also use the initial solution as the initial seed for both SA and Hill Climbing. Figure 5.5 shows pseudo-code for the initial solution. Simple hill climbing algorithm evaluates initial state and then loop through neighbor nodes until there is no better solution than the current solution. Simple hill climbing algorithm can quickly converge to local optimums or plateau. An enhanced hill climbing algorithm was used to avoid above disadvantages in simple hill climbing algorithm. It loops through some iterations and finds a better solution. If the new solution is better than the previous solution, it is assigned as best one. Then again loop through some iteration to find a better solution. Likewise, it loops through until there is no better solution than the previous solution. This mechanism provides flexibility to traverse through a relatively larger search space than in pure hill climbing and reduce the possibility of optimum local convergence. Figure 5.6 shows pseudo-code for enhanced hill climbing algorithm.

```
shuffle(drivers) -> i
jobs -> j

loop j -> [1: size(jobs)]

   if j < size(drivers)
      i = j
   else
      i = j - size(drivers)


   if driver is eligible
      job j -> driver i

end loop
```

Figure 5.5. The pseudo-code of the initial solution.

```
loop (improvement)

   loop
      if value(new) <= value(current)
            current = new
   end loop

   if value(current) <= value(best)
      best = current
   else
      improvement = false

end loop
```

Figure 5.6. The pseudo-code of enhanced hill climbing algorithm.

Table 5.6 shows results against different time windows on Monday for both datasets with hill climbing algorithm, and Table 5.8 shows results against different time windows across a week with hill climbing algorithm. Table 5.7 shows results against different time windows on Monday for both datasets with initial solution. Figure 5.7 illustrates job coverage against various algorithm on Monday for dataset 1. It shows that SA-based solution still gives a better solution regarding job coverage and hill climbing also gives an acceptable solution, but it is less than SA performance in each time window. The initial solution is unable to provide satisfactory results compared to other two algorithms.

Table 5.9 shows min, max, and average results against different algorithms on Monday for dataset 1 through different simulation runs. Figure 5.8 illustrates the min-max range of job coverage against different algorithms. This implies SA provide a more stable solution than other algorithms as its min-max range is relatively low in each case than other algorithms.

Table 5.6. Results against different time windows on Monday for dataset 1 and 2 with hill climbing algorithm.

Time Window (h)	0		±1		±2		±3		±4	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	3	6	71	66	65	62	61	57	52	49
Execution Time (s)	25	36	21	25	22	30	21	29	22	27
Cost (×100)	5	17	163	332	158	163	149	154	139	136
Profit (×100)	10	27	184	169	167	156	144	144	111	113

Table 5.7. Results against different time windows on Monday for dataset 1 and 2 with initial solution.

Time Window (h)	0		±1		±2		±3		±4	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	1	2	31	30	34	31	33	31	29	27
Cost (×100)	0.78	4	71	76	78	69	78	78	72	63
Profit (×100)	1.52	4.6	83	85	76	70	72	49	53	50

Table 5.8. Results against different time windows across a week with hill climbing algorithm.

	Tuesday									
Time Window (h)	0		±1		±2		±3		±4	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	3	6	59	55	52	50	49	47	43	41
Execution Time (s)	24	18	20	13	21	13	22	13	23	13
Cost (×100)	5	12	129	127	127	126	120	115	98	98
	Wednesday									
Time Window (h)	0		±1		±2		±3		±4	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	3	6	55	51	46	44	42	39	37	34
Execution Time (s)	28	16	21	13	22	14	24	14	24	14
Cost (×100)	5	13	119	111	110	106	96	93	87	81
	Thursday									
Time Window (h)	0		±1		±2		±3		±4	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	3	6	63	60	55	54	51	49	47	41
Execution Time (s)	27	16	22	13	23	14	24	13	24	14
Cost (×100)	4	13	138	133	130	130	126	118	118	98
	Friday									
Time Window (h)	0		±1		±2		±3		±4	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	2	5	48	45	42	39	38	37	33	31
Execution Time (s)	26	15	21	12	21	14	21	13	21	13
Cost (×100)	3	11	108	10	99	95	87	89	79	73
	Saturday									
Time Window (h)	0		±1		±2		±3		±4	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	2	5	56	50	46	46	42	40	38	37
Execution Time (s)	23	16	18	12	21	12	21	12	20	12
Cost (×100)	4	14	126	120	118	111	104	103	103	97
	Sunday									
Time Window (h)	0		±1		±2		±3		±4	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	2	4	16	15	14	13	13	13	13	13
Execution Time (s)	21	12	21	12	21	11	21	11	22	12
Cost (×100)	3	7	32	33	31	26	32	26	23	23

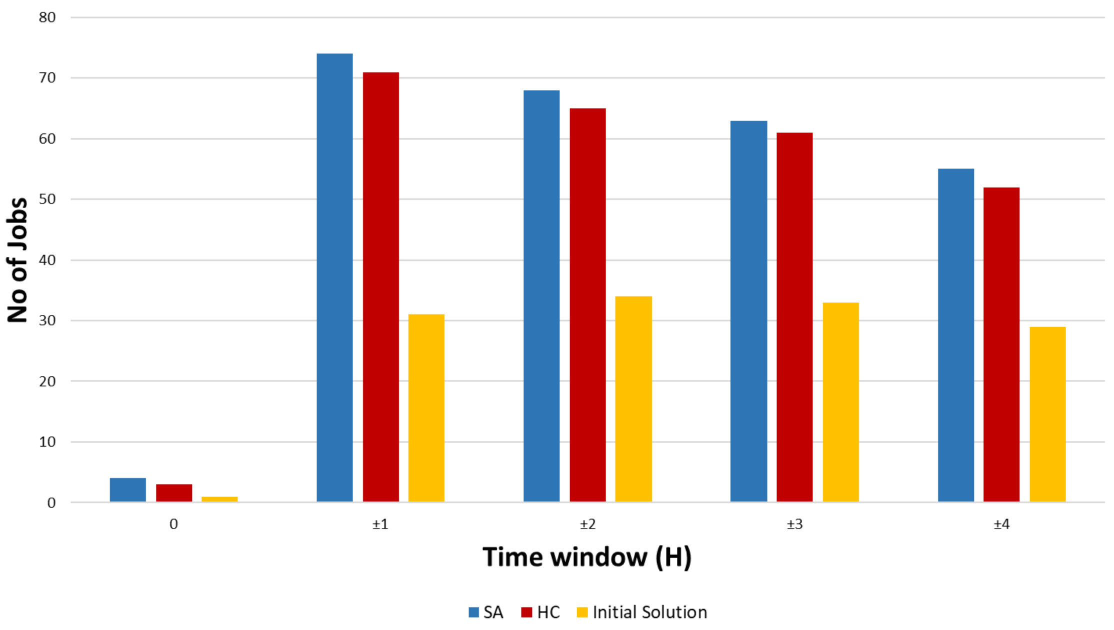


Figure 5.7. Job coverage against various algorithms.

Table 5.9. Min, max, average results against different algorithms on Monday for dataset 1.

SA – Simulated annealing; HC – Hill climbing; IN – Initial solution;

Time Window (h)	0			±1			±2			±3				±4	
Dataset	SA	HC	IN	SA	HC	IN	SA	HC	IN	SA	HC	IN	SA	HC	IN
Job Coverage															
Min	4	2	0	74	69	27	68	64	32	62	59	31	54	51	28
Average	4	3	1	74	71	31	68	65	34	63	61	33	55	52	29
Max	4	4	2	74	73	35	69	67	37	63	63	36	55	54	30

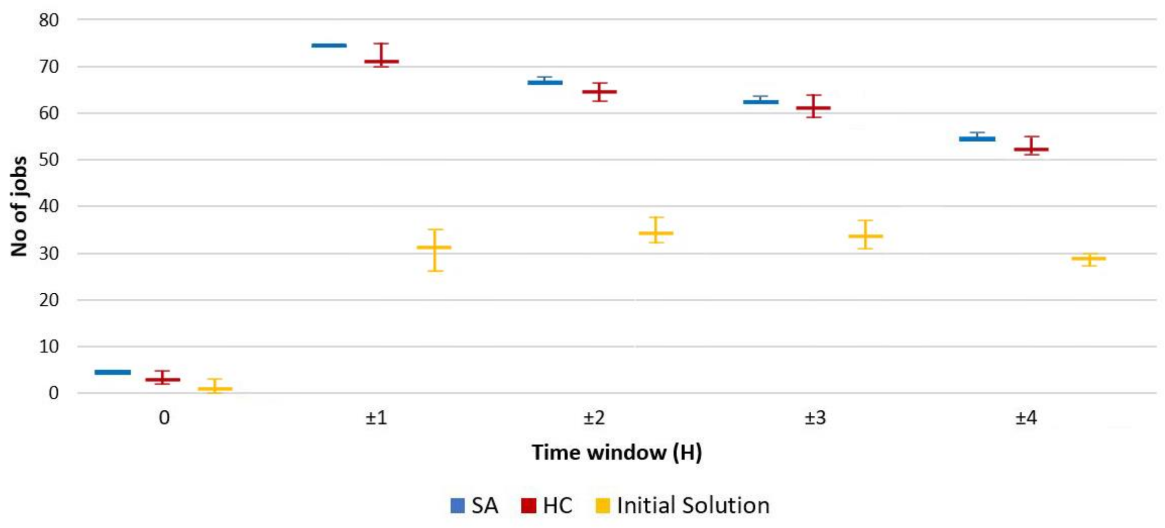


Figure 5.8. Job coverage against various algorithms min-max range.

5.2.2 Effects of the unavoidable delays and issues

More than one job can be assigned to a driver depending on his/her availability. However, if a job gets delayed due to reasons such as an accident, breakdown, traffic, and customer delay, all subsequent jobs of that driver gets affected. If the driver can take the vehicle home, as it needs to be delivered on the following day, such delays would impact jobs in the following day as well. Moreover, there are cases where the driver may suddenly become unavailable due to sickness or even without knowing a reason. In those scenarios, it is hard to assign another driver because they may be already assigned to other jobs. Thus, this could result in a chain reaction. Table 5.10 shows the number of affected jobs w.r.t. different job delays in both datasets. Figure 5.9 shows the impact of delayed jobs against different time delays. When the delay is increased, only a few jobs in the overall solution get affected as jobs are not tightly packed. Results show that only 5% and 7% of jobs get affected even with 5 hours delay in dataset 1 and dataset 2, respectively.

In some cases, VD company may get to know some unavoidable delays or issues soon after they find the next day schedule, such as driver unavailability. In these scenarios, we can regenerate the schedule without affecting to other available drivers. We checked the impact of the delayed jobs by regenerating the solution. Table 5.11 shows the impact of 5% of jobs delayed and Table 5.12 shows the impact of 10% of jobs delayed. Figure 5.10 shows the impact of delayed jobs against different delays. When 5% and 10% jobs get delayed, only 4% and 11% of jobs get affected maximumly.

Table 5.10. Impact of 5% of delayed jobs for dataset 1 and 2.

Delay (H)	1		2		3		4		5	
Dataset	1	2	1	2	1	2	1	2	1	2
No of jobs affected	1	1	2	2	3	3	3	4	4	5
% of jobs affected	1	1	3	3	4	4	4	6	5	7

Table 5.11. Impact of 5% of delayed jobs for dataset 1 and 2 with regenerating the solution.

Delay (H)	0		1		2		3		4	
Dataset	1	2	1	2	1	2	1	2	1	2
Delayed Distance (km)	-	-	285	429	463	464	380	462	406	497
Job Coverage	74	70	74	70	74	69	73	69	73	69
% of jobs affected	0	0	1	1	4	4	4	4	4	4
Execution Time (s)	205	244	206	275	226	279	203	264	215	273
Cost (×100)	170	166	171	172	160	170	162	166	162	163
Profit (×100)	205	206	210	208	205	203	207	201	207	207

Table 5.12. Impact of 10% of delayed jobs for dataset 1 and 2 with regenerating the solution.

Delay (H)	0		1		2		3		4	
Dataset	1	2	1	2	1	2	1	2	1	2
Delayed Distance (km)	-	-	807	1097	784	895	825	885	937	970
Job Coverage	74	70	72	69	68	65	69	62	68	64
% of jobs affected	0	0	2	1	8	7	7	11	8	8
Execution Time (s)	205	244	237	308	242	287	236	298	234	291
Cost (×100)	170	166	167	171	151	159	154	152	149	156
Profit (×100)	205	206	207	207	203	197	204	192	198	195

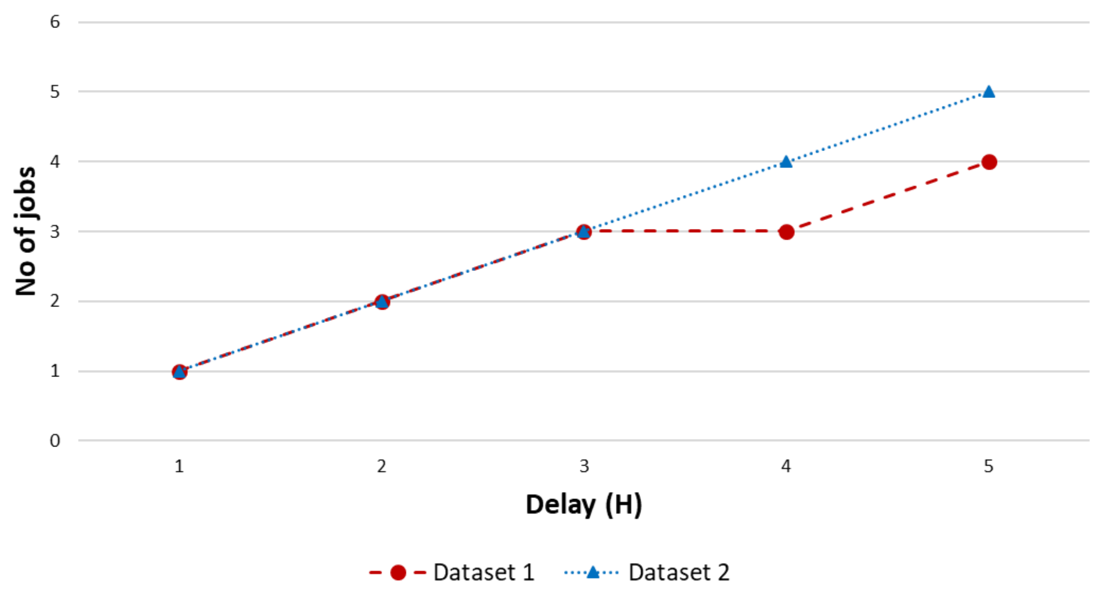


Figure 5.9. Impact of delayed jobs against different delays.

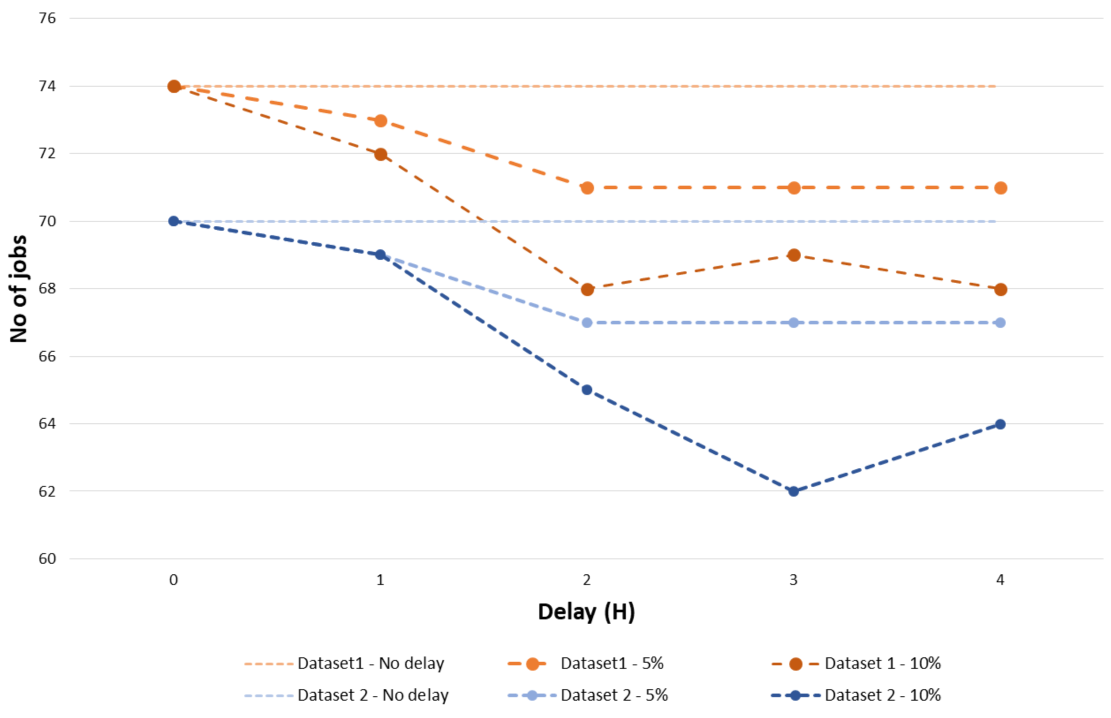


Figure 5.10. Impact of delayed jobs against different delays by regenerating the solution.

5.2.3 Effects of public transportation use of drivers

Drivers may want to use public transport to reach the vehicle pickup location and return from a delivery. The initial travel time factor set to 1.2 and Table 5.13 shows results against different travel time factors on Monday for dataset 1 and 2 with ±1H time window. Figure 5.11 illustrates job coverage variation against different travel time factors. It shows that there is no significant influence on job coverage with travel time factor and only a few jobs can be affected when the travel time factor increased. Initially, cost factor set to 0.7 and Table 5.14 shows the results against different cost factors. Figure 5.12 shows the profit against different travel cost factors. When the cost factor increases, the overall job cost gradually increases because VD company has to cover the cost associated with the driver using public transportation to return to the next job or home. Consequently, the profit gradually decreases with the increase of public transport cost factor.

Table 5.13. Results against different travel time factors on Monday for dataset 1 and 2 with ±1H time window.

Travel Time Factor	1.1		1.2		1.3		1.4		1.5	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	74	70	74	70	74	69	73	69	73	69
Execution Time (s)	228	279	224	280	222	272	226	274	221	274
Cost (×100)	170	177	173	179	163	170	154	166	155	163
Profit (×100)	207	207	206	204	207	207	210	204	211	206

Table 5.14. Results against different public transportation cost factors on Monday for dataset 1 and 2 with ±1H time window.

Travel Cost Factor	0.5		0.6		0.7		0.8		0.9	
Dataset	1	2	1	2	1	2	1	2	1	2
Job Coverage	74	70	74	70	74	70	74	70	74	70
Execution Time (s)	218	279	214	263	218	265	221	266	215	260
Cost (×100)	145	146	154	159	161	174	178	189	192	200
Profit (×100)	231	224	216	211	202	204	188	183	179	174

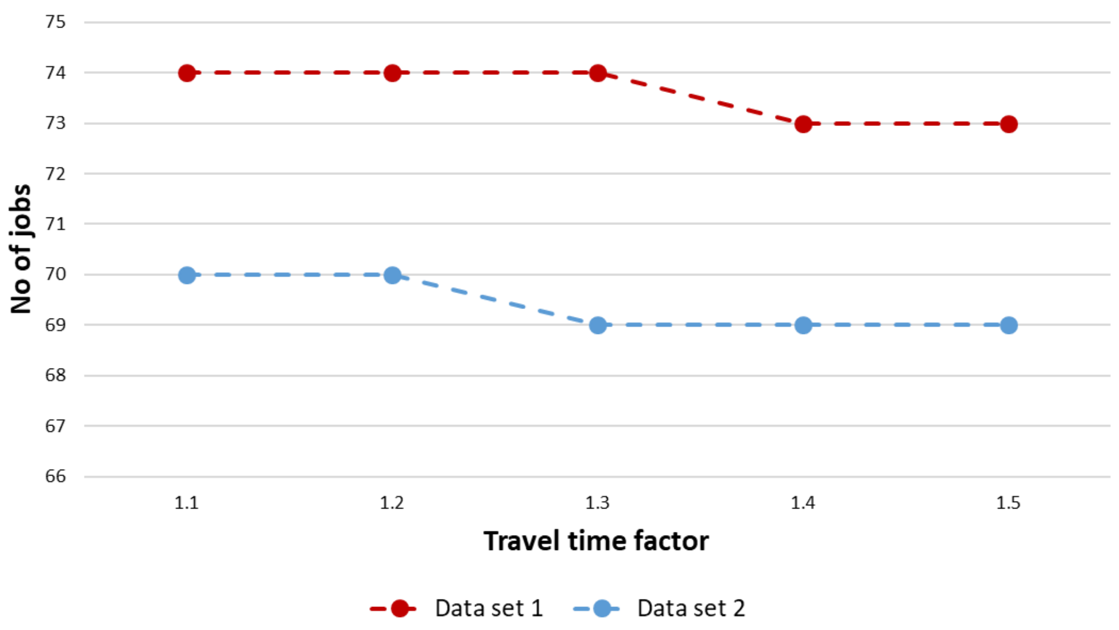


Figure 5.11. Job coverage against different travel time factors.

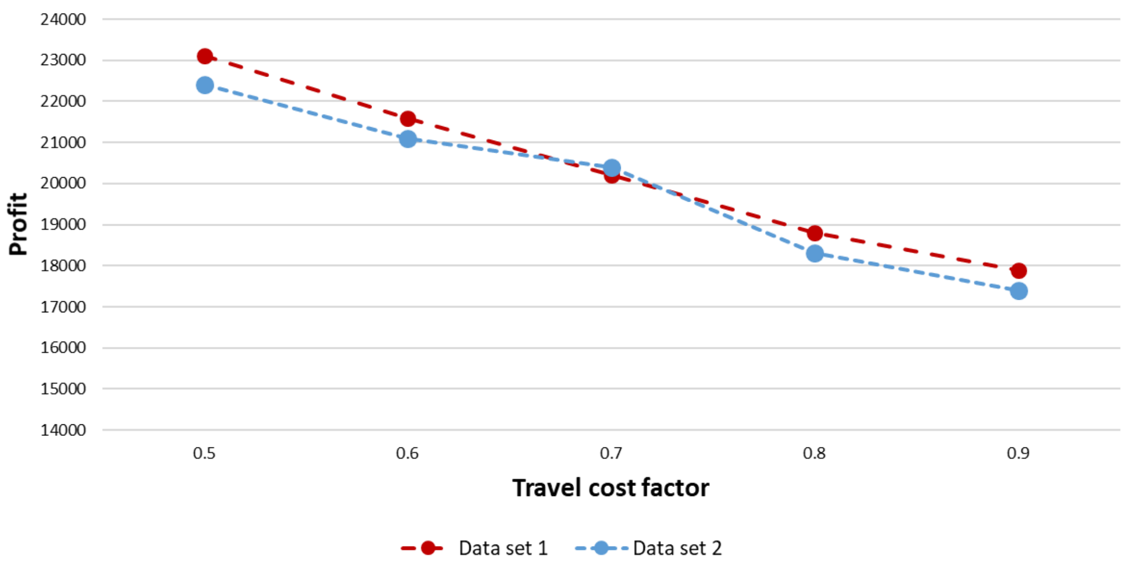


Figure 5.12. Profit against different travel cost factors.

5.2.4 Income distribution of drivers

Usually, drivers are paid only for the number of kilometers the vehicle is driven, and subsistence expenses are reimbursed. Hence, a driver's income is proportional to the total distance of assigned jobs. Figure 5.13 and 5.15 show the weekly average of a driver's income w.r.t. to his/her average weekly availability for dataset 1 and 2, respectively.

The Gini coefficient [27] is a measure of inequality of a distribution. The Gini coefficient is equal to half of the relative mean difference. The Gini coefficient is often used to measure income inequality. Here, zero corresponds to perfect income equality (i.e., everyone has the same income) and one corresponds to perfect income inequality (i.e., one person has all the income, while everyone else has zero income). The Gini coefficient can be used to measure wealth inequality. It is also commonly used for the measurement of the discriminatory power of rating systems in the credit risk management. We calculated Gini coefficient of *income : availability* ratio to inspect income distribution of drivers w.r.t their availability. Figure 5.14 and 5.16 respectively show the *income* : *availability* ratio for dataset 1 and dataset 2 respectively. Gini coefficient of all drivers is 0.194 and 0.218 for dataset 1 and dataset 2 respectively. Thus, further confirms that driver income is balanced based on their willingness to contribute.

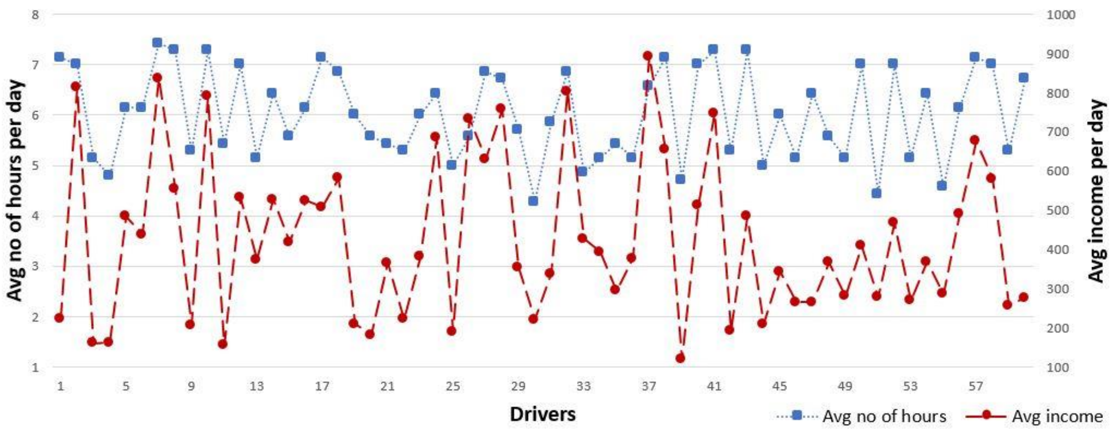


Figure 5.13. Weekly average of driver availability and income with ±1H time window for dataset 1.

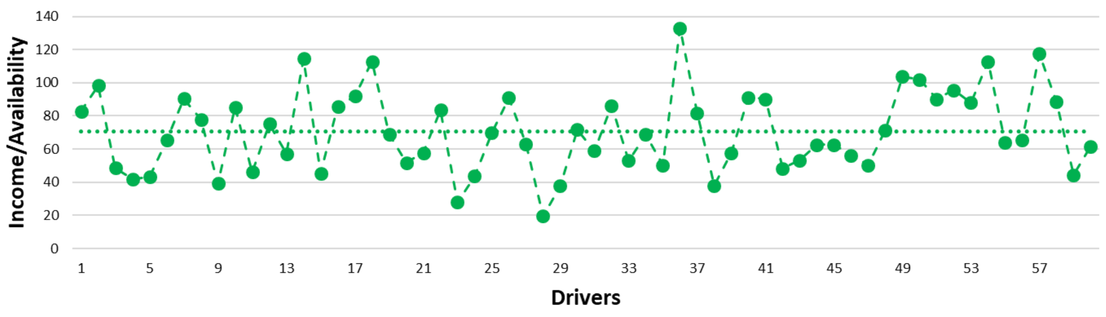


Figure 5.14. Weekly income/availability ratio with ±1H time window for dataset 1.

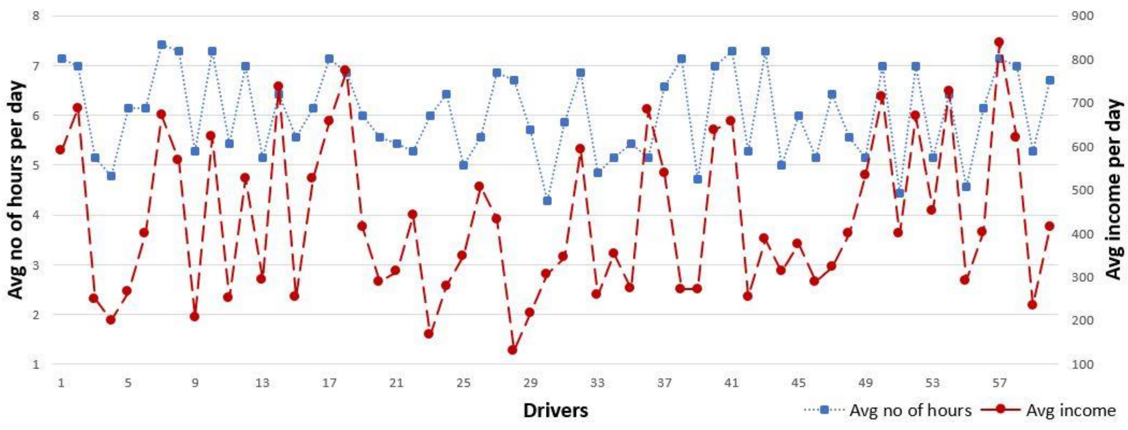


Figure 5.15. Weekly average of driver availability and income with ±1H time window for dataset 2.

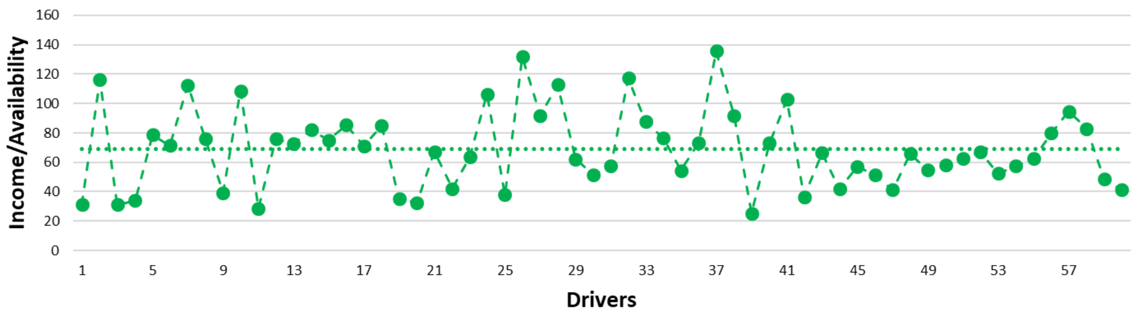


Figure 5.16. Weekly income/availability ratio with ±1H time window for dataset 2.

6. SUMMARY AND FUTURE WORK

6.1 Conclusion

Vehicle delivery is a major business in many countries where third-party drivers are used to delivering vehicles when relocate, sold, or while returning from rental cars. A vehicle delivery company works in a certain geographical area and to fit the job locations drivers are also geographically dispersed. The driver scheduling is usually carried out by an experienced scheduling manager who schedules the next day's schedule at the end of the previous day. As the number of jobs getting increased, the process becomes tiresome, error-prone, and sub-optimal as it is nontrivial to find an appropriate driver while maintaining conflicting goals of the customer, driver, and company.

Therefore, vehicle delivery companies require a reliable and scalable solution to optimize driver allocation to increase the efficiency and reduce the company's overall cost. Unlike taxis and rental cars services, vehicle delivery consists of various driver, customer, and job-related parameters, which makes the process more complicated.

We identified all related parameters and built up required constraints most of which are unique to the vehicle delivery process. We have done a literature survey to figure out existing optimization algorithms which can approach global optimality in a chaotic, noisy environment with many constraints and easily customizable. We proposed a rule and Simulated Annealing based technique for the driver scheduling problem in the vehicle delivery industry.

Simulation results were derived using a workload trace from a real vehicle delivery company. We used two different datasets each includes 80 jobs and 60 drivers, which would be a reasonable maximum traffic per day in the company. Our solution was capable of covering 82.5% and 78.5% of jobs respectively while minimizing overall cost. Moreover, drivers' income was equitably distributed according to their availability as Gini coefficient of *income : availability* for both datasets were 0.194 and 0.218, respectively.

The proposed solution was compared with two other optimization algorithms, where our solution outperformed Hill Climbing and Initial solution with better job coverage and a more stable solution. Time windows are used to tolerate unexpected delays in the process. It is also proved that only a few jobs get affected due to a delay in prior jobs. Even if we reschedule the whole jobs due to some known issues, only a few jobs get affected.

Therefore, we can conclude that our solution is capable of covering a considerable amount of jobs while minimizing overall cost and equitably distributing drivers' income based on their availability within acceptable computation time. Moreover, the solution can tolerate unexpected delays in the process without a considerable impact on the majority of jobs.

6.2 Future Work

In our solution, unavoidable delays are managed using flexible time windows. However, several practical situations are difficult to capture under a time window, e.g., excessive traffic, temporary road closed, the road under immediate construction, accidents, and breakdowns. Therefore, we need a mechanism to tolerate such events better. We believe this can be addressed by a kind of mechanism presented in [28]. The approach allocates available road space (or equivalently travel time) on designated roads to vehicles for the duration of their intended journey based on potentially prioritized requests. This approach is similar to how road space is now dedicated to public transport vehicles (via bus lanes) or multi-occupied private vehicles (via carpool lanes). If there is a mechanism to predict traffic in a particular area at a given time, our solution can be improved to consider those traffic when scheduling next day's jobs. In [29], a GPS-based traffic prediction approach using machine learning was proposed.

Customer satisfaction is the most prioritize objective in our solution. It is essential to facilitate the addition, update, and cancellation of last-minute job requests arriving at least within the day. Therefore, the solution needs to be improved further to capture last-minute job requests. We believe this objective can be addressed by a mechanism that always tries to allocate independent jobs which do not rely on already allocated jobs. Therefore, we require a mechanism to evaluate the chain reaction effect of job allocation.

Currently, our solution is only capable of finding the driver schedule when the total job set is available. It means the company already agreed to process all jobs in the next day. Although, the company may not be able to process all jobs as agreed. This is true for even manual scheduling and has a negative impact towards company's reputation. Therefore, we require a real-time driver scheduling mechanism to provide real-time feedback to the customer about the feasibility of processing the job without conflicting with already scheduled jobs. This would enable a company to accept only the jobs that can be completed within agreed timeline. Even though there are already allocated jobs; we can still change the drivers among scheduled jobs to find the most optimized solution as each driver needs to know their next day's schedule only the previous evening.

References

1. A. A. Bertossi, P. Carraresi, and G. Gallo, “On Some Matching Problems Arising in Vehicle Scheduling Models,” *Networks*, vol. 17, no. 3, 1987, pp. 271-281.

2. A. Wren et al., “A Flexible System for Scheduling Drivers,” *J. of Scheduling*, vol. 6, Sep. 2003, pp. 437-455.

3. S. Fores L. Proll, and A. Wren, “An Improved ILP System for Driver Scheduling,” *Computer-Aided Transit Scheduling*, Springer Verlag, 1999, pp. 43-61.

4. B. Laurent and J.K. Hao, “Simultaneous Vehicle and Driver Scheduling: a Case Study in a Limousine Rental Company,” *Computers & Industrial Eng.*, vol. 53, 2007, pp. 542-558.

5. S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, 1st ed. John Wiley & Sons, New York, USA, 1990.

6. F. Glover, “Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems,” *Discrete Applied Mathematics*, vol. 65, no. 1-3, 1996, pp. 223-253.

7. L. Hong, Y. Wang, L. Shi, and L. Sujian, “A Column Generation Based Hyper-Heuristic to the Bus Driver Scheduling Problem,” *Discrete Dynamics in Nature and Society*, vol. 2015, 2015, pp. 1-10.

8. E. Burke et al., “Hyper heuristics: and emerging direction in modern search technology,” *Handbook of Metaheuristics*, Springer, Berlin, Germany, vol. 57 of Int. Series in Operations Research & Manage. Sci., 2003, pp. 457–474.

9. M. Desrochers and F. Soumis, “A column generation approach to the urban transit crew scheduling problem,” *Transportation Science*, vol. 23, 1989, pp. 1–13.

10. R. S. K. Kwan and A. Kwan, “Effective search space control for large and/or complex driver scheduling problems,” *Annals of Operations Research*, vol. 155, no. 1, 2007, pp. 417–435.

11. D. L. Davis, E.L. Gillenwater, and J.D. Johnson “An Artificial Neural Syst. Framework for Delivery Truck Scheduling,” in *Proc. 23rd Annual Hawaii Int. Conference on System Sciences*, Kailua-Kona, HI, USA, vol. 3, 1990, pp 327-333.

12. M. Maghrebi, C. Sammut, and T.S. Waller, “Feasibility Study of Automatically Performing the Concrete Delivery Dispatching through Mach. Learning Techniques,” *Eng. Construction and Architectural Management*, vol. 22, no. 5, 2015, pp. 573-590.

13. C. W. Feng and H. T. Wu, “Integrating fmGA and CYCLONE to optimize the schedule of dispatching RMC trucks”, *Automation in Construction*, vol. 15, no. 2, 2006, pp. 186-199.

14. C. W. Feng, T. M. Cheng, and H.T. Wu, "Optimizing the schedule of dispatching RMC trucks through genetic algorithms," *Automation in Construction*, vol. 13, no. 3, 2004, pp. 327-340.

15. Y. Zhang, M. Li, and Z. Lui, "Vehicle scheduling and dispatching of ready mixed concrete," in Proc. *4th Int. Workshop on Advanced Computational Intell.*, Oct. 2011, pp. 465-472.

16. S. Garcia, A. Fernández, J. Luengo, and F. Herrera "A study of statistical techniques and performance measures for genetics-based mach. learning: accuracy and interpretability", *Soft Computing*, Vol. 13, 2009, pp. 959-977.

17. M. Koubâa, S. Dhouib, D. Dhouib, and A. E. Mhamedi, "Truck Driver Scheduling Problem: Literature Review", *Int. Federation of Automatic Control*, IFAC-PapersOnLine 49-12, 2016, pp. 1950–1955.

18. Google Maps API, [Online]. Available: https://developers.google.com/maps/. [Accessed: Nov. 08, 2017].

19. Y. Xiang, S. Gubian, and F. Martin, "Generalized Simulated Annealing," *Computational Optimization in Eng. - Paradigms and Applicat.*, 2017, doi: 10.5772/66071.

20. Google Distance Matrix API, [Online]. Available: https://developers.google.com/maps/documentation/distance-matrix/. [Accessed: Oct. 05, 2017].

21. E. Aarts, J. Korst, and W. Michiels, "Simulated Annealing," *Search Methodologies*, Burke E.K., Kendall G. ed. Springer, Boston, MA, 2005, pp. 187-210.

22. S. Anily and A. Federgruen, "Simulated Annealing Methods with General Acceptance Probabilities," in *J. of Applied Probability*, Sep. 1987, vol. 24, pp. 657-667.

23. Simulated Annealing [Online]. Available: http://www.cs.ubbcluj.ro/~csatol/mestint/pdfs/Busetti_AnnealingIntro.pdf. [Accessed: Oct. 05, 2017].

24. Y. Nourani and B. Andresen, "A Comparison of Simulated Annealing Cooling Strategies," *J. Phys. A: Math. and Gen*. vol. 31, 1998, pp. 8373-8385.

25. Simulated Annealing for beginners [Online]. Available: http://www.theprojectspot.com/tutorial-post/simulated-annealing-algorithm-for-beginners/6. [Accessed: Oct. 05, 2017].

26. D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by Simulated Annealing: An Experimental Evaluation; Part 1, Graph Partitioning," *Operational Research*, vol. 77, 1989, pp. 865-892.

27. Income inequality and dualism, [Online]. Available: http://www.unc.edu/~nielsen/special/s2/s2.htm. [Accessed: Oct. 05, 2017].

28. V. Cahill et al., “The managed motorway: Real-time vehicle scheduling: A research agenda,” in *Proc. 9th Workshop on Mobile Computing Syst. and Applicat.*, Napa Valley, California, USA, doi: 10.1145/1411759.1411771.

29. J. Rzeszótko, and S. H. Nguyen, “Machine Learning for Traffic Prediction,” *J. Fundamenta Informaticae - Concurrency Specification and Programming*, vol. 119, Aug. 2012, pp. 407-420.