MS-E2177 Seminar on Case Studies in Operations Research
Final Report for SOK

Amir Sultanbekov (project manager)
Pontus Heir
Hanna Karras
Kristo Laurila
Alvar Wilhelmsson

May 29, 2023

Contents

1 Introduction

1.1 Background

Suomen Osuuskauppojen Keskuskunta (SOK) is a cooperative organization that operates in the retail industry in Finland. SOK is the leading player in the Finnish retail market, accounting for nearly half of the grocery stores' market share. The company is responsible for the strategy of S Group, which comprises over 1,900 stores in Finland, including grocery stores, department stores, and service stations.

Maintaining a well-balanced labor force is crucial for SOK to ensure smooth operations across all its business units. In grocery stores, for example, the need for employees fluctuates seasonally, and the timing of vacations is therefore important to schedule accordingly. A poorly scheduled vacation plan can have adverse effects on employees' well-being, and work quality, and ultimately, increase personnel costs. For instance, if there are too few employees during peak periods, SOK may need to hire part-time workers to fill the gaps, which can significantly increase personnel costs. Moreover, inadequate staffing levels can lead to longer waiting times, decreased customer satisfaction, and even loss of business. Therefore, by developing an optimized vacation scheduling tool that can match workforce demand and supply, SOK aims to improve employee well-being, increase operational efficiency, and reduce personnel costs.

Maintaining a well-balanced labor force is crucial for SOK to ensure smooth operations across all its business units. In grocery stores, for example, the need for employees fluctuates seasonally, and the timing of vacations is therefore important to schedule accordingly. A poorly scheduled vacation plan can result in weeks with too few employees, which can have an adverse effect on employees' well-being and work quality, and ultimately, increase personnel costs.

To address this concern, we work with SOK to develop an optimized vacation scheduling tool that can match workforce demand and supply. The tool's main objective is to minimize weeks where there are insufficient employees while ensuring that there are always enough workers with the necessary skills to meet demand. Our model takes into account various factors, such as complying with laws and collective agreements, accommodating different employee skills, enabling vacations of different lengths, and considering employee vacation time preferences. The planning scope of the project is to develop a model that covers the summer months, as this is the peak holiday period for SOK's grocery stores.

The scope of the project involves creating an optimization model for vacation and absence planning on a weekly basis for approximately 100 employees over a one-year period, focusing specifically on the summer months. The tool should be user-friendly so that someone without optimization expertise can create vacation schedules using the model. The project only considers those employees

who work according to the work list, with supervisors or external contract workers not being part of the vacation schedule.

This project is a significant undertaking for SOK, and it represents the company's commitment to investing in innovative solutions that improve employee well-being and increase operational efficiency. By leveraging advanced optimization techniques, SOK aims to create a tool that can be scaled across all its commercial establishments and help to set a new standard for employee vacation planning in the retail industry.

1.2 Objectives

To develop an effective vacation optimization tool, there are several important considerations to keep in mind. First and foremost, the tool must be based on accurate and reliable data. This may include information on employee availability, skill sets, vacation preferences, and historical workload data. Gathering and analyzing this data can help to inform the model's design and identify potential constraints that need to be considered. The model should also take into account relevant laws and regulations related to labor and vacation time, as well as collective agreements with employee representatives.

In addition to data collection and legal considerations, it is important to collaborate with stakeholders to ensure that the optimization tool meets their needs. This may involve working closely with store managers and human resources personnel to understand their pain points and incorporate their feedback into the model. By engaging with stakeholders in this way, it is possible to ensure that the optimization tool is designed to meet the specific needs of SOK's business units and that it will be effectively used once it is deployed.

Once the optimization model has been developed, it is important to evaluate its effectiveness and refine it as needed. This may involve running simulations using different inputs or adjusting the model's constraints to better reflect real-world scenarios. For example, it may be necessary to adjust the vacation schedule in response to unexpected fluctuations in demand or to accommodate changes in employee availability or preferences. By refining the optimization model in this way, it is possible to ensure that it continues to meet the evolving needs of SOK's business units over time.

To ensure that business units can use the optimization tool effectively, it may be necessary to provide support to help them understand how to use the tool and interpret its outputs. This can help to ensure that the model is implemented successfully and achieves the desired results. It is also important to ensure that the optimization tool is user-friendly and accessible to all employees, regardless of their technical skills or level of expertise.

Expanding on the importance of providing support for using the optimization

tool, it is crucial to note that the successful implementation of the tool depends on how well the employees can use and understand it. Since not everyone may have the technical skills or expertise to operate the tool, providing support and training can help to bridge the knowledge gap and ensure that the model is used effectively.

Moreover, it is important to ensure that the optimization tool is user-friendly and accessible to all employees. If the tool is complex or difficult to use, it may deter employees from using it, which can hinder the successful implementation of the model. Therefore, the tool should be designed with a user-friendly interface that makes it easy for employees to operate and interpret its outputs.

By ensuring that the optimization tool is user-friendly and providing support for its use, SOK can effectively leverage the tool's benefits, such as improving operational efficiency, reducing costs, and enhancing overall productivity. Additionally, when employees feel confident in using the tool, they can make better decisions based on the model's recommendations, which can lead to improved outcomes, such as increased customer satisfaction and reduced personnel costs.

Overall, the development of an effective vacation optimization tool for SOK's business units represents an important opportunity for our team to apply our skills in operations research to a real-world problem. By working closely with stakeholders and engaging in careful data analysis and model development, we are confident that we can create a tool that will help SOK's business units to optimize their vacation schedules and improve employee well-being, work quality, and reduce personnel costs.

1.3 Optimization theory

Optimization theory is a branch of applied mathematics that deals with the development of mathematical models to find optimal solutions to problems. The goal is to identify the best possible choice among a set of alternatives, given a set of constraints. Optimization problems arise in many areas of science and engineering, such as finance, logistics, engineering design, and resource allocation.

Mixed Integer Linear Programming (MILP) is a powerful optimization technique. It is a type of linear programming where some or all of the variables are constrained to be integer values. MILP problems can model a wide range of real-world problems, such as scheduling, network flow, and resource allocation.

MILP problems can be formulated as a set of linear constraints and a linear objective function, with the added constraint that some or all of the variables must be integers. The solution space for MILP problems is often discrete, meaning that the variables can only take on specific values. The challenge in solving MILP problems is to find the optimal solution within this discrete solution space.

One popular approach to solving MILP problems is branch and bound. This approach involves dividing the problem into smaller sub-problems and solving each sub-problem separately. The method proceeds by iteratively branching on variables, creating a tree-like search space of possible solutions. The algorithm uses bounds to eliminate sub-trees that are guaranteed to not contain optimal solutions.

MILP solvers have become increasingly powerful and efficient in recent years, enabling them to solve large-scale optimization problems that were previously thought to be intractable. These solvers can handle problems with millions of variables and constraints, making them a valuable tool for solving complex real-world problems.

In the context of the SOK vacation scheduling project, MILP is used to optimize the vacation scheduling plan to ensure that workforce demand is matched with supply while minimizing the number of weeks with too few employees. The MILP model takes into account various constraints, such as employees' skills, vacation time preferences, and the need to comply with laws and collective agreements. By using MILP, the project team can develop an optimal vacation schedule that meets the needs of the business and its employees. MILP will be discussed more in Section 2.

In this project, we use Julia programming language and its optimization package, JUMP, to solve the mixed-integer linear programming problem for vacation scheduling. JUMP is a powerful modeling and optimization package that allows us to define and solve complex optimization problems with ease. One of the main advantages of JUMP is its ability to interface with different solvers, providing flexibility and allowing us to choose the best solver for the problem at hand.

In this project, we choose to use the HiGHS solver, which is a state-of-the-art open-source solver for linear and mixed-integer linear programming. HiGHS is known for its excellent performance and scalability, making it an excellent choice for large-scale optimization problems. HiGHS was chosen over other solvers such as Gurobi, CPLEX, and MOSEK, as HiGHS license and usability matched the requirements for our project.

The combination of JUMP and HiGHS provides an efficient and robust solution to the vacation scheduling problem. By using these tools, we are able to find a high-quality vacation schedule that satisfies all the constraints and objectives in a reasonable amount of time. Furthermore, the flexibility provided by JUMP and the ability to switch solvers easily allows us to compare different solver performances and choose the most appropriate one for future projects.

2 Literature Review

Literature specifically about vacation optimization is scarce, however, there is a multitude of scientific papers concerning the optimization of scheduling and rostering in general [10].

2.1 Scheduling and rostering

The process of creating optimized work schedules for employees, known as personnel scheduling and rostering, has gained increasing attention in recent years. Ernst et al. [8] review hundreds of papers on having the appropriate personnel available at any given time, which is a crucial aspect for most organizations in meeting their customers' needs. In essence, the problem of rostering involves the allocation of qualified staff to meet time-sensitive service requirements while complying with workplace regulations and seeking to fulfill individual work preferences. These optimization problems are typically very complex and heavily constrained. Because different industries have distinct requirements, various rostering models exist, necessitating distinct solution methods to produce effective and practical solutions.

Van den Bergh et al. [16] provide a recent review of papers on personnel scheduling problems and found out that the primary focus is on staffing and/or scheduling workers based on fixed inputs. Based on the reviews, Van den Bergh et al. recommend that researchers incorporate a range of factors into their personnel scheduling problem, including demand forecasting, hiring and firing, machine scheduling, and multiple locations. By controlling these variables, operational advantages can be achieved. Many important aspects of the personnel scheduling problem are often overlooked, which limits the solution method's applicability in real-life problems where these characteristics are present. Therefore, it would be beneficial to incorporate as many aspects as possible, such as break placement, different skills, and flexible worker contracts. In recent years, companies have increasingly considered employee preferences, for example, requests for specific working days or shifts, assignments to a particular location or working partner, and preferred durations or start times, to satisfy their workforce and allow for flexible personal life management. While the literature on personnel scheduling has noted the growing trend toward flexibility, there are still significant opportunities to develop algorithms that can efficiently manage these preferences. However, the studies revealed that personnel scheduling problems have numerous variations with regard to both hard and soft constraints. However, the impact of these constraints on the problem's complexity has received little attention. Conducting a more focused theoretical investigation would enable researchers to better comprehend the effects of various constraints and provide an opportunity to develop suitable algorithms.

Koutsopoulos and Wilson [12] examine operator workforce planning in the transit industry, notably also considering vacation planning. Two fundamental problems were found in it. Firstly, the problem was determining the appropriate workforce size by taking into consideration related factors such as hiring decisions and vacation allocation. Secondly, the issue of maximizing the utilization of the available manpower was addressed through the introduction of two new models, tactical and operational models. The first model focuses on the day-to-day management of absences during a given week and allocates extra-board resources to minimize expected overtime while the operational model tackles the challenge of determining report times for extra-board operators on a specific day.

Mixed integer programming (MIP) and integer programming (IP) are established techniques for solving scheduling problems. Azmat et al. [2] introduce MIPs with the objectives of distributing working hours equitably across the workforce, minimizing annual overtime hours, and creating a yearly schedule for the small-company manufacturing workforce in Switzerland. Two holiday scenarios were described. In the first scenario, the workforce is given the opportunity to select their preferred holiday weeks from a predetermined set of options. The second holiday scenario utilizes a MIP formulation to assign holiday weeks. For both scenarios, two MIPs were presented. Three out of four MIPs were able to find feasible solutions for all test problems, with the maximum workload difference between workers being 15.4 hours at most, which is less than 1% of the normal work length per worker over a year, excluding holiday weeks.

Van den Broek et al. [4] examine the integration of personnel scheduling into the planning of railway yards, specifically within the context of the train unit shunting problem. The objective was to extend the problem to involve a conflict-free schedule for all yard activities. The main challenge arose from the significant walking distances between activities, as railway yards often span several kilometers of track. To address this problem, two efficient heuristics for staff assignment were proposed. These heuristics were integrated into a local search framework, allowing for the generation of feasible solutions for the train unit shunting problem with staff requirements. Notably, this was the first algorithm developed to solve the complete version of this problem. Additionally, a dynamic programming method was introduced to assign staff members as passengers to train movements, reducing their walking time. Furthermore, several ILP-based approaches were outlined to find robust solutions for the staff assignment problem, with these solutions used to evaluate the quality of the heuristic-generated solutions. Through experimentation on a dataset comprising 300 instances of the train unit shunting problem with staff scheduling on a real-world railway yard, the best-performing heuristic integrated into the local search approach demonstrates a success rate of 97% within an average runtime of three minutes.

2.2 Vacation planning

Chong et al. [5] develop a quantitative model to assign leaves for flying military squadrons with the objective of maximizing crew preferences. The process is limited by manning constraints, which impose restrictions on the number of crews that can be on leave at the same time. The assignment process is based on a point system that enables each crew to bid for their preferred periods. The periods are then assigned to the highest bidder, ensuring that each crew has an influence in the leave assignment process, and the outcome is entirely determined by the crews themselves. The method involves a two-phase partially open bidding system. During Phase I, crews registered their preferred periods on a posted calendar without assigning points to them. Crews were allowed to modify their period requests during this phase to avoid conflicts with other crews. The juggling process was closed at the end of Phase I, after which crews could no longer change their period requests. Crews were given a week between the end of Phase I and the start of Phase II to determine how many other crews they were competing with. In Phase II, crews submitted their points via closed bid.

Dewess [6] report a model for vacation planning for a German public transportation company. The model aims to create socially acceptable annual holiday plans while considering various legal, company, and driver-related constraints. These constraints include holiday entitlements, driver qualifications, and connections, among others. The model assigns benefit values to each application based on social criteria, such as family situations and application priorities. To solve this problem, a two-stage heuristic algorithm was used. In the first stage, the algorithm assumes that the applications for leave are approved, resolves capacity conflicts, and arranges the applications to obtain a feasible solution with a high benefit. In the second stage, the algorithm attempts to improve the solution further. Computational experiments indicate that the proposed algorithm can efficiently handle problem instances with up to 10,000 drivers within a reasonable amount of time.

2.3 Days-off scheduling and planning

The literature on workforce scheduling using integer programming approaches has placed considerable attention on addressing shift and days-off scheduling problems. These problems are commonly encountered in managerial scenarios where the goal is to schedule full-time employees in a manner that minimizes labor hours while effectively meeting the variable workforce requirements [14].

The scheduling of days off for staff members is a key concern in workforce planning. Two types of constraints need to be considered. The first type involves individual factors such as industry regulations, labor contracts, workplace agreements, and personal preferences. These constraints include ensuring em-

ployees work within specified upper and lower limits, have at least one day off per week, and limit the number of consecutive working days. The second type of constraint focuses on daily staffing requirements for each shift throughout the planning period. These constraints ensure that there are enough workers with the necessary skills for each shift, taking into account the predetermined staffing levels and the skills possessed by employees [11].

Elshafei et al. [7] introduce a dynamic programming algorithm to address scheduling problems related to employee days off in a single-shift and compressed workweek setting. The problem involved constraints on various factors such as the maximum work stretch, maximum number of workdays per week, and minimum number of consecutive off days per week. Notably, the problem featured a unique cost structure that is dependent on the work sequence, where the daily wage of each employee is influenced by the previous work assignments in the preceding days. This characteristic caused traditional integer programming models to be impractical for solving this problem effectively. To overcome these challenges, an efficient dynamic programming algorithm was developed to determine the optimal assignment of days off that minimizes the overall labor cost. Finally, the algorithm was successfully applied to a real-life employee scheduling problem and was extensively tested through a series of computational experiments.

3 Data and Optimization Model

This chapter introduces constants, variables, constraints, and objective function for the optimization model and examples are shown with the data. Data is gathered from the example Prisma unit, but all confidential information is encrypted, and information is randomized in an Excel file, and all the examples in the report are based on it.

As discussed in Introduction, the structure of the work can be divided into three parts. The first part is the initial data that was received from the Prisma unit in Excel files, which was collected and modified into a suitable format in one input Excel file so that it is more convenient for the model to read all the necessary data in one place. The second part is the most technical and mathematical part of the work, which is clearly the most significant regarding the project. In this section, the mathematical modeling of vacation optimization is studied and the Julia code is reviewed in the next section (Results). The final part of the structure of the work is the final result in Excel, where optimal vacations and statistics have been compiled for our client in a visualized form.

The model has employees, weeks, and skills with skill levels. Employees are the ones for whom the optimal vacations are planned, and they are modeled by $e = 1, 2, ..., E$. In the project, we use employee identification using an encrypted employee ID, which is a series of numbers but in this example data the personnel base is 20 employees. From a practical point of view, however, it is possible to create a solution based on the employee's first and last name. The unit can be divided into several departments $d = 1, 2, ...$ which all have their own employees, but in this example case, only one department is used.

Weeks are the period over which the vacations are placed, and are modeled by $w = 1, 2, ..., W$. In our project, the first week's number is 18, the first week of May, and the last is 35, the last week of August, since the optimization is done for the summer period. However, the practical solution has been implemented so that the user can define the desired parameters for the weeks and other variables.

In order to plan vacations so that there are always enough skilled employees in the unit, the skill and skill level are taken into account with $s = 1, 2, ..., S$. Every skill has a skill level which is in this example case from 1 to 3, meaning that an employee with skill level 3 on skill s has the highest knowledge regarding the skill. In addition, if the employee has skill level 3, it means she also has all skill levels below that skill level, that is levels 2 and 1 of that skill. In data handling, the skill level is coded to each skill, so, for example, one skill can be $skill_1 level_3$, and the number of skills s is the number of different skills multiplied by the number of highest skill level.

3.1 Constants

The model uses several constants, which are read into the model from the input data. Every employee has contractual hours, which means the minimum number of hours that employees work during the work week. The constant is modeled with H_e. Every employee has also a vacation allowance V_e measured in weeks. In the Excel input side, this is measured in days but changed to a weekly basis in data handling. This constant determines how many days an employee has a vacation, and the vacation can not be longer than that constant. A matrix is defined as all skills and employees, including which skills the employee has at which skill level. This is formulated as binary constant $x_{e,s}$. In input Excel, the matrix is not binary, but data handling changes the size of the matrix to ExS by reading the skill levels of the cells. Figure 1 presents the example data for employees' information.

$$x_{e,s} = \begin{cases} 1, \text{ if employee } e \text{ has skill } s \\ 0, \text{ otherwise} \end{cases}$$

employee	contract_h_week	vacation_summer	skill1	skill2	skill3	skill4
1	30	12	1	0	3	0
2	25	12	0	2	3	0
3	24	24	3	0	1	0
4	30	12	0	1	1	1
5	25	12	1	0	1	0
6	25	24	1	0	3	0
7	30	24	0	2	1	2
8	30	12	1	3	0	3
9	20	24	2	1	1	0
10	20	12	0	1	2	0
11	24	12	2	0	1	1
12	30	24	1	2	1	1
13	30	24	2	0	1	1
14	24	12	2	1	0	1
15	20	24	0	0	0	3
16	24	18	1	0	0	0
17	20	12	1	1	3	2
18	25	24	1	2	1	0
19	20	24	2	3	2	1
20	25	12	3	0	3	0

Figure 1: Worker data of twenty employees

The model optimizes the next summer's vacations based on last year's va-

cations (Figure 2.), thus the constant $Z_{e,w}$ refers to when each employee can have a long vacation this year. The constant is calculated during data processing from the past year's vacation plan and the vacation plan works with the circular method, which means that each employee is allowed to start their long vacation between four to eight weeks after the last year's start of the long vacation. This is a binary matrix for each employee for each week and the employee's vacation is often divided into at least two separate vacations, one of which is called a long vacation if it is a vacation consisting of at least two consecutive weeks. If the employee had no vacation last year, all weeks between June - August are allowed. The user can also write in the input Excel when none of the employees are allowed to start their vacation, for example, a busy week can be locked so that all employees are at work that week.

$$Z_{e,w} = \begin{cases} 1, \text{ if start of the long vacation is allowed for employee } e \text{ on week } w \\ 0, \text{ otherwise} \end{cases}$$

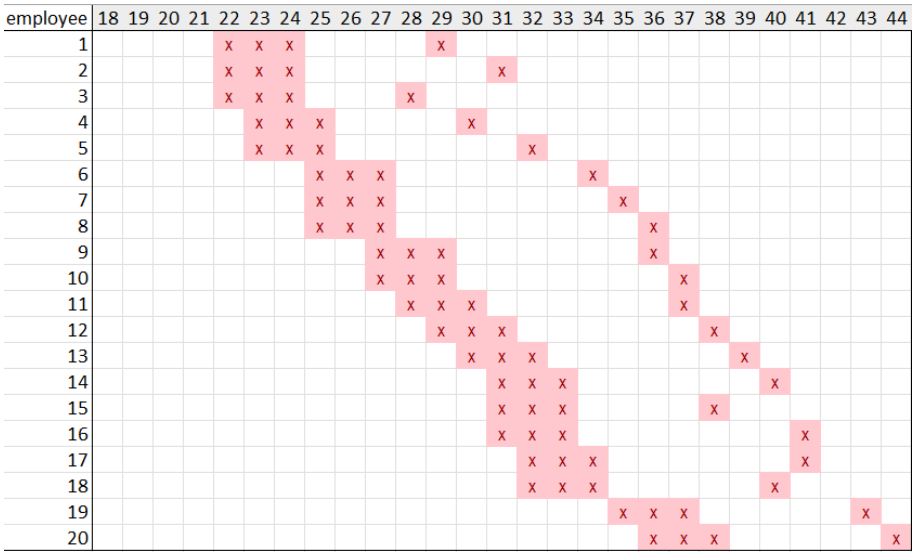


Figure 2: Vacation weeks of twenty employees from the last year

In addition to allocating vacations by the optimizer, the employee can request a week of vacation for themselves, which the user marks in the input data. Such a week is forced as a holiday for the employee and the information can be found in the binary matrix $f_{e,w}$. Requested vacation weeks consumes vacation days and user can also do manual corrections to vacation weeks for each employee by this constant.

$$f_{e,w} = \begin{cases} 1, \text{ if employee } e \text{ must have vacation on week } w \\ 0, \text{ otherwise} \end{cases}$$

In addition to holidays, employees can have absences, such as sick leave or parental leave, which can be marked by the user manually to input data. These absences do not consume the employee's vacation days but reduce the unit's total working hours. This constant is $p_{e,w}$, which is a binary matrix.

The model operates with two types of working hours. Other working hours are those for which the employee is required to have a certain level of competence in a certain skill. These hours are defined in the input data, as how many hours of work are required for each skill and skill level in total working hours during the week. This constant is modeled by r_s and it is the same for all weeks. Another of these constants models how many hours of general work per week the unit should do, $g_{w,d}$. This constant is calculated in data handling, where the required amount of skilled work is subtracted from the total workload forecast for each week.

Minimum vacation weeks α_e is a constant that is used to refer to how many weeks each employee can have their vacation. In the case example, α_e is at most three and at lowest the number of weeks that each employee has in the input data V_e. Finally, the last constant in our model is big M, which is a widely used method in linear programming, which makes the simplex algorithm solve the problem more efficiently. In the project, the value of 1000 is used for it.

3.2 Decision variables

The objective function obtains different values by changing the values of decision variables. In this project, there are six decision variables in total. First decision variable is $v_{e,w}$ which denotes if employee e is on vacation on week w, which is binary variable:

$$v_{e,w} = \begin{cases} 1, & \text{if employee } e \text{ is on vacation on week } w \\ 0, & \text{otherwise} \end{cases}$$

The number of working hours for each skill is measured on a weekly basis for each employee. In other words, this variable $y_{e,w,s} \geq 0$ describes the working hours for each employee e on each skill s on each week w. Naturally, if an employee does not have a certain skill level, the variable will get a value of zero for that employee with that skill for all weeks. On the other hand, also the amount of working hours for general work for each employee e on each week w is monitored with variable $u_{e,w} \geq 0$. General skills are work that does not require any skill from the employee.

It is important to plan the beginnings of long vacations so that rotating vacations can be successful, so the variable $z_{e,w}$ is modeled as a binary variable if

employee e starts the long vacation on week w, i.e.,

$$z_{e,w} = \begin{cases} 1, \text{ if employee } e \text{ starts the long vacation on week } w \\ 0, \text{ otherwise} \end{cases}$$

The last two decision variables are D and $o_{e,w}$. The largest week deficit of required general work hours, that is general hours that are unfulfilled, is defined in variable D. Unfulfilled general hours are calculated in such a way that the required hours of skilled work are subtracted from the weekly workload forecast and finally, the hours of general work, which are covered by the employees on site in total, are subtracted. The other variable is $o_{e,w} \geq 0$ which describes how many overtime hours there are for each employee e for each week w. Overtime working hours can be accumulated for employees who have skills that are in short supply. In this case, overtime hours ensure that the unit always has sufficient expertise on site.

3.3 Constraints

In this subsection all 16 constraints are explained and some examples are given to make the equations easier to understand.

$$\sum_{w \in W} v_{e,w} = V_e, \quad \forall e \in E. \tag{1}$$

The sum of vacation weeks per employee must be equal to the vacation allowance per employee. If an employee should have three weeks of vacation in total during the vacation period, then three weeks off is allocated to them.

$$\sum_{w \in W} z_{e,w} * Z_{e,w} = 1, \quad \forall e \in E. \tag{2}$$

Out of all the possible vacation starts, only one start is selected for each employee. The sum of the product of the allowed start of the long vacation and the actual start of the long vacation needs to be one for all employees. This basically ensures that each employee can start their long vacation only once. For example, if an employee has allowed starts on weeks 20, 26, and 27 and starts the long vacation on week 26, then the constraint is not violated.

$$r_s = \sum_{e \in E} y_{e,w,s} * x_{e,s}, \quad \forall w \in W, \forall s \in S. \tag{3}$$

Specialized work must be fulfilled. A sum of all employee's skilled work hours multiplied by the employee's skill base must be equal to the required hours per skill. r_s is the same for all weeks. If the employee does not have a specific skill then $x_{e,s} = 0$, and if an employee does not work at all for a specific skill for a

specific week, then $y_{e,w,s} = 0$.

$$\sum_{s \in S} y_{e,w,s} + u_{e,w} = H_e * (1 - v_{e,w}) + o_{e,w}, \quad \forall e \in E, \forall w \in W. \tag{4}$$

The employee's working hours match the amount of work planned. All work hours done by an employee must be equal to the sum of the employee's contract hours and overtime hours. $(1 - v_{e,w})$ term ensures that no working hours are required from an employee who is on vacation. For example, if an employee has a 30-hour contract, works skilled work for a total of 12 hours, and general work for 22 hours in one week, then overtime work of 4 hours is allocated to them. Thus,

$$\sum_{w \in W} z_{e,w} = 1, \quad \forall e \in E. \tag{5}$$

The long vacation can be started exactly once. This constraint ensures that all employees can start their long vacation only once but also that every employee must start their long vacation at some point, i.e.,

$$z_{e,w} \leq Z_{e,w}, \quad \forall e \in E, \forall w \in W. \tag{6}$$

All possible starts of the long vacation must be included in constant $Z_{e,w}$, which is all the allowed starts for the long vacation. This constraint verifies that no employee can start their long vacation on a week that is not possible for them. For example if possible startings are $Z_{e_1,22} = Z_{e_1,26} = Z_{e_1,27} = 1$ then $z_{e_1,23} = 1$ is not possible.

$$v_{e,w} = f_{e,w}, \quad \forall e \in E, \forall w \in W. \tag{7}$$

Employees must have a vacation on weeks that are manually marked as "fixed vacation" for the employee in the data. These fixed vacations can be weeks that the employer has granted requested by the employee. This constraint is also used if a user wants manually put someone on vacation on a specific week to adjust the optimized vacation model.

$$v_{e,w} = p_{e,w}, \quad \forall e \in E, \forall w \in W. \tag{8}$$

An employee is having a "vacation" if they are on sick leave, for example. This constraint works in a similar way as a constraint (7).

$$D \geq g_{w,d} - \sum_{e \in E} u_{e,w}, \quad \forall w \in W. \tag{9}$$

The deficit of general work must be equal to or lower than the sum of total general hours worked by the employees subtracted from estimated general work hours.

$$y_{e,w,s} \leq M * x_{e,s}, \quad \forall e \in E, \forall w \in W, \forall s \in S. \tag{10}$$

Amount of skilled working hours per employee needs to be lower or equal to a big number if a significant employee has significant skill.

$$o_{e,w} \leq 10, \quad \forall e \in E, \forall w \in W. \tag{11}$$

Overtime per employee can not exceed 10 hours per week. According to the collective agreement in the retail sector [1], additional work is up to 40 hours of work per week and overtime is work that exceeds this limit. The 10 hours of additional work is determined by the project team and is justified by maintaining the well-being of the employees.

$$\sum_{e \in E} o_{e,w} \leq 60, \quad \forall w \in W. \tag{12}$$

The sum of overtime for all employees must be equal to or lower than 60 hours per week.

$$o_{e,w} \leq M * v_{e,w}, \quad \forall e \in E, \forall w \in W. \tag{13}$$

The amount of overtime work for all employees and all weeks must be equal or lower than a big number if an employee is on vacation on that week. In other words, if an employee is not on vacation, overtime should be zero.

$$z_{e,w} \leq v_{e,(w+\alpha_e)}, \quad \forall e \in E, \forall w \in H_w. \tag{14}$$

An employee with three or more vacation weeks should have three vacation weeks in a row. If the employee has fewer vacation weeks than three they should have as many as possible in a row. The constraint uses the variable z that describes the start of the long vacation and ensures that the vacation structure, that is the weeks following the start of the long vacation are also considered vacation weeks.

$$1 - v_{e,(w+\alpha_e+b-1)} - z_{e,w} \geq 0, \quad \forall e \in E, \forall w \in W, \forall b \in B_e. \tag{15}$$

This constraint puts an artificial gap between the end of the long vacation and a possible one-week vacation. B_e is an employee-specific constant that specifies the length of the gap between the long and short vacation.

$$1 - v_{e,(w-b)} - z_{e,w} \geq 0, \quad \forall e \in E, \forall w \in W, \forall b = B_e. \tag{16}$$

This constraint does the same but ensures the same as constraint 15 except it guarantees that there are no vacations B_e weeks before the long vacation.

3.4 Objective function

The objective function is discussed in more detail in the next section, but it has the following form.

$$\min_{v_{e,w}} \quad D + M^2 \sum_{e \in E, w \in W} o_{e,w} \tag{17}$$

4 Results

4.1 Different objective functions

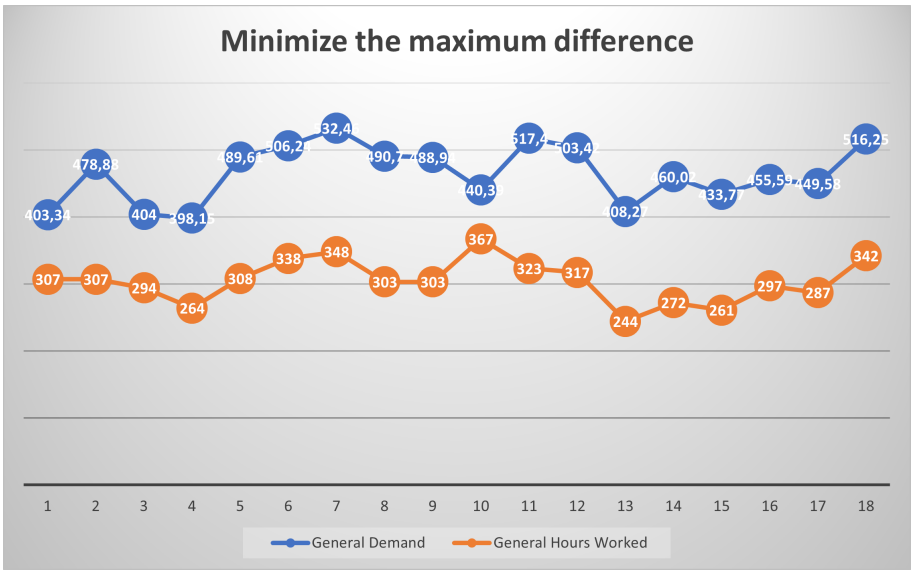


Figure 3: Demand in hours and actual hours worked when minimizing the maximum difference

As discussed in Chapter 3.4, we chose the objective function:

$$\text{min: } D + M * \sum_{e \in E, w \in W} o_{e,w}, \tag{18}$$

Here the goal is to minimize the peak, that is minimize the maximum difference in available employee hours compared to the needed number of employee work hours. We started by doing the inverse, that is maximizing the smallest difference, which gives the objective function.

$$\text{max: } D - M * \sum_{e \in E, w \in W} o_{e,w}, \tag{19}$$

As can be seen by comparing figures 3 and 4, both versions do create a distribution of available employees that follows the needed number of working hours.

However, we noticed that the original maximization idea had some properties we did not like. As it raised the minimum difference it tended to leave quite large peaks, which generated poor results. Here it can be seen in the form of the fourth week where the number of hours worked becomes too low at 209 compared to the demand. Thus we ended up switching to the minimizing version

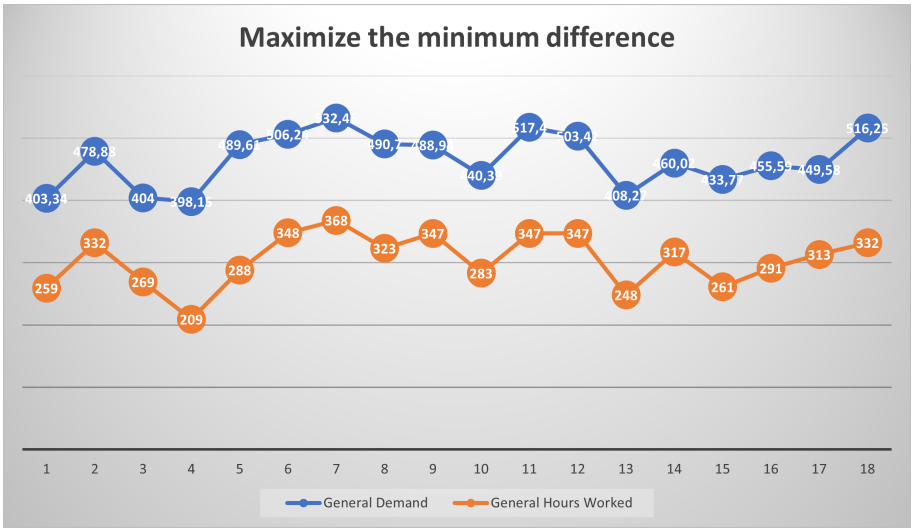


Figure 4: Demand in hours and actual hours worked when maximizing the minimum difference

seemed to better generate more even solutions, that is the difference in supply and demand was more consistent throughout the time period W. We do not compare the objective values as they do not measure the same property.

4.2 Balancing real-world needs and optimality

The results are heavily impacted by constraints we set to enforce fairness and to ensure we follow both the laws governing employee rights as well as all rules present in the union agreement. Furthermore, the employee's needs must be considered in the model to ensure employee retention.

The biggest impact of these new constraints is the one pertaining to when an employee can have their union-mandated long vacation as well as the one determining the gap between the three-week vacation and the one-week vacation. These constraints constrain the set of possible solutions a lot. This can be clearly seen in figure 5, where the weeks the long vacation can start on have been marked as green. This limits the optimality of the model quite a lot, that is it means the variance in the difference between hours worked and demand varies a lot between weeks. We further limit the starting time of the vacation to the weeks 18-33 as per the customer's request.

The constraint modeling the gap between the long and short vacation also has

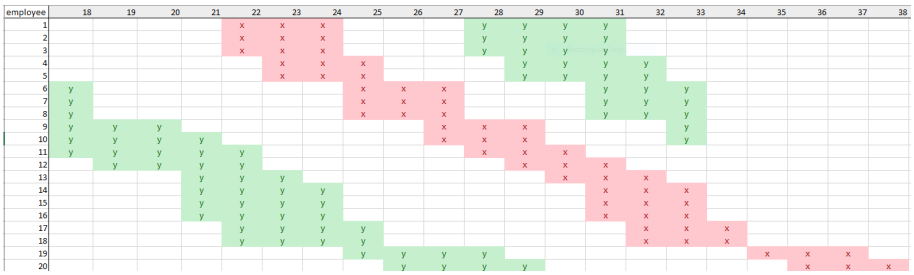


Figure 5: Each possible starting week of the long vacation is marked in green for each employee. Red marks the employees' long vacation last year.

a clear impact, as when solving without it the model manages to find a more optimal solution. However, the impact is a lot smaller than that of the aforementioned constraints regarding the longer vacation period, as at most one week is affected depending on the number of vacation weeks the employee has.

Finally, an addition to the model affords the user the possibility to manually set a certain week as "locked", that is no employee can be on vacation that week, and to manually set an employee as either on vacation or otherwise unavailable but not on vacation. The impact of these additional constraints is of course heavily dependent on the number of additional limitations set, and the number of employees affected. For example, locking a whole week, generally means the difference in hours worked and hours needed that week is a lot smaller than other weeks.

4.3 Julia code

The resulting product is a Julia [3] software application divided into three different parts, with the input and output handled through Excel. It is divided into three working parts, the data parser, the optimization model, and the output generator as well as a main file that works as a bridge between the three code parts. This was done in order to ease future development and increase readability as we considered it a natural way to divide the code into segments. Furthermore, a customer requirement was that all Julia-related software used to be open-source and free. All Julia packages used in this project are licensed under the MIT license, which means it is open source and, thus, can be used freely in commercial applications.

4.3.1 Parser and output generator

The Julia code that parses the input Excel is primarily based on the package XLSX.jl [15], which allows one to both read and write to and from Excel files. Thus same Julia package is also used to display the results in the output Excel. In addition to loading and displaying all relevant data, we carry out post-processing on the data transforming it into a suitable form for the optimization model itself. This includes transferring relevant information into binary variables, the loaded skillsets, and fixing issues rising from the additions of the aforementioned locked vacations and other absences. For example, if the whole summer is set as absence except for the first four weeks the model is not solvable unless we remove a few constraints.

We make the following changes to ensure such issues do not arise. In practice, we treat being absent from work in the same way as being on vacation, as the impact on the vacation plan does not change. We also remove the constraint on a minimum gap between vacations if more than one vacation or other absence is specified. For each specified vacation we lower the number of needed vacation weeks in a row. That is if we have four vacation weeks out of which two are locked, we only require a two-week long vacation. Finally, for each specified absence unrelated to vacations we add a vacation week to ensure the employee gets his full vacation in addition to the absences, if possible to fulfill within the given time period.

4.3.2 Optimization Model

The optimization model is implemented as described in Chapter 3 using the mathematical modeling language JuMP [13]. JuMP requires a separate solver to solve the modeled problem, here we use HiGHS solver [9]. The solver solves the problem using the branch and bound optimization method. We set the solver to use multi-threading to increase the solution speed, as it helps in the initial symmetry detection.

As the binary elements of the model, make it an MLP the solution time varies a lot depending specifically on the number of binary elements. The number of possible skills an employee can have means there is quite a lot of variance in the solution time, with models having either a large number of employees or possible skills being close to unsolvable. An acceptable solution to this was to take the best possible solution found by the model after a set amount of time. As the model does not have to be completely optimal this was considered acceptable by the client, as it gives a viable solution.

Therefore, the change discussed in Chapter 4.1 has an even bigger on the solution. The current implementation starts with searching for a way to decrease

the highest difference of hours required on a summer week and the schedule by the model the number of available hours. Thus, when the optimization run is terminated prematurely before the optimal solution is found, the model would be having the smallest found peak of not-fulfilled hours compared to the initial solution, which would ensure that the week with the smallest number of missing general working hours would be as even with other weeks as possible - at the cost of having possibly left a few weeks with extremely unbalanced uncovered demand.

Over the experiments with the real data of the testing SOK unit, the model converged to below 5% difference between the upper and lower bounds of the optimization model in under 60 seconds for every combination of skills and user-added constraints and settings. This means that the difference between the highest peak of the unfulfilled working hours of the presented vacation schedule and the theoretical best result is under 5%. Given that the demand forecasts contain a higher degree of uncertainty, it is reasonable to save computational resources and time while presenting a solution that satisfies all set constraints.

4.3.3 Structuring variables

While reading the data from Excel to the optimization model, the input information is grouped into three structs. A struct in Julia is a user-defined composite type that allows grouping related data together. It provides a way to define custom data structures with their own fields and functions, allowing for efficient memory layout and performance optimizations.

Utilizing a struct allows grouping different variables into one container, which is named to reflect its contents. Then, the parser function returns a handful of containers to the main function and these containers can be conveniently passed on to the optimization. In the optimization part, the required variables are read directly from the structs. The grouping is influenced by the layout of the data in the input Excel.

The following three containers were utilized to read the data from Excel. Fields of each container are present in the respective bullet lists along with their mathematical notation from Chapter 3.1 or other explanations where applicable.

- Employee data:
  - Employees, e
  - Employee Skills, $x_{e,s}$
  - Department, a list of departments

  - Department Skills, a list of all skills and levels present per department
  - Contract Hours, V_e
  - Vacation Weeks, H_e
  - Worker Name
  - Minimum Vacation Weeks, a variable that limits the length of the long vacation per employee. If the employee has four weeks of vacation, he should take three weeks in a row as the "long vacation period" as discussed earlier. If the employee has fewer vacation weeks, he needs to take all his vacation in one continuous period.

- Last year vacations:
  - Start of last vacation, week number, when the employee had his long vacation last year
  - First year at the firm, a boolean variable to check whether the employee had any vacation last year
  - On leave, $p_{e,w}$
  - Vacation wish, $f_{e,w}$
  - Allowed starts, $Z_{e,w}$

- Other data:
  - Skill, s
  - Week, w
  - Department, a dictionary of which departments each skill belongs in specialized work requirements
  - Required hours per week, $r_{w,s}$
  - Required general hours per week, g_w
  - No vacation weeks, a list of weeks, during which no vacations are allowed

4.4 Quality of results

Figure 6 shows the generated vacation plan. One can clearly see how large of an impact last year's long vacations have by comparing it to Figure 2. The figure manages to find a solution better than simply pushing all vacations forward by four weeks, as can be seen by the broken pattern. Now looking at Figure 7, the ideal scenario would of course be a straight line. However, if we ignore the three dips where the difference is clearly smaller than the other weeks, It is quite close to a line as the trendline shows. This means the result is at least somewhat close to an ideal solution. A positive regarding the larger outliers being small values is the fact that in order to cover the deficit stores generally hire summer workers. Thus, one can hire in accordance with the trendline or largest peak and simply be slightly overstaffed on weeks 19,27, and 31, in comparison to being understaffed if the situation was the opposite.

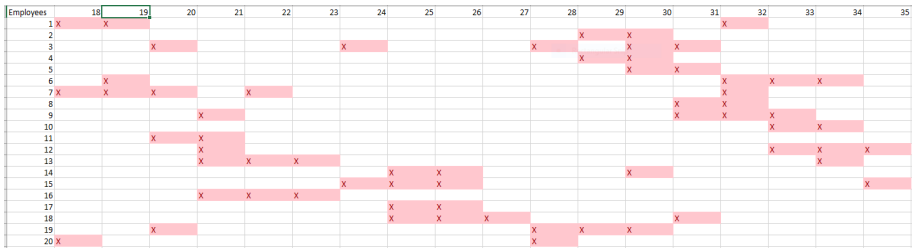


Figure 6: The generated vacation plan.

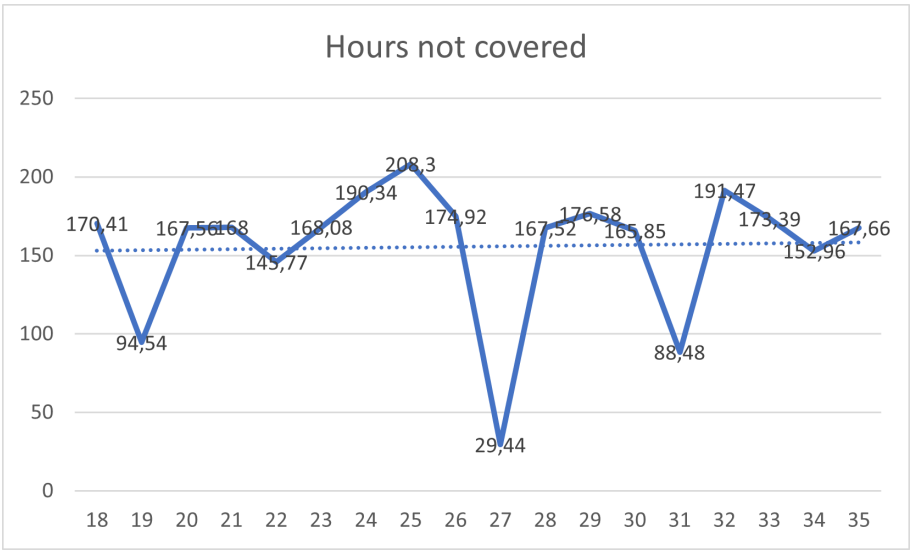


Figure 7: Graph displaying the number of hours not covered each week.

4.5 Output UI

4.5.1 Dashboard

As discussed earlier, the output user interface is implemented using Excel to flatten the learning curve and allow the end users to focus on analyzing the results and not on learning how to use a new interface. The overview of the resulting dashboard is present in Figure 8. The output can be discussed in the following sections:

- vacation schedule, in the top-left corner with vacation weeks colored in red
- employee's skills, in the top-right corner with each present skill colored in green

- resulting statistics, below the vacation schedule
- skills of employees at work and on vacations, below the statistics
- hours worked per required specialized work skills, in the bottom-left in gray
- charts, in the bottom-right

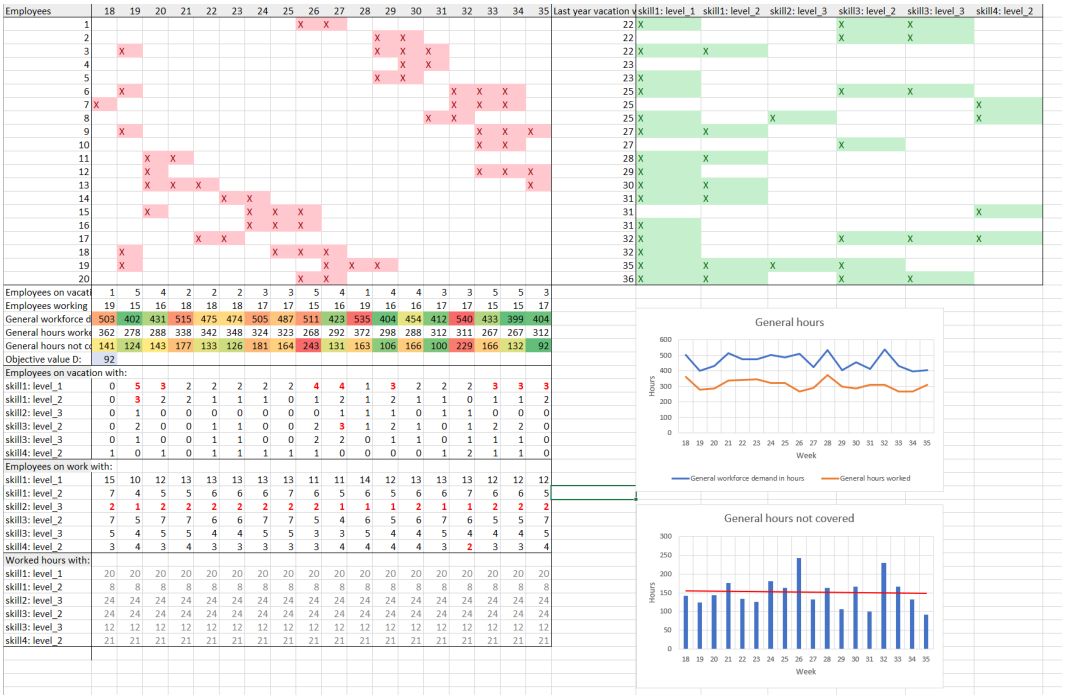
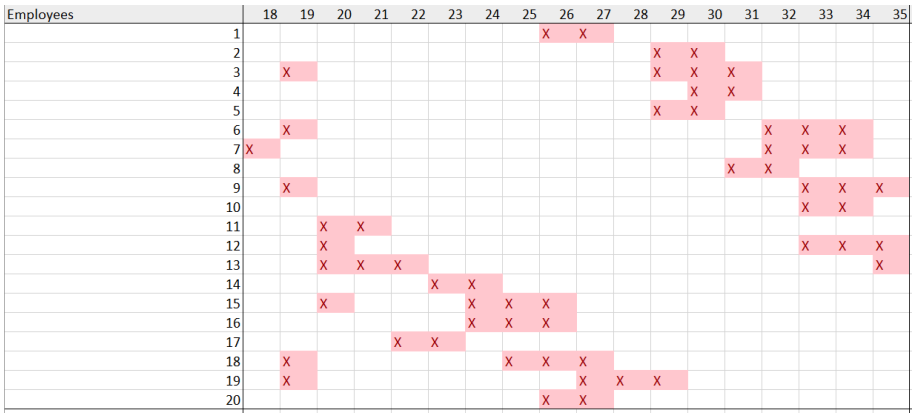


Figure 8: Excel output dashboard

This dashboard is created in Excel separately for every department of the company. This allows to keep the data neatly together without overwhelming the user with the number of different skill and level combinations. Moreover, printing each department's results separately fits well with the concept of running the optimization for each department iteratively.

All the data presented is combined from several optimization output variables into one matrix, which is written to Excel using the XLSX library. The library does not support, however, the formatting. Therefore, an Excel file only with the data is generated after the code finishes running. The formatting is added later by the user using the conditional formatting function in MS Excel.

The vacation schedule is a table of employee numbers (later - names) in rows and weeks used in the optimization - in columns. The weeks, on which the employee is on vacation, are marked with X. This data comes from the decision variable $v_{e,w}$ of the optimization model. The last column features the week number when the employee started long vacations last year. This information is used to ensure that the current optimization schedule aligns with the rule that the long vacation is moved by a predefined interval every year. Moreover, it helps to make manual adjustments to the plan while adhering to the aforementioned rule. A closer look at the anonymized output can be explored from Figure 9 below.



Employees	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1									X	X								
2												X	X					
3		X										X	X	X				
4													X	X				
5												X	X					
6		X													X	X	X	
7	X														X	X	X	
8														X	X			
9		X														X	X	X
10																X	X	
11			X	X														
12			X													X	X	X
13			X	X	X													X
14						X	X											
15			X				X	X	X									
16							X	X	X									
17					X	X												
18		X						X	X	X								
19		X								X	X	X						
20									X	X								

Figure 9: Excel output: vacation schedule

Next, on the right from the vacation schedule, the skills of each employee are marked with X. If the employee has level 2 of a skill, then he also is seen as having level 1 of this skill. This allows to account that experienced employees are able to do less specialized work if needed and the same concept is carried over to the output file. This information helps the end user to make manual adjustments to the optimized vacation schedule. For this, it is mandatory to know the skills of each employee to ensure that the required specialized hours are always covered after the required changes. A closer look is available from Figure 10.

Furthermore, the statistics feature on a weekly basis: how many employees are on vacation and how many are working, how many there are forecasted general working hours in hours (coming from the input), how many general work hours the department is scheduled by the program to output and the difference between these two numbers for each week. Adding conditional formatting for the required workforce demand and unfulfilled general working hours allows one to notice easily, which weeks could turn out to be problematic. Lastly, the objective value is printed out, which is the same as the highest difference between the required and worked hours.

skill1: level_1	skill1: level_2	skill2: level_3	skill3: level_2	skill3: level_3	skill4: level_2
X			X	X	
			X	X	
X	X				
X					
X			X	X	
					X
X		X			X
X	X				
			X		
X	X				
X					
X	X				
X	X				
					X
X					
X			X	X	X
X					
X	X	X	X		
X	X		X	X	

Figure 10: Excel output: skills

Climbing down the Excel sheet, the user is presented with the number of employees that possess each skill and who are either on vacation or at work. First, the number of "skills on vacation" is printed, followed by the number of "skills at work". Naturally, a single employee possessing several skills would be present in several of these numbers and together they reflect how many employees have a certain skill. Again, this extra information allows the end user to make manual changes and adjust the schedules while maintaining the required specialized work hours. Moreover, it is important for the end user to know that each week there will be a certain minimum number of workers with a certain skill. In case of unexpected absence or a peak in demand, there would be a second worker to cover the need.

Finally, in gray there is a number of hours that the department is working on the required specialized skills. The numbers are printed from the model output variable $y_{e,w,s}$ and are the same as the input variable r_s. This information complements the previously described information on the general working hours to create a holistic picture of available and scheduled working hours of the department.

The last, manually added, piece of the user dashboard is the two charts that visualize the required and worked hours. The top chart shows the trend per each summer week with the forecasted workload in blue and planned by the model working capacity in orange. It is easy to notice that the model strives to copy the peaks in demand by having fewer workers on vacation on these weeks. The

lower chart focuses on the difference between these two numbers and illustrates the uncovered general work hours. The format of the chart was chosen to be a bar chart to make it easier to compare different weeks between each other. Furthermore, the red line shows the average unfulfilled number of hours over the whole summer period. The ideal result would be that every week would be equal to this average and, thus, a fixed number of extra summer employees would perfectly cover the demand and vacations. The charts are provided in Figure 11.

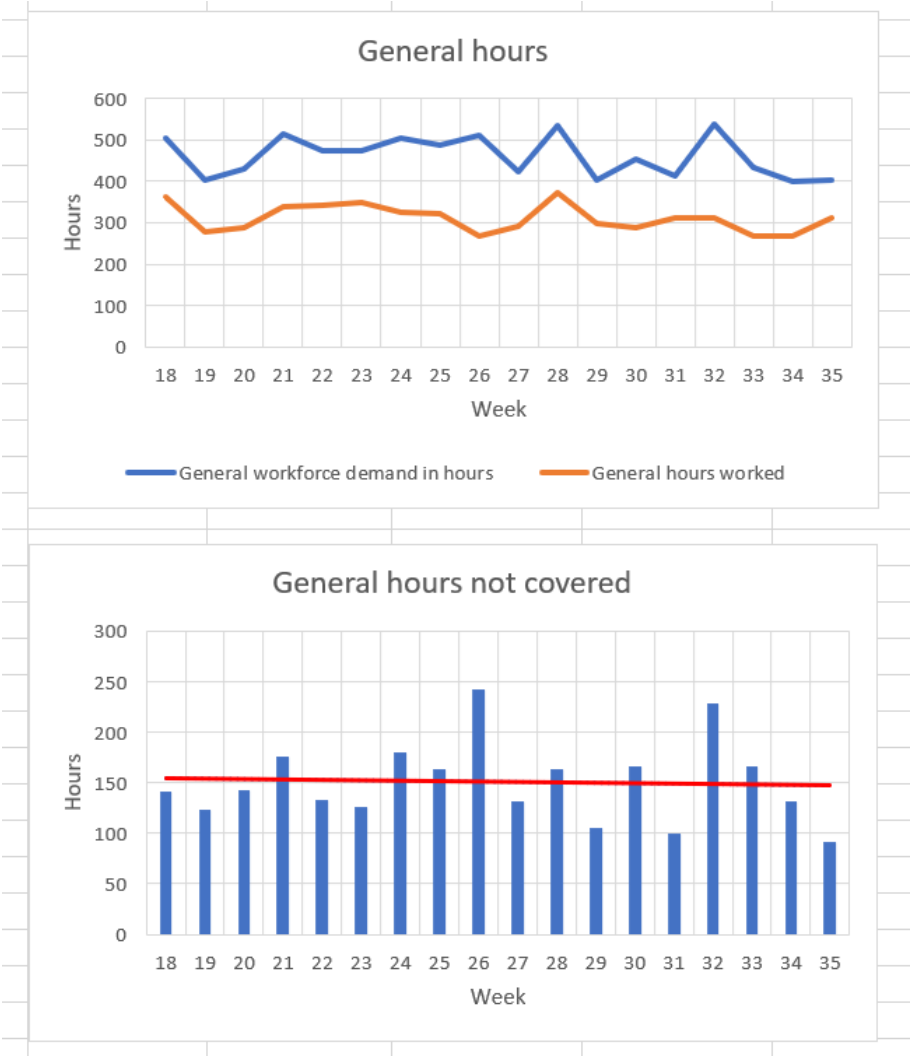
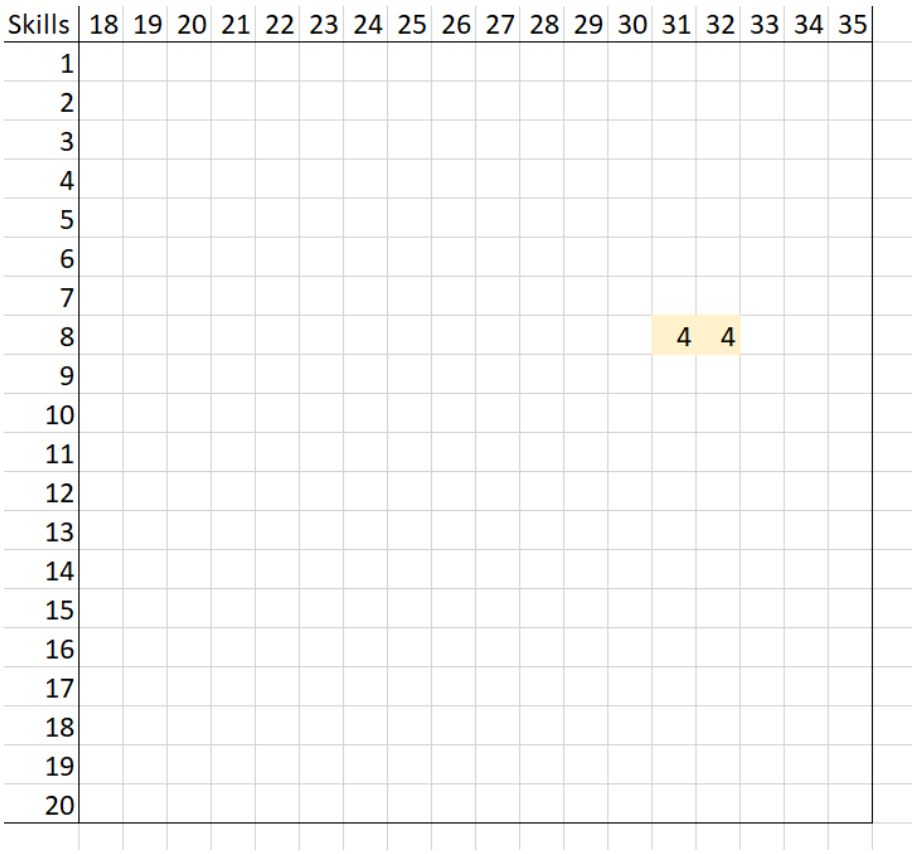


Figure 11: Excel output: charts

4.5.2 Overtime

For each department, a second sheet is printed. This worksheet contains the number of overtime hours required from each employee to ensure that all constraints are satisfied. It is possible and was seen in the testing data, that some departments may not have enough skilled workers to cover the required specialized work hours, which are included as a hard constraint. Therefore, the planning tool is allowed to set some workers on overtime but at a considerably high cost as described earlier. Thus, in reality, overtime is scheduled only in mandatory cases. A snapshot of the overtime sheet of the user interface is in Figure 12



Skills	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1																		
2																		
3																		
4																		
5																		
6																		
7																		
8														4	4			
9																		
10																		
11																		
12																		
13																		
14																		
15																		
16																		
17																		
18																		
19																		
20																		

Figure 12: Excel output: overtime

5 Discussion

The goal of the project was to create an optimization tool for SOK to reduce planning costs and improve the quality of vacation planning by ensuring that given constraints are always met. This goal was met with positive feedback both from the project's enablers at SOK and end users at the Prisma unit.

The created tool is by far not complete. It can be improved in a variety of ways that were discussed during the implementation of the project. For example, by creating a possibility to plan a long vacation in the length of two weeks in a row instead of limiting a long vacation to three weeks only. Further research can be implemented to analyze suitable ways to ensure that employees get vacation weeks that are perceived as advantageous, like having a vacation in July instead of September. Discussed options were to rotate vacations in a fixed frame where the employee who is next on the list always would have a vacation after the employees above him. Moreover, adjusting the parameters is important to ensure reasonable results.

Furthermore, additional constraints can be added to account for law requirements, like that vacations cannot start on public holidays, or consider employees who have less than a full week of vacation. Currently, vacation weeks are calculated by rounding up any starting weeks of vacation allowance, i.e. 10 vacation days would be 2 weeks in the model.

Additionally, the tool can be modified to reflect the different processes and requirements of different business entities and units collaborating with SOK. The current model is tailored to the requirements of the testing unit and the realities of their business. However, the model was built with space for generalization and further development.

On the other side, as discussed in the literature review, there are just a handful of research papers that focus on the same applications of optimization. Therefore, the solution of this project to create a MILP program that could effectively create an output of acceptable quality in under a minute per optimization run has used several unusual ways to model the objective and constraints.

6 Conclusions

This report summarizes four months of work done on developing a vacation optimization tool for SOK. The project started with gathering information on the vacation laws and regulations, and requirements of SOK and the Prisma unit and stemmed from scientific research. Next, a theoretical optimization model was developed and confirmed with SOK. The biggest part of the project consisted of implementing the model in Julia and creating a parser that would reliably read the information from the Excel file to Julia.

The collaboration with all parties has thrived throughout the project. SOK representatives, Antti Punkka and Teemu Kinnunen, have offered invaluable help in ideating on how to solve the challenges that have arisen during the project and facilitated excellent communication with the end users to ensure that the tool is designed in the required way. We had exactly enough online meetings with the end users to go through the ways how they operate and what tool would help them in vacation planning. Moreover, from the Aalto side, Ahti Salo and Jerry Aunula offered valuable feedback on the project deliverables and presentations, in addition to organizing the opportunity to work on this course.

Overall, the objectives of the project were achieved. The output vacation plans were checked to satisfy the constraints and, thus, should be valid. Hopefully, the code will be further developed by SOK and put into production to create saving for the next year's summer vacation planning.

A Self Assessment

A.1 How closely did the actual implementation of the project follow the initial project plan?

The goal of this project was to create a vacation optimization tool for a single Prisma store unit, so that it could be scaled to multiple S Group establishments. The tool should be easy to use and produce better results than with hand and faster. The schedules that our tool produces fulfil all the constraints and the optimization process takes only seconds. How easy-to-use the tool finally was, is to be seen, because no real life testing in the Prisma unit was excecuted during the project. However, feedback from vacation planners of the Prisma seemed like they would be able to use it with good-enough guiding.

Tasks in our project plan were mostly executed by the original schedule. However, some of the suggestions by the customers could not be implemented into the final model, because the suggestions for improvement were suggested in the final presentation for the customer, so there was no time. Nevertheless, the tool was a success and after the project the S Group will be continuing working on the tool.

A.2 In what ways was this project successful?

The primary objective of developing an optimized vacation planning tool was successfully achieved. The final product may be considered a minimum viable version, but it is a solution that can be effectively utilized by those experienced with the methods and coding language.

Communication with the client played a crucial role in the success of the project. From the outset, we established a clear and open line of communication with the client, which allowed us to better understand their needs and expectations. The client was always responsive and proactive in providing feedback and input, which helped us to refine the model and ensure that it aligned with their goals.

Throughout the project, we maintained regular communication with the client to keep them informed of our progress and any issues that arose. We also provided regular updates on the model's performance and sought feedback on how it could be improved. The client was consistently engaged and showed a genuine interest in the success of the project, which made working with them a pleasure.

Finally the teamwork has been great. Every team member has aided the project with their unique expertise and perspectives.

A.3 In what ways was this project unsuccessful?

The project had a few things that could have been improved upon. The perhaps biggest issue with the end product is the fact that the excel is only updated numerically. All formatting has to be done by hand. This drawback of the product is mostly due to time-constraints as well as the general difficulty of doing so through the chosen Julia package. This is in the end however a rather minor issue as even a inexperienced excel user can quite easily add the cosmetic changes.

Another minor issue that came up such is the case by case handling of manually inserted absences and vacations being handled by removing constraints instead of modifying the constraints heavily. This however is quite a rare use case and should not have a major impact on the final product.

Finally some small details such as the code being quite rigid and impossible to modify without touching the source code, concerning for example the design of the input excel, and still running rather slowly and likely possible to optimize further remained unfixed.

A.4 What could have been done better?

There are a few things we could have done differently in hindsight. The schedule of the project plan could have been even more precise or the schedule could have been followed even better, because the course, so to speak, ended too early compared to what we reached in our project. By estimation we finished about 90 % of all the work we had planned. We could have improved our model and added new functionalities, but we had to limit them because of the course schedule. However, this was not a completely critical problem, as SOK was satisfied with the results we achieved, and plan to develop the solution even further. So, as agreed at the beginning of the course, we got a good starting point for creating optimal schedule plan for SOK.

We could also have started the coding part earlier, because learning Julia context took some time, and also in the latter half of the course, the team members were more busy with other courses as well. At the beginning of the course, getting data from the client also served as a bottleneck for a couple weeks, because at first we weren't sure what all the information we needed to complete the project, and it also took time to understand and clean the data.

References

[1] Palvelualojen ammattiliitto. "Key provisions of the collective agreement in the retail sector". In: (2022). URL: https://www.lukusali.fi/pam/#/reader/703f7638-d290-11ec-b1f0-00155d64030a (visited on 02/24/2023).

[2] C. S. Azmat, T. Hürlimann, and M. Widmer. "Mixed Integer Programming to Schedule a Single-Shift Workforce under Annualized Hours". In: *Annals of Operations Research* 128.1 (2004), pp. 199–215.

[3] J. Bezanson et al. "Julia: A fresh approach to numerical computing". In: *SIAM review* 59.1 (2017), pp. 65–98. URL: https://doi.org/10.1137/141000671.

[4] R. Van den Broek, H. Hoogeveen, and M. Van den Akker. "Personnel Scheduling on Railway Yards". In: *20th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*. Vol. 85. 2020.

[5] P. S. Chong and M. W. Strevell. "A vacation scheduling algorithm for military flight crews: Maximizing satisfaction while maintaining military preparedness". In: *Journal of Operations Management* 5.2 (1985), pp. 205–211.

[6] S. Dewess. "Socially acceptable annual holiday planning for the crew of a local public transport company in Germany". In: *Public Transport* 2.1 (2010), pp. 25–49.

[7] M. Elshafei and H. K. Alfares. "A dynamic programming algorithm for days-off scheduling with sequence dependent labor costs". In: *Journal of Scheduling* 11.2 (2008), pp. 85–93.

[8] A. T. Ernst et al. "An Annotated Bibliography of Personnel Scheduling and Rostering". In: *Annals of Operations Research* 127.1 (2004), pp. 21–144.

[9] Q. Huangfu and J. A. J. Hall. "Parallelizing the dual revised simplex method". In: *SIAM review* 10.1 (2018), pp. 1867–2957. URL: https://doi.org/10.1007/s12532-017-0130-5.

[10] T. Kinnunen. *Cost-efficient vacation planning with variable workforce demand and manpower*. Espoo, 2016. URL: https://sal.aalto.fi/publications/pdf-files/tkin16_public.pdf.

[11] A. Klinkert. "Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT'08)". In: 2008.

[12] H. N. Koutsopoulos and N. H.M. Wilson. "Operator workforce planning in the transit industry". In: *Transportation Research Part A: General* 21.2 (1987). Special Issue Managing transportation, pp. 127–138.

[13] M. Lubin et al. "JuMP 1.0: Recent improvements to a modeling language for mathematical optimization". In: *Mathematical Programming Computation* (2023). In press.

[14] J. G. Morris and M. J. Showalter. “Simple Approaches to Shift, Days-Off and Tour Scheduling Problems”. In: *Management Science* (1983), pp. 942–950.

[15] F. Noronha. *XLSX*. URL: `https://felipenoris.github.io/XLSX.jl/stable/?ref=morioh.com&utm_source=morioh.com`.

[16] J. Van den Bergh et al. “Personnel scheduling: A literature review”. In: *European Journal of Operational Research* 226.3 (2013), pp. 367–385.