Dynamic scheduling with stochastic customers for a real-life application with multiple depots

Master thesis

J.M. van Beem

TU Delft

Dynamic scheduling with stochastic customers for a real life application with multiple depots

By

J.M. van Beem

Master thesis

in partial fulfillment of the requirements for the degree of

Master of Science
in Mechanical Engineering

at the Department Maritime and Transport Technology of Faculty Mechanical, Maritime and Materials Engineering of Delft University of Technology

Student number:	4487400
MSc track:	Multi-Machine Engineering
Report number:	2023.MME.8841
Supervisor:	dr. B. Atasoy

TU Delft

Abstract

Efficient scheduling of services for customers poses significant challenges in various real-life systems, such as accounting for changes in resource availability, stochastic customers, and optimizing for multiple contradictive objectives. Rule-based scheduling models can be limited and produce bad results.

This thesis focuses on a case study of a company installing heating and cooling systems at individuals homes. This includes dealing with different installation difficulty levels, multiple depots, and teams with varying skill levels located at each depot. Installation completion times depend on installer skill and installation difficulty. The schedule consists of all working days of a week, excluding national holidays, and teams communicate their availability by pre-notifying vacation or off days, thus reducing scheduling options. The goal for an optimized schedule is to minimize operational cost. From this goal, the two objectives are derived: (1) minimizing transportation cost and (2) maximizing human resource utilization. Two extra requirements are constructed regarding client satisfaction: (1) adding a new client to the schedule in the range of seconds and (2) a fixed date assignment once a client is scheduled. This will prevent possible re-optimization from altering the agreed upon date with the client.

Next, scientific literature on the topic of dynamic scheduling with stochastic customers is studied. A combination of a dynamic scheduling model based on optimization in combination with re-optimization for dynamic events is derived as a promising method. Both risk minimization (RM) and chance constrained programming (CCP) are studied for providing a promising solution for the requirements and objectives. Risk minimization is chosen as the most promising method for modelling the multiple objectives. Finally, multiple solution methods, such as exact, metaheuristics and learning based algorithms are discussed. Since the problem size is relatively small, exact methods are chosen as the most promising solution method.

The methodology defines the dynamic scheduling model with a number of constraints, such as the consecutively of multi period installations and consideration for team capabilities. This mathematical model is implemented using linear programming to be solved with a MILP solver. A total of four decision variables are constructed. Two of which represent the client-team and client-period assignment and two for constructing a number of constraints. The objective function consist parts: (1) transportation cost, divided into time and distance with accompanying weight factors and (2) a risk cost function with accompanying weight factor to maximize the human resource utilization. Next the re-optimization model is defined, which is based on the dynamic scheduling model. A constraint for client-period assignment is implemented in order to ensure the fulfillment of the second requirement. Another two decision variables are added to ensure only scheduling changes lowering the objective function can be executed. Finally, the real-life implementation of both models is discussed in detail, including proposing structures with multiple modules.

A number of experiments are designed in order to investigate the performance of the proposed methods on real-life and synthetic scenarios. The first experiments for the dynamic scheduling model vary the number of teams and depots, gathering insights into the models performance. Next, a promising risk cost function is determined by a pareto analysis and investigating the impact of different risk cost functions. The last experiments regarding the dynamic scheduling model increase the problem size up to 250 clients, 7 depots and 18 teams. Next, the performance of the re-optimization model is investigated by re-optimizing a selection of schedules produced by earlier experiments. The optimal deterministic solution is discussed, followed by recommendations for real life implementation.

The first results showed the potential of the proposed models by decreasing the travel cost by up 17 percent, but also emphasized the importance of trade-off between the multiple objectives. The analysis of the risk cost function showed the most promising cost function to be the logarithmic function. Next, the larger experiments showed an improvement of up to 25 percent in terms of travel cost, and a reduction of 21 periods of unused human resources. The re-optimization model showed a decrease of up to 6.7 percent for travel cost. The computational performance for both models exhibits a close to quadric increase in time when increasing the number of teams. Finally the complexity of finding the optimal deterministic solution is emphasized by the inability to find the solution withing 22 hours of computational time.

The methodology and results are discussed with the most important points of discussion being the possibility for significantly different results with different client data sets and the lack of consideration of the risk cost function for the re-optimization model. Future recommendations include the implementation of flexible clients, the grouping of clients and a variable cost function per team as the most promising directions.

Preface

This thesis serves as the final project for completing my masters degree at TU Delft. I have learned a lot managing a project this size. It was an insightful experience to see the project take shape as I made progression. The goals and results of the project were adjusted when new insights or information was revealed. There were times I struggled, but in the end I enjoyed the processes of overcoming the challenges and improving each time.

I want to thank Bilge for her excellent guidance during the entire project. She provided useful insights, and we could have discussions about difficult constraints and implementation of the models. Her feedback on my ideas, thesis and models helped me to get a clear direction of execution for this work. I want to thank Sander Wapperom from the company DeWarmte B.V. for the opportunity to do a case-study at his company. I was given the trust and responsibility to implement the first version of my model into the company workflow after only a couple of weeks of development. I learned a lot by using the models to get direct results and feedback from other employees. Lastly, I want to thank Yimeng Zhang for his critical questions and useful feedback during official meetings.

Joël van Beem
Delft, July 2023

Contents

1 Introduction

Efficient scheduling of services for customers poses a big challenge in real life situations for various systems. Changes in resource availability, predicting future customer requests and customer cancellations are some examples. Stakeholders can have different and sometimes even contradictive requirements for an optimized planning. Finding the optimal solution for such a planning should include all of the requirements and constraints. Manually optimizing such a schedule is possible when the problem size is small and the number of constraint or requirements is low. For larger or more complicated problems a computer model is needed to optimize the planning. A rule-based method can be used, but the limitations of such a model can lead to a sub optimal planning.

1.1 Scope

The problem definition for this thesis is based on a company installing heating and cooling systems at individuals homes. Different type of installations are carried out by the company, resulting in different installation difficulty levels. Multiple depots are available where the systems are stored and the teams depart from. The workforce consists of multiple teams with accompanying vans, each team located at one of the depots. Teams can have a different level of experience or skill level. Installations can have a different completion time, caused by the skill level of the installer, or the difficulty level of the installation. The schedule consists of all the working days in a week, excluding the national holidays. The teams can determine their own availability by communication their vacation or off days in advance. This reduces the number of available options in the schedule.

One of the companies goals is generating profit, as a result, the minimization of operational cost is one of the goals for an optimized schedule. This cost is simplified by breaking it down into two main contribution factors: transportation cost and human resource cost. Minimizing the transportation cost can be done by reducing the time and distance traveled by the vans. The human resource cost is constant, even when no work is available for the installers. As a result, the maximization of human resources is the second goal for minimizing the operational cost. The combination of these two goals results in a multi objective scheduling problem. The two objectives are as follows:

- Minimize transportation cost
- Maximize human resource utilization

Another important aspect of creating an schedule is customer satisfaction. The goals for this aspect are scheduling the client within seconds and not changing the assigned day(s) once the client is scheduled. This results in two extra requirements listed below

- The scheduling of a new client must be realized within seconds after receiving the request
- The assigned period of a client cannot be changed by re-optimization

1.2 Research questions

A main research question is derived based on the scope. Sub questions divide the main research questions into smaller sections, to be answered individually in this thesis.

"How can the dynamic scheduling with stochastic customers be solved efficiently using a real transportation network ?"

In the way to answer the main research question, the following sub questions are considered:

1. What is a promising method for the dynamic scheduling problem to be modelled?
2. How can the dynamic scheduling problem be solved efficiently?
3. How can the re-optimization of the schedule be modelled and solved?
4. What are the requirements for integrating the models with a real life system?
5. What is the performance of the proposed method compared to the rule-based model under different scenarios?

1.3 Methodology

The scientific methodology applied for developing the proposed methods is shown in Figure 1. After determining a promising type of scheduling model, the model is designed and tested. This iterative process resulted in the proposed final model presented in Chapter 3.

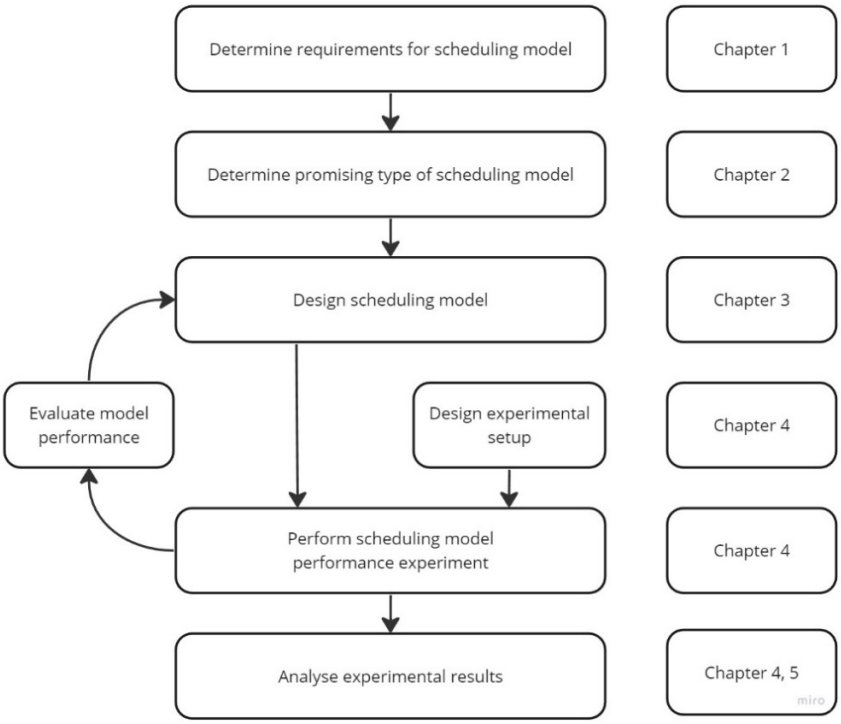

Figure 1: Scientific methodology for the scheduling problem

1.4 Thesis outline

First a literature review in Chapter 2 is conducted to investigate the current proceedings in the scheduling field. Relevant variants with their accompanying solutions and future directions will be discussed. In Chapter 3 the current rule based scheduling solution is explained and its shortcomings elaborated. Chapter 3 continues with the new optimization model for both the dynamic client scheduling and the re-optimization of the schedule. To incorporate the models into the real-life environment, a structure is proposed for the communication, integration and data storage. To evaluate the models, numerous simulations are done in Chapter 4, both with real and synthetic scenarios. The results, including important findings are discussed in Chapter 5, followed by future recommendations and a conclusion.

2 Theory on modelling approaches

The logistics industry is rapidly gaining importance with the increasing connectivity of our digital world. To meet this demand, scheduling problems (SP) aim to optimize the allocation of resources for a set of clients to a set of vehicles or teams. Each client can have different attributes such as: type , travel time and procession time. Generally, solving this problem to optimality can result in a 5-20% reduction in operational cost (Toth & Vigo, 2002). Scheduling can be divided into two classes, predictive and reactive scheduling (P. Burke & Prosser, 1991). In predictive scheduling all information about the clients is deterministic and known in advance. On the other hand, in reactive scheduling not all the information is known in advance, such as the arrival time of new clients or their service time. This increases the difficulty to model and solve such a problem.

Trends in research include the minimization of global warming or reducing emissions (Nura & Abdullahi, 2022). Another trend is finding good quality solutions within a reasonable time by developing efficient solution algorithms. The SP is a NP-Hard problem (Lenstra & Kan, 1981), meaning it gets increasingly more difficult when the number of vehicles and/or clients increases. This led researchers to study innovative methods for solving the SP, such as meta heuristics and learning based optimization, with genetic algorithms being the most widely used method (Chaudhry & Khan, 2016).

Most research test the feasibility and effectiveness of the proposed model and algorithm on standardized or adapted test cases. This leads to an easy comparability between different models and solution algorithms. A big opportunity lies in applying the models and solving methods to a real-life case to investigate its effectiveness and value to real-life problems.

This literature review is performed considering the problem of a company that needs to install systems at individuals' homes. Installations are executed with multiple vans, with visits planned for a single or multiple days. The vans have multiple depots to depart from. The scope of the literature review is determined based on this explained case and the relevant variants of the SP are investigated together with the solution methods proposed for them. The objective of this literature review is to get a clear understanding of the current solutions to relevant problems in the literature.

First the fundamentals of the scheduling problem are explained in detail, to be extended with the relevant variants according to the defined scope. For each variant a comparison is made, explaining the tradeoffs and benefits for each variant. Next, the solution methods for solving the scheduling problem are explained and compared. Throughout the review, gaps and future directions will be explained and elaborated on.

2.1 Variants of the scheduling problem

The scheduling problem is defined by the allocation of resources in order to minimize or maximize an objective function. There are three main categories of approaches to the scheduling problem: conventional, rule-based and distributed solving (Suresh & Chaudhuri, 1993). The conventional approach consists of developing a mathematical model and optimizing for the objective function. Rule based is constraint-directed and rule-based. Those rules are often found by simulation or experimentation. Lastly, distributed solving implements uses a multi-agent approach. Each agent has its own set of tasks and responsibility. Construction of different layers with agents using bi-directional communication often part of the implementation. Conventional modeling arises as the most promising approach to the scheduling problem defined in chapter 1.1. This choice is based on the unfavorable results produced by previously developed rule-based model. Distributed solving is often used for more complex processes, involving multiple decision layers.

2.1.1 Multi objective problems

The objective function often includes minimizing the total completion time of all the tasks to be done (Hartmann & Briskorn, 2022). Other frequently used objective functions can include the maximization of resource utilization or the minimization of total operational costs. A multi objective function often has multiple contradicting objectives. An example of this is both minimizing the environmental impact and maximizing profit. In almost all cases, if the environmental impact is minimized, the profit will also go down. To mitigate this problem, a tradeoff needs to be made between the different objectives.

A Pareto set is often used to get useful insights into the tradeoffs between the different objectives.. Each solution in this set is non-dominated, meaning that for all the solutions inside of the pareto set, there exists no solution that can improve on all objectives. The pareto frontier, see Figure 2, is a graphical representation of this set, showcasing the optimal tradeoffs between the different objectives. Using this pareto frontier has multiple benefits, such as the trade-off analysis or a sensitivity analysis. The latter can be used to asses changing parameters in the model and evaluate how the frontier shifts. Useful insights can be extracted and informed adjustment can be made.

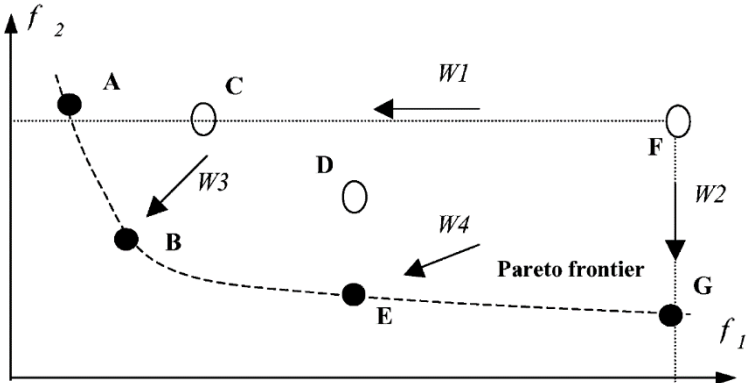

Figure 2 Pareto frontier (Kacem et al., 2002)

2.1.2 Stochastic scheduling

In many real life applications, not all of the information about the SP is known in advance resulting in a stochastic SP (SSP). Travel time, new customer arrival time and the customer job duration are some examples of stochasticity in the SP. It is important to make a distinguishment between dynamic and stochastic information. Dynamic information only becomes available during the execution of the schedule, for example, the next customer cancelled their order unexpectedly. Stochasticity indicates the uncertainty of the data itself, e.g., the travel time between client 1 and 2 is not known exactly. In other words, stochasticity is about anticipation to uncertainty and a dynamic problem is about reacting to changes in the environment. Reacting to changes can be real-time, but is often at specific time points. A multi period schedule could be reoptimized at the end of each period, based on information collected during the previous period. To make matters a bit more complicated, the dynamic information can also be stochastic. For example, dynamic information, revealed during the execution of an schedule could be: the completion time of the current customer is expected to take one to two hours longer. To summarize:

- Stochastic = take the uncertainty of the environment into account
- Dynamic = react to changes in the environment
- Stochastic- Dynamic = react to changes in the environment while taking the uncertainty of the environment into account

In this chapter, only the stochastic nature of the client data such as, travel time and the arrival time is taken into account.

Literature shows two different methods for modeling stochasticity: Risk minimization (RM) and Chance constrained programming (CCP) (Ehmke et al., 2015). A simple overview is shown in Figure 3: Two methods of modelling stochasticity: RM and CCP.

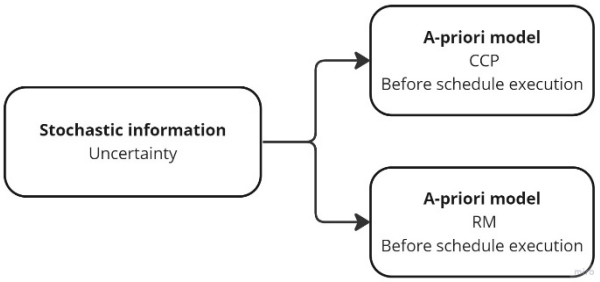

Figure 3: Two methods of modelling stochasticity: RM and CCP

2.1.2.1 Risk minimization

The risk or probability of the uncertainty in data causing a constraint to be violated can be calculated based on the probability distribution on the data. This risk can be included into the objective function in order to minimize it. Combining this risk minimizing objective with the operational cost minimizing objective results in contradiction. As an extreme example, if one want to minimize the risk of open spots in the schedule, one could always plan a new client into the spot closest to today. This will result in the other part in the objective function to be much higher. A practical example of implementing RM in scheduling is the use of a cost function (Im et al., 2013). This cost function can relate the risk of keeping an open spot to other parts of the objective function, such as travel time and distance. Tuning this cost function, either by simulation or calculation is of great importance to the optimization of the entire scheduling problem.

2.1.2.2 Chance constrained programming

With change constrained programming, the probability of constrain violation is bounded. This method ensures clients are not exposed to risk above a certain level. A possibility with this approach is to distinguish between premium and standard customers with a difference in allowable risk exposure. Compared to RM, CCP is much more computationally intensive because of the extra constraint(s), while RM only adds to the objective function.

Table 1: Comparison for modelling approaches for SSP

	Pro	**Con**
RM	-Lower computational time -Easier to implement -Easier to combine with other constraints	-Possible big risk variation between customers -No bounds on maximum risk
CCP	-Amount of maximum risk can be controlled	-More complicated -More computational intense

2.1.2.3 Challenges in stochastic scheduling

Most research is focused on testing the developed algorithm on benchmark or generated problems (Chaudhry & Khan, 2016). A big opportunity lies in solving a practical problem and testing its solution. Another gap and promising future direction is increasing the number of constraints in the scheduling problem (Miyata & Nagano, 2019). Another gap and future direction is the use of multi-objective functions (Miyata & Nagano, 2019). When taking stochasticity into account, it is of great importance that the data and distributions used for the modal are of good quality. If the data can not be reliably fitted to a distribution or model it can often even make the scheduling model perform worse than not including the stochasticity (Pitt & Myung, 2002).

2.1.3 Dynamic scheduling

Reacting to the changes in the environment can be done by making the scheduling problem dynamic. Examples of such dynamic events are a change in the constraints of the resources or an increase in availability. This new information can be used to execute a re-optimization of the schedule. Another dynamic aspect of the scheduling problem is the dynamic reveal of job or client arrival times. This results in the need for inserting this new client into the new schedule. This dynamic client arrival differs from the other dynamic events, because there is no need to re-optimize the entire schedule. There are different methods to determine when to update the model and what to do when the model is updated. Both are explained and compared in this chapter.

2.1.3.1 Re-optimization

Re-optimization can be done for adjusting the current schedule to better fit the newly available information. This could be the change of a resource constraint, a client cancelling or an increase in availability of one of the installation team members. The first question we need to ask ourselves is: what do we want to re-optimize?

Regarding the scope of this literature review, only 1 parameter is suitable for re-optimization. This parameter is the assignment of a client to a specific installation team. The period or installation date in other words is already communicated with the client and cannot be changed anymore. The second question we need to ask ourselves is: when to re-optimize?

Literature shows four main methods for when to re-optimize the model (Zhang et al., 2022). The methods are explained below to be compared in Table 2.

Periodic update – A period is defined which splits up the schedule in multiple periods. This method makes the problem effectively a normal schedule. Defining a new problem at the beginning of each period. Periodic update was first implemented by Kuo et al., (2016)

Dynamic update – When new information is available, the model is immediately reevaluated. One could argue that this method had the most reevaluations to do, but if there is only one new piece of information, the periodic update will have more reevaluation moments. Dynamic update was first implemented by Taillard & Badeau, (1997)

Customer update – When the clients job is finished, the re-optimization is triggered. This method ensures re-optimization only when no client is being served. This could improve safety and communication between the driver and the planners. Customer update is applied by Campos et al., (2008) and Pillac et al., (2012)

Key point update – New information is collected at predefined key points. An example of such a key point could be at break times for employees driving the vehicles or running the machines. Maybe some repairs at the machine are dynamically revealed just before the break . Updating at a strategic key point could also have a time advantage for calculations the best schedule, if the point is combined with lunch time for example, a calculation time of 10 or 20 minutes is acceptable in that case. Key point update is implemented by AbdAllah et al., (2017)

Table 2: Comparison for dynamic update moment for DSP

	Pro	Con
Periodic update	-Easy adjustment for the number of updates -Known update time	-Receive crucial updates late -Hard to predict update location
Dynamic update	-Respond direct to update	-Likely to require the most computation time
Customer update	-Predictable update place -Easier implementation for the driver side	-Update intervals can vary widely – high/low distance -Receive crucial updates late
Key point update	-Choose the best point (time, location) to update	-Receive crucial updates late

2.1.3.2 Dynamic job arrival times

Another aspect of dynamic scheduling is the possibility of dynamic job arrival times. This means that the number of jobs and when they arrive are not known in advance. This greatly increases the difficulty of the scheduling problem. This is the real-life situation in most cases. Multiple methods can be adopted to solve this challenge (Mohan et al., 2019).

Stochastic information about the number of jobs and expected arrival times can be used to make informed decisions when not all of the information is know in advance. It is of great importance that the information used in the model is of good quality.

Job prioritization can be implemented to make sure dynamically arriving high priority job can be executed when they arrive. This would mean low priority job will be put on hold for a certain time in order to leave resources available for a possibly arriving high priority job.

A online scheduling model will make a decision once a new job arrives with the limited information available. This means scheduling the job as soon as it arrives. This could result in a non optimal schedule because of the lack of overall schedule information being used.

2.1.3.3 Challenges in dynamic scheduling

One of the big challenges in dynamic scheduling is finding a solution within reasonable computational time (Hartmann & Briskorn, 2022). These re-optimization calculations for the dynamic problem often need to be done within the order of minutes and sometimes even seconds. Mohan et al., 2019 states that a big challenge and opportunity lies in the combination of exact algorithms and heuristics for solving the DSP. The algorithms could benefit from each others advantages. Another challenge is making the DSP networked (Mohan et al., 2019). This means creating multiple agent that are interconnected and communicate with each other using a network. These agents could represent processes, machines or many other parts of the problem. The agents make their decisions based on constraints and interdependencies between agents.

2.2 Solution methods for the scheduling problem.

Firstly, the different types of solving are explained, to be extended in their own chapters with more detail on useability and challenges. Lastly the different approaches are compared. Since the SP is closely related to the Vehicle routing problem (VRP) references and solutions from both problems are used

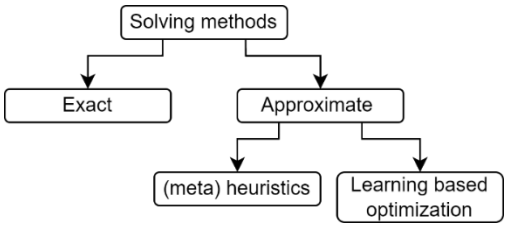

Figure 4: Classification of solving methods for the SP . Adapted form (Li et al., 2022)

Approaches to solving the SP can be divided into two main categories: Exact/optimal solutions and approximate solutions as seen in Figure 4. Exact solutions are generated by an exact algorithm, resulting in the best solution possible. This type of a method is computationally intensive, therefore approximate solution methods were developed for large scale problems. An approximate solution is generated for example by a (meta)heuristic algorithm or a learning-based optimization (LBO) algorithm. Heuristics make use of problem specific tricks to get very close to the optimal solution in the fraction of the time of exact algorithms. LBO uses machine learning to learn a mapping function from the input to the output of the VRP. A simple comparison on a capacitated SP with 20 clients can be seen in Table 3. Note that this data is based on a single instance, results can vary greatly based on problem size and complexity, but it illustrates the huge difference between exact and approximate solutions in terms of calculation time.

Table 3: Comparison of different solution methods (Li et al., 2022)

	Objective (lower is better)	Calculation time (s) (lower is better)
Exact (ILP)	5.74	1800
Approximate heuristic (LNS)	6.12	1.25
Approximate LBO	6.41	0.25

2.2.1 Exact solution methods

An exact solution is generated by an exact algorithm, such as direct tree search or integer linear programming (ILP). In simple terms, an exact algorithm checks all the possible solutions to exactly find the best and optimal solution. This method takes a lot of time which increases exponentially when the problem size increases, since the SP is an NP-Hard problem (Lenstra & Kan, 1981). When the problem size increases beyond 200 clients, with only a capacity constraint, exact methods are struggling (Pecin et al., 2017).

Research into exact algorithms can be divided into two categories: decreasing computational time and increasing the complexity of the SP. Pecin et al., (2017) adopted a problem specific branch-cut-and-price (BCP) algorithm to solve a SP with up to 360 clients, almost doubling the previous limit of 200 clients. This result was achieved by combining different improvements by different authors on the algorithm. A generic BCP algorithm was proposed by Pessoa et al., (2020), able to solve many different types of SPs. Which showed very good results, in some instances even better than exact problem specific solvers. Computational time is also greatly reduced, a 100 client SP which took a couple hours a few years ago, can now be solved in minutes (Pessoa et al., 2020). SPs with many constraints are researched at a small size. This research can give insight on the optimal solution and how to develop heuristics approximating the optimal solution. High performing exact algorithms are difficult to extend when the problem formulation changes (Fakhravar, 2022). This could make the implementation challenging when a company wants to extend the algorithm in the future.

2.2.2 Approximate solution methods

This section studies the relevant approximate solution methods for the dynamic scheduling problem. First, heuristics are investigated followed by the combination of metaheuristics and exact algorithms. Finally the learning based algorithms are studied.

2.2.2.1 Heuristics

Heuristics are simple, rule-based strategies that can be used to solve problems quickly and efficiently. They involve making decisions based on experience or intuition rather than a rigorous analysis of all available options. Meta heuristics are more complex problem-solving techniques which use heuristic methods as building blocks for generating higher quality solutions, generally improving 3-7% (Laporte, 2001). Meta heuristics employ additional search techniques such as simulated annealing, tabu search and genetic algorithms in order to explore the solution space more thoroughly and find better solutions than those found using basic heuristic methods alone. A very simple example of a heuristic rule: the 2 client points furthers away from each other probably won't be on the same route. Rules like this nudge the search in a certain direction, greatly reducing the search time. However, it is unknown if the solution found is also the optimal solution. To combat this problem, many researchers (Li et al., 2022), (Daniele & Paolo, 2014) compare the solutions obtained by the heuristic to the known optimal solution. For new problems, with unknown optimal solution an estimation of the optimality gap can be made.

Heuristics are mostly problem specific since they rely on rules, based on certain characteristics of the SP. Most heuristics only tackle up to 3 variants at the same time (Konstantakopoulos et al., 2022), but Penna et al., (2019) developed an algorithm for 6 variants, including backhauls, multiple depots, split deliveries, open routes, duration limits, and time windows.

A big challenge is the development of a (meta)heuristics able to solve a wide varieties of SPs (Vidal et al., 2020). Furthermore, multi constraint, multi objective SPs remain a big challenge to solve with (meta)heuristics (Zhang et al., 2022)

2.2.2.2 Combining metaheuristic and exact algorithms

Heuristics and exact algorithms are combined in order make use of the advantages of both type of algorithms. Puchinger & Raidl, (2005) proposed two different classifications: Collaborative – and integrated combinations. With collaborative combination, information is exchanged between two or more algorithms. Klau et al., (2004) implemented the following: firstly finding non optimal solutions with a metaheuristic algorithm. Next, using the non optimal solution as a starting point, an exact algorithm is used to find the optimal solution. This way, speed (the advantage of the metaheuristic) and precision (the advantage of the exact algorithm) are used. Integrated combination consists of one master algorithm, this can be both an exact or a metaheuristic algorithm. To improve certain parts or steps within the master algorithm, one or more slave algorithms are implemented. Burke et al., (2001) implemented an exact algorithm in the local search part of a local and variable neighborhood search algorithm. This approach results in a known optimum within the local search.

2.2.2.3 Learning based optimization algorithms

Learning based optimization algorithms use machine learning, a form of artificial intelligence (AI) to approximate a mapping function between the in and output of a SP. This is a relatively new field compared to the other methods and it can be divided into step-by-step and end-to-end approaches. The former is generating a solution and iteratively improving it whereas the latter directly outputs a feasible solution. This approach is relatively new and is currently only suitable for small scale problems with one or two constraints. Another implementation of LBO is selecting or generating a heuristic (Li et al., 2022), this is considered as a hyper-heuristic. Okulewicz & Mańdziuk, (2020) implemented such a hyper-heuristic to solve a dynamic SP with an improvement of 2,8% over pervious methods.

2.3 Summary

The literature review explores the scheduling problem (SP) witch a focus on dynamic scheduling with stochastic customers for a real-life application. The review discusses multiple aspects of the SP, including different variants, solution methods, its relevance and challenges.

The logistics industry is gaining importance in the digital and connected world, and the SP aims to optimize the allocation of resources to meet the increasing demand. Solving the SP to optimally can result in a significant reduction in operational costs and increase efficiency. The SP can be divided into two classes: predictive scheduling, where all information known in advance, and reactive scheduling, where not all information is known in advance, making the problem more challenging to model and solve.

Trends in research include minimizing global warming and reducing emissions, as well as developing efficient models accompanied by solution algorithms to find high-quality solutions within a reasonable time. The SP is known as an NP-Hard problem, leading researchers to explore innovative solution methods such as metaheuristics and learning-based optimization, with genetic algorithms being widely used. Most research in the field tests proposed models and algorithms on standardized or adapted test cases for comparability. However, there is a significant opportunity to apply these models and solution methods to real-life cases to investigate their effectiveness and relevance.

The literature review focuses on a specific case of a company installing heating and cooling systems at individuals' homes using multiple vans and depots. The scope of the review includes relevant variants of the SP and the solution methods proposed for them. The review outlines the research approach, starting with an explanation of the fundamentals of the scheduling problem and its relevant variants. It then discusses solution methods for the SP and compares them. Throughout the review, gaps and future directions in the literature are identified and discussed. The stochastic nature of the scheduling problem is explored, highlighting the relevant aspects.

Two methods for modeling stochasticity, risk minimization (RM) and chance constrained programming (CCP), are discussed and compared. The challenges in stochastic scheduling are highlighted, including the lack of real-life problem-solving and the need to increase the number of constraints in the scheduling problem. The use of multi-objective functions and the importance of high-quality data for modeling stochasticity are also emphasized.

Dynamic scheduling is addressed, considering the reactions to changes in the environment and the dynamic job arrival times. Different methods for re-optimizing the model are discussed, along with the challenges of finding solutions within reasonable computational time. The literature review concludes by discussing the different approaches to solving the scheduling problem, including exact/optimal solutions and approximate solutions. The advantages and challenges of each approach are examined and compared. Furthermore, references and solutions from the Vehicle Routing Problem are also considered.

Overall, the literature review provides insights into the scheduling problem, focusing on dynamic scheduling with stochastic customers for a real-life application. It explores relevant variants, solution methods, challenges, and future directions, aiming to contribute to the understanding of the current models, solutions and gaps in the literature.

3 Methodology

The rule-based model is first explained, followed by its shortcomings. A dynamic scheduling model is proposed to provide a solution for these limitations. Different types of risk cost functions are developed to optimize the trade-off between multiple objectives. Finally, a re-optimization model is proposed in order to further improve the scheduling performance. After defining these models, recommendations on the implementation are given. This implementation refers to when to do the re-optimization and the integration of the proposed models into a real-life system.

3.1 Rule-based scheduling model

The rule-based model works based on a set of conditions. Those conditions are based on the constraint deducted from the limitations and parameters of the clients and the installation teams. All of the open spots in the schedule are first retrieved from the open schedule with all the information of the new client. Next the conditions are applied to all of the open spots to reduce the number of options to one. The five conditions shown in Figure 5 are explained below.

Condition 1: The first condition checks if the distance and driving time to a client does not exceed the limitations of the teams. Condition 1 also ensures the lowest time and distance are selected, including distances and times within a certain threshold. Simulations with earlier versions of the rule-based model showed undesirable results for not including this threshold.

Condition 2: The second condition checks if each team possesses the required skill level to match or exceed the difficulty level of the client.

Condition 3: The third condition checks if the length of the client installation does not exceed the maximum installation length the team can execute.

Condition 4: The fourth condition picks all of the earliest dates in the remaining options. If there is only one option, condition 5 is skipped and the suggestion is sent straight to the customer.

Condition 5: The fifth and last condition calculates the overall team utilization, since fair distribution among the teams is required. If there are multiple options remaining from condition 4, the team with the lowest utilization is selected to execute the clients installation.

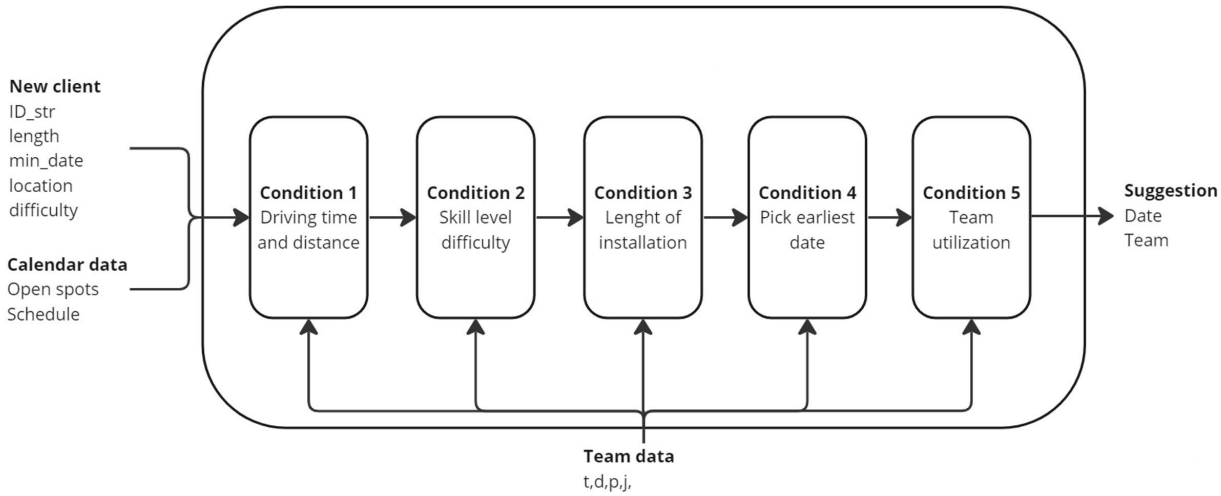

Figure 5: Rule-based scheduling model structure

3.1.1 Limitations of the rule-based model

Multiple problems arise while using and testing the rule-based model. Since the model only checks one condition at a time, it is impossible to optimize for multiple objectives at the same time. We could take an extreme case as an example:

- Two teams are available, one at the Utrecht depot and one at the Delft depot
- All new clients are significantly closer to the Utrecht depot, exceeding the location threshold

With this example, no open spots for the Delft team will get accepted by Condition one. Therefore all new clients will get assigned to open spots belonging to the Utrecht team. This results in the Utrecht team having a full schedule and the Delft team a complexly empty schedule. This of course is an undesirable result. The threshold somewhat mitigates this problem, but it can still have a significant negative impact on the overall scheduling performance.

Another problem arises regarding the minimization of overall travel time and distance. Condition 4 simply picks the earliest date, even though another location within the threshold will result in a lower time and distance

The final problem is regarding the risk of unutilized human resources. The rule-based model has no tools to make a consideration between reducing the travel cost and maximizing the human resource utilization. To summarize, the problems with the rule-based model are:

- Poor results in extreme client cases
- No consideration between risk of lowering human resource utilization and minimizing time and distance

3.2 Dynamic scheduling model

In order to address the limitations of the rule-based model, a dynamic scheduling model based (DSM) on optimization is proposed. The dynamic scheduling model is first formulated as a mathematical model with accompanying indices, (sub)sets, variables and constraints. An open-source mixed integer linear programming (MILP) exact solver called PULP is selected as solution method. This requires all the constraints to be linear. The constraints are implemented one by one in order to iteratively develop the model.

3.2.1 Indices and (sub)sets

Firstly, the client ID i is constructed, it consists of a client number m and a type or difficulty j as shown in Table 5. This notation is chosen for the model and solver to be able to distinguish between clients with the same number but with a different type. The teams are indexed by s with the total number of teams denoted by k. The model considers a planning horizon denoted by f with periods denoted by p. In order to formulate a number of consecutively constraints, multiple subsets of p, indexed by team s are created. In combination with the alternative periods q_s, these sets will be used for the consecutive constraints for multi period installations. Set Q_s includes one up to $e_s = f$ or smaller and is based on the input variable w_{sp}. The construction and relation between the period sets P, $\widehat{P}_s$ and Q are best explained with a simple example, shown in Table 4. The purpose of $\hat{P}_s$ is relating q_s to p, the first entry in the subset $\hat{p}_s$ relates the first entry in the alternative period q to the real period p.

Table 4: Example of a 4 period schedule with 1 team, showing the relation of p, $\hat{p}_s$ and q

q_s	1	$q_s =\ e_s = \sum_{p=1}^{f} w_{sp} = 2$		
$\hat{p}_s$		2	3	
p	1	2	3	4
w_{sp}	$w_{11} = 0$	$w_{12} = 1$	$w_{13} = 1$	$w_{14} = 0$
$s = 1$	XXX	$A - 12$		XXX

Table 5: Indices and sub sets for the dynamic scheduling model

$I = i(i = 1,2, \dots, n)$	Client ID: $i = jm \ \ (j = 1)$ $(m = 123) \ \ (i = 1123)$	n	Total no. of clients IDs
$S = s(s = 1,2, \dots, k)$	Team number	k	Total no. of teams
$J = j(j = 1,2, \dots, l)$	Type	l	Total no. of types
$M = m(m = 1,2, \dots, o)$	Client number	o	Total no. of client numbers
$P = p(p = 1,2, \dots, f)$	Period	f	Horizon: number of periods considered for the planning
$\hat{P}_s \subseteq P$	Subset with rule: if $w_{sp} = 1$ include p in $\hat{P}_s$		Periods which are free or where another team is planned (used to link x to $\hat{x}$)
$Q_s = q_s(q_s = 1,2, \dots, e_s)$	With $e_s = \sum_{p=1}^{f} w_{sp}$		Periods q

3.2.2 Input variables

The first 8 input parameters, shown in Table 7 are generated based on client and team data inputs. The cost for an open spot C_{sp} can be different for each team and period. The definition of the function determining this cost is one of the most important aspect for optimizing the schedule. This function influences the consideration between minimizing the time and distance versus the increased risk of lower human resource utilization. An in depth explanation of the cost function can be found in Chapter 3.5.

Both the binary input parameters wc_{sp} and w_{sp} are derived directly from the current schedule. An illustrated example in Table 6 explains the derivation of the two binary input parameters. Vacation days or days where the team is not available are indicated by: XXX. A scheduled client is denoted by its ID with first the type in the form of a string, followed by the client number m. In the first period, team one is not available, resulting in the value of zero for both wc_{sp} and w_{sp}. A client is scheduled in the second period, resulting in no availability for team one, therefore $wc_{12} = 0$. However, team one is scheduled for a client, so $w_{12} = 1$.

Table 6: Example of 4 period schedule with 1 team, w and wc

	$w_{11} = 0$	$w_{12} = 1$	$w_{13} = 1$	$w_{14} = 0$
	$wc_{11} = 0$	$wc_{12} = 0$	$wc_{13} = 1$	$wc_{14} = 0$
	$p = 1$	$p = 2$	$p = 3$	$p = 4$
$s = 1$	XXX	$A - 12$		XXX

Table 7: Input parameters of the dynamic scheduling model

d_{is}	Distance of traveling from client i to team s	D_s	Maximum travel distance for team s
t_{is}	Time of travelling from client i to team s	T_s	Maximum travel time for team s
h_i	Difficulty for client i	H_s	Skill level of team s
l_i	Number of periods for client i	L_s	Maximum periods for a client for team s
C_{sp}	Risk cost function value for team s on period p		
wc_{sp}	Binary parameter indicating if team s is available on period p		
w_{sp}	Binary parameter indicating if team s is available or scheduled for a client on period p		

3.2.3 Decision variables

A total of 4 decision variables are needed to implement all of the necessary constraints. The decisions for the period and team assigned to a new client are indicated by x_{spi}. With a value of 1 if the client is visited on period p by team s. For the alternative periods q_s, the alternative binary decision variable $\hat{x}_{sq_si}$ is constructed. This variable is defined in the same way as x_{spi}.

x_{spi}	Binary decision variable indicating if team s is planned to service client i in period p
$\hat{x}_{sq_si}$	Binary decision variable indicating if team s is planned to service client i in period q_s
y_{si}	Binary decision variable indicating if team s is servicing client i
r_{sq_si}	Binary decision variable indicating if $\hat{x}_{sq_si} = 1$, has already occurred for team s, client i in period q_s

3.2.4 Objective function

The objective function is based on the scope defined in chapter 1.1. Maximizing human resource utilization is implemented by the risk cost function: aC_{sp}. The cost function has its own weight factor: a in order to balance the trade-off related to the other objectives in the objective function. The travel time and distance cost: $bd_{is} + ct_{is}$ are both represented by different factors with their own weight factor. This weight factor for the travel cost can be deducted from the real-life costs. The cost function weight factor can be determined by experimental results for multiple objectives.

$$minimize \sum_{s=1}^{k} \sum_{p=1}^{f} \sum_{i=1}^{n} x_{spi} * (aC_{sp} + bd_{is} + ct_{is})$$

3.2.5 Constraints

A total of 13 constraints are constructed and verified, each constraint is explained separately. The first six constraint do not need further elaboration then simply stating the function of the constraint. Constraint 7 up to 13 are further elaborated by examples and the explanation of the necessity of an extra decision variables.

Every client needs to be visited the correct number of periods.

$$\sum_{s=1}^{k} \sum_{p=1}^{f} x_{spi} = l_i, i \in (1,2,\dots,n) \qquad (1)$$

Every team can only service 1 client in 1 period

$$\sum_{i=1}^{n} x_{spi} = 1, s \in (1,2,\dots,k), p \in (1,2,\dots,f) \qquad (2)$$

The difficulty level of an installation cannot exceed the skill level of a team

$$H_s \geq h_i * x_{spi}, s \in (1,2,\dots,k), p \in (1,2,\dots,f), i \in (1,2,\dots,n) \qquad (3)$$

The driving distance to a client cannot exceed the max team driving distance

$$D_s \geq d_i * x_{spi}, s \in (1,2,\dots,k), p \in (1,2,\dots,f), i \in (1,2,\dots,n) \qquad (4)$$

The driving time to a client cannot exceed the max team driving time

$$T_s \geq t_i * x_{spi}, s \in (1,2,\dots,k), p \in (1,2,\dots,f), i \in (1,2,\dots,n) \qquad (5)$$

The number of periods for a client cannot exceed the maximum number periods a team can service

$$\sum_{p=1}^{f} x_{spi} \leq L_s, s \in (1,2,\dots,k), i \in (1,2,\dots,n) \qquad (6)$$

A client cannot be scheduled on vacation or off day. If $wc_{sp} = 0$, team s is not available on that specific period, as a result, x_{spi} can only be 0 as well. If $wc_{sp} = 1$, x_{spi} can be either 0 or 1, making it possible to schedule a client at that period.

$$\sum_{i=1}^{n} x_{spi} \leq wc_{sp}, s \in (1,2,\dots,k), p \in (1,2,\dots,f) \qquad (7)$$

A client requiring multiple periods can only be scheduled for one team. An extra binary decision variable y_{si} is created, which has the value 1 if client i is assigned to team s. The constraint enforces only 1 team assignment per client. This constraint is not possible with using the other decision variable x_{spi} to our knowledge. One could sum over all the periods for each team and count the number of nonzero sums, but this would also require an extra decision variable to indicate f a sum is nonzero or not.

$$\sum_{s=1}^{k} y_{si} = 1, i \in (1,2,\dots,n) \qquad (8)$$

All periods of an client must be done by the same team. Constraint 9 is an extension of constraint 8, linking x_{spi} and y_{si}.

$$x_{spi} \leq y_{si}, s \in (1,2,\dots,k), p \in (1,2,\dots,f), i \in (1,2,\dots,n) \qquad (9)$$

All periods of a client have the be consecutive, meaning no other installation is allowed in between the periods of a single client. Constraint 10 and 11 ensure $r_{sq_si} = 1$ when $\hat{x}_{sq_si}$ has been 1 in the current or past periods.

$$r_{s(q_s+1)i} \geq r_{sq_si}, s \in (1,2,\dots,k), q_s(q_s = 1,2,\dots,e_s - 1), i \in (1,2,\dots,n) \qquad (10)$$

$$r_{sq_si} \geq \hat{x}_{sq_si}, s \in (1,2,\dots,k), q_s(q_s = 1,2,\dots,e_s), i \in (1,2,\dots,n) \qquad (11)$$

Constraint 12 ensures all of the nonzero values of $\hat{x}_{sq_si}$ are consecutive. With q_s indexing only the periods where either an installation is scheduled, or there is an option in the schedule.

$$r_{s(q_s+1)i} + r_{sq_si} + \hat{x}_{s(q_s+1)i} - \hat{x}_{sq_si} \leq 2,$$
$$s \in (1,2,\dots,k), q_s(q_s = 1,2,\dots,e_s - 1), i \in (1,2,\dots,n) \qquad (12)$$

The best way to illustrate this constraint is though an example of 2 scenarios. Table 8 shows a schedule which does not satisfy constraint 12, client: $A-2$ is scheduled in between two periods of client $A-1$. If we fill in constraint 12 for $q_s = 2$, we get the equation below. In Table 9, the planning does satisfy constraint 12.

$$r_{s(q_s+1)i} + r_{sq_si} + \hat{x}_{s(q_s+1)i} - \hat{x}_{sq_si} = 1 + 1 + 1 - 1 = 3 <\neq 2$$

Table 8: Example of a schedule not satisfying constraint 12

		$r_{111} = 1$	$r_{121} = 1$		$r_{131} = 1$
		$\hat{x}_{111} = 1$	$\hat{x}_{121} = 0$		$\hat{x}_{131} = 1$
		$q = 1$	$q = 2$		$q = 3$
	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
$s = 1$	XXX	$A-1$	$A-2$	XXX	$A-1$

Table 9: Example of a schedule satisfying constraint 12

		$r_{111} = 1$	$r_{121} = 1$		$r_{131} = 1$
		$\hat{x}_{111} = 0$	$\hat{x}_{121} = 1$		$\hat{x}_{131} = 1$
		$q = 1$	$q = 2$		$q = 3$
	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
$s = 1$	XXX	$A-2$	$A-1$	XXX	$A-1$

The last constraint links $\hat{x}_{sqi}$ to x_{spi} using the subset of P: $\hat{P}_s$.

$$x_{spi} = \hat{x}_{sq_si}, s \in (1,2,\dots,k), p \in \hat{p}_s,\ q_s(q_s = 1,2,\dots,e_s), i \in (1,2,\dots,n) \qquad (13)$$

3.3 Re-optimization model

A re-optimization model is proposed to further improve the performance of the scheduling model. This model can re-optimize the client-team assignment, with no other variables to be changed. The re-optimization model uses all of the constraints, indices and variables the dynamic scheduling model uses, with a few additions explained below.

3.3.1 Input parameters

Information about the schedule is inserted into the re-optimization model using two input parameters. Firstly z_{pi}, which indicates if client i is scheduled for period p. Secondly the integer variable u_i, indicating the assignment of team number to the client. If client 1 is scheduled to team 2 on period 4, the two input variables would be: $z_{41} = 1$ and $u_1 = 2$.

z_{pi} Binary parameter indication if client I is serviced in period p

u_i Integer parameter indicating which team s is assigned to client i in the un-optimized planning

3.3.2 Decision variables

Only one decision needs to be made: the client-team assignment. However, two decision variables are needed in order to get an optimized schedule. The first decision variable uz_i is created to represent the absolute value between the assigned team number before and after the re-optimization. If this variable would be used in the objective function, it would still be possible to make to make unnecessary team switches. The observed phenomenon arises from the absence of distinctiveness in the transition of a single client with a difference of 3 team numbers, in contrast to the transitions of three clients altering their team by a single unit in an upward or downward direction.

uz_i Integer decision variable, indicating the difference between the newly assigned team s and the previously assigned team s for client i

uc_i Binary decision variable, indicating if client i is assigned a new team s

3.3.3 Objective function

One term is added to the objective function of the dynamic scheduling model: the minimization of the number of new team assignments.

$$minimize \sum_{s=1}^{k} \sum_{p=1}^{f} \sum_{i=1}^{n} x_{spi} * (aC_{sp} + bd_{is} + ct_{is}) + \sum_{i=1}^{n} uc_i$$

3.3.4 Constraints

The client-period assignment must be constrained. An appointment with the client has been made on a specific period, this period can not be changed anymore. Constraint 1 ensures the period assignment is constrained, but the team assignment can still be changed.

$$\sum_{k=1}^{s} x_{spi} = z_{pi} \;\; s \in (1,2,\dots,k), p \in (1,2,\dots,f), i \in (1,2,\dots,n) \qquad (1)$$

Changing the team assignment is only allowed if it does benefit the objective function. Constraints 2 and 3 ensure the absolute value of the difference between the old scheduled team u_i and the new team s is constructed. By also minimizing the integer decision variable: uz_i, no unnecessary team switches are made.

$$y_{si} * (u_i - s) \leq uz_i, s \in (1,2,\dots,k), i \in (1,2,\dots,n) \qquad (2)$$
$$-y_{si} * (u_i - s) \leq uz_i, s \in (1,2,\dots,k), i \in (1,2,\dots,n) \qquad (3)$$

The last constraint is a binary variable for minimizing the total number of team switches. Since the integer variable uz_i does not differentiate between 4 times a 1 team difference and 1 time a 4 team difference, binary decision variable uc_i is introduced. This variable has the value 1 if uz_i is nonzero and the value 0 if $uz_i$is also 0.

$$uc_i * k \geq uz_i \; (4)$$

3.4 Deterministic model

The optimal solution for the deterministic scenario can be investigated by creating a deterministic model. This model could be used to compare the dynamic and stochastic solution to a deterministic solution. The deterministic model will be based on the re-optimization model with a few adjustments.

First, constraint 1 from the re-optimization model will be removed in order to give the deterministic model the option to change the client-period assignment. However, a client cannot be assigned to any period in the schedule. A client cannot be scheduled before its know arrival time or period. As a result, an new input parameter and constraint are created.

3.4.1 Input parameter

The new binary input parameter at_{pi} is created. This parameter has the value 0 if the arrival period is higher than the current period. If the arrival period is the same or smaller than the current period, the value of at_{pi} is 1.

at_{pi} Binary parameter indication if the arrival period has ben passed at the current period

3.4.2 Constraint

The extra constraint will ensure no client can be scheduled before its arrival period. The decision variable x_{spi} can only equate the value 1 for periods after the arrival period of the client

$$x_{spi} \leq at_{pi} \;\; s \in (1,2,\dots,k), p \in (1,2,\dots,f), i \in (1,2,\dots,n) \qquad (1)$$

3.4.3 Objective function

The objective function is identical to the objective function of the dynamic scheduling model in Chapter 3.2

$$minimize \sum_{s=1}^{k} \sum_{p=1}^{f} \sum_{i=1}^{n} x_{spi} * (aC_{sp} + bd_{is} + ct_{is})$$

3.5 Risk cost function

The risk cost function contains the relation between the risk of lower team utilization and minimizing the travel time and distance. Finding a robust and reliable relation between these two contradicting objectives is important. In order to investigate this relation, different types of cost functions are constructed. Each type of cost function can be adjusted by changing its parameters. Another important aspect of the cost function is that it cannot contain any sections with a derivative of zero. A section with a derivative of zero will result in clients being inserted into the planning at random spots, increasing the risk of open spots and a higher objective function. This is caused by a constant value of the objective function for the section in the cost function where the derivative is zero. To mitigate this problem, a very small term is added to the cost function. This term increases slightly when p increases. As a result, the cost for the function in Figure 7 has the value 1 for $p = 30$ and $1{,}001$ for $p = 40$. This is true for any value $p > h$, up to $p = f$. With f being the horizon the schedule considers for the optimization and h defined as the value of p where the risk cost function equals 1.

The trade-off influenced by the cost function can be best explained by an example. Figure 6 shows a linear and a logarithmic cost function with their relative costs of 0.14 and 0.47 at $p = 5$. This cost value is multiplied by the cost weight factor, in this case 800. This results in a relative weighted cost of 112 and 376 for linear and logarithmic respectively. In this example, a new client must be scheduled with two available options. One option with a travel cost of 300 at $p = 1$, the other option containing a travel cost of 100 at $p = 5$. Combining these costs in Table 10 results in the linear cost function favoring option 2 and the logarithmic cost function favoring option 1.

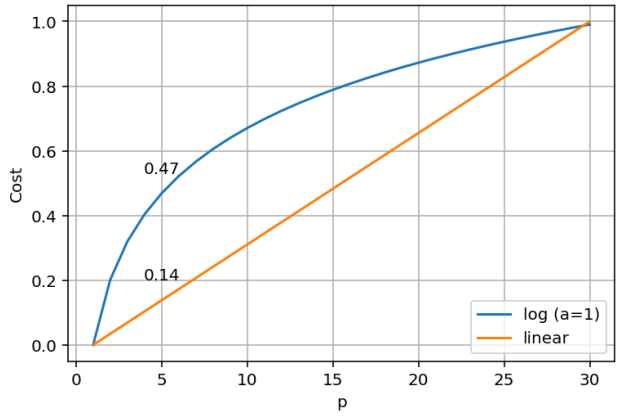

Figure 6: Cost function comparison between linear and logarithmic with cost values at p=5

Table 10: Costs for 2 options and 2 cost functions

	Linear		Log	
	Opt 1	*Opt 2*	*Opt 1*	*Opt 2*
Travel cost	300	100	300	100
Risk cost function	0	112	0	376
Total cost	300	**212**	**300**	476

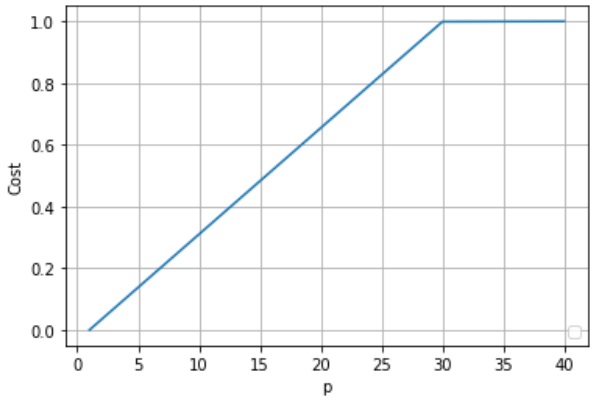

Figure 7: Linear cost function with (h=30) up to p=40

3.5.1 Linear cost function

The linear cost function is constructed using a horizon for when the cost should be one. The cost for inserting a client at $p = 1$ is zero and the cost at $p = h$ is one. The equation below shows the complete formulation. Figure 9 shows a graphical representation of the linear cost function.

$$Cost = \begin{cases} \dfrac{p-1}{h} & h \geq p > 0 \\ 1 + \dfrac{p-h}{M} & p > h \end{cases}$$

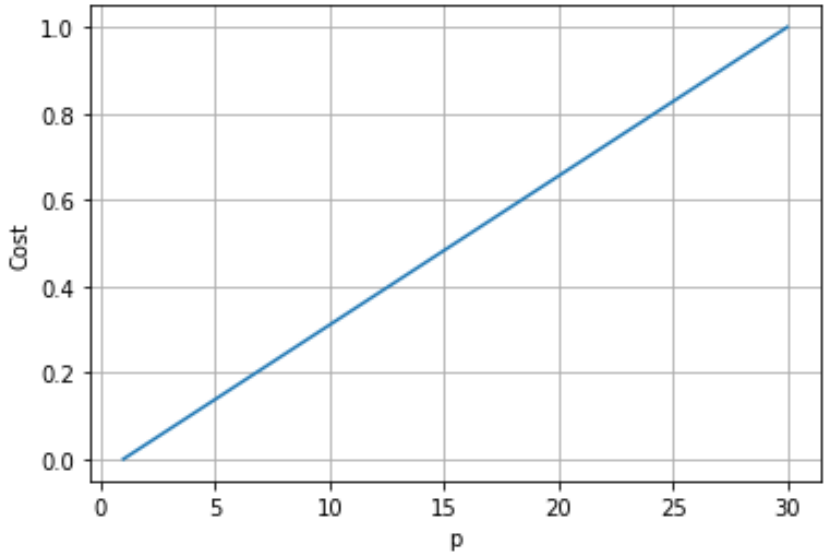

Figure 8: Linear cost function (h=30)

3.5.2 Logarithmic cost function

The logarithmic cost function will increase the cost significantly at low values of p and increase less at higher values of p. Increasing a will result in an decreasing cost at $p = 1$, this decrease will favor inserting a new client in an open spot if it is only a few periods in the future. Figure 10a shows the logarithmic cost function with $a = 1$. Comparison of this function to the function with $a = 5$ in Figure 9b presents a graphical representation of the decreased cost at $p = 1$. The equation below shows the complete formulation.

$$Cost = \begin{cases} \dfrac{\log(a * (h-1)+1) - \log(a)}{\log(a * h) - \log(a)} & h \geq p > 0 \\ 1 + \dfrac{p-h}{M} & p > h \end{cases}$$

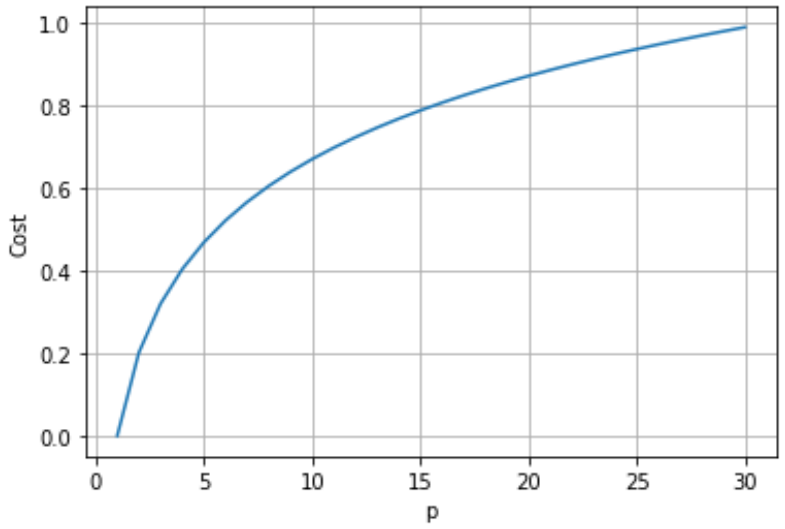

Figure 10a: Logarithmic cost function (a=1) (h=30)

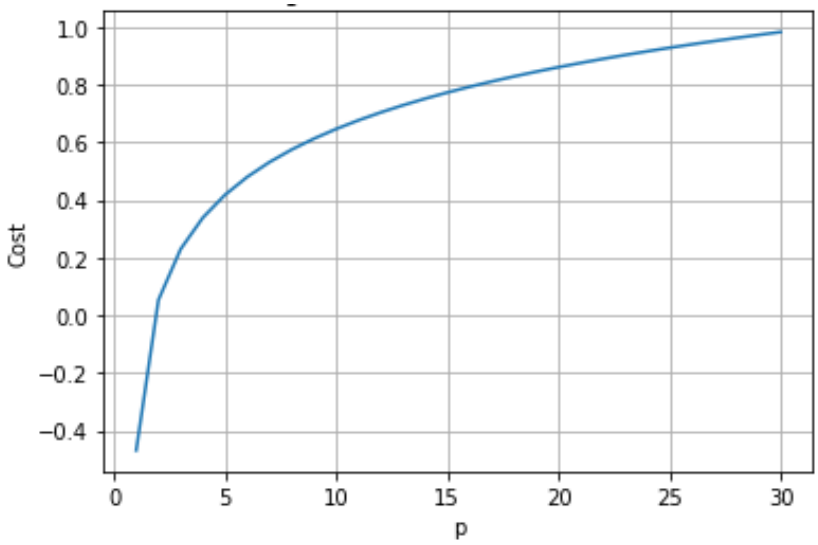

Figure 9b: Logarithmic cost function (a=5) (h=30)

3.5.3 Piecewise linear cost function

The piecewise linear cost function is an alternative to the linear cost function with more customization possibilities. Each line segment can be adjusted to emphasize specific trad-offs or objective priorities. This approach is more suited to real-life applications. A visual representation of this function is shown in Figure 10 The complete formulation is shown in the equation below.

$$Cost = \begin{cases} 0 & 0 < p < a \\ c * \dfrac{p-a}{b-a} & b \geq p \geq a \\ \dfrac{p-b}{h-b} * (1-c) + c & h \geq p > b \\ 1 + \dfrac{p-h}{M} & p > h \end{cases}$$

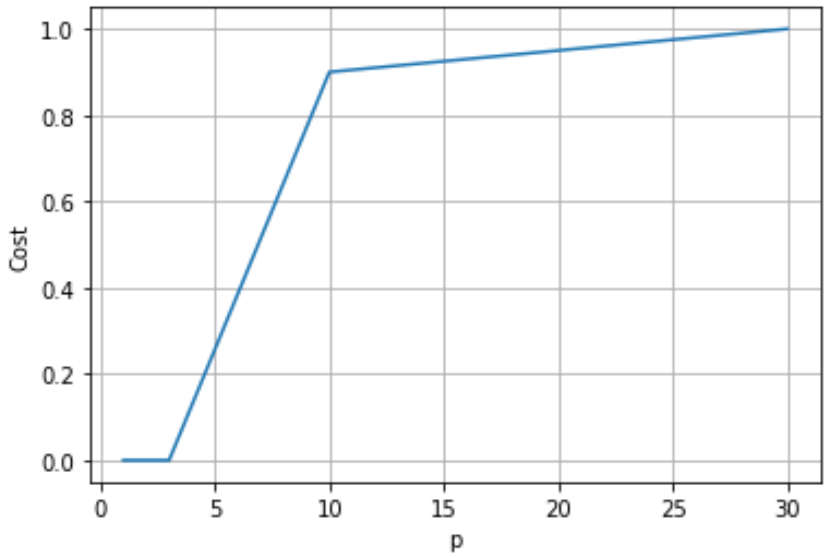

Figure 10: Piecewise linear cost function (a=3) (b=10) (c=0,9) (h=30)

3.5.4 Piecewise logarithmic cost function

The piecewise logarithmic cost function is created to investigate the combination of the linear and logarithmic functions. Parameter a has the same functionality as the parameter a defined in the logarithmic cost function in chapter 3.5.3. Parameter b defines the p value at which the cost equals c. A graphical representation of this cost function is shown in Figure 11. The complete formulation is shown in the equation below.

$$Cost = \begin{cases} \dfrac{\log(a * (h-1)+1) - \log(a)}{\log(a*h) - \log(a)} & h \geq p > 0 \\ \dfrac{p-b}{h-b} * (1-c) + c & h \geq p > b \\ 1 + \dfrac{p-h}{M} & p > h \end{cases}$$

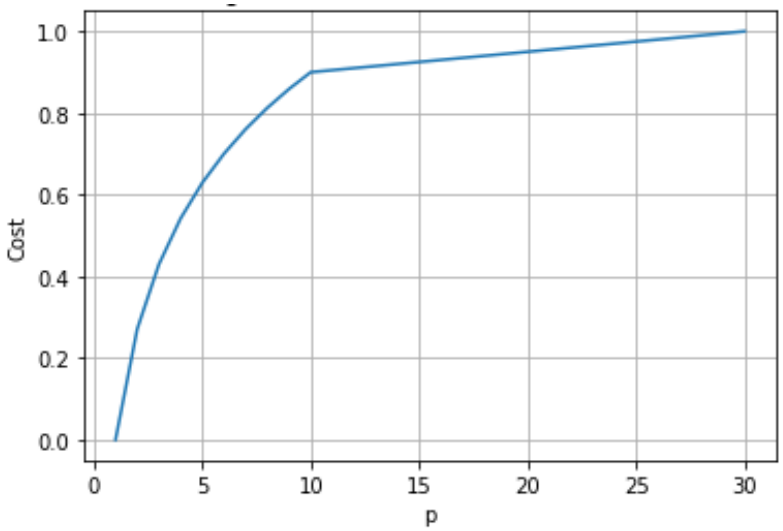

Figure 11: Piecewise logarithmic cost function with (a=1) (b=10) (c=0,9) (h=30)

3.6 Summary

This scientific methodology proposes a solution to the limitations of a rule-based scheduling model by proposing a dynamic scheduling model based on optimization. The rule based model uses a set of Conditions in order to schedule clients. As a result, the use of these Conditions introduce several limitations, including lack of consideration between multiple objectives and poor results for real-life scenarios.

The dynamic scheduling model addresses these limitations by formulating a mathematical model with an objective function, containing multiple objectives. The model uses a mixed-integer linear programming solver to find the optimal value for the objective function. The model considers client and team data inputs to generate input parameters, including distance and time of travel, skill levels, and maximum installation lengths.

The objective function of the model aims to balance multiple objectives, including maximizing human resource utilization and minimizing travel time and distance. Constraints are implemented to ensure that each client is visited for the correct number of periods, teams can only service one client per period, difficulty levels are matched with team skill levels, and driving distances and times do not exceed team limitations. Additional constraints enforce consecutive periods for each client and ensure that all periods of a client are done by the same team.

The re-optimization model can change the client-team assignment in order to further reduce the travel cost. A number of decision variables and constraints are added to the formulation of the dynamic scheduling model to create the re-optimization model.

The deterministic model will receive all of the information at once through the addition of a input parameter, containing the insertion dates of all of the clients. This model is similar to the re-optimization model, but with removal of the client-period assignment constraint.

Different risk cost functions are designed, including linear, logarithmic and piecewise variants. The parameters in the cost function can be adjusted, such as the value of p where the cost equals one and the length of each section in the piecewise variants.

Overall, this methodology provides a systematic approach to address the limitations of a rule-based scheduling model through a dynamic scheduling model based on optimization. The proposed models takes into account various constraints and objectives to effectively schedule clients while considering factors such as the trade-off between different objectives, skill levels, and travel limitations.

4 Case study

A case study is conducted to evaluate the performance of the proposed methods under different scenarios. The exact scenario at the case-study company will be investigated, combined with artificial scenarios. First the scope is determined, followed by the construction of the key performance indicators (KPI). The proposed models are verified using a wide variety of tests. Next, the risk cost function is explained and defined. Different types of this cost function are formulated. Multiple experiments are conducted to investigate different aspects of the model performance. The objective function parameters and the risk cot function types are compared in different experiment to determine the desired trade-off between the multiple objectives. Next, the re-optimization model is evaluated by re-optimizing schedules from the previous experiments. Data about the models performance is analyzed and compared. Finally finding the optimal deterministic solution is discussed.

4.1 Overview

The definition of the scope for the case study is an extension of the problem definition defined in chapter 1.1. Specifics about locations, team parameters and clients are elaborated on. This scope is based on the case study at a start-up installing two types of heating and cooling systems with different technologies. The two types are: the Pump-AO, referred to in the schedule with: “A” and the Heat-cycle, referred to with: “C”. The first system utilizes the more common air-water heat pump principle. As a result, this system generally requires less time and a lower skill level to install compared to the second type of system. The second system uses a water-water heat pump with filtered sewage water as a heat source. The installation teams are all driving electric vehicles and are spread across the depots. Three teams are located at Utrecht and three at Delft, the seventh and last team is located at Geldrop.

- 7 electric vans with installation teams
- 3 depots at Utrecht, Delft and Geldrop
- 2 different types of installations: Pomp-AO (skill level: 1) and Heat-cycle (skill level: 2)
-

The schedule is realized in the online version of google sheets, shown in Figure 12. This implementation realizes easy access to all of the employees, in combination with easy manual adjustments and historical data. The rule-based model is already implemented in this real-life environment. New schedule entries are generated by the rule based model. Employees can enter the required data into the rule based model. Future goals include the client being able to directly schedule their own installation using the dynamic scheduling model. As a result, the following steps are required to be handled by the implemented model.

- 1: A new client sends a request for planning a new installation
- 2: A date proposal is sent to the customer
- 3: If the customer declines, a new date proposal is sent
- 4: If the customer declines multiple times, the client can choose from any available spot
- 5: The accepted date is inserted into the schedule contained in google sheets
-

A	BT	BU	BV	BW	BX	BY	BZ	CA	CB	CC	CD	CE	CF	CG	CH	CI	CJ
Jump to Today	12/3	13/3	14/3	15/3	16/3	17/3	18/3	19/3	20/3	21/3	22/3	23/3	24/3	25/3	26/3	27/3	28/3
Team 1 (DELFT)	0	A-139			A-83	A-83	0	0		A-96	A-96	A-85	A-85	0	0		
Team 2 (DELFT)	0	A-121	A-121	RES	RES	A-67	0	0	A-67	A-92	A-92	A-86	A-86	0	0		
Team 3 (DELFT)	0	A-148	A-148	A-88	A-88		0	0	A-97	A-97	A-105	A-105	A-105	0	0	A-94	A-94
Team 4 (UTRECHT)	0	C-160	C-160	C-160			0	0		C-163	C-163	A-169		0	0	A-100	A-100
Team 5 (UTRECHT)	0	A-79	A-79	A-80	A-80		0	0	A-164	A-164	A-132	A-132		0	0	A-48	A-48
Team 6 (UTRECHT) RB	0	XXXX	XXXX	XXXX	XXXX	XXXX	0	0	XXXX	XXXX	XXXX	XXXX	XXXX	0	0	XXXX	XXXX

Figure 12 Schedule in google sheets: A or C denoting the type, followed by the client number

These steps result in a number of requirements for the real-life implementation of the proposed models. The client cannot received the same date suggestion multiple times. This introduces a problem if multiple teams are available on the same date or period. The multiple succeeding solutions generated by the dynamic scheduling model could share the same period. A function called declined dates is introduced as a solution to this problem. This function contains a list of declined suggested dates for each new client. The implemented dynamic scheduling model can use this list of declined dates for the elimination of options in the schedule sharing this same date. Furthermore, a client can suggest to be scheduled after a specific date. The minimal date function provides a solution for this requirement. All dates before the minimal date are eliminated before the options for the schedule are inserted into the proposed model. In conclusion, two functions for real-life implementation need to be considered: Declined dates and minimal date

4.2 Verification

The verification is divided into two sections: verification of the constraints and of the functions required for real-life implementation. First the verification of the constraints is explained, followed by an explanation for the verification for the real-life implementation. Finally the results of the verification are discussed.

The proposed models have undergone verification through multiple tests, which are documented in detail in appendix 6.2. Each constraint is subjected to a minimum of two test in order to assess the performance under different scenarios. To ensure the validity of each tested scenario, only one input parameter is adjusted in between tests. The verification of constraint 3 in chapter 3.2.5 is explained in detail as an example. This constraint ensures the difficulty level of a client cannot exceed the skill level of the scheduled installation team. A test scenario is considered where no available team possesses the required skill level for the new client. This test is expected to result in no possible solution. This test scenario is slightly adjusted for the next test for this constraint

Functions for the real-life implementation of the proposed models are tested. The declined date s function is tested by decline multiple succeeding suggestions generated by the proposed model. The scenario for this test is constructed such that multiple teams are available on the same date. The expected result is no repetition of already suggested dates or periods.

The realized results for all of the test scenarios mentioned above did correspond to the expected results. No undesirable results were found, as a result, the proposed models are successfully verified

4.3 KPIs

The performance of the models will be measured and compared using four different KPIs, provided in Table 13. These KPIs are derived from the problem definition and the scope of the case study. Both travel time and distance are derived from the objective of minimizing the operational cost. Distance and time are separated for the reason different costs being associated to them. Since most of the travelling will be done during or close to rush hour, the separation of time is even more significant. Driving to the city center of Amsterdam can be much shorter in distance compared to driving to a small village, while the time to reach the center of Amsterdam can be greater. Both the rule-based and dynamic scheduling model take the travel time and distance with traffic predictions into account.

The third KPI is the number of open spots in the schedule up to the date of the last client insertion into the schedule. An open spot means a team is not working on that day and thus costing money without creating revenue, in other words: Unutilized resources (UR). Lastly the makespan KPI is created. This value represents the risk of unutilized resources (RUR). Once the schedule is finished the value of this KPI becomes the highest period in which a client is planned. A higher makespan means a higher chance of an open spot in the schedule. This is best illustrated with an simple example.Table 11 and Table 12 show a schedule with a high and a low makespan respectively, with the current day at period 4. Imagine a new client request arrives which can only be scheduled with the Utrecht team. The schedule in Table 11 will now have unutilized human resources (open spot) when the current day passes, while the schedule in Table 12 will not have an open spot. The open spot and makespan KPIs are defined as the unutilized resources and the risk of not utilizing those human resources.

Table 11: Schedule with high makespan (p=8)

p	1	2	3	4 (today)	5	6	7	8
Team 1 - Delft	A-9	A-9	XXX	XXX				
Team 2 - Utrecht	A-15	A-15	XXX	XXX	A-32	A-32	A-45	A-63

Table 12: Schedule with low makespan (p=6)

p	1	2	3	4 (today)	5	6	7	8
Team 1 - Delft	A-9	A-9	XXX	XXX	A-45	A-63		
Team 2 - Utrecht	A-15	A-15	XXX	XXX	A-32	A-32		

Table 13: The four KPIs used for evaluation of the models

KPI	**Definition**	**Measuring range**
Travel distance (km)	Travel cost	All clients
Travel time (hrs)	Travel cost	All clients
Open spots (periods)	Unutilized resources (UR)	Up to date of last new client insertion
Makespan (periods)	Risk of a lower utilization of resources (RUR)	Last period in the schedule where a client is scheduled

4.4 Dynamic scheduling experiments

The proposed models are evaluated using a number of different experiments. Real-life scenarios are simulated for the first two experiments. These experiments are conducted to gather useful insights about the dynamic scheduling models behavior. Next, the impact of the risk cost function in different scenarios is investigated. This influence is further investigated by a pareto analysis in the next chapter. The scenarios are expanded in order to investigate the performance on larger and more complicated scenarios in the next chapter. Finally a very large scenario with 250 clients and 18 teams is simulated in order to get insights into the performance of the proposed models with large problems.

4.4.1 Number of teams and depots variation

In the first experiments the number of teams and depots are variated for both the rule-based and optimization model. Starting with the simplest scenario with only one depot and two teams. The last scenario has seven teams distributed over three depots as defined by the scope in chapter 1.1. The input parameters subjected to change can be found in Table 14. No constraints on team parameters are included in this experiment. The clients do not decline any of the suggested dates, meaning the first solution of the model is accepted and inserted into the schedule. In order to balance the relative size of the client set and the number of teams, the number of clients per period is increased as the of teams increases. All of the input parameters that are changed in-between experiments are in bold text. The risk cost function used for these first experiments is the linear cost function with the parameters shown in Table 15, combined with the weight factors for the objective function. Real life client data is used for the duration and location of the clients. The clients are inserted into the models starting from period one. Once the first client is scheduled, the next client is inserted until all of the clients for the experiment are scheduled. After the scheduling the KPIs are calculated and presented in tables and graphs, including important findings.

Table 14: Varying input parameters for the team and depot variation experiments

	x-1	**x-2**	**x-3**	**x-4**	**x-5**
Number of depots	1	**2**	2	2	**3**
Number of teams	2	2	**4**	**6**	**7**
Team – depot assignment	1,1	**1,2**	**1,1,1,2**	**1,1,1,2,2,2**	**1,1,1,2,2,2,3**
Arrival frequency of new clients	1 per day	1 per day	**2 per day**	2 per day	2 per day

Table 15: Fixed input parameters for the team and depot variation experiments

Risk cost function type	Linear
Risk cost function h	30
Risk cost function weight factor (objective a)	800
Travel distance cost (objective b)	0,8
Travel time cost (objective c)	100

4.4.1.1 Without vacation or off days

The first experiment is performed on an empty schedule without any vacation or off days. A simple representation is given in Table 16, with an empty cell representing an option in the schedule. A 0 is representing the weekends.

Table 16: Part of an open schedule without vacation or off days

p	1	2	3	4	5	6	7	8
Team 1						0	0	
Team 2						0	0	

Table 17 and Figure 13 show the results of the first experiment. The results for experiment 1-1 are identical for both the rule-based and the optimization model, as expected. There is only one depot with two teams with no difference between them in terms of objective function influencing parameters. This results in every decision being an optimal decision as long as the client is scheduled with the lowest period possible. Both model succeeded in performing this.

Table 17: Results for experiment 1: team and depot variation without vacation or off days

	Travel distance (km)		**Diff (%)**	**Travel time (hrs)**		**Diff (%)**	**Open spots (periods)**		**Diff (%)**	**Make span (periods)**		**Diff (%)**
	R-B	*DSM*		*R-B*	*DSM*		*R-B*	*DSM*		*R-B*	*DSM*	
1-1	4708	4708	0,0%	55,8	55,8	0,0%	0	0	0	75	75	0,0%
1-2	3601	3400	**-5,6%**	43,4	41,0	**-5,5%**	0	0	0	75	75	0,0%
1-3	3807	3827	0,5%	45,7	45,9	0,4%	0	0	0	55	49	**-12,8%**
1-4	3526	3281	**-6,9%**	42,6	39,6	**-7,0%**	0	1	1	31	33	8,7%
1-5	3138	2828	**-9,9%**	38,9	35,3	**-9,3%**	0	4	4	27	31	21,1%

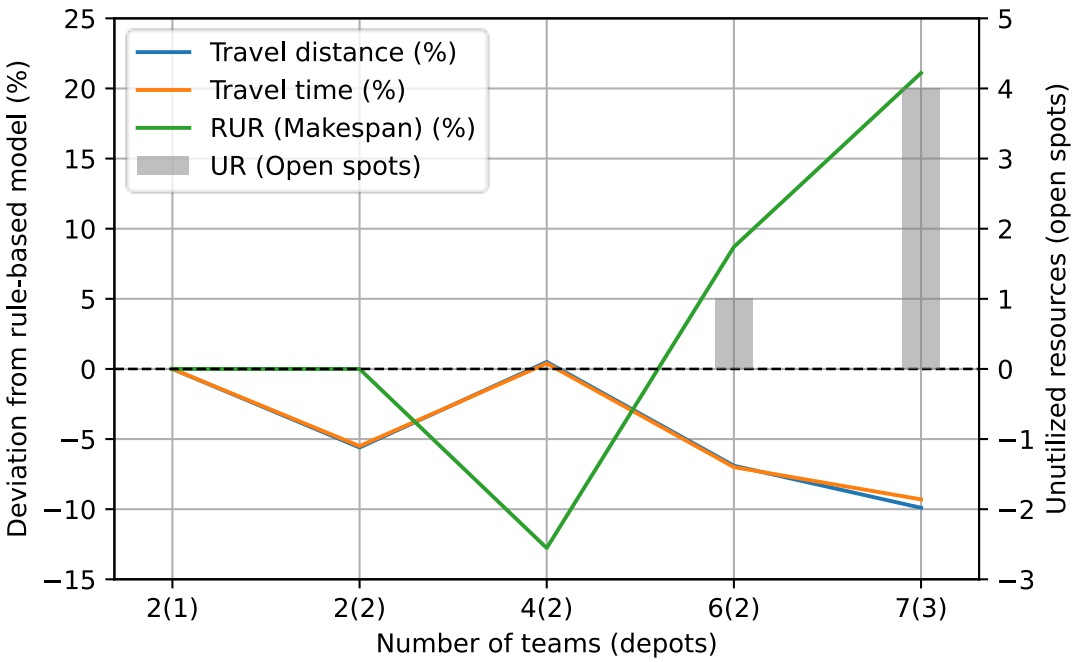

Figure 13: Graph of results for varying depots and teams for an schedule without vacation days

For experiment 1-2 an extra depot is added with one team at each depot, here a difference between the models is expected. The optimization model shows around a five percent lower travel time and distance compared to the rule-based model. This can be explained by the optimization model being able to schedule a new client at the depot with lower travel cost at the cost of a higher risk of lower team utilization. The rule-based model cannot make this tradeoff since it can only select for one condition or objective at a time in chronological order.

Experiment 1-3 shows interesting results with higher travel cost and a lower makespan. With this experiment the tradeoff between travel cost and the risk of open spot gets very clear. The cost function for the optimization model for both 1-2 and 1-3 is the same, while the relative number of teams per depot differs from 50-50 to 75-25. This greatly influences the risk of an open spot per team. This tradeoff with be investigated further in chapter 4.4.2.

The last two parts of the experiment: 1-4 and 1-5 show improvements in travel cost at the expense of the makespan. The optimization model also resulted in 1 and 4 open spots respectively. This can be explained by the number of clients almost matching the number of available spots in the schedule. When the insertion period of new clients is very close to the available spots, the risk of an open spot increases. This result can also be contributed to the linear cost function, which makes no distinction between periods far in the future compared to periods near in the future in terms of relative cost difference.

Overall the results for experiment 1 show the capability of the optimization model to lower the travel cost or to lower the makespan. The importance of a well defined and reliable cost function is shown clearly in the results.

4.4.1.2 With vacation or off days

In the second experiment, the schedule is changed to represent the real-life use case. This includes vacation days. This decreases the total number of available spots for a given number of periods. The rest of the input parameters is the same. A simple representation of an schedule with vacation days is given in Table 18, with a 0 representing the weekends and three Xs representing the added vacation or free days compared to Table 16 in the previous Chapter 4.4.1.1.

Table 18: Part of an open schedule without vacation or off days

p	1	2	3	4	5	6	7	8
Team 1			XXX			0	0	XXX
Team 2				XXX		0	0	

The second experiment shows similar results to the first experiment, shown in Table 19 and Figure 14. A slightly smaller improvement is made in terms of travel cost, but the trend is the same. Experiment 2-2 shows an improvement in terms of one open spot. This can be explained by the rule-based model utilizing one team significantly compared to the other team at a certain point in time. An example of this can be found in chapter 4.3.

Experiment 2-4 and 2-5 show a great improvement in terms of open spots for the optimization model compared to the results form experiment 1. The average amount open options per period is lower in the schedule for experiment 2 compared to the schedule of experiment 1. This decreases the risk for open spots in the schedule, since new clients are scheduled further into the schedule. This effect can also be identified in the larger makespan throughout experiment 2 compared to 1. The optimization model has a number of constraints and decision variables to ensure all possible combinations are an option.

Table 19 Results for experiment 2: team and depot variation with vacation or off days

	Travel distance (km)		**Diff (%)**	**Travel time (hrs)**		**Diff (%)**	**Open spots (periods)**		**Diff (%)**	**Make span (periods)**		**Diff (%)**
	R-B	*DSM*		*R-B*	*DSM*		*R-B*	*DSM*		*R-B*	*DSM*	
2-1	4708	4708	0,0%	55,8	55,8	0,0%	0	0		87	87	0,0%
2-2	3583	3432	**-4,2%**	43,1	41,3	**-4,2%**	0	0		87	87	0,0%
2-3	3560	3827	7,5%	42,6	45,9	7,7%	1	0		75	56	**-25,3%**
2-4	3647	3322	**-8,9%**	43,8	40,1	**-8,4%**	0	0		38	39	2,6%
2-5	3520	2883	**-18,1%**	42,9	35,8	**-16,6%**	0	0		34	38	10,5%

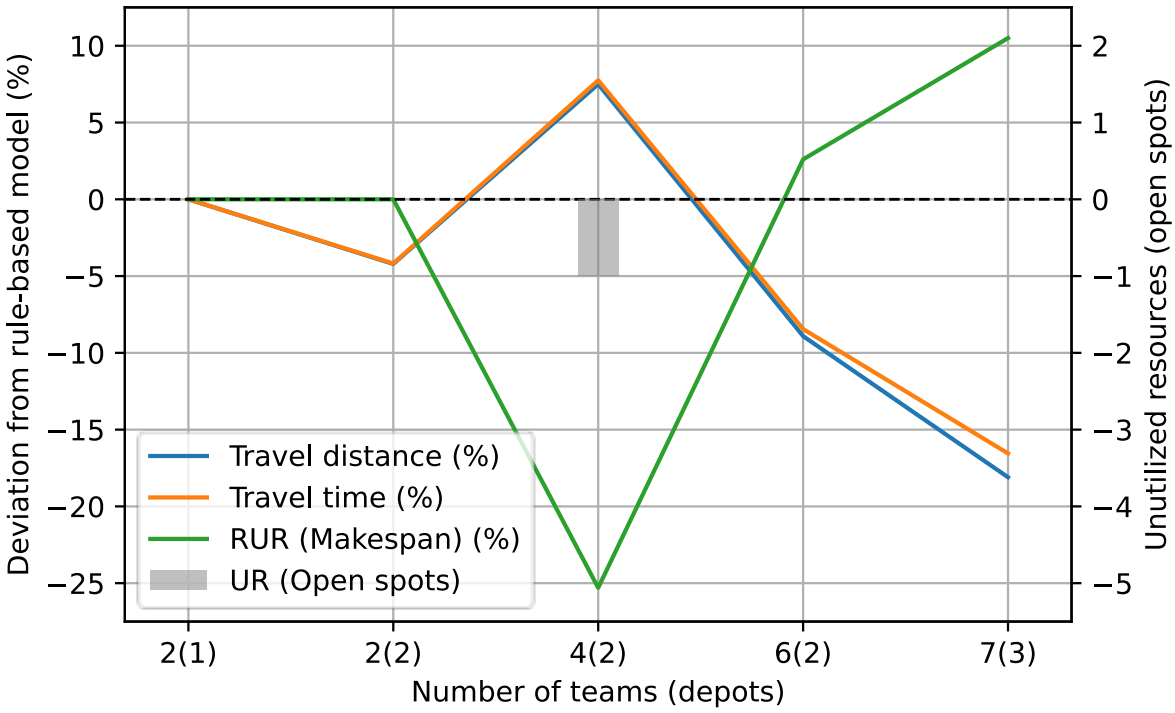

Figure 14: Graph of results for varying depots and teams for a schedule with vacation days

4.4.2 The impact of different risk cost functions

The tradeoff between the risk unutilized resources (UR) and minimizing the travel cost is investigated using different cost functions. A total of two experiments is conducted on the most complicated scenario in the first experiment and the second experiment. The different types of cost functions are explained in detail in chapter 1.1. All 4 different functions are included in the experiment, the logarithmic function is included twice, but with a different parameter. This function is expected to generate the best results. The benchmark case of experiment 1-5 is chosen for the reason of the optimization model resulting in four open spots, while the rule-based model did not have any. Both the rule-based and optimization results of the benchmark case are compared with the results of all the cost functions. The varying input parameters can be found in Table 20. The objective function weight factors are identical to the previous experiments. All of the other input parameters are identical and thus not repeated in. All of the cost functions are plotted in Figure 15 with the linear cost used in the precious experiments as reference. The steepness of the cost function closer to $p = 1$ is an indication of a higher risk of unutilized resources (RUR).

Table 20: Varying input parameters for the risk cost functions experiments

Experiment	**x-1**	**x-2**	**x-3**	**x-4**	**x-5**
Benchmark	1-5	1-5	1-5	1-5	1-5
Risk cost function type	Log	Log	PW lin	PW log	PW log
Risk cost function a	1	5	3	1	1
Risk cost function b	-	-	10	10	20
Risk cost function c	-	-	0,95	0,95	0,95
Risk cost function h	30	30	30	30	30

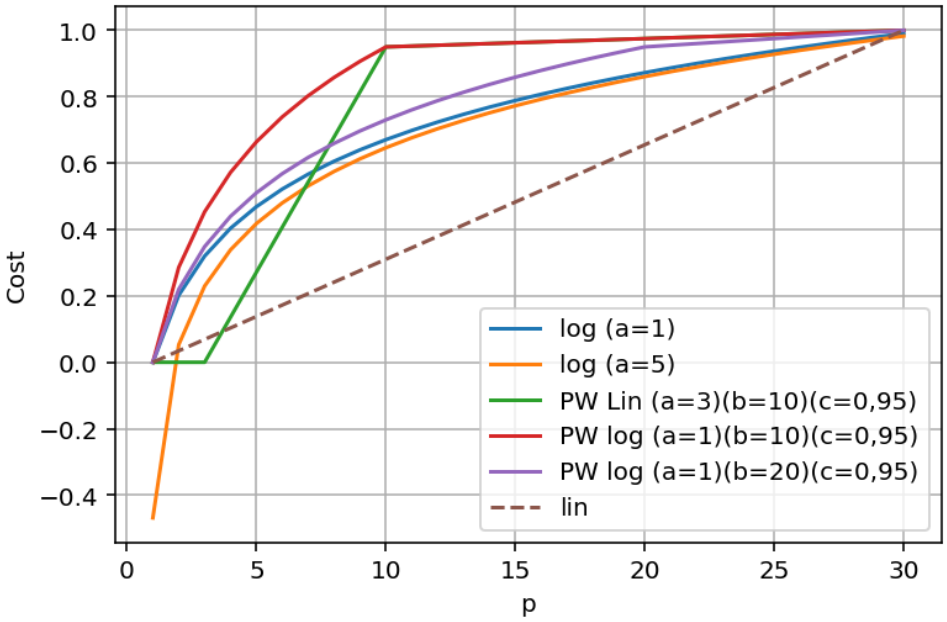

Figure 15: Cost function plots for experiment 3 and 4 (linear cost for reference)

Experiments 3-1, 3-2 and 3-5 show identical results, finding a “middle ground” between the rule-based model and the optimization with the linear cost function. As shown in Table 21 and Figure 16, the travel cost increase, but open spots and makespan decrease compared to the linear cost optimization. This can be explained by the cost function causing the optimization to strongly favor options with a lower period as the period approaches one. The cost function in experiment 3-2 even has a negative cost for period 1, minimizing the risks of open spots even further. Table 21 shows identical results in terms of travel distance for both logarithmic cost functions and the piecewise logarithmic function with the b parameter with a value of 20. This could be explained by all three cost functions providing a trade-off characteristic within a certain range for all the new clients presented.

Table 21 Results for experiment 3: cost functions comparison

	Travel distance (km)			**Travel time (hrs)**			**Open spots (periods)**			**Make span (periods)**		
	R-B	*Opti linear*	*DSM*	*R-B*	*Opti linear*	*DSM*	*R-B*	*Opti linear*	*DSM*	*R-B*	*Opti linear*	*DSM*
bm	3138	2828		38,9	35,3		0	4		27	31	
3-1	**-4,5%**	6,0%	2998	**-4,4%**	5,4%	37,2	0	-4	0	0,0%	**-12,9%**	27
3-2	**-4,5%**	6,0%	2998	**-4,4%**	5,4%	37,2	0	-4	0	0,0%	**-12,9%**	27
3-3	7,4%	19,2%	3370	5,9%	16,7%	41,2	0	-4	0	0,0%	**-12,9%**	27
3-4	4,3%	15,7%	3273	3,3%	13,9%	40,2	0	-4	0	0,0%	**-12,9%**	27
3-5	**-4,5%**	6,0%	2998	**-4,4%**	5,4%	37,2	0	-4	0	0,0%	**-12,9%**	27

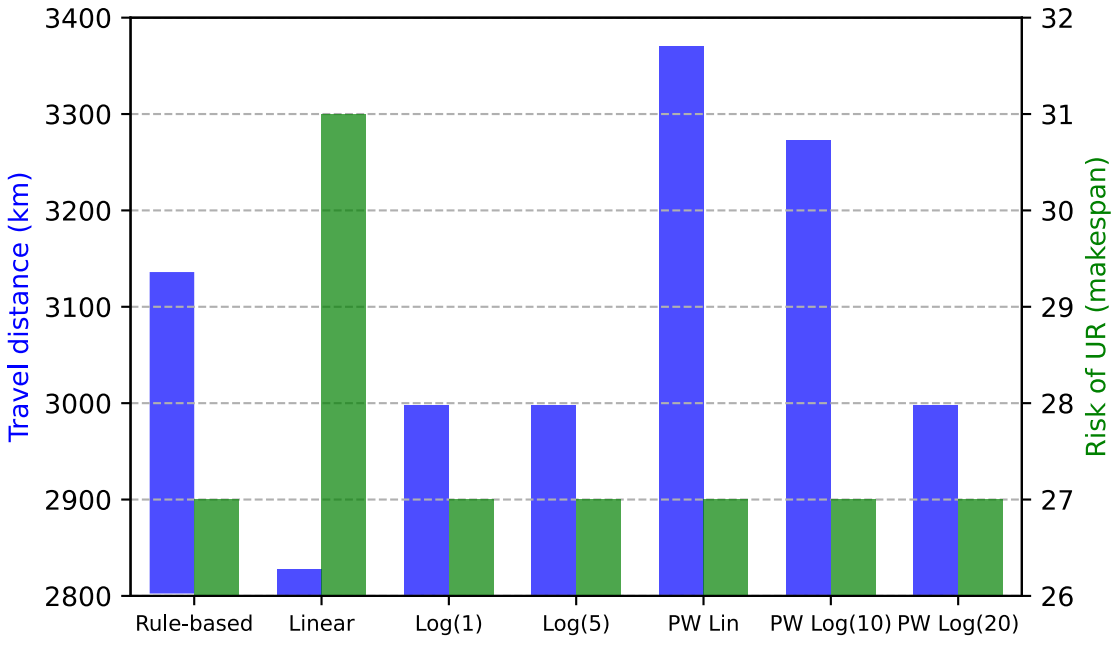

Figure 16: Results for experiment 3: cost functions comparison

In experiment 4, all cost functions improve over the rule-based model in term of RUR and travel cost, see Table 22 and Figure 18. No improvements on travel cost are made over the optimization model with the new cost functions. This result is identical to the previous experiments and as expected. The makespan KPI is decreased overall with four periods.

The piecewise linear cost function performs the worst, followed by the piecewise logarithmic function. This can be explained by strongly favoring minimizing the risks for open spots, especially for the piecewise linear function. With this function the sacrifice made for the trade-off is the same for 2 options at $p = 10$ and $p = 8$ versus $p = 3$ and $p = 1$. Logically, the risk of creating an open spot at $p = 1$ is much greater than at $p = 8$, but this cost function does not differentiate and increases the travel cost to avoid a potentially very low risk, this is unnecessary. For the piecewise logarithmic function with $b = 10$, there is some deviation between lower and higher periods, but the overall risk avoidance is the highest of all functions.

Both logarithmic functions perform the same regarding the KPIs. This can be explained by both functions having almost identical shapes with one exception: the log(5) function has a negative cost for $p = 1$. This negative cost makes the potential sacrifice in travel cost the biggest of all the functions and thus very unlikely to result in open spots if the total number of clients is sufficient. The reason both functions do not differ in result can be explained by the absence of the need to make a bigger sacrifice in travel cost than the log(1) function already supplied.

In order to find the best fitting cost the results from experiment 3 and 4 are combined. From experiment 3, cost function 1,2 and 5 are selected based on their result on decreasing the open spots to zero and their identical travel cost results. From experiment 4, cost function 1, the logarithmic cost with $a = 1$ is selected based on the best performance on travel cost, while having identical performance in the other KPIs compared to the other cost functions.

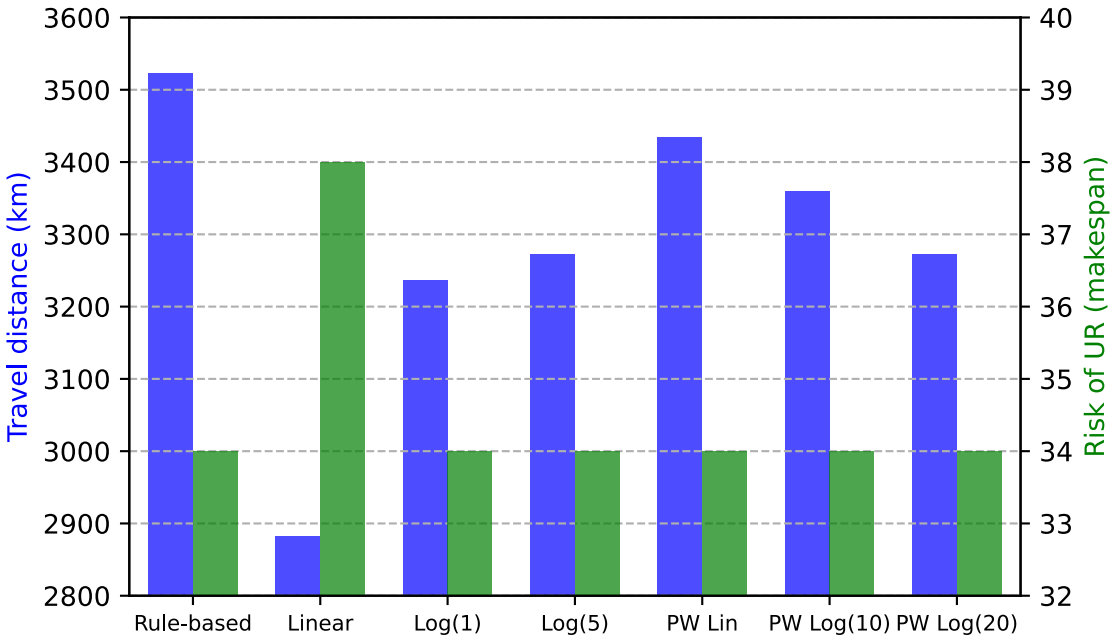

Figure 17: Results for experiment 4: cost functions comparison

Table 22 Results for experiment 4: cost functions comparison

	Travel distance (km)			**Travel time (hrs)**			**Open spots (periods)**			**Make span (periods)**		
	R-B	*DSM*	*DSM-c*	*R-B*	*DSM*	*DSM-c*	*R-B*	*DSM*	*DSM-c*	*R-B*	*DSM*	*DSM-c*
bm	3520	2883		42,9	35,8		4	0		34	38	
4-1	**-8,1%**	12,2%	3236	**-7,5%**	10,9%	39,7	0	0	0	0,0%	**-2,9%**	34
4-2	**-7,0%**	13,5%	3273	**-7,0%**	11,5%	39,9	0	0	0	0,0%	**-2,9%**	34
4-3	**-2,4%**	19,1%	3435	**-1,9%**	17,6%	42,1	0	0	0	0,0%	**-2,9%**	34
4-4	**-4,6%**	16,5%	3359	**-5,4%**	13,4%	40,6	0	0	0	0,0%	**-2,9%**	34
4-5	**-7,0%**	13,5%	3273	**-7,0%**	11,5%	39,9	0	0	0	0,0%	**-2,9%**	34

4.4.3 Pareto analysis

The interaction between the two contradicting objectives is investigated using a pareto analysis. A pareto front will be constructed from multiple simulations. The weight factor of the objective to maximize the human resource utilization is varied, this influences the tradeoff between the two objectives. Table 23 shows the different values for the cost function weight factor. Experiment 1-3, from chapter 4.4.1 is chosen as a benchmark case for the reason of its deviating behaviors compared to the other simulations in experiment 1. The rest of the input parameters and setting are identical to experiment 1.

Table 23 Varying input parameters for pareto analysis experiments

Experiment	**5-1**	**5-2**	**5-3**	**5-4**
Benchmark	1-3	1-3	1-3	1-3
Cost function weight factor	**100**	**250**	**400**	**800**

The results shown in Table 24 show a clear trade-off between the RUR and the travel cost. A higher open spot cost function as input parameter results in a higher travel cost and a lower makespan KPI. These data point are inserted into a graph in Figure 18 and fitted with a second order polynomial curve. A good fit, shown by the blue line, is found for the four data points. Any point on this pareto front represents a solution where it is impossible to decrease one objective without increasing the other. The result for the rule-based model is represented by the green dot. From this solution, improvements can be made in either 1 of the objectives, or both. If we move down from the rule-based solution, the height of the makespan is reduced, without increasing the travel time. If we move left from the rule-based solution, the travel time is reduced without increasing the makespan. The last possibility would be to move anywhere in between down and left to improve on both solutions. This pareto front proves the capability of the optimization model to give better solutions compared to the rule-based model. However, it is very important to understand the trade-off between the different objective to make an informed decision on the cost function and accompanying weight factor.

With this knowledge, a cost weight factor could be chosen for similar situations as the one of this experiment. The choice can be made between a factor of 800, favoring a low makespan and a factor 400, favoring a low travel time and cost. When implementing this model into a real-life situation it is important to generate a new pareto front for the current configuration of teams, depots and constraints. Based on this new front, a selection between new weight factors can be made

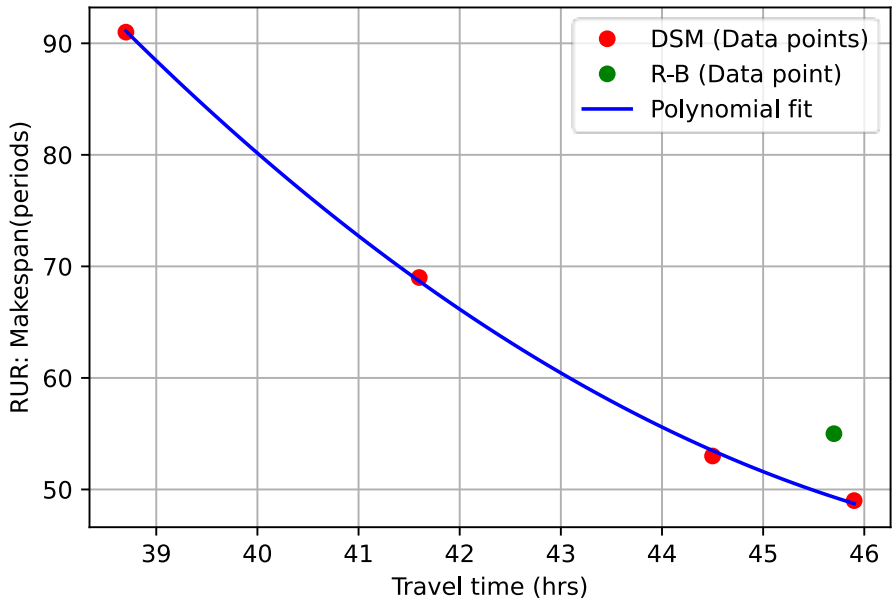

Figure 18: Results for the optimization model with different weights for 1 objective (risk cost function)

Table 24: Results of pareto analysis

	Travel distance (km)	**Travel time (hrs)**	**Open spots (periods)**	**Make span (periods)**
Optimization model: cost function weight				
100	3200	38,7	6	91
250	3425	41,6	0	69
400	3724	44,5	0	53
800	3827	45,9	0	49
Rule-based	3807	45,7	0	55

4.4.4 Combining team constraints and client differentiation

The combination of more clients, different clients and input parameters for the teams are investigated in this experiment. Clients input parameters include a varying number of periods and difficulty level. Table 25 shows all of the input parameters for each team individually. Team input parameters are separated by commas, with ",-," implying no limit or constraint for the particular team on that place in the series. For example: -,-,75,- implies team 1,2, and 4 don't have a limit on that particular constraint, but team 3 has a 75 kilometer driving distance limit. The best performing cost function from previous experiments 3 and 4 is selected. For the clients, both one and two period clients are selected at random from the real-life client database.

Table 25: Varying input parameters for experiment 6: Combining team constraints and client differentiation

Experiment	**6-1**	**6-2**	**6-3**
Number of depots	2	2	**3**
Number of teams	4	**6**	**7**
Team – depot assignment	1,1,2,2	**1,1,1,2,2,2**	**1,1,1,2,2,2,3**
Team max driving time (min)	60,-,-,-	60,-,-,-,-,-	60,-,-,-,-,-,45
Team max driving distance (km)	-,-,75,-	-,-,75,-,-,-	-,-,75,-,-,-,60
Team max periods	2,2,2,1	2,2,2,1,2,2	2,2,2,1,2,2,1

Table 26 shows the results of the three simulations, with the first one, 6-1 showing the most improvement over the rule-based model. This can be explained if we look at the cost function in Chapter 3.5. The client insertion rate per period is higher than the available periods per day. This causes new client to be inserted further from the current period. On the cost function graph in Chapter 3.5.2, higher periods are on the right side, with a lower derivative of the cost function. The maximum sacrifice of travel cost made is lower and therefore a bigger advantage over the rule-based model can be made. This comes at the small cost of the makespan being one period higher.

Both 6-2 and 6-3 operate more on the right side of the cost function and therefore do not increase on the makespan KPI. Improvements are made on travel cost consistently as well as a significant improvement in the number of open spots. The biggest challenge for the number of open spots was the 7th team with both time, distance and the number of periods per client limitations. This is the where the majority of the open spots occurred for both models. In the client set, there were exactly enough clients that satisfied the constraints given for team 7. The optimization model only left two of the possible 23 spots open for team 7 compared to the 7 open spots for the rule-based model. The low performance of the rule-based model can be explained by the elimination of team 7 in one of the first conditions based on the driving distance.

Table 26: Results for experiment 6: Combining team constraints and client differentiation

	Travel distance (km)		**Diff (%)**	**Travel time (hrs)**		**Diff (%)**	**Open spots (periods)**		**Diff (%)**	**Make span (periods)**		**Diff (%)**
	R-B	*DSM*		*R-B*	*DSM*		*R-B*	*DSM*		*R-B*	*DSM*	
6-1	10504	9026	**-14,1%**	128,8	111,7	**-13,3%**	0	0	0	89	90	1,1%
6-2	8768	8426	**-3,9%**	109,9	105,9	**-3,6%**	2	0	**-2**	55	55	0,0%
6-3	8526	7903	**-7,3%**	107,1	100,2	**-6,4%**	7	2	**-5**	53	53	0,0%

4.4.5 Large scale problem

The performance of the dynamic scheduling model on a large scenario is investigated. A total number of 250 clients from the case study are used for this simulation, as shown in Table 27. Multiple constraints for team driving time, distance and the maximum number op periods are added to increase the difficulty of the simulation. Both types of clients are used, with different required skill levels. Another addition over previous experiments is the ability of a client declining a solution or suggested date, resulting in the request for a new date. This increases the difficulty of optimizing for a lower risk of unutilized resources (RUR). The same risk cost function is used as in the previous experiment, the reason for this is the excellent performance in previous experiments. The schedule used for this simulation is an extension of the type one used in previous experiments. The schedule is copied multiple times to accommodate 18 teams, distributed over 8 different depots. The depot locations are spread randomly throughout the Netherlands, but with a higher density of depots where the population density is higher.

Table 27: Input parameters for the large simulation

Number of depots	7
Number of teams	18
Number of clients	250
Type of clients	1 (“A”), 2(“C”)
Periods per client	1 or 2
Declined suggestions per client	0 or 1
Insertion rate of new clients	14 per day
Objective function weight a	800
Risk cost function type	Log
Risk cost function a	1
Risk cost function h	30

The results of the large simulation are shown in Table 28. The optimization model is performing significantly better on all of the KPIs. An improvement of over 25 percent on transportation cost combined with no open spots in the schedule exceeds all previous results.

This improvement could be explained by how the rule-based model deals with multiple options for assigning a depot location. The earliest possible option is selected from the set of available depots within the driving time threshold of 2000 second. If all 8 depots are within the threshold, the chance of picking the closest e.g. with the lowest transportation cost is only 12,5%. This effect is much less pronounced when the total number of depots is only two or three. The dynamic scheduling model can be a trade-off decision between the closest depot and the earliest time, therefore having a significantly improved decision.

Table 28: Results of the large simulation

	Travel distance (km)		**Diff (%)**	**Travel time (hrs)**		**Diff (%)**	**Open spots (periods)**		**Diff (%)**	**Make span (periods)**		**Diff (%)**
	R-B	*DSM*		*R-B*	*DSM*		*R-B*	*DSM*		*R-B*	*DSM*	
	23156	17265	**-25,4%**	265,2	197,5	**-25,5%**	21	0	**-21**	62	57	**-8,1%**

4.5 Re-optimization experiments

A selection of resulting schedules from the previous experiments is made in order to investigate the performance of the re-optimization model. The most demanding experiment form the depots and team variation experiments: 1-5 is chosen as a first schedule. Next, two iterations on 1-5: 3-3 and 3-5 are chosen. Those experiments used a different cost function in order to investigate the performance and gather insights about the risk-cost trade-off. All of the experiments from 4 are chosen to investigate the effectiveness on varying team and depot input parameters The entire schedule generated by the dynamic scheduling model is re-optimized. After the re-optimization the new KPIs are calculated and compared with the old KPIs. The number of open spots and the makespan will not change. This is a result of the client-period assignment being constraint for the re-optimization.

Table 29 shows the results of the re-optimization experiments with an improvement over all of the experiments. An improvement of 0,2% up to 6,7% is made, with an average of 2,6%. This experiment shows the capabilities of the re-optimization model for different scenarios.

Table 29: Results of the re-optimiztion model on different schedules from previous experiments.

	Travel distance (km)		**Diff (%)**	**Travel time (hrs)**		**Diff (%)**
	Old	*Re-opt*		*Old*	*Re-opt*	
1-5	3138	3080	**-1,8%**	38,9	38,1	**-2,1%**
3-3	3370	3144	**-6,7%**	41,2	39	**-5,3%**
3-5	2998	2964	**-1,1%**	37,2	36,9	**-0,8%**
4-1	3236	3206	**-0,9%**	39,7	39,3	**-1,0%**
4-2	3273	3242	**-0,9%**	39,9	39,4	**-1,3%**
4-3	3435	3331	**-3,0%**	42,1	40,9	**-2,9%**
4-4	3359	3319	**-1,2%**	40,6	40,5	**-0,2%**
4-5	3273	3242	**-0,9%**	39,9	39,4	**-1,3%**

4.6 Computational performance

The computational performance is investigated by comparing the data on computational time for a different number of teams and depots. Settings for the team parameters and depot location are identical to the large simulation in chapter 4.4.5. The influence of the number of teams and the number of depots is investigated. Two important factors influence the overall computation time. These two factors are the data gathering and the solving itself. The data gathering includes reading the databases and requesting data from the APIs followed by converting this gathered data into a usable format for the solver to use. Examples of this are the creation of the input variables wc_{sp} and wc_{sp}. Performance is investigated for both the dynamic scheduling model as well as for the re-optimization model. Lastly, a comparison with the rule-based model in terms of solving time is made.

No significant differences in computational time were found regarding the number of depots. The number of teams does have a very significant impact on the computational time. Figure 19, shows the results for 3 teams up to 18 teams. The computational time is multiplied by a factor slightly lower than four if the number of teams doubles. The polynomial fit, generated in python is represented by the following equation:

$$t = 0.0324029x^2 - 0.221052x + 0.82781$$

With x being the number of teams and t the required computational time. When using the dynamic scheduling model for a large scale problem it is important to consider the rapidly increasing computational time.

The data gathering time, shown in green in Figure 19 increases almost linear with the number of teams increasing. This can be explained by the size of the matrices and sets also increasing linear with the number of teams. If the number of teams is doubles, the program in python also has to go though double the loops to created the matrices.

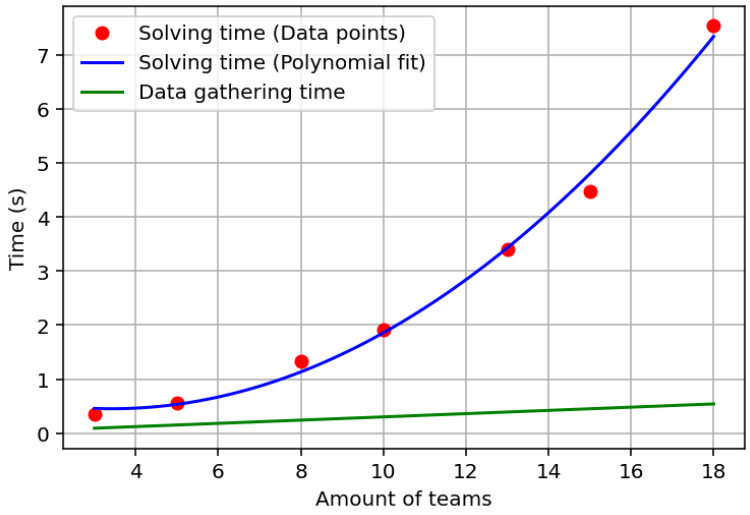

Figure 19: Computational time with an increasing number of teams for the dynamic scheduling model

The computational performance of the re-optimization is measured by the number of total client that are re-optimized. Figure 20 shows the results for solving and data gathering time. The behavior data gathering time is very similar to the behavior with the dynamic scheduling model. The time increases almost linear with the number of clients increasing. The polynomial fitted to the solving time data points is defined by the equation:

$$t = 0.00232892x^2 - 0.0537021x + 2.44445$$

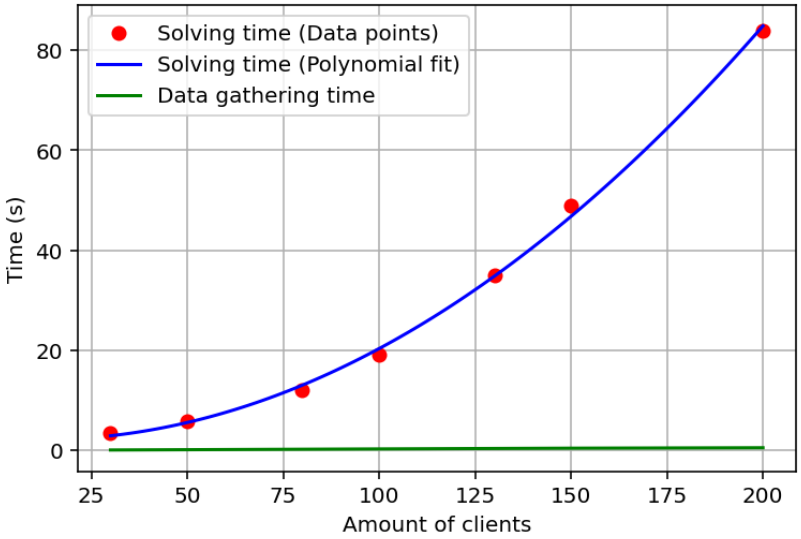

Figure 20: Computational time with an increasing number of clients for re-optimization

4.7 Optimal deterministic solution

The difference between the deterministic optimal solution and the results from the dynamic scheduling model in combination with the re-optimization are investigated. This could give useful insights about the performance of both proposed models.

The schedule from the third experiment in Chapter 4.4.2, with the piecewise linear cost function is chosen for the reason of significantly worse performance compared to the other experiments in the chapter. Once the schedule is inserted into the solver, a time limit of 24 hours is set. No optimal solution is found within this time unfortunately.

This could be explained by the number of options increasing significantly. The schedule contained a total of 100 clients. Chapter 4.6 stated that the re-optimization of 100 clients takes an average of around 20 second. However, not only the team-client assignment can be decided, also the period-client assignment. This could increase the computational time to a equivalent of 100*90 = 9000 clients, when the model considers an horizon of 90 periods. If we substitute this number into the equation of the polynomial from Chapter 4.6 we get:

$$0.00232892 * 9000^2 \ - 0.0537021 * 9000 \ + \ 2.44445 \approx 187.000 \ s$$

The absolute validity of this comparison may be subject to discussion, yet it may offer insight into the progression of computational time as the number of options increases. The calculated computational time equates to over 2 entire days of computation.

4.8 Real-life implementation

In this section, the real-life implementation of both models is explained. Data gathering and the communication between different data bases and algorithms are illustrated. For the re-optimization a recommendation on when and how to re-optimize is proposed

4.8.1 Dynamic scheduling model

In order to implement the model into a real-life organization, multiple cooperating python scripts using a database and a variety of APIs are developed, see Figure 21. Starting at the bottom left, a new client or job can be inserted by providing the relevant information to the client planner module.

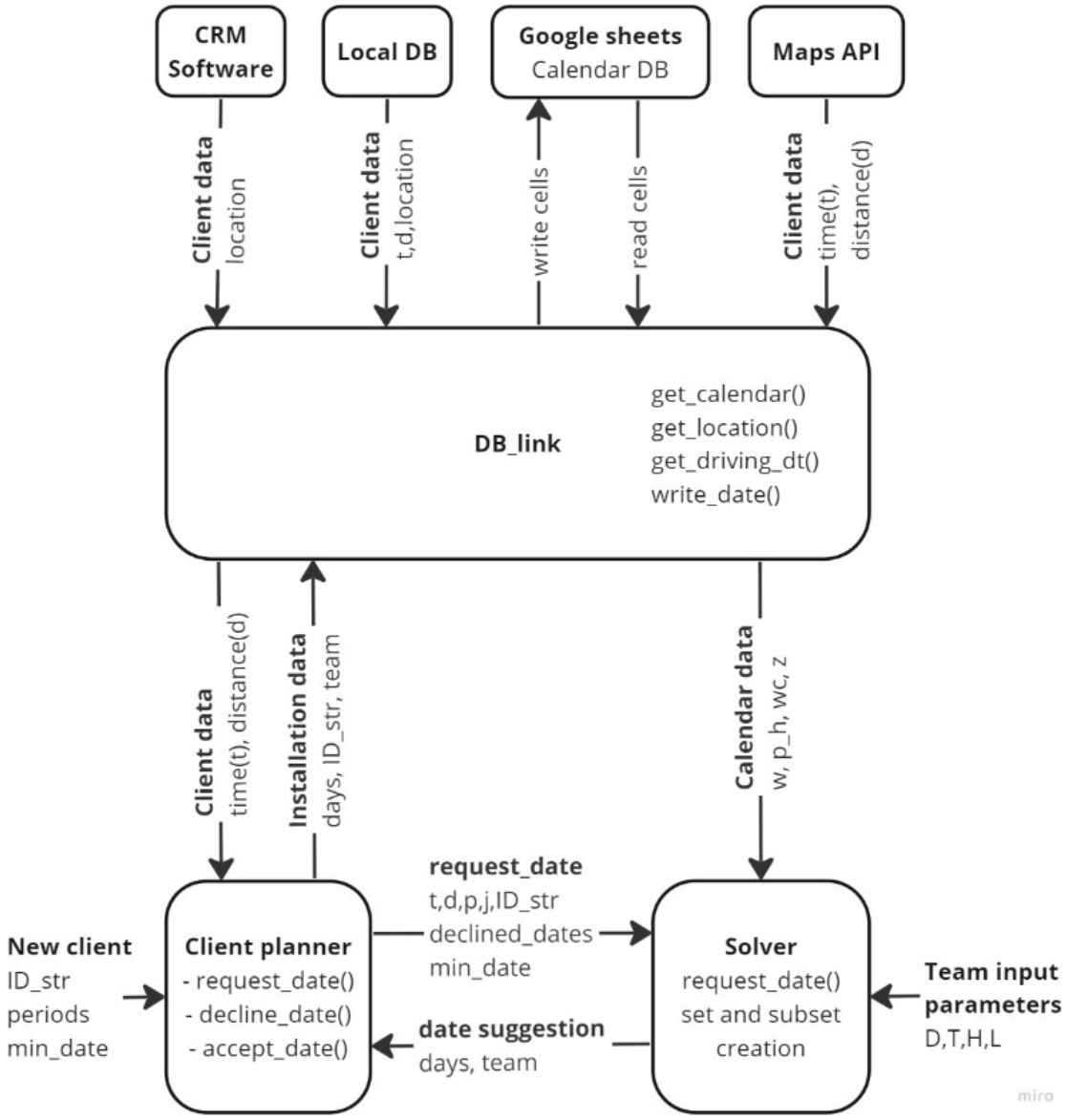

Figure 21: Real-life implementation structure of the dynamic scheduling model

The client planner module will create an object for the new client with all the relevant client information. It will receive the time, distance and location from the DB link module. It also converts the prefix in the ID string to the type j, with "A" being type 1 and "C" being type 2. This could be expanded or changed for future needs of course.

A timestamp is being created for the day after 1 January 2023 in order to link the periods in the solver to real word dates. This timestamp is being updated with every new request date to ensure correct period, real date matching.

After receiving or converting all the correct data, the client planner will execute the function: request date. This data stream is shown explicitly in the diagram since it is one of the most important ones.

The solver receives the request date data and the schedule data and creates all of the sets and sub sets needed to solve the scheduling problem. The solver is a MILP programmed in python using the open source PULP solver. Once the solver has found the optimal solution, it is sent to the client planner, where the date suggestion can be accepted or declined. If the date is declined, the suggestion is added to the declined dates attribute in the client planner. The solver can interpret this declined dates and converts them to $wc_{sp} = w_{sp} = 0$ for all the teams. This ensures no date can be suggested twice, since this would be unfavorable from the sales stakeholder goal. If a date is accepted, it will be written to the schedule stored in google sheets. This operation is executed by the DB link

The DB link handles all of the communication, conversion and data gathering between the different modules APIs and databases. One of the most important functions is the location and driving distance and time function. If a request for a location is made, the module first checks in chronological order: Local DB, CRM Software and as last the user can input the location. The local DB is used to speed up the gathering of locations when re-optimization is executed, see next chapter. Once the location is known the driving time and distance during the rush hours are gathered from the google maps API. The next important function is the conversion of the schedule data to input variables for the mathematical model inside the solver. A series of filters and conditional statements convert the cell data to the correct format.

4.8.2 Re-optimization model

First, the update policy of the re-optimization is proposed using the literature review in Chapter 2.1.3 as a reference. Next, the structure of the real-life implementation is explained. The literature review in explored multiple options of implementing a re-optimization model. Periodic update, key-point update or client update are not relevant for this specific implementation. After a certain time span, as with periodic update, there might not be any new information available and thus nothing to re-optimize. The re-optimization could be most effective using the dynamic update policy. This policy can be triggered by any of the following 3 dynamic events

Event 1: An entire week is scheduled, meaning no more open spots to be filled in that particular week. The probability of possible improvements is very high, as shown in chapter 0. The re-optimization model will reconsider every client, team assignment in order to optimize for the lowest objective function possible.

Event 2: A manual adjustment in the schedule is made by one of the employees of the company. This could be the cancellation of a client, the addition or removal of vacation days or the manual addition of a new client. After such an manual adjustment is made, the re-optimization model can be triggered to find a better solution for the current schedule with the newly added information

Event 3: A team input parameter is adjusted, such as the maximum driving time. Such a dynamic event can have a great influence on the performance of the schedule. Once a parameter is adjusted, the re-optimization model can re-optimize for a certain range of periods for which the parameter(s) are changed

The re-optimization model can be implemented using the structure in Figure 22. Once the re-optimization in triggered by one of the three events explained above, the solver will gather the necessary data for the specified range of periods. The DB Link script ensures all the data is gathered and checked, with sources being the different APIs or databases.

Once all the required data is gathered, the sets and sub sets are created by the solver module. These sets include the client-team and client-period assignments of the current, pre optimization schedule. The solver finds the lowest possible objective function and finds the resulting team changes. If no beneficial changes are found, the solver terminates and waits for the next trigger

If the solver finds team changers that improve the objective function, the suggested team switches are sent to the DB link script to write it to the google sheets. The DB link script first removes all of the old assignments of the clients. Next, the new assignments are inserted into the google sheets and the solver is terminated, waiting for the next trigger.

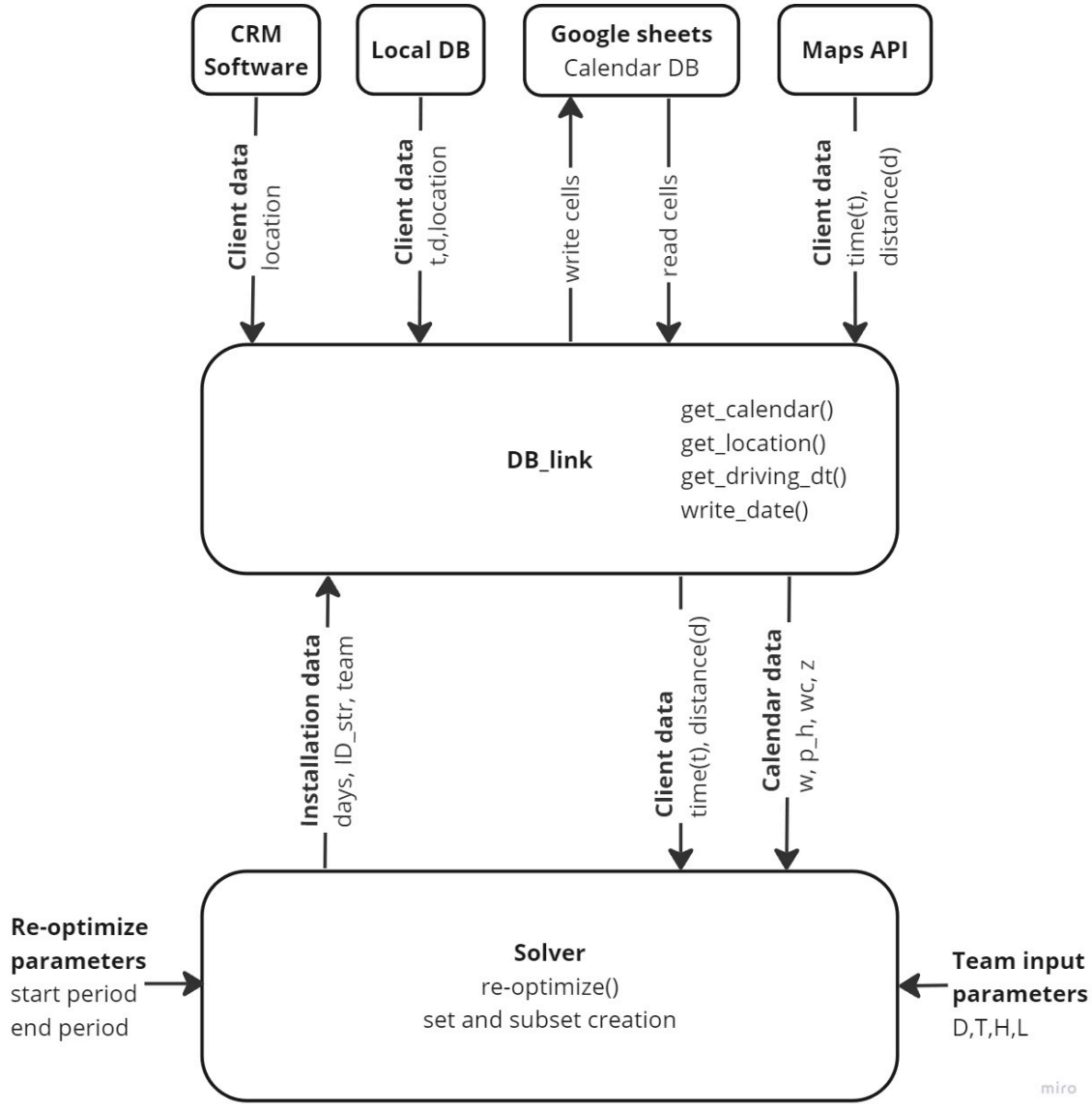

Figure 22: Real-life implementation structure of the re-optimization model

5 Conclusions and Future Recommendations

Multiple aspects of the methodology and the case study will be discussed in this chapter. Shortcomings, observations and potential areas to be improved upon will be highlighted. First the methodology is discussed, followed by the case study and accompanying results and finally the future recommendations. Next, future recommendations for model improvements, or promising directions are made. Finally, the chapter is concluded with answering the research question with its accompanying sub questions.

5.1 Conclusion

The sub questions to the main research question: "How can the dynamic scheduling with stochastic customers be solved efficiently using a real transportation network?" are answered. The main research question can be answered after formulation the answers to the sub questions.

What is a promising method for the dynamic scheduling problem to be modelled?

Multiple approaches for modeling the dynamic scheduling problem are explored in chapter 2 by reviewing relevant literature. Different aspects of the problem are investigated with their accompanying modeling methods. A promising method is proposed based on the combination of the reviewed literature and the problem definition. As a result, an optimization model with the dynamic insertion of clients is chosen over a rule-based or agent based approach. This method provides a promising method for modelling the dynamic scheduling problem

How can the dynamic scheduling problem be solved efficiently?

Different solution methods for the proposed dynamic scheduling model are reviewed in chapter 2.2. Exact, (meta) heuristics, learning based algorithms and a combination of these options are explored. Solution methods are reviewed in depth for their underlying working principles and application scenarios. Finally the methods are compared based on complexity and computational performance. A exact solution method is chosen based on the combination of the literature review and the problem definition. The overall complexity and size of the problem is relatively small, rendering exacts solution methods the most promising option in terms of computational performance with respect to time.

How can the re-optimization of the schedule be modelled and solved?

Dynamic scheduling modelling methods are investigated regarding the implementation of re-optimization. Different methods for re-optimizing a schedule are: key-point, client, periodic and dynamic updating. After careful consideration, including requirements from the problem definition and case study, dynamic update is chosen as the promising method. Re-optimizing the schedule is only required when a dynamic event takes place. Next, the solution methods are investigated. Re-optimization of a complete schedule increases the complexity in relation to the stepwise insertion of clients with the dynamic scheduling model. However, exact solution methods remain a viable option. The availability of computational time is greater compared to the dynamic scheduling model, thus poses the added complexity and increased computational time no problem to meeting the requirements.

What are the requirements for integrating the models with a real life system?

The real-life implementation of the dynamic scheduling model and re-optimization model has been explained supported by an illustration of the implementation structure. The dynamic scheduling model involves the use of multiple cooperating Python scripts, a database, and various APIs to handle data gathering and communication between different modules. The client planner module receives new client information, retrieves relevant data from the database, and executes functions to generate date suggestions. The solver, implemented as a Mixed Integer Linear Programming (MILP) model using the

PULP solver, receives the request for a new date and schedule data, creates necessary sets and subsets, and finds an optimal solution for the scheduling problem. The client planner module accepts or declines the date suggestion, updates the declined dates, and communicates with the DB link module to write the accepted dates to the schedule stored in Google Sheets.

The DB link module plays a crucial role in handling communication and data gathering between different modules, APIs, and databases. It retrieves client location information from the local database, CRM software, or user input and obtains driving time and distance during rush hours from the Google Maps API. Additionally, it converts the schedule data stored in google sheets to the correct format for input variables in the solver through filters and conditional statements.

For the re-optimization model, the literature review identified the dynamic update policy as the most promising approach. Three dynamic events trigger the re-optimization: when an entire week is scheduled, when a manual adjustment is made to the schedule, or when a team input parameter is adjusted. In each event, the solver gathers necessary data, creates sets and sub-sets, and optimizes for the lowest objective function based on the specified range of periods. If beneficial team changes are found, they are sent to the DB link script, which updates the assignments in Google Sheets. If no beneficial changes are found, the solver waits for the next dynamic event.

What is the performance of the proposed methods under different scenarios?

The performance of the proposed methods was evaluated under different scenarios. In the first set of experiments the variation of the number of teams and depots are explored. The optimization model outperformed the rule-based model in most scenarios by reducing the travel cost up to 17 percent. This can be contributed to the optimization models' ability to make a trade-off between the risk of an open spot and travel cost. The rule-based model cannot make this trade-off due to its chronological order of decision making. It is important to note the optimization model decreased the travel cost at the expense of the risk of lower human resource utilization. As a result, the importance of a properly tuned cost function by performing worse compared to the rule-based model on specific scenarios is emphasized.

In the second set of experiments in chapter 4.4.2 and 4.4.3 the cost functions definition and weight factor are investigated. Logarithmic and a piecewise combination of logarithmic and linear showed consistently improved performance compared to the linear cost function. The two promising cost functions better represented the desired trade-off between human resource utilization and travel cost. A pareto analysis is conducted, resulting in a graphical representation of the trade-off between travel cost and the risk lower human resource utilization.

In the third and fourth set of experiments, the models performance on larger and more complex scenarios is investigated. The optimization model showed the largest improvement of 25% for travel cost for the largest scenario. This can again be contributed to the ability to make a calculated trade-oof between the multiple objectives, where the rule-based model cannot.

The re-optimization model showed a consistent improvement over the existing schedule. An improvement of 0,2% up to 6,7% for the travel cost was made. No improvements over the open spots or risk of lower human resource utilization could be made. This is a result of the re-optimization model definition based on the problem formulation.

How can the dynamic scheduling with stochastic customers be solved efficiently using a real transportation network ?

In conclusion, the dynamic scheduling model can significantly improve the schedule for a real transportation network. The re-optimization model can further improve this performance by a small number of reduction in travel cost. Finding suitable parameters for the risk cost function is of great

importance to balance the trade-off between the multiple objective. The models can be implemented in a real-life scenario using multiple python modules and APIs.

5.2 Discussion of methodology

The time for development and implementation was 6 weeks for the rule-based model, compared to 24 weeks for the optimization model. This could have given the optimization model an advantage. If more time was available for the rule-based model improvements could be made to its shortcomings. Selecting the depot with the lowest travel cost without selecting the earliest period is one of the main factors for its underperformance. This factor became more significant when to number of depots increased. Efforts could be made to design a condition that could combine both travel cost and the risk of unutilized resources.

The computational time of the dynamic scheduling model is could possibly be decreased by a more simple formulation. The model is formulated with four decision variables, this greatly increases computational time. The re-optimization model is formulated with 6 decision variables, increasing the computational time even more.

The re-optimization method could interfere with the influence of the risk cost function. The utilization of teams will be altered if team-client assignments are changed by the re-optimization. it is important to recognize that the cost function has been meticulously tuned using the Pareto analyses presented in chapter 4.4.3. Pareto analyses involve evaluating and optimizing for multiple objectives simultaneously, considering trade-offs between conflicting objectives. The careful calibration of the cost function ensures that it adequately reflects the projects priorities and goals. However, introducing the re-optimization method may introduce changes to the system that could potentially disrupt the delicate balance achieved through the Pareto analyses. As a result, the influence and effectiveness of the cost function may be compromised, necessitating a careful evaluation of the implications of re-optimization on the overall project outcomes.

The current implementation of the re-optimization model introduces a risk of resulting in a non-valid schedule. A multi period client could be only partially included in the re-optimization when a start or end period in between the client periods is selected. As a result, the re-optimization can only safely be used for the entire existing schedule. Efforts could be made to check if a client had multiple periods, follow by including all of the clients periods into the re-optimization.

5.3 Discussion of case study and results

All of the experiments are conducted on a set of real-life clients. This client set contained 250 clients with their locations and arrival date. A chronological or random selection was made if the experiment required less than 250 clients. A different client set could result in significantly different performance when comparing the rule-based to the optimization model. Efforts could be made to generate new client sets or select a different set of real clients.

In scientific experiments or studies, it is important to consider various factors that can potentially have an impact on the outcomes. One such factor is the ratio of clients per day to the number of teams. This ratio is not constant throughout the experiments, meaning that the number of clients arriving per day may vary in relation to the number of available teams. This fluctuation in the ratio could have a significant impact on the performance of the models being studied. This effect could be investigated further in order to get insights about this impact

5.4 Future recommendations

Throughout the development of the proposed models and writing the thesis, multiple areas to improve upon are identified. A number of recommendations stems from problem encountered during the testing or implementation phase. Other recommendations are insights or new ideas to further improve the proposed models.

Client self scheduling: A (part of) all the available options in the schedule could be presented to the client, if multiple options from the dynamic scheduling model are declined. The challenge is determining which options to present to the client. This could be based on stochastic customer information as an example, but multiple options should be explored to determine the best method

Flexible clients: Improvements in the scheduling model could be made by introducing flexible clients. These clients will be assigned a range of periods, where the exact period is yet unknown. The exact period assignment of a flexible client can be communicated later. The flexibility of these clients can improve the performance of both the dynamic scheduling and the re-optimization models. Since the period assignment for a flexible client is not fixed, the number of available options for the scheduling model is larger. As a result, better decisions can be made. The re-optimization model can also consider an increased number of options, expanding its decision capabilities from only team assignments to both team and period assignments.

Prediction of new client locations: Using client location data to predict where new clients are most likely to be located could potentially increase the performance of the models. However, careful consideration and testing needs to be done in order to verify the effectiveness of this implementation.

Variable cost function per team: The result in chapter 4.4.1 show significantly different results when the ratio of teams per depot changes. This could possibly be resolved by implementing a variable cost function per team. This cost function could be different for every team, instead of identical for each team such as the current cost function. Information about a number of input parameters should be included. The number of teams and the distribution of the teams over the depots is one of the most important factors. Other factors include information about the team constraints such as maximum driving distance or the maximum number of periods for one installation. Lastly, the overall future team utilization could also be taken into consideration. A lower number of available periods meaning a higher utilization.

Flexible constraints: Team constraints are implemented as rigid, meaning absolutely no violation of the constraints can be made. This could eliminate good solutions which are only a very small violation of the team constraints. Most MILP solvers allow for the implementation of flexible constraints. A penalty can be set in case of the violation of a constraint, to ensure a constraint is only violated if the overall solution improves

Multiple periods in one day: The current implementation uses one period for one day. This fulfills the requirements of the case study at the moment of conducting this research. However, future requirements can include the separation of one day into multiple periods. The installation time will get shorter with increasing experience, opening up the possibility of multiple installations in one day. This could be implemented by expanding the current schedule for multiple cells or entries per day. Constraints could be added to differentiate between morning and afternoon periods for example.

Grouping clients: Clients with relatively close locations could be grouped together. Grouping could be implemented in a model with a single period per day or multiple periods per day. In a single period per day this could be beneficial if the transportation cost is very high for all of the clients in a group. A team could stay close to the client location in-between days to save on transportation costs. With multiple

periods per day the teams could drive from one location to the other in a short number of time, leaving enough time for the installations themselves.

Dynamic constraints: Team constraints can change over time, different agreements can be made about cost, maximum driving time or other parameters. This can be implemented by adding the team index to the already existing team input parameters. Careful consideration is needed when implementing dynamic constraints with the current model formulation. Matching the period of the dynamic constraints to the period notation of the schedule has to be taken into account. The schedule always starts at $p = 1$ with the current formulation. A conversion between the actual period, e.g. day of the year, to the schedule period needs to be made.

6 Appendix

6.1 Scientific paper

Dynamic scheduling with stochastic customers for a real-life application with multiple depots

J.M. van Beem
Supervisor: Dr. B. Atasoy

***Abstract*— This paper presents a dynamic scheduling model based on optimization and a re-optimization model in order to address the limitations of a rule-based scheduling model for a real-life application. In the problem at hand, a number of teams depart from a number of depots to install systems at clients' homes. The two contradicting goals are minimizing transportation costs and maximizing human resource utilization. Experiments show a reduction of up to 25 % in transportation costs while also realizing improved resource utilization. The importance of defining a risk cost function to balance the trade-off between the objectives is emphasized with experiments and a Pareto analysis. Re-optimizing schedules generated by the dynamic scheduling model result in an improvement in travel cost of up to 6.7 %. Computational time increases almost quadratically with the number of teams. Finally, future recommendations include the implementation of flexible clients and grouping clients as the most promising directions.**

I. Introduction

Efficient scheduling of services for customers poses a big challenge in real-life situations for various systems. Changes in resource availability, predicting future customer requests, and customer cancellations are some examples [6]. Stakeholders can have different and sometimes even contradicting requirements for an optimized schedule. The optimal solution for such a schedule should include all requirements and constraints. Manually optimizing such a schedule is possible when the problem size is small and the number of constraints or requirements is low [7]. For larger or more complicated problems, a computer model is needed to optimize the planning. A rule-based method can be used, but the limitations of such a model can lead to a sub-optimal schedule.

This paper focuses on a case study at a company installing heating and cooling systems at individuals homes. Different type of installations are carried out by the company, resulting in different installation difficulty levels. Multiple depots are available where the systems are stored and the teams depart from. The workforce consists of multiple teams with accompanying vans, each team located at one of the depots. Teams can have a different level of experience or skill level. Installations can have a different completion time, caused by the skill level of the installer, or the difficulty level of the installation. The schedule consists of all the working days in a week, excluding the national holidays. The teams can determine their own availability by communication their vacation or off days in advance. This reduces the number of available options in the schedule.

One of the companies goals is generating profit, as a result, the minimization of operational cost is one of the goals for an optimized schedule. This cost is simplified by breaking it down into two main contribution factors: transportation cost and human resource cost. Minimizing the transportation cost can be done by reducing the time and distance traveled by the vans. The human resource cost is constant, even when no work is available for the installers. As a result, the maximization of human resources is the second goal for minimizing the operational cost. The combination of these two goals results in a multi objective scheduling problem. The two objectives are as follows:

- Minimizing transportation costs
- Maximizing human resource utilization

Another important aspect of creating an schedule is customer satisfaction. The goals for this aspect are scheduling the client within seconds and not changing the assigned day(s) once the client is scheduled. This results in two extra requirements listed below

- Scheduling of a new client within seconds
- Fixed client-period assignment

A research question is defined based on the problem definition and requirements: **"How can the dynamic scheduling with stochastic customers be solved efficiently using a real transportation network ?"**

The remainder of this paper is organised as follows: Section II presents a literature review to investigate the current proceedings in the scheduling field. Relevant variants with their accompanying solutions are discussed as will as the contribution of this paper to the current literature. In Section III the current rule based scheduling solution is explained and its shortcomings elaborated. Section III continues with defining a optimization model for both the dynamic client scheduling and the re-optimization of the schedule. To incorporate the models into the real-life environment, a structure is proposed for the communication, integration and data storage. To evaluate the models, numerous simulations are done in Section IV, both with real and synthetic scenarios. Section V presents a discussion about the proposed methods and results, followed by future recommendations and a conclusion.

II. Literature Review

The scientific literature on the topic of dynamic scheduling with stochastic customers is studied as it directly related to the problem at hand. The scheduling problem is defined by the allocation of resources in order to minimize or maximize an objective function. There are three main categories of approaches to the scheduling problem: conventional, rule-based, and distributed solving [1]. The conventional approach involves developing a mathematical model and optimizing for the objective function. Rule-based is constraint-directed and rule-based. Those rules are often found by simulation or experimentation. Lastly, distributed solving implements use a multi-agent approach. Each agent has its own set of tasks and responsibilities. The construction of different layers with agents using bi-directional communication is often part of the implementation. Conventional modeling arises as the most promising approach to the scheduling problem defined in the introduction [5]. This choice is based on the unfavorable results produced by the previously developed rule-based model. Distributed solving is often used for more complex processes involving multiple decision layers [4]. A combination of a dynamic scheduling model based on optimization and re-optimization for dynamic events arises as a promising method.

The objective function often includes minimizing the total completion time of all the tasks to be done [2]. Other frequently used objective functions can include the maximization of resource utilization or the minimization of total operational costs. A multi objective function often has multiple contradicting objectives. An example of this is both minimizing the environmental impact and maximizing profit. In almost all cases, if the environmental impact is minimized, the profit will also go down. To mitigate this problem, a trade-off needs to be made between the different objectives. A Pareto analysis is often used to get useful insights into the trade-offs between the different objectives.. Each solution in this set is non-dominated, meaning that for all the solutions inside of the Pareto analysis, there exists no solution that can improve on all objectives. Using this Pareto analysis has multiple benefits, such as the trade-off analysis or a sensitivity analysis. The latter can be used to asses changing parameters in the model and evaluate how the frontier shifts. Useful insights can be extracted and informed adjustment can be made.

Both risk minimization (RM) and chance-constrained programming (CCP) are studied as promising solutions for the requirements and objectives [3]. Risk minimization arises the most promising method for modeling multiple objectives. Finally, multiple solution methods are discussed, such as exact, meta-heuristics, and learning-based algorithms. Since the problem size is relatively small, exact methods are chosen as the most promising solution method.

Four types of modeling dynamic events are studied: periodic, key point, dynamic and client update [8]. The dynamic update is chosen as the promising method after careful consideration, including requirements from the problem definition and case study. Re-optimizing the schedule is only required when a dynamic event takes place.

Two models based on optimization are chosen as a promising method for providing a solution for the problem at hand. The first model is a multi objective dynamic scheduling model, which will schedule new clients within seconds. The second is a re-optimization model that can change client-team assignment in order to further reduce travel costs. Stochastic customers can be modeled with risk minimization by including the risk in the objective function. The trade-off between the multiple objectives can be investigated with a Pareto analysis Dynamic events can be modeled by re-optimizing for dynamic events. Exact methods are chosen as a promising solution methods, resulting from the relatively small problem size.

The contribution of this paper to the current literature is twofold. Firstly, existing models are tailored to the specific case study presented in this paper. This produces insights into the models behaviours and limitations. Secondly, a novel constraint is formulated and evaluated in order to ensure consecutive multi period installation scheduling.

III. Methodology

This scientific methodology proposes a solution to the limitations of a rule-based scheduling model by proposing a dynamic scheduling model based on optimization. The rule-based model uses a set of conditions in order to schedule clients. As a result, using these Conditions introduces several limitations, including a lack of consideration between multiple objectives and poor results for real-life scenarios.

A. Rule-based model

The rule-based model (R-B) is developed and implemented to provide a solution for the limitations of manual scheduling. This model is in use by the company from the case study for over 6 months at the time of writing. This model works by scheduling clients based on five conditions, with each condition reducing the number of options until the best option is found. The five conditions are:

- Travel time and distance limitations
- Skill level limitations
- Installation length limitations
- Selection of earliest option
- Selection of lowest team utilization

Multiple problems arise when using or simulating a scenario for the rule-based model. Since the model only checks one condition at a time, it is impossible to optimize for multiple objectives at the same time. Two main limitations are identified: Poor results in specific cases and no consideration between two objectives.

B. Dynamic scheduling model

The dynamic scheduling model (DSM) addresses these limitations by formulating a mathematical model with an objective function containing multiple objectives. The model uses a mixed-integer linear programming solver to find the optimal value for the objective function. The model considers client and team data inputs to generate input parameters, including distance and time of travel, skill levels, and maximum installation lengths.

First, a set of periods $P = p(p = 1, 2, \ldots, f)$ is defined, with f being the horizon considered for the schedule. Next, a set of teams $S = s(s = 1, 2, \ldots, k)$ is constructed, with k being the total amount of teams. Client set $I = i(i = 1, 2, \ldots, n)$, with n being the total number of clients is defined. The input variable w_{sp} indicates if team s is available or scheduled for another client by taking the value one. The decision variable x_{spi}, shown in equation 1 indicates if client i is scheduled to be serviced by team s in period p.

The objective function in equation 1 aims to balance multiple objectives, including maximizing human resource utilization and minimizing travel time and distance. The objectives have a weight factor in order to balance their trade-off. The risk cost function in section III-D is represented by aC_{sp} is used to balance this trade-off with respect to period p. The travel cost is represented by the distance and time with their own weight factors: $bd_{is} + ct_{is}$ Constraints are implemented to ensure that each client is visited for the correct number of periods, teams can only service one client per period, difficulty levels are matched with team skill levels, and driving distances and times do not exceed team limitations.

$$\text{minimize} \sum_{s=1}^{k} \sum_{p=1}^{f} \sum_{i=1}^{n} x_{spi} * (aC_{sp} + bd_{is} + ct_{is}) \quad (1)$$

The consecutive constraint ensures all the periods of a multi period installation can have weekends or vacation periods in between them, but no other installation periods. Alternative period q_s is created, along with subset $\hat{P}_s \subseteq P$, linking q_s to p. Input variable e_s is defined as $e_s = \sum_{p=1}^{f} w_{sp}$. Finally, two decision variables are created r_{sq_si} which is constrained to have the value one for any p after the value of x_{spi} has been one and $\hat{x}_{sq_si}$, which is x_{spi}, but with the alternative period. With these, the following two constraints can be formulated:

$$\begin{gathered} r_{s(q_s+1)i} + r_{sq_si} + \hat{x}_{s(q_s+1)i} - \hat{x}_{sq_si} \leq 2 \\ s \in (1, ..k), q_s \left(q_s = 1, ..e_s - 1\right), i \in (1, ..n) \end{gathered} \quad (2)$$

$$\begin{gathered} x_{spi} = \hat{x}_{sq_si} \\ s \in (1, ..k), p \in \hat{p}_s, q_s \left(q_s = 1, ..e_s\right), i \in (1, ..n) \end{gathered} \quad (3)$$

Equation 2 ensures all nonzero values of $\hat{x}_{sq_si}$ are consecutive by eliminating the possibility of $\hat{x}_{s(q_s+1)i} = 1$, while $\hat{x}_{sq_si} = 0$. Equation 3 relates x_{spi} to $\hat{x}_{sq_si}$, ensuring the periods and alternative periods are both representing the same day in the schedule.

C. Re-optimization model

The re-optimization model (ROM) is based on the DSM, with a few adjustments. All clients for a certain range of periods are inserted at once, providing the ROM with all the information needed for re-optimization. The model can change the client-team assignment to further reduce the travel cost. Client-period assignment is represented by input variable z_{pi} in equation 4. This constraint ensures the re-optimization model cannot alter the client-period assignment. Furthermore, the number of client-team assignments changes is represented by the decision variable uc_i shown in equation 5. Client-team changes that do not reduce the travel cost are undesirable, therefor an extra term is added to the objective function. This term is shown in equation 1, representing the sum of total client-team assignment changes.

$$\sum_{k=1}^{s} x_{spi} = z_{pi} s \in (1, ..k), p \in (1, ..f), i \in (1, ..n) \quad (4)$$

$$\sum_{i=1}^{n} uc_i \quad (5)$$

D. Risk cost function

Different risk cost functions are designed, including linear, logarithmic, and piece-wise variants, shown in Figure 1. The parameters in the cost function can be adjusted, such as the value of p where the cost equals one and the length of each section in the piece-wise variants. The risk cost function has an impact on the trade-off between multiple objectives. This impact is different from the weight factors in the objective function. The risk cost function alters the weight in the objective function based on the amount of periods an available option in the schedule is. This allows for a lowering in the risk of unutilized resources. If two options in the schedule exist, one at $p = 1$ and one at $p = 30$, the relative weights become 0 and a. This results in the possibility to schedule a new client in the closest spot, regardless the travel cost.

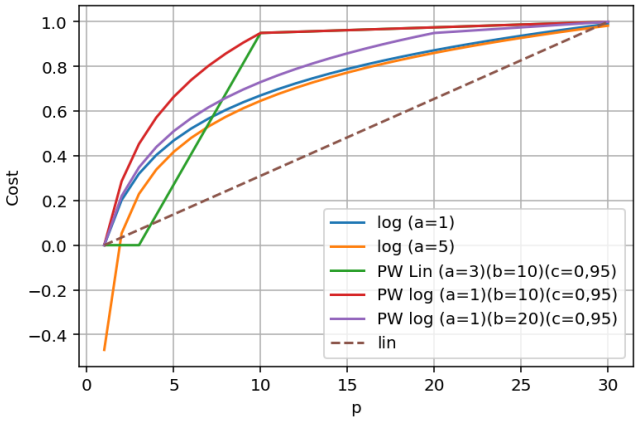

Fig. 1: Different risk cost functions

E. KPIs

The performance of the models will be measured using four different KPIs. The risk of unutilized resources (RUR) is defined as the period of the last client in the schedule. The unutilized resources (UR) are defined as the number of open spots in the schedule up to the arrival time of the last client. A list of the KPIs is shown below:

- Travel time
- Travel distance
- UR (open spots)
- RUR (makespan)

IV. Case Study

The proposed models are evaluated using a number of different experiments. A data-set containing 250 real clients is gathered from the companies database. The client data includes: client arrival date, type, location, preferred installation date, the number of declined date proposals by the client and installation length. The clients are inserted into the schedule by the dynamic scheduling model one at a time. Two types of schedules are used: firstly a schedule with full availability on weekdays. The second type of schedule includes historical off days and vacation days of the installation teams. The situation from the case study is defined as follows:

- 7 electric vans with installation teams
- Amount of teams per depot: (depot 1: 3), (depot 2: 3), (depot 3: 1)
- 3 depots at different locations
- 2 different installation difficulty levels

The scheduling of a new client is implemented by following a number of steps. This approach was used for manual scheduling, the rule based model as well as for the new proposed model. The steps are listed below:

- A new client sends a request for scheduling a new installation
- A date proposal is sent to the customer
- If the customer declines, a new date proposal is sent
- The accepted date is inserted into the schedule

The case study scenario is used for experiments in Chapter IV-A. This includes the amount, location and skill levels of the teams to be an exact copy of the situation at the case study. These experiments are conducted to gather useful insights into the dynamic scheduling models' behavior. Next, the impact of the risk cost function in different scenarios is investigated in Chapter IV-B . This influence is further investigated by a Pareto analysis. Chapter IV-C investigates an artificially expanded scenario by adding more teams and depots in order to investigate the performance on larger and more complicated scenario. Chapter IV-D shows the results for the re-optimization model, followed by the computational performance of the dynamic scheduling model in Chapter IV-E. Finally, the method for real life implementation is presented in Chapter IV-F along with the structured design.

TABLE I: Makespan and UR for the rule-based model compared to the dynamic scheduling model: Chapter IV-A

	Makespan		**Open spots**	
	R-B	*DSM*	*R-B*	*DSM*
2(1)	87	87	0	0
2(2)	87	87	0	0
4(2)	75	56	1	0
6(2)	38	39	0	0
7(3)	34	38	0	0

A. Number of teams and depots variation

The performance of the dynamic scheduling model is compared to the rule-based model on a different number of teams and depots. The first scenario only includes one depot with two teams and is expanded up to the actual situation at the case study with seven teams and three depots. The linear risk cost function, shown in Figure 1 is used. A total of 50 clients from the case study are inserted individually into the schedule. Figure 2 shows the potential of the proposed model to improve on both objectives.

The third experiment, with four teams and two depots, shows interesting results with higher travel costs and a lower RUR. This can be explained by looking at the first condition from the rule based model in combination with the team-depot distribution of three to one. The rule-based model can completely eliminate the possibility of one depot with the first condition. This means only one depot is left as an option for scheduling, as a result the makespan is increased significantly by one depot being assigned new clients further into the future. I shows this effect clearly with only a small reduction in makespan, while the amount of teams is doubled from experiment 2(2) to 4(2). This experiment clarifies the importance of defining a fitting trade-off between travel cost and the risk of unutilized resources (RUR).

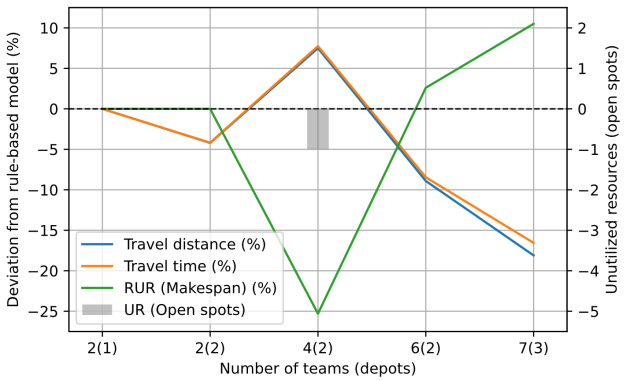

Fig. 2: Results for varying the number of teams and depots

B. Impact of the risk cost function

The trade-off between the objectives is investigated by analyzing the impact of different risk cost functions. The third experiment with four teams and two depots from section A is chosen as the benchmark. Figure 3 shows a great improvement for the RUR however, all alternative cost

functions result in a higher travel distance. This can be explained by examining the different risk cost functions in Figure 1. All alternative cost functions show a higher cost difference between lower periods, resulting in the dominance of the objective minimizing RUR. The logarithmic risk cost function arises as the most promising variant, balancing the trade-off between the objectives.

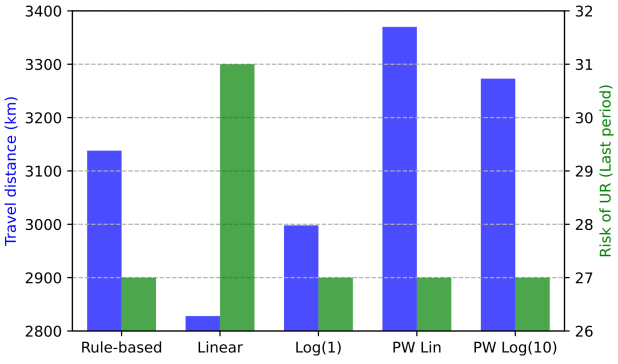

Fig. 3: Results for different risk cost functions

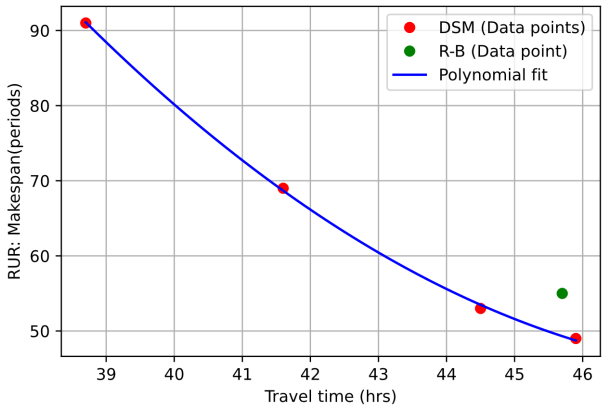

Fig. 4: Pareto analysis for cost weight factor

C. Performance on a large scale problem

The performance of the dynamic scheduling model on a large problem is investigated by designing a simulation with 250 clients, 18 teams, and 7 depots. The logarithmic risk cost was implemented due to the excellent performance in previous experiments. Team input parameters such as maximum driving distance and a maximum number of periods per client are included to increase this large problem's difficulty. The results of the dynamic scheduling model compared to the rule-based model for the large simulation are shown in Table II. The increased performance compared to previous experiments can be explained by the increased benefit of an optimized trade-off between the objectives for a larger amount of depots. The dynamic scheduling model can reduce travel costs by scheduling clients on a team with lower travel costs.

TABLE II: Performance of the DSM vs R-B for a large scale problem

	R-B	***DSM***	***Difference***
Travel time (hrs)	*265,2*	*197,5*	*-25,4%*
Travel distance (km)	23156	17265	-25,5%
UR: open spots (periods)	21	0	-21
RUR: makespan (periods)	62	57	-8,1%

TABLE III: Results of re-optimizing a selection of schedules generated by the DSM

No of teams	**Travel distance (km)**			**Travel time (hrs)**		
	Old	*Re-opt*	*Difference*	*Old*	*Re-opt*	*Difference*
2 (exp 1)	*3138*	*3080*	***-1,8%***	38,9	38,1	**-2,1%**
2 (exp 2)	3370	3144	**-6,7%**	41,2	39	**-5,3%**
4 (exp 1)	2998	2964	**-1,1%**	37,2	36,9	**-0,8%**
4 (exp 2)	3236	3206	**-0,9%**	39,7	39,3	**-1,0%**
6 (exp 1)	3273	3242	**-0,9%**	39,9	39,4	**-1,3%**
7 (exp 1)	3435	3331	**-3,0%**	42,1	40,9	**-2,9%**
7 (exp 2)	3359	3319	**-1,2%**	40,6	40,5	**-0,2%**
7 (exp 3)	3273	3242	**-0,9%**	39,9	39,4	**-1,3%**

D. Re-optimization

The performance of the re-optimization model is evaluated by re-optimizing complete schedules generated by previous experiments. No input parameters are changed, and the re-optimization model can exclusively alter the client-team assignment. The model showed a consistent improvement in travel cost of up to 6.7 %, show in Table III. No improvements on UR and RUR are made; this is a result of the fixed client-period assignment in equation 4.

E. Computational performance

The computational performance is investigated by comparing the data on computational time for a different number of teams and depots. The influence of the number of teams and number of depots is investigated. Two important factors influence the overall computation time. These two factors are the data gathering and the solving itself. The results in Figure 5 show an almost quadratic increase in solving time as the number of teams increases. The number of depots has no significant impact on performance. The data gathering time is almost linear to the number of teams in Figure 5.

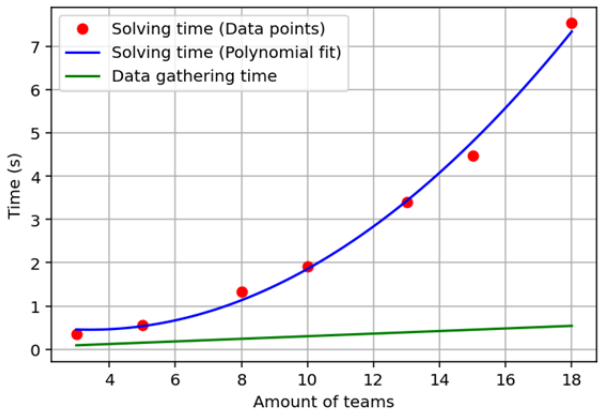

Fig. 5: Computational performance for the DSM

F. Real-life implementation

Both proposed models are implemented in a real-world environment. In order to implement the dynamic scheduling model into a real-life organization, multiple cooperating python scripts using a database and a variety of APIs are developed, see Figure 6. Starting at the bottom left, a new client or job can be inserted by providing the relevant information to the client planner module.The client planner module will create an object for the new client with all the relevant client information. It will receive the time, distance and location from the DB link module. It also converts the prefix in the ID string to the type j, with "A" being type one and "C" being type two. This could be expanded or changed for future needs of course.

A timestamp is being created for the day after 1 January 2023 in order to link the periods in the solver to real word dates. This timestamp is being updated with every new request date to ensure correct period, real date matching. After receiving or converting all the correct data, the client planner will execute the function: request date. This data stream is shown explicitly in the diagram since it is one of the most important ones.

The solver receives the request date data and the schedule data and creates all of the sets and sub sets needed to solve the scheduling problem. The solver is a MILP programmed in python using the open source PULP solver. Once the solver has found the optimal solution, it is sent to the client planner, where the date suggestion can be accepted or declined. If the date is declined, the suggestion is added to the declined dates attribute in the client planner. The solver can interpret this declined dates and inserts it into the input variables. This ensures no date can be suggested twice, since this would be unfavorable from the sales stakeholder goal. If a date is accepted, it will be written to the schedule stored in google sheets. This operation is executed by the DB link

The DB link handles all of the communication, conversion and data gathering between the different modules APIs and databases. One of the most important functions is the location and driving distance and time function. If a request for a location is made, the module first checks in chronological order: Local DB, CRM Software and as last the user can input the location. The local DB is used to speed up the gathering of locations when re-optimization is executed, see next chapter. Once the location is known the driving time and distance during the rush hours are gathered from the google maps API. The next important function is the conversion of the schedule data to input variables for the mathematical model inside the solver. A series of filters and conditional statements convert the cell data to the correct format.

The re-optimization model is implemented in a similar way to the DSM in Figure 6. The literature review identified the dynamic update policy as the most promising approach for the re-optimization model. Three dynamic events trigger the re-optimization: when an entire week is scheduled, when a manual adjustment is made to the schedule, or when a team input parameter is adjusted. In each event, the solver gathers necessary data, creates sets and sub-sets, and optimizes for the lowest objective function based on the specified range of periods. If beneficial team changes are found, the schedule is updated.

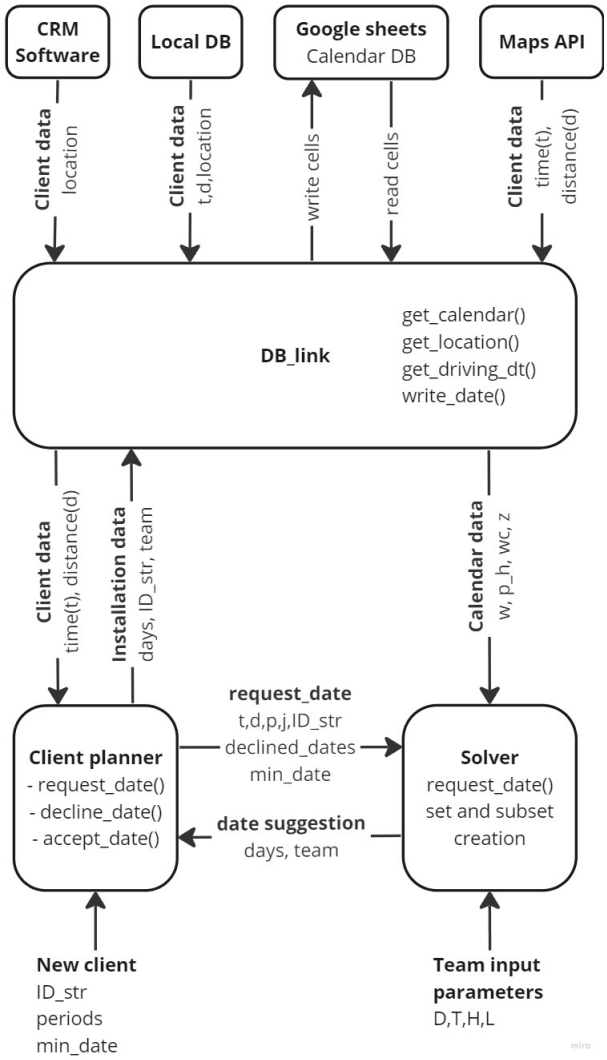

Fig. 6: Real life implementation structure

V. Conclusions and Future recommendations

First, two discussion points are highlighted, followed by the two most promising future recommendations.

The development time for the optimization models was six times greater than the rule-based model. Efforts could be made to improve the performance of the rule-based model.

The re-optimization model does not consider the trade-off between multiple objectives. This could influence decisions made by the dynamic scheduling model and disturb the fine-tuned balance realized by the risk cost function

The first future recommendation is the implementation of flexible clients. These clients will be assigned a range of periods where the exact period is unknown. The exact period assignment of a flexible client can be communicated later. The flexibility of these clients can improve the performance

of both the dynamic scheduling and the re-optimization models. Since the period assignment for a flexible client is not fixed, the number of available options for the scheduling model is larger.

The grouping of clients could be implemented for clients with relatively close locations. Grouping could be implemented in a model with single or multiple periods per day. In a single period per day, this could be beneficial if the transportation cost is very high for all clients in a group. A team could stay close to the client location in-between days to save on transportation costs. With multiple periods per day, the teams could drive from one location to the other quickly, leaving enough time for the installations themselves.

The main research question defined in Section I is answered:

In conclusion, the dynamic scheduling model can significantly improve the schedule for a real transportation network. The re-optimization model can further improve this performance with a small reduction in travel cost. Finding suitable parameters for the risk cost function is important to balance the multiple objectives' trade-offs. The models can be implemented in a real-life scenario using multiple Python modules and APIs.

REFERENCES

[1] P. Burke and P. Prosser. "A Distributed Asynchronous System for Predictive and Reactive Scheduling". In: *Artificial Intelligence in Engineering* 6.3 (July 1991), pp. 106–124. ISSN: 09541810. DOI: 10.1016/0954-1810(91)90034-L.

[2] Sönke Hartmann and Dirk Briskorn. "An updated survey of variants and extensions of the resource-constrained project scheduling problem". In: *European Journal of Operational Research* 297.1 (2022), pp. 1–14. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2021.05.004. URL: https://www.sciencedirect.com/science/article/pii/S0377221721003982.

[3] Sungjin Im, Benjamin Moseley, and Kirk Pruhs. "Online Scheduling with General Cost Functions". In: (2013).

[4] Zuo-Jun Max Shen et al. "Dynamic Scheduling with Uncertain Job Types". In: *European Journal of Operational Research* 309.3 (Sept. 16, 2023), pp. 1047–1060. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2023.02.013.

[5] Man-Wen Tian et al. "Risk-based stochastic scheduling of energy hub system in the presence of heating network and thermal energy management". In: *Applied Thermal Engineering* 159 (Aug. 1, 2019), p. 113825. ISSN: 1359-4311. DOI: 10.1016/j.applthermaleng.2019.113825. URL: https://www.sciencedirect.com/science/article/pii/S1359431119303606.

[6] Jin Xie et al. "Review on Flexible Job Shop Scheduling". In: *IET Collaborative Intelligent Manufacturing* 1.3 (2019), pp. 67–77. ISSN: 2516-8398. DOI: 10.1049/iet-cim.2018.0009.

[7] Zhiqiang Zeng. "Multi-object optimization of flexible flow shop scheduling with batch process — Consideration total electricity consumption and material wastage". In: *Journal of Cleaner Production* 183 (2018), pp. 925–939. DOI: 10.1016/J.JCLEPRO.2018.02.224.

[8] Haifei Zhang et al. "Review of Vehicle Routing Problems: Models, Classification and Solving Algorithms". In: *Archives of Computational Methods in Engineering* 29.1 (Jan. 1, 2022), pp. 195–221. ISSN: 1886-1784. DOI: 10.1007/s11831-021-09574-x.

6.2 Model verification

Constraint	Client input	Schedule input	Team input	Accept/decline	Expected output	Realized output
C1	("A-177", 1, 0)	-	Only team 1	Accept	X_1_1_1177 = 1	X_1_1_1177 = 1
C1	("A-177", 2, 0)	-	Only team 1	Accept	X_1_1_1177 = 1 X_1_2_1177 = 1	X_1_1_1177 = 1 X_1_2_1177 = 1
C1	("A-177", 3, 0)	-	Only team 1	Accept	X_1_1_1177 = 1 X_1_2_1177 = 1 X_1_3_1177 = 1	X_1_1_1177 = 1 X_1_2_1177 = 1 X_1_3_1177 = 1
C2	("A-177", 1, 0) ("A-178", 1, 0)	-	Only team 1	Accept	X_1_1_1177 = 1 X_1_2_1178 = 1	X_1_1_1177 = 1 X_1_2_1178 = 1
C2	("A-178", 1, 0) ("A-177", 1, 0)	-	Only team 1	Accept	X_1_1_1178 = 1 X_1_2_1177 = 1	X_1_1_1178 = 1 X_1_2_1177 = 1
C3	("C-177", 1, 0)	-	Only team 1 and 2 $H_1 = 1, H_2 = 2$	Accept	X_2_1_2177 = 1	X_2_1_2177 = 1
C3	("A-177", 1, 0)	-	Only team 1 $H_1 = 2$	Accept	X_1_1_1177 = 1	X_1_1_1177 = 1
C4	("A-177", 1, 0) $d_{1177,1} = 176$ $d_{1177,2} = 176$	-	Only team 1 and 2 $D_1 = 200, D_2 = 100$	Accept	X_1_1_1177 = 1	X_1_1_1177 = 1
C4	("A-177", 1, 0) $d_{1177,1} = 176$	-	Only team 1 $D_1 = 100$	Accept	No solution	No solution
C5	("A-177", 1, 0)	-	Only team 1 and 2	Accept	X_1_1_1177 = 1	X_1_1_1177 = 1

	$t_{1177,1} = 1.94$ $t_{1177,2} = 1.94$		$T_1 = 2, T_2 = 1$			
C5	("A-177", 1, 0) $t_{1177,1} = 1.94$	-	Only team 1 $T_1 = 1$	Accept	No solution	No solution
C6	("A-177", 2, 0)	-	Only team 1 and 2 $L_1 = 2, L_2 = 1$	Accept	X_1_1_1177 = 1	X_1_1_1177 = 1
C6	("A-177", 2, 0)	-	Only team 1 $L_1 = 1$	Accept	No solution	No solution
C7	("A-177", 1, 0)	$wc_{1,1} = w_{1,1} = 0$	Only team 1	Accept	X_1_2_1177 = 1	X_1_2_1177 = 1
C7	("A-177", 1, 0)	$wc_{1,1} = w_{1,1} = 0$	Only team 1 and 2	Accept	X_2_1_1177 = 1	X_2_1_1177 = 1
C7	("A-177", 2, 0)	$wc_{1,2} = 0$	Only team 1	Accept	X_1_1_1177 = 1 X_1_3_1177 = 1	X_1_1_1177 = 1 X_1_3_1177 = 1
C8 + C9	("A-177", 2, 0)	$wc_{1,1} = w_{1,1} = 0$ $wc_{2,2} = wc_2 = 0$	Only team 1 and 2	Accept	X_2_1_1177 = 1 X_2_3_1177 = 1	X_2_1_1177 = 1 X_2_3_1177 = 1
C8 + C9	("A-177", 3, 0)	$wc_{1,1} = w_{1,1} = 0$ $wc_{2,2} = wc_2 = 0$ $wc_{2,3} = w_{2,3} = 0$	Only team 1 and 2	Accept	X_1_1_1177 = 1 X_1_3_1177 = 1 X_1_4_1177 = 1	X_1_1_1177 = 1 X_1_3_1177 = 1 X_1_4_1177 = 1
C9 – C13	("A-177", 2, 0)	$w_{1,2} = 1, w = 0$	Only team 1	Accept	X_1_3_1177 = 1 X_1_4_1177 = 1	X_1_3_1177 = 1 X_1_4_1177 = 1
C9 – C13	("A-177", 2, 0)	$w_{1,2} = 1, w = 0$	Only team 1	Accept	X_1_4_1177 = 1 X_1_5_1177 = 1	X_1_4_1177 = 1 X_1_5_1177 = 1

		$w_{1,3} = wc_1 = 0$				
C9 – C13	("A-177", 2, 0)	$w_{1,2} = wc_1 = 0$ $w_{1,3} = 1, w = 0$	Only team 1	Accept	X_1_4_1177 = 1 X_1_5_1177 = 1	X_1_4_1177 = 1 X_1_5_1177 = 1
C9 – C13	("A-177", 9, 0)	$w_{1,9} = 1, w = 0$	Only team 1	Accept	X_1_10_1177… …. X_1_18_1177 = 1	X_1_10_1177… …. X_1_18_1177 = 1

Table 30

Constraint	Client input	Schedule input	Team input	Accept/decline	Expected output	Realized output
Declined days	("A-177", 1, 0)	-	Only team 1	Decline Decline Accept	X_1_1_1177 , X_1_2_1177 , X_1_3_1177	
Declined days	("A-177", 1, 0) $d_{1177,1} = 176$ $d_{1177,2} = 100$	-	Only team 1 and 2	Decline Accept	X_2_1_1177 , X_1_1_1177	
Min date	("A-177", 1, [today+10days])	-	Only team 1	Accept		

7 Bibliography

AbdAllah, A., Essam, D., & Sarker, R. (2017). On Solving Periodic Re-Optimization Dynamic Vehicle Routing Problems. *Applied Soft Computing*, *55*. https://doi.org/10.1016/j.asoc.2017.01.047

Burke, E. K., Cowling, P. I., & Keuthen, R. (2001). Effective Local and Guided Variable Neighbourhood Search Methods for the Asymmetric Travelling Salesman Problem. In E. J. W. Boers (Ed.), *Applications of Evolutionary Computing* (Vol. 2037, pp. 203–212). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45365-2_21

Burke, P., & Prosser, P. (1991). A distributed asynchronous system for predictive and reactive scheduling. *Artificial Intelligence in Engineering*, *6*(3), 106–124. https://doi.org/10.1016/0954-1810(91)90034-L

Campos, V., Corberán, Á., & Mota, E. (2008). A Scatter Search Algorithm for the Split Delivery Vehicle Routing Problem. In *Studies in Computational Intelligence* (Vol. 144, pp. 137–152). https://doi.org/10.1007/978-3-540-69390-1_7

Chaudhry, I. A., & Khan, A. A. (2016). A research survey: Review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, *23*(3), 551–591. https://doi.org/10.1111/itor.12199

Daniele, V., & Paolo, T. (Eds.). (2014). *Vehicle Routing—Problems methods and applications*. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611973594

Ehmke, J. F., Campbell, A. M., & Urban, T. L. (2015). Ensuring service levels in routing problems with time windows and stochastic travel times. *European Journal of Operational Research*, *240*(2), 539–550. https://doi.org/10.1016/j.ejor.2014.06.045

Fakhravar, H. (2022). *Combining heuristics and Exact Algorithms: A Review* (arXiv:2202.02799). arXiv. https://doi.org/10.48550/arXiv.2202.02799

Hartmann, S., & Briskorn, D. (2022). An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, *297*(1), 1–14. https://doi.org/10.1016/j.ejor.2021.05.004

Im, S., Moseley, B., & Pruhs, K. (2013). *Online Scheduling with General Cost Functions*.

Klau, G., Ljubic, I., Moser, A., Mutzel, P., Neuner, P., Pferschy, U., Raidl, G., & Weiskircher, R. (2004). *Combining a Memetic Algorithm with Integer Programming to Solve the Prize-Collecting Steiner Tree Problem*. *3102*, 1304–1315. https://doi.org/10.1007/978-3-540-24854-5_125

Konstantakopoulos, G. D., Gayialis, S. P., & Kechagias, E. P. (2022). Vehicle routing problem and related algorithms for logistics distribution: A literature review and classification. *Operational Research*, *22*(3), 2033–2062. https://doi.org/10.1007/s12351-020-00600-7

Kuo, R. J., Wibowo, B. S., & Zulvia, F. E. (2016). Application of a fuzzy ant colony system to solve the dynamic vehicle routing problem with uncertain service time. *Applied Mathematical Modelling*, *40*(23), 9990–10001. https://doi.org/10.1016/j.apm.2016.06.025

Laporte, G. (2001). *Classical heuristics for the capacitated VRP*. *109-128.*(The Vehicle Routing Problem).

Lenstra, J. K., & Kan, A. H. G. R. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, *11*(2), 221–227. https://doi.org/10.1002/net.3230110211

Li, B., Wu, G., He, Y., Fan, M., & Pedrycz, W. (2022). An Overview and Experimental Study of Learning-Based Optimization Algorithms for the Vehicle Routing Problem. *IEEE/CAA Journal of Automatica Sinica*, *9*(7), 1115–1138. https://doi.org/10.1109/JAS.2022.105677

Miyata, H. H., & Nagano, M. S. (2019). The blocking flow shop scheduling problem: A comprehensive and conceptual review. *Expert Systems with Applications*, *137*, 130–156. https://doi.org/10.1016/j.eswa.2019.06.069

Mohan, J., Lanka, K., & Rao, A. N. (2019). A Review of Dynamic Job Shop Scheduling Techniques. *Procedia Manufacturing*, *30*, 34–39. https://doi.org/10.1016/j.promfg.2019.02.006

Nura, A., & Abdullahi, S. (2022). A systematic review of multi-depot vehicle routing problems. *Systematic Literature Review and Meta-Analysis Journal*, *3*(2), Article 2. https://doi.org/10.54480/slrm.v3i2.37

Okulewicz, M., & Mańdziuk, J. (2020). *A Particle Swarm Optimization hyper-heuristic for the Dynamic Vehicle Routing Problem* (arXiv:2006.08809). arXiv. http://arxiv.org/abs/2006.08809

Pecin, D., Pessoa, A., Poggi, M., & Uchoa, E. (2017). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, *9*(1), 61–100. https://doi.org/10.1007/s12532-016-0108-8

Penna, P. H. V., Subramanian, A., Ochi, L. S., Vidal, T., & Prins, C. (2019). A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet. *Annals of Operations Research*, *273*(1), 5–74. https://doi.org/10.1007/s10479-017-2642-9

Pessoa, A., Sadykov, R., Uchoa, E., & Vanderbeck, F. (2020). A generic exact solver for vehicle routing and related problems. *Mathematical Programming*, *183*(1), 483–523. https://doi.org/10.1007/s10107-020-01523-z

Pillac, V., Guéret, C., & Medaglia, A. L. (2012). An event-driven optimization framework for dynamic vehicle routing. *Decision Support Systems*, *54*(1), 414–423. https://doi.org/10.1016/j.dss.2012.06.007

Pitt, M. A., & Myung, I. J. (2002). When a good fit can be bad. *Trends in Cognitive Sciences*, *6*(10), 421–425. https://doi.org/10.1016/S1364-6613(02)01964-2

Puchinger, J., & Raidl, G. R. (2005). Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification. In J. Mira & J. R. Álvarez (Eds.), *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach* (pp. 41–53). Springer. https://doi.org/10.1007/11499305_5

Suresh, V., & Chaudhuri, D. (1993). Dynamic scheduling—A survey of research. *International Journal of Production Economics*, *32*(1), 53–63. https://doi.org/10.1016/0925-5273(93)90007-8

Taillard, E., & Badeau, P. (1997). *A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows | Transportation Science*. https://pubsonline-informs-org.tudelft.idm.oclc.org/doi/abs/10.1287/trsc.31.2.170

Toth, P., & Vigo, D. (Eds.). (2002). *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898718515

Vidal, T., Laporte, G., & Matl, P. (2020). A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research*, *286*(2), 401–416. https://doi.org/10.1016/j.ejor.2019.10.010

Zhang, H., Ge, H., Yang, J., & Tong, Y. (2022). Review of Vehicle Routing Problems: Models, Classification and Solving Algorithms. *Archives of Computational Methods in Engineering*, *29*(1), 195–221. https://doi.org/10.1007/s11831-021-09574-x