A sensitivity analysis approach to improve workforce optimization results by simulating employee training

Emil Nyman

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 31.12.2021

Supervisor

Prof. Antti Punkka

Advisors

D.Sc.Comp. Paul Saikko

M.Sc.Comp. Laur Pulliainen

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Emil Nyman

Title A sensitivity analysis approach to improve workforce optimization results by simulating employee training

Degree programme Mathematics and Operations Research

Major Systems and Operations Research **Code of major** SCI3055

Supervisor Prof. Antti Punkka

Advisors D.Sc.Comp. Paul Saikko, M.Sc.Comp. Laur Pulliainen

Date 31.12.2021 **Number of pages** 41+8 **Language** English

Abstract

Efficient scheduling in retail operations is achieved through workforce optimization which is essential in managing the large expense of personnel costs. In addition to regulations and contractual terms, the employee skills determining which tasks an employee is able to complete play a significant role in the scheduling process. An optimal workforce schedule ensures the personnel is fully utilized while complying with rules and regulations. However, a schedule is rarely optimal due to the complex restrictions leading to employee idling.

This thesis proposes a simulation based sensitivity analysis algorithm to improve the results of an existing workforce optimization solution. The algorithm recommends new skills to be trained to employees in attempt to lower the idle hours of employees by enabling them to work on additional tasks. Missing employee-skill recommendations are determined with a Multiple One-At-a-Time sensitivity analysis method which uses Latin Hypercube Sampling to explore the space of possibilities. Different scenarios representing stores or store departments are used to test the algorithm.

Improvements in test scenarios are measured by the amount of idle hours that can be decreased. Results show a large variation between both the amount that can be improved and the run times of the algorithm. Some test cases decrease idle hours close to zero while others can not be improved. In addition, the differences between the fastest and slowest execution time is almost thousandfold. Based on the results and the initial data of test cases, observations are made to help identify whether there is a possibility for improvement or not although definitive conclusions can not be made. The thesis is concluded with suggestions on how the algorithm could be improved.

Keywords Workforce, skills, optimization, simulation, sensitivity analysis

Aalto University
School of Science

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Emil Nyman

Työn nimi Herkkyysanalyysimenetelmä työvoimaoptimoinnin tulosten parantamiseksi simuloimalla työntekijöiden koulutusta

Koulutusohjelma Matematiikka ja operaatiotutkimus

Pääaine Systeemi- ja operaatiotutkimus **Pääaineen koodi** SCI3055

Työn valvoja Prof. Antti Punkka

Työn ohjaaja FT Paul Saikko, FM Laur Pulliainen

Päivämäärä 31.12.2021 **Sivumäärä** 41+8 **Kieli** Englanti

Tiivistelmä

Tehokas aikataulutus vähittäiskaupan alalla saavutetaan työvoiman optimoinnilla, joka on olennaisen tärkeää suurten henkilöstökustannusten hallinnassa. Säädösten ja sopimusehtojen lisäksi työntekijöiden taidot, jotka määrittävät, mitä tehtäviä työntekijä pystyy suorittamaan, ovat merkittävässä asemassa aikataulutusprosessissa. Optimaalinen työvoiman aikataulutus varmistaa sen, että henkilöstöä hyödynnetään täysimääräisesti noudatetaan sääntöjä ja määräyksiä. Aikataulut ovat kuitenkin harvoin optimaalisia, koska monimutkaiset rajoitukset johtavat työntekijöiden joutilaana toimimiseen.

Työssä ehdotetaan simulointiin perustuvaa herkkyysanalyysialgoritmia, jolla parannetaan olemassa olevan työvoiman optimointiratkaisun tuloksia. Algoritmi suosittelee työntekijöille koulutettavia uusia taitoja, joilla pyritään vähentämään työntekijöiden joutilaanaolemisaikaa heidän pystyessä työskentelemään useamman työtehtävien parissa. Puuttuvia taitoja suositellaan moninkertaisesti toistamalla yksi kerrallaan -herkkyysanalyysimenetelmää, jossa hyödynnetään latinalaista hyperkuutiopoimintametodia otosten keräämiseen syöteavaruudesta. Algoritmin testaamiseen käytetään todellisia myymälöitä tai myymäläosastoja edustavia skenaarioita.

Testiskenaarioiden tuloksien parannusta mitataan sillä, kuinka paljon joutilaanaolotunteja voidaan vähentää. Tulokset osoittavat suurta vaihtelua sekä parannettavissa olevien joutilaanaolotuntien määrässä että algoritmin suoritusajoissa. Joissakin testitapauksissa joutilaanaolotunnit vähenevät lähes nollaa, kun taas toisissa tapauksissa niitä ei voida lainkaan vähentä. Lisäksi, erot nopeimman ja hitaimman suoritusajan välillä ovat lähes tuhatkertaiset. Tulosten ja testitapausten lähtötietojen perusteella havainnoidaan tunnistettavia ominaisuuksia tapauksista, joissa on parannusmahdollisuuksia, vaikka lopullisia johtopäätöksiä niistä ei pystytäkään tekemään. Työn lopussa esitetään algoritmin parannusehdotuksia.

Avainsanat Työvoima, taidot, optimointi, simulointi, herkkyysanalyysi

Contents

Abstract ii

Abstract (in Finnish) iii

Contents iv

Glossary v

1 Introduction 1

2 Background 3
2.1 Workforce scheduling principles . . . 3
2.2 Retail workforce scheduling . . . 4
2.2.1 Challenges . . . 5
2.2.2 Workforce schedule optimization . . . 6
2.3 Experimental design in simulation . . . 7
2.3.1 Sensitivity analysis . . . 8
2.3.2 Latin Hypercube Sampling . . . 10
2.4 Literature review . . . 12
2.4.1 Sensitivity analysis in scheduling problems . . . 12
2.4.2 Skills in scheduling problems . . . 14

3 Research material and methods 16
3.1 Underlying workforce optimization solution . . . 16
3.2 Test data . . . 17
3.3 Sampling of new skills . . . 18
3.4 Key Performance Indicators . . . 20
3.5 Multiple One-At-a-Time algorithm . . . 20

4 Results 25
4.1 Execution of the MOAT algorithm . . . 25
4.2 Comparison of MOAT stopping criteria . . . 26
4.3 Optimized addition of skills . . . 28
4.4 Decreases in idle values . . . 29

5 Discussion 32

6 Conclusions 36

A Appendix 42

Glossary

workforce	human resources available in an organization to complete working tasks
workload	the amount of work required to be completed
shift	a period of time when an employee is assigned to their work task
workforce management	process meant to maximize the competency and performance of an organization
workforce planning	process meant to ensure that an organization has the proper amount of working capital available at any given time
workforce scheduling	process of defining working hours for the available workforce to meet the current and future workload
workforce optimization	optimizing workforce schedules
footfall	the number of customers arriving to the store at a given time

1 Introduction

In retail, a large and multi-skilled personnel is needed to keep the daily business running steadily. Varying numbers of cashiers, shelving workers and salespersons are needed at different times of the day depending on a multitude of factors, such as large incoming deliveries, peak hour footfall or customer attracting events. This essential group of people accounts for up to 15 % of a company's turnover in labor costs (Skorupa et al., 2019), which in combination with the increasingly competitive retail environment (Hodson et al., 2012), highlights the need of using workforce planning as a strategical tool to reduce expenditures.

Store managers own the important task of balancing personnel costs with the workforce need and the available staff while fulfilling contractual terms, labour laws and employee preferences. Workforce management tools help retailers in creating shifts more efficiently while also providing visibility in most important key performance indicators. However, the use of aforementioned tools does not drastically decrease the need for manual work in shift planning. It is time consuming and difficult to achieve optimality in shifts so that all constraints are met and there is no over- or understaffing (Ernst et al., 2004). To overcome this challenge, workforce optimization is deployed.

Workforce optimization provides retailers with automated shift planning which accounts for the timing, location, type and the needed quantity of workforce. Three main components must be focused on to achieve optimal results: workload forecasting based on incoming deliveries, customer footfall and online orders; workload optimization where store specific tasks and staffing rules are considered; and shift optimization which is based on the available employees, their skills, regulations and the needed workforce (Halme, 2020b). Together these components are used to create an optimized shift schedule which complies with the given restrictions and preferences in the best way possible giving the store manager a good starting point in creating shifts.

Achieving optimal and feasible shifts with a workforce optimization model requires an amplitude of input information, such as contractually agreed working hours and limitations outlined by any collective labour agreements. In addition, each employee has a set of skills which determine the tasks they can be assigned to. This information is used to form and constrain an often complex optimization model in which the relationships between the outputs and the inputs are not always clear. However, it is known that training employees to multiple skills increase scheduling possibilities (Disselkamp, 2013) and help overcome short demand peaks in certain tasks (De Bruecker et al., 2015). This creates a desire for model users to perform manual what-if scenarios by slightly changing the input information in attempt to analyze consequences or find a more optimal solution.

This thesis develops a systematic simulation based sensitivity analysis approach to improve commercial workforce optimization solution by proposing training of addition skills for certain employees. The underlying solution is considered as a

black-box optimization tool which is evaluated with input files that are modified based on a Multiple One-At-a-Time algorithm. Results are analyzed to provide insight on what can be achieved with modifications in the skill sets and how they affect the optimality of the solution. The sensitivity analysis is performed on test data to achieve the most realistic results possible. Based on these analyses, model users are given suggestions on actions that are identified to make the solution more optimal.

The remainder of this thesis is structured as follows. Section 2 introduces the challenges of optimizing workforce schedules in the retail setting, concepts needed in developing the sensitivity analysis approach, and the existing literature on scheduling problems including sensitivity analysis and skills. In Section 3, the underlying workforce optimization solution is explained and the sensitivity analysis approach is developed. Section 4 presents the results from executing the developed approach on test scenarios which are further discussed in Section 5. The thesis is concluded in Chapter 6 with recommendations for future research directions.

2 Background

This section discusses the background of workforce scheduling, introduces experimental design methods in simulation and reviews the existing Literature on workforce scheduling problems.

2.1 Workforce scheduling principles

The topic of workforce scheduling was introduced in a toll booths and traffic delays study by Edie (1954) and the first influential approach on the subject was proposed by Holt et al. (1955) a year later. Optimization models and algorithms have since been developed, taking into account more complex problems. Kesavan and Mani (2015) report that in addition to manufacturing, application areas include: transportation systems, health care systems, civic services, call centers, financial services, hospitality and tourism, venue management and retail. However, there is no type of scheduling model that would be used in general for each application. They find that early research focused on determining workforce requirements for manufacturing environments in the context of a long time horizon aggregate planning. The planning horizon of up to 12 months considered employment, production and inventory levels in simplified manners to minimize overall costs.

In the last few decades, among the growth in popularity of IT systems in general, also workforce management (WFM) systems and tools have become increasingly popular (Hota and Ghosh, 2013). They have expanded from simple mechanical timekeeping solutions to cloud-based computer systems including extensive management tools. Disselkamp (2013) describes modern WFM systems as having six important principles which they follow: aligning workforce with organizational goals; controlling the dynamic and unpredictable nature of an organization; timing events happening in the organization; informing each stakeholder with meaningful and cultivated data; providing visibility across the whole organization; and executing the five preceding principles in an efficient matter. Thus, these systems cover many aspects of managing human resources and are essential in controlling workforce costs by providing, among others, automated workforce planning and scheduling.

Baker (1976) proposed that workforce schedules can be classified in three main groups. In the first group, days off scheduling problems in which working and resting days are assigned while employees must have a set amount of resting days during a week. In the second group, shift scheduling problems where daily shifts are created so that the needed capacity is fulfilled according to availability of the workforce. Start and end times as well as the lengths of the shifts are defined for each employee. The third group includes tour scheduling problems in which features of the first and second group problems are combined. The majority of more recent workforce scheduling problems fall into the third category in which long operating weeks and days create a need for both shift assignment as well as resting days appointment (Van den Bergh et al., 2013).

Scheduling problems classified as touring problems include appointing shifts and

resting days but, they can further be divided into smaller subproblems due to their complexity. Ernst et al. (2004) proposed an interesting approach in which the workforce scheduling process is divided into six modules:

1. Demand modelling which determines the amount of workforce needed for a planning period based on fluctuating customer footfall, mandatory daily tasks and the minimum shift requirements.
2. Days off scheduling where resting days are determined.
3. Shift scheduling which considers the total amount of workforce and tasks to be done to meet the demand while also accounting for possible breaks.
4. Constructing work schedules over a planning horizon which satisfies a set of constraints.
5. Assigning tasks to be completed during a shift to the lines of work.
6. Assigning employees to lines of work if not already done in tandem with the fourth module.

Other approaches to scheduling problems can be thought as modifications of the described module process. For example, Talarico and Duque (2015) approach the problem with three steps: demand planning, optimal workforce generation and shift assignment while Defraeye and Van Nieuwenhuyse (2016) add one step more: demand forecasting, workforce requirements, shift scheduling and assigning workforce to shifts.

Scheduling is viewed from different perspectives depending on which organizational department it is done for. When creating schedules, employees have their own preferences, desires and differences in productivity levels which they would take into account. On the other hand, human resource departments must comply to certain regulations and company wide rules while financial departments are strictly interested in staying in budget. On the contrary, operation managers want as much workforce scheduled to their operations as needed (Kesavan and Mani, 2015). The regulatory side of scheduling needs to be respected at all times but taking other perspectives into account is a balancing act. Fisher et al. (2020) view workforce planning and scheduling as a two-part hierarchy where first workforce is allocated and then the allocated amount is used as a base to create schedules. Nonetheless, every aspect has to be considered and employees have to be scheduled to each task based on their skills and qualifications as well as preferences and work time restrictions to achieve effective schedules.

2.2 Retail workforce scheduling

Large portions of the scientific literature concentrate on workforce problems in healthcare, transportation or manufacturing (Van den Bergh et al., 2013) and before Melachrinoudis and Olafsson (1992) presented a scheduling system for cashiers, the retail sector had received little attention on the subject. In the recent decades,

however, the increased attention on retail performance has increased the interest in workforce scheduling and optimization (Kesavan and Mani, 2015).

2.2.1 Challenges

Workforce scheduling in retail is considered more challenging than other industries due to the constantly fluctuating workforce need which is one of the most significant factors retailers have to account for (Parisio and Jones, 2015). The fluctuation is created by different factors, such as rush hours demanding more cashier work and incoming deliveries requiring shelving work, which have to be handled to ensure smooth store operation. Thus, forecasting the needed workforce for each task is an important part of workforce scheduling. Typical tasks in retail include cashier work, shelving, clerical duties at a service counter or a department of the store and store opening or closing routines. They can be divided into fixed and volume-based tasks from which the latter constitutes of 80 % of the workforce (Halme, 2020a). Fixed tasks always need a similar amount of workforce, but the workforce need for volume-based tasks is harder to estimate as multiple factors need to be accounted for. In a retail setting where each customer generally leaves with a purchase, historical sales data can be used to create an estimate of future customer traffic. However, some retail sectors, such as clothing stores, see less purchases per customer and thus store traffic data can be used instead (Chuang et al., 2016; Olivares et al., 2020). Customer footfall forecasts and incoming deliveries need to be combined to create a common forecast for workforce need. This is done by using efficiency multipliers which indicate the amount of delivery boxes that can be shelved or the number of customers that a cashier can handle in a certain time period.

De Bruecker et al. (2015) point out that the fluctuation in workforce need should be handled without incurring extra costs and thus implementing carefully planned personnel schedules are needed. They note that to avoid the fluctuations and uncertainty of workforce demand, many retailers hire seasonal and part-time workers. Seasonal workers are employed to work full-time for some weeks during high seasons, such as Christmas, whereas part-time workers are employed year-round with a variable amount of working hours per day and week. Fisher et al. conclude that these two types of employees bring retailers with benefits of having lower workforce costs than full time employees and bringing workforce flexibility due to their more adjustable working times.

In addition to workforce flexibility from the employer viewpoint, Parisio and Jones (2015) note that flexible working hours and contracts enable employee wishes to be granted more easily. A healthy mixture of stability and flexibility has to be achieved to strike a balance in efficiency and benefit both parties. Too much flexibility increases the scheduling problem complexity which means that finding a feasible solution becomes harder (Van den Bergh et al., 2013). In addition, as benefits of increasing workforce at a store are uncertain and hard to measure while the additional costs are fixed, retailers tend to focus too much on simple cost measures while leaving the store understaffed (Fisher et al., 2020).

2.2.2 Workforce schedule optimization

In their literature review, Van den Bergh et al. (2013) find that the objective of a scheduling problem is commonly to minimize workforce cost rather than minimizing the actual number of employees. They report that minimizing costs gives a wider range of possibilities where each category or action, such as overtime or day of week wage, has a different cost. Van den Bergh et al. categorize most solutions as mathematical programming which include integer, linear, dynamic and goal programming as well as improvement heuristics. Less frequently used solutions include constraint programming, queuing and simulation.

The creation of a schedule is driven by constant as well as dynamic factors which can be categorized as either soft or hard constraints depending on the objective. Hard constraints must be satisfied by the solution while soft constraints only incur a penalty to the objective value and thus, they can be violated if a better solution can be found. Jones and Nolde (2013) note that solving the problem until mathematical optimality is generally not feasible nor necessary and in their scheduling problem solution, constraints are split into a hierarchy based on their criticality and used as a hard or soft constraint respectively. This method is used to allow the creation of a feasible schedule to be calculated while no schedule would strictly meet all desired restrictions.

In their literature review, Van den Bergh et al. (2013) observe similar mixing of hard and soft constraints as Jones and Nolde (2013) and in addition, they highlight the simultaneous usage of a single parameter as both a soft and a hard constraint. The most common hard constraint, used in 75 % of all research papers reviewed by Van den Bergh et al. (2013), is a coverage constraint which specifies that enough workforce has to be present during a shift. On the other hand, the coverage constraint is also one of the most popular soft constraints because the workforce should be minimized as well. They are commonly used together to achieve sufficient workforce at each time period while minimizing overstaffing. Furthermore, more advanced solutions incorporate lunch breaks and other short breaks into the coverage constraints to keep a constant workforce available.

In addition to simple workforce coverage constraints, also employee skills are commonly considered in scheduling problems (Van den Bergh et al., 2013). The definition of employee skills is the ability to perform a task in a satisfying manner and they can be divided into two classes: hierarchical and categorical skills. Hierarchical skills are those where an employee with a low skill level completes less work in the same time that a very skilled employee does. A higher skilled employee has more experience and has received more training on the subject. A lower skilled employee can usually substitute a higher skilled one with the cost of lower efficiency. Within categorical skills, the difference in skill level is disregarded and a task can be completed as long as an employee has received a sufficient training. In essence, in categorical skills there are no better or worse performing employees but rather employees who can complete a task or who can not. (De Bruecker et al., 2015).

In the literature, employee skills are viewed as fixed components that can not be modified due to training possibilities often being left out of scheduling models (Van den Bergh et al., 2013). Skills are either viewed as hard constraints in tasks that require a special skill or as a soft constraint to favor employees with a better skillset for the task. Disselkamp (2013) suggests that cross-training employees, meaning that an employee possesses multiple different categorical skills, and splitting the workload into smaller tasks increases the scheduling possibilities. De Bruecker et al. (2015) add that cross-training increases flexibility and thus short demand peaks can be overcome more easily. However, they mention that while cross-training improves employee productivity, efficiency is decreased due to more frequent task changing which interrupts the workflow. Fisher et al. (2019) find that cross-training a skill via online module increased sales by 1.8 %.

Government regulations and collective labor agreements such as maximum working hours during a certain time span and required resting times which must be complied with are commonly used as hard constraints. However, maximum working hours and resting times can simultaneously be used as soft constraints when a certain level is desired while regulatory constraints function as definitive limits (Van den Bergh et al., 2013). For example, the model proposed by Bürgy et al. (2019) allow employee shifts to be extended by an hour which is still under the regulatory maximum hours but is more than the employee has agreed as normal working hours with the employer. Other time-related constraints found from literature include limiting consecutive working and non-working days, maximum amount of overtime and amount of weekend shifts during a predefined period (Van den Bergh et al., 2013). In addition to complying with laws and regulations the aforementioned constraints are intended to create fairness and keeping employees satisfied as stable scheduling is shown to increase productivity within employees (Williams et al., 2018).

Due to having such a large spectrum of constraints and aspects to account for, optimization problems are commonly complex and highly variable between use cases. Constraints and objectives as well as the solution type differ largely between application areas. This makes it hard to compare different optimization solutions and the goodness in their results as Van den Bergh et al. (2013) note in their literature review. Only few research papers compare their results to other studies within the same field and even then the comparison is done to a very limited amount of recent solutions. Ernst et al. (2004) observe similar aspects in commercial scheduling software which can provide considerable optimization capabilities but with the price of being specialized to a targeted application area with no easy way to apply it to other industries. On the other hand, software solutions capable of broad application commonly lack automated scheduling and function more as a tool for manual functions and reporting where the scheduling is done as a manual process.

2.3 Experimental design in simulation

Imitating real-life processes using mathematical computer models is called simulation. It is widely used in trying to gain understanding on how the underlying process

behaves in different situations. This leads to experimenting with the model parameters on how they affect the outcome of the simulation model and thus on how they would affect the real-life situation (Law et al., 2014). When the alternative model inputs are not known in advance and the goal of experimenting has less structure, for example to find most influential input parameter, a more structured approach is needed. Designing simulation experiments is more effective than unsystematically trying different input configurations in trying to achieve the goal of the study (Kelton and Barton, 2003).

In simulations, input factors are categorized as controllable and uncontrollable based on whether they can be influenced by managerial decisions in the corresponding real-world situation (Law et al., 2014). Thus, when optimizing simulated processes, the controllable factors, such as the number of employees in a store, are generally focused on. Law et al. (2014) explain that uncontrollable factors in the real-world become controllable when exercising a simulation model and are important in simulating impacts of unusual scenarios. They add that in experimental simulation, however, the experimental and fixed factors are decided depending on the goal of the study. Finding the experimental factors that have the most influence on the simulation output is done with sensitivity analysis.

2.3.1 Sensitivity analysis

Sensitivity analysis studies the changes in the output of a model when inputs are varied in the plausible value region (Saltelli, 2002). It can be applied to any system that transforms inputs into outputs but commonly the system deploys a complex mathematical model for which additional insights are sought. According to Razavi et al. (2021), the motivations for conducting sensitivity analysis include the desire to find the most critical input that deserves the most attention and further analysis or to identify interesting regions in the input space where input interactions or small movements alter the output. They also view that inputs can be classified based on the underlying system and may consist of model parameters, constraints or boundary conditions which can be either discrete or continuous variables, or triggers that activate varying parts of the model. Outputs generally consist of objective, error or model response functions that represent the effects of different input components.

Figure 1 demonstrates the sensitivity analysis process on a high level as described by Razavi et al. (2021). The system processes inputs $\theta_1 \dots \theta_n$ into outputs Z in box (b) which are received by the sensitivity analysis tool in box (a) that creates new inputs based on the received data. The tool quantifies the information, such as input interactions and the contribution in variability of an input to output variability, it has gathered in box (c). There is a lot of information that can be quantified and thus achieving meaningful results requires the sensitivity analysis to have a clear objective (Saltelli et al., 2008).

The study of sensitivity analysis can be divided in local and global methods. Local sensitivity analysis is the more known, simple and intuitive method where one or multiple inputs are varied around a nominal point in the input space hence the

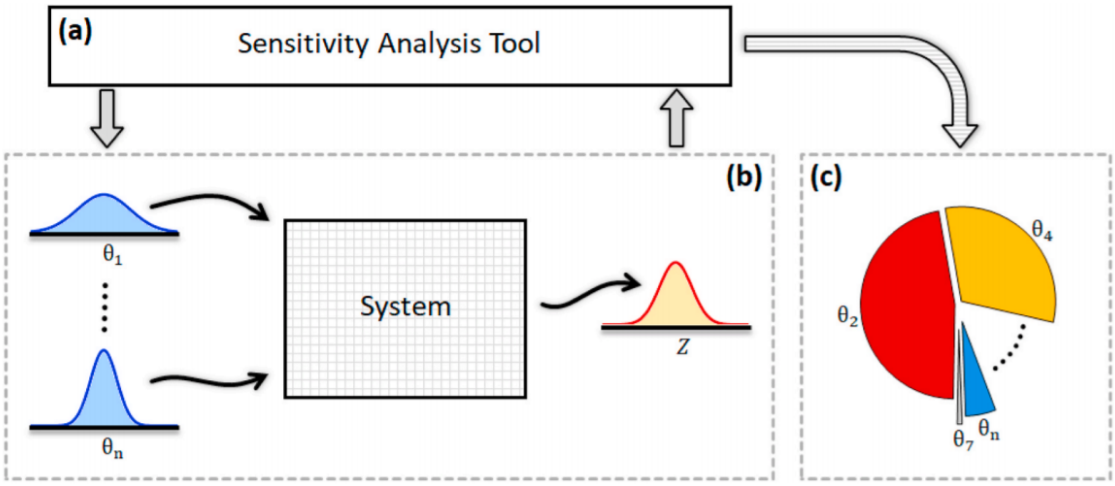

Figure 1: A high-level process chart for a typical sensitivity analysis. (Razavi et al., 2021).

naming of the method (Saltelli et al., 2008). Saltelli and Annoni (2010) note that when the model output is explored only in the neighborhood of a pre-determined set of boundaries, local approaches provide sufficient model sensitivity information as there is no need for information over the whole input space. However, the local method has a vast drawback where if the underlying model is not linear or the linearity is not known, the exploration of the rest of the input space is restricted which can lead to false assumption of important factors being noninfluential (Saltelli et al., 2008). In contrast, global sensitivity analysis attempts to provide a thorough overview of how inputs interact with each other and affect the output in the whole input space regardless of the underlying model (Razavi et al., 2021) while also preventing false assumptions of noninfluential factors (Saltelli et al., 2008).

While global methods are preferred in general sensitivity analysis, the most popular method is a local one-at-a-time (OAT) method (Saltelli and Annoni, 2010). In this method, one input parameter is changed at a time while keeping others fixed while returning to the baseline point before changing another parameter. The baseline point is commonly an established point where the properties are well-known and which already gives satisfactory results from the model. Thus, all input sensitivities and their performances are referred to the baseline point. The popularity of the OAT method can be explained by the intuitivity of changing one moving part around a safe baseline point for which the change of the output is entirely caused by the difference in the changed parameter. Similarly, a lack of effect on the output when changing an input parameter immediately implies that the parameter is noninfluential. In addition, the chances of making changes that would make the model totally unfeasible is small when only the surrounding of the baseline values are explored.

OAT is based on assumptions of input factor independence and the linearity of the underlying model and in most research papers reviewed by Saltelli and Annoni (2010),

these assumptions are not justified. They also note that multiple research papers have found interaction effects between parameters only while conducting global sensitivity analysis opposed to OAT where it is impossible to discover interactions due to the one-at-a-time characteristic. Although OAT has shortcomings, it still remains a popular sensitivity analysis method due to its simplicity and its integral part of decision making systems where the impact of potential changes to the status quo are of interest (Razavi et al., 2021).

2.3.2 Latin Hypercube Sampling

When a model has several tens or even hundreds of inputs and the execution time of the model is relatively long, an OAT method might not be feasible due to runtime limitations. To combat this issue, Space Filling Designs (SFD) are introduced to sample input possibilities so that the whole space of input values is evenly explored with a smaller amount of model executions (Damblin et al., 2013). A SFD is a more sophisticated version of random sampling where, depending on the sampling method, prior sampled points affect the next points to be sampled.

Latin Hypercube Sampling (LHS) is a SFD which attempts to populate the whole input space by taking an even amount of random sampling points from each segment of a divided input space (McKay et al., 2000). It is an extension of quota sampling, a common method for selecting survey participants, in which the target group is divided into smaller sub-groups from which the interviewer is free to choose the subjects based on the given proportion (Moser and Stuart, 1953). In LHS, the count of samples n must first be decided which is then used to divide each dimension r to n segments creating an r-dimensional hypercube when $r \geq 2$ (Law et al., 2014). Each segment is turned into a standard uniform distribution from where a point is randomly chosen and all chosen points are moved to the diagonal of the cube. The dimensions are then independently permuted so that each hyperplane only contains a single sample. The hypercube achieves the Latin Square quality due to having exactly one sample per hyperplane, hence it is called a Latin Hypercube (Law et al., 2014).

A two-dimensional input space is depicted in Figure 2 where each dimension is divided to account for a sample size of $n = 5$ creating a 5×5 matrix. The matrix on the left includes samples points on the diagonal of the created matrix for which permutations for each dimension form the matrix on the right-hand side. The right-hand side contains the final sample locations where each row and column only contain a single sample.

To further augment the LHS method, a maximin strategy is deployed to select the best sample set. The maximin strategy maximizes the minimum distance between a pair of samples in a set of a Latin Hypercube samples and thus ensures a good space filling character for the sample set. The distance between a sample pair is calculated

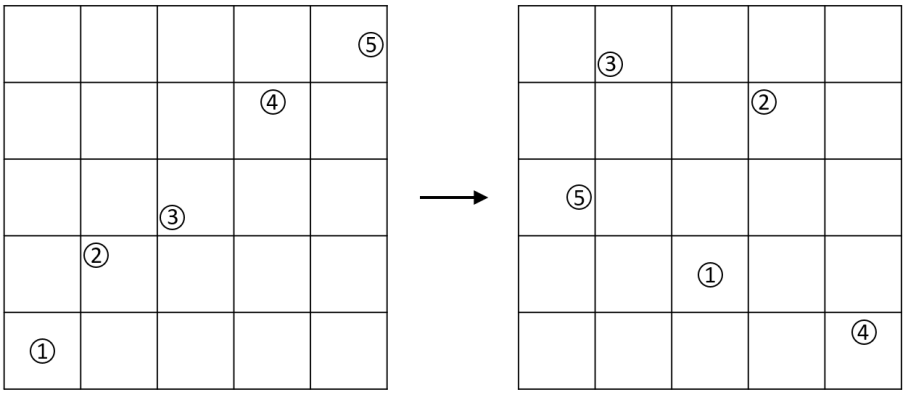

Figure 2: A two-dimensional input space containing latin hypercube sample locations before and after each dimension is permuted when $n = 5$.

using Euclidean distance

$$d(x_i, x_j) = \left[\sum_{k=1}^{r} (x_{ik} - x_{jk})^2 \right]^{1/2}, \tag{1}$$

where x_i and x_j are two samples from the sample set $X = x_1, x_2, x_3, \ldots, x_n$ where each sample has r dimensions. The distances are calculated between each pair of samples in a sample set forming a distance matrix D_{ij} from which the minimum distance **min** D_{ij} is calculated.

LHS algorithms output sample values for each dimension from a continuous uniform distribution between 0 and 1 which are then scaled to accommodate the values used as inputs. If scaling is done for discrete values and the number of samples n is larger than the possible values of a discrete variable, the latin square quality is lost. This occurs due to the samples being generated in n segments for each dimension and then rounded to the nearest integer which will result in the input rows and columns having more than one sample each. However, the original objective of LHS being a SFD is only amplified due to there being more samples than in a regular LHS. Figure 3 shows a two-dimensional Latin Hypercube before and after the samples are distributed to the integer values of the axes. On the left-hand side plot, the samples are distributed according to the gray grid which represents the LHS grid with $n = 6$. On the right-hand side, the samples are rounded to the axis values $\{x_i, x_j \in \mathbb{Z} : x_i \in [0, 4], x_j \in [0, 3]\}$ which are depicted with dashed lines resulting in integer value samples. It must be noted that since regular rounding is done and uniform distributions are used in every instance, the border values have a smaller chance of receiving the sample. For example, in Figure 3 rounded values for x_j are as follows.

$$\text{round}(x_j) = \begin{cases} 0, & \text{if } x_j \in [0, 0.5[\\ 1, & \text{if } x_j \in [0.5, 1.5[\\ 2, & \text{if } x_j \in [1.5, 2.5[\\ 3, & \text{if } x_j \in [2.5, 3] \end{cases},$$

where values 0 and 3 are achieved from a smaller set of the axis length. Thus, when creating integer samples using LHS, the value ranges of axes must be extended by 1 in both ends to ensure uniform probabilities for each value. The samples rounded to the extended values are then discarded.

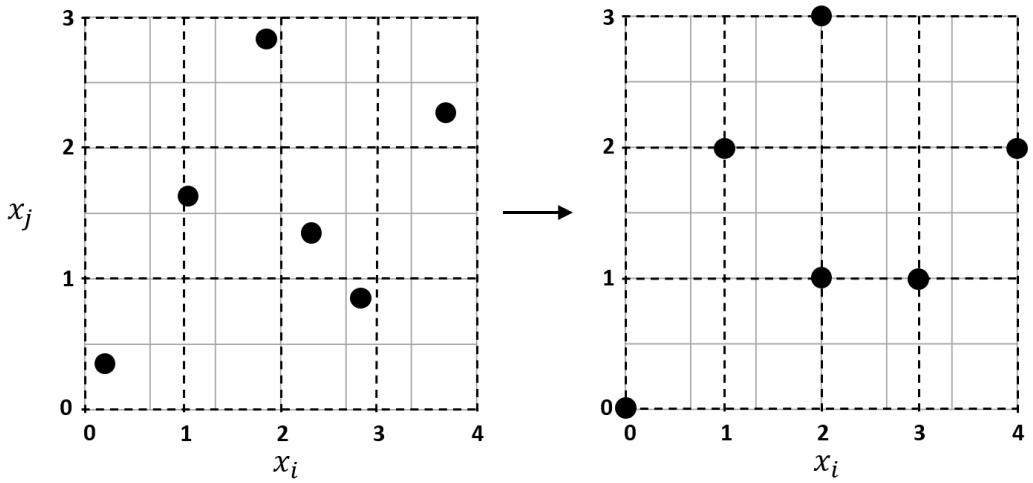

Figure 3: A two-dimensional input space containing latin hypercube sample locations before and after samples are scaled to integer values when $n = 6$ and $\{x_i, x_j \in \mathbb{Z} : x_i \in [0, 4], x_j \in [0, 3]\}$.

2.4 Literature review

The workforce scheduling problem literature mainly consists of studies proposing their own mathematical model to solve a given problem. A majority of the studies do not compare their results to other studies but rather to simple models (Özder et al., 2020) and only some include basic sensitivity analysis to further explore the proposed model. Skills are widely included in scheduling problems with varying context as they bring real life implications to purely mathematical models. De Bruecker et al. (2015) have reviewed the literature that focus on problems in which skills are taken into account. This section consists first of a general review of sensitivity analyses conducted for workforce scheduling problems and then a review interesting studies where skill has been incorporated into scheduling problems.

2.4.1 Sensitivity analysis in scheduling problems

Sensitivity analysis on workforce optimization models has been proposed in few studies, of which most analyses are done with simple what-if scenarios. These

scenarios are created by deviating one or multiple input parameters from the baseline values and inspecting the output differences to the baseline solution and thus they can be considered as an OAT method. The most important reason for the lack of sensitivity analyses is the large computational time the models require to calculate optimal results. Computing times can, for example, vary from 20 seconds of the metaheuristic solution by Talarico and Duque (2015) to over 30 minutes taken by the stochastic programming based solution by Parisio and Jones (2015).

Talarico and Duque (2015) incorporate sensitivity analysis in the proposed model to solve a workforce scheduling problem for an Italian supermarket chain. They propose a metaheuristic model to solve the problem which includes an ideal mix of full-time and part-time workers. They perform a sensitivity analysis on the configuration parameters of the proposed model by investigating each possible combination of parameters and determine the most impactful parameters while also considering the computing time. Using analysis of variance (ANOVA) they find that all the parameters in their heuristic model have a significant impact on the value of the objective function which is to minimize the total personnel cost. A statistical analysis is then performed on these parameters to measure the effects of parameters to the solution quality and to choose the best parameters possible for the final solution. During multiple scenario analyses, on average, they observed an improvement of roughly 13 % in the objective function value with optimized parameters compared to initial values generated by their algorithm.

In a manufacturing environment, Othman et al. (2012) develop a linear multi objective integer programming model which takes human aspects such as personality, fatigue and recovery time into consideration when scheduling workforce. They highlight that the model users must be aware of how recommendations given by the model change when input values are varied and thus it is important to conduct sensitivity analysis. In the first part, they create test scenarios in which the weights of the parameters of the objective function are modified to simulate different company objectives such as cost minimization or the minimization of employee idle times. They conclude that changing weights enables the model to better comply with company policies and rules by, for example, weighting the minimization of idle time increases workforce utilization. In the second part of the sensitivity analysis, they study the effects of recovery allowance, maximum endurance time and maximum fatigue inputs to the output of the model by studying three scenarios in which the weights of the parameters are shifted. They conclude that fatigue levels have minimal effect on the objective function which attempts to minimize costs, but it gives information on the amount of breaks a worker should have.

A workforce scheduling problem with fluctuating demand, days-off requirements and different daily wage rates where weekly costs are minimized is formulated by Mohamad and Said (2013) as an integer linear programming problem. They perform linear programming problem sensitivity analysis to determine non-binding variables that can be changed without changing the objective function but affect the solution. In addition, they find the range in which objective coefficients affect the objective

value but not the solution. The sensitivity analysis is performed one-at-a-time and simultaneous changes are not addressed in their work.

An integer linear programming approach is also proposed by Razali et al. (2018), who solve a preference based workforce scheduling problem in a small retail store with the objective of minimizing workforce costs. They perform a sensitivity analysis by creating five scenarios where the store workforce is decreased from the previous scenario. They conclude that decreasing from the full available workforce increased the weekly costs in all scenarios due to overtime costs. On the contrary, instead of removing workforce, Mohammadian et al. (2019) perform sensitivity analysis by adding available workforce in the healthcare sector. They propose a goal programming method based on preferences to schedule nurse shifts with sparse nurse availability due to an ongoing nursing shortage in the healthcare sector. Sensitivity analysis is made on the proposed model by calculating different scenarios in which the number of nurses available at a certain time are increased one at a time. Due to the shortage, decreasing the available nurses would cause a decrease in the optimality of the solution and thus analysis with less nurses is not considered. The analysis shows an increase in the deviation of fulfilling nurse off days based on their interests which is explained by the increased number of nurses and their preferences. However, they conclude that increasing nurses minimizes the differences in the working days of nurses and eliminating the need of overtime work.

2.4.2 Skills in scheduling problems

Henao et al. (2015) analyze the impact of adding multi-skilled employees to a workforce scheduling problem. They propose a mixed integer linear programming model to decide which skill or skills should be trained to which employee and their working tasks for a planning period of a week. The setting of the study is in a home improvement store where a skill covers the ability to work at a specific department. Each employee can be trained for any department in the store but initially each is specialized, only having one skill. Metrics used to measure the goodness of a result are weekly personnel demand coverage and weekly overstaffing. The model is executed in six scenarios in which the amount of departments in the stores vary. The results show that there are always economic benefits in training employee to be multi-skilled when the training costs are low and it also improves coverage levels and especially when there is slight understaffing. They also find that the coverage did not increase when overstaffing is either very high or very low in each department. The staffing structures are found to have the best coverage to cost ratio when there is a balance of multi-skilled and specialized employees.

Focusing on workforce planning for airport ground staff, Zeng et al. (2019) propose a branch-and-price approach for solving the underlying model. The objective of the study is to find the smallest employee mix with which daily demand profiles from a large airport in China can be covered. Due to growing airport sizes and high turnover rates, the majority of employees are always new while the other part consists of employees with more experience. Thus, a hierarchical skill system is deployed where

experienced employees are able to work more efficiently than newcomers. They also allow downgrading, which in this context means that an experienced employee can also work on a lower level task than their competence would allow. They experiment the proposed model with 10 scenarios where each have different demand profiles, amount employees and levels per skill. Their results show that the proposed model gives better coverage compared to a previously used model although the solution time for each scenario differed drastically due to the chosen branch-and-price method.

In contrary to hierarchical skills, Avramidis et al. (2010) include categorical skill in their proposed simulation based scheduling algorithm with the goal of minimizing employee costs in a telephone call center while reaching the service level target. Each employee has one or multiple skills which represent which type of call they can handle. Every employee with a particular skill finishes the tasks related to that skill as efficiently as any other employee because there is no hierarchy in the skills. They model arriving calls with stationary Poisson processes in scenarios where the size of the calling center in term of employees vary. The results of the proposed algorithm is compared to a two-step scheduling, where first staffing levels are decided and based on them shifts are created with linear programming. Results show that that the proposed model is superior to the two-step method in terms of employee costs in all call center sizes but increasingly in the larger ones. However, the proposed model did not always return a feasible solution while the two-step method always resulted in feasible schedules.

3 Research material and methods

This section presents the methods used in this thesis. First, a high level description of the workforce optimization model is given to give an understanding of the basic functionalities of the underlying process. Then, the test data is introduced alongside a statistical method for generating samples. Finally, the selection and implementation of the sensitivity analysis approach as well as key performance indicators to measure the goodness of results are explained.

3.1 Underlying workforce optimization solution

The existing workforce optimization solution is presented as a black-box solution. Thus, only the needed inputs and the resulting outputs are explored instead of exploring the mathematical models behind the solution. The target of the solution is to create shifts that match the workload as closely as possible with the best possible employee skill, while conforming to contractual requirements and collective agreements. The optimization part of the solution is divided into two main parts: long term planning (LTP) and intraday optimization. LTP assigns rest and work days to employees and determines the shift lengths for each working day of a planning period that commonly spans multiple months. This is done in two steps where first the working days are chosen and then their lengths are determined. The solution uses a workforce forecast that indicates the total amount of work per working task for each day. Both the working days and working hours optimizations use adaptive greedy algorithms which iterate over all possibilities to maximize a reward utility function until a better solution cannot be found. The intraday optimization uses simulated annealing to create optimal shifts and calculations are done for one planning week at a time. It uses the results of LTP as inputs and decides the start time of each shift, lunch break slot for each working day and tasks for each employee on each shift. Intraday optimization results are complete employee schedules that comply with every restriction, but which can still be modified by the manager. Figure 4 depicts a simplified process of the optimization solution. In this thesis, the focus is on the LTP module.

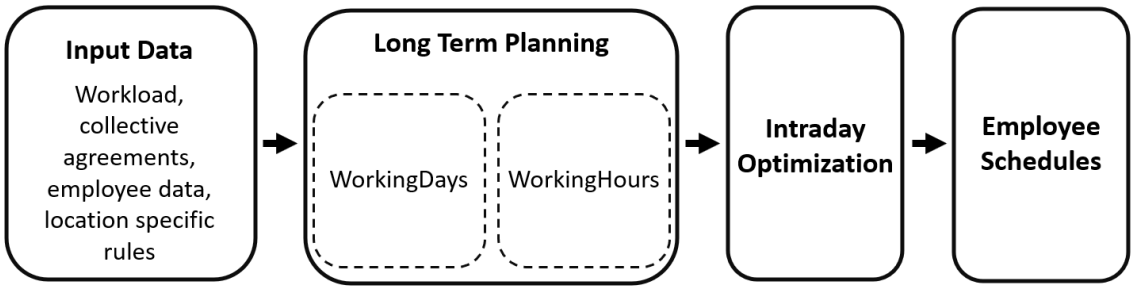

Figure 4: The underlying workforce optimization solution process.

A multitude of input parameters are needed for the solution to work as desired and

for it to output meaningful results. They are used as constraints and as penalizing or rewarding factors in both parts of the optimization solution. Some of the most important parameters include the preferred base shift duration, shift length restrictions, average weekly hours and balancing periods. Values for these constraints are collected from parameters defined in the employee contract, collective labour agreement and store level operations. When needed, the values can be manually changed by the model configurer.

In addition to the aforementioned parameters, each employee has a skillset which determines the skills they possess and thus which tasks they can work on. Tasks and skills are related in such way that there exists only one skill per task to be completed, for example, the task of shelving requires the shelving skill. In LTP skills are used as penalizing or rewarding factors when assigning working days and their lengths to employees. The intraday optimization uses the skills as constraints when assigning employees to tasks. The skillset of an employee in the optimization input is not necessarily an exhaustive list of the employees competences, but rather a list of tasks on which the employee is desired to work on. Adding a skill to an employee means that they are able to work on the task related to the added skill. Any costs and time required to train an employee to a task for which they do not yet have competences are not considered.

Skills have levels from 1 to 3 which indicate the priority of the skill which the employee should be using. In addition to skill priorities, the tasks are also given priorities representing the importance of the task. For example, an employee can have a skill level of 3 in "cashier" and a skill level of 1 in "meat counter" from which, depending on other employees and task priority, the system will prefer cashier as a task for the employee. An employee can not be assigned to a task for which they do not possess the skill for. A time period in which an employee is not assigned to any particular task is counted as idle time.

3.2 Test data

The test data used in experiments represent 20 different planning units where each of them has one input file containing all needed information. A planning unit might represent an entire store, independent floors or departments within a store depending on user needs. Thus, there are large differences in amount of configured employees and skills between planning units. The smallest planning units in the test data set have roughly 10 employees and a few different skills while largest planning unit has over 100 employees and over 15 unique skills. Each data file includes workload forecasts for the planning period of the scenario which ranges from 1 to 6 months with the most common length being 4 months. The working days optimization in LTP uses day-level forecasts of each work type which are further disaggregated to smaller intervals later in the intraday algorithm. The workload forecasts are pre-calculated and will not be analyzed in this thesis. Each file also includes all parameters that are used in the optimization including input parameters for each employee and store specific inputs. Employees have a skillset where skills that they

possess are defined with a level from 1 to 3. The skills for each employee can be adjusted by the store manager and commonly trial and error methods are used to achieve desirable optimization results.

Baseline statistics for each test case can be seen in Table 2. The skill count is the amount of different skills and thus also the amount of tasks that exist in the test case. Skills per employee is the average amount of skills an employee possesses. A value of under 1 means that there are employees within the test case who do not have any skills. Workload hours are all the forecasted working hours in all tasks combined for the whole planning period. Idle hours depict the total amount of estimated idle time between all employees for the whole planning period.

Table 2: Baseline statistics for each test case.

Test case	Employee count	Skills count	Skills per employee	Workload hours	Idle hours	Workload hours per idle hours
1	65	10	4.6	44908	1105	38.7
2	91	12	0.9	25350	3221	7.7
3	30	5	1.0	24892	666	35.8
4	86	11	2.4	22679	4569	4.9
5	60	10	3.9	9833	3070	3.3
6	80	10	4.8	8560	1881	4.4
7	14	2	0.0	1071	2932	0.4
8	80	10	4.8	8560	1861	4.5
9	19	5	2.5	3216	779	4.1
10	15	3	0.6	3419	1104	3.0
11	112	14	0.9	28039	2301	11.4
12	14	5	3.3	3502	588	7.0
13	25	5	1.6	4101	3972	1.0
14	14	5	3.3	3502	596	6.9
15	33	14	2.0	7484	740	9.9
16	10	2	0.2	918	1500	0.6
17	78	10	0.9	4784	3886	1.2
18	33	14	2.0	7484	741	9.9
19	30	14	2.1	7543	812	8.9
20	27	4	1.0	3895	167	23.4

3.3 Sampling of new skills

The best way to execute OAT is to try every single possibility of deviating from the baseline input values, but with larger setups this becomes infeasible due to the excessive amount of iterations that would be needed. To combat this issue, Latin Hypercube Sampling is used to choose what skill should be added to which employee in each OAT iteration. LHS explores the employee-skill space evenly while keeping the number of samples and thus the iteration count manageable. The Latin

Hypercube samples are generated in a discrete two dimensional space of size $e \times s$ where e is the total amount of employees and s is the total amount of different skills in the test scenario. The samples drawn from LHS are thus indices of the matrix dimensions which are used to select the employee-skill combinations.

An example of the employee-skill matrix is shown in Table 3 where rows depict the skillset of an employee. In the baseline situation each cell is either empty if the employee does not have the skill or marked with the skill level if the employee has the skill in their skillset. A generated sample is thus an employee-skill combination to use in an OAT iteration. If the employee already has the sampled skill in their skillset, the sample is skipped as an employee can not have the same skill twice. Table 4 depicts the situation where an employee-skill sample has been added to the baseline situation. The sample adds skill 2 with level 1 is added to employee number 3.

Table 3: A two dimensional skillset matrix depicting the skill levels of each employee.

	skill 1	skill 2	skill 3	...	skill s
employee 1		3			
employee 2	3		1		
employee 3					3
...					
employee e		2	3		

Choosing the amount of samples to be generated in this setting is a difficult task. Loeppky et al. (2009) suggest that $n = 10d$ sample points, where d is the amount of dimensions in the input space, would be sufficient for initial results for a computer experiment. In our setting, however, the two-dimensional input matrix is merely a way of spreading out all possible input parameters where each cell is an input parameter that is either 0 or 1. Choosing $n = 10d$ sample points would mean that 20 samples are taken and while it might be sufficient for smaller planning units with only few employees and tasks, it is not enough to discover the possibilities of a larger planning unit where 20 sample points would only cover 1 to 2 % of all possibilities. Thus, the amount of samples are scaled according to the size of the planning unit. In the test cases, a task is underemployed more often than an employee is as there are generally less tasks than employees. Hence, skills are used as the scaling factor for the sample size. The amount of samples generated is $n = 15s$, where s is the number of unique skills in the planning unit. On average, this means that each skill is added to 15 different employees one at a time.

Table 4: A two dimensional skillset matrix after skill 2 with level 1 is added to employee 3.

	skill 1	skill 2	skill 3	...	skill s
employee 1		3			
employee 2	3		1		
employee 3		**1**			3
...					
employee e		2	3		

3.4 Key Performance Indicators

LTP has integrated key performance indicators (KPIs) which are automatically saved in the output file of the solution and can be extracted for further use. Two KPIs were chosen to measure the performance of the solutions: TotalIdle and TotalScore. TotalIdle refers to the total amount of idle work there is in the solution for the whole planning period. Idle work refers to a time during the working shift of an employee when they are not working on a specific task but where they would be available to work. This means that any breaks are not counted as idle work. TotalIdle is chosen as a KPI due to it directly addressing the total duration that the employees are not efficiently working. It is measured in ticks of 15 minutes, the base duration in the solution, which can easily be translated to a monetary value to use in further decision making. Improving the TotalIdle means lowering the total value closer to zero. When TotalIdle is zero, each employee is assigned to a task at all times when they are available to work during the planning period. Due to the quantifiability of the TotalIdle, it can also be compared between planning units and different test cases. When comparing test cases, one must remember to account for other differences too, such as the amount of employees and the length of the planning period.

TotalScore accounts for the task priority, the working day length deviation from the desired working day length as well as the TotalIdle. It is the value of the function which is maximized in the LTP algorithm when assigning working hours to employees. Thus, it gives a broader view of the solution quality compared to the TotalIdle, where effects on other aspects than idle time can be seen. The TotalScore can not directly be compared between different planning units as their employee counts, contractual agreements and planning unit settings can greatly vary. It is only useful when comparing results of different optimization runs for the same planning unit.

3.5 Multiple One-At-a-Time algorithm

Conducting a proper sensitivity analysis requires certain considerations to be made so that the correct approach is used. Saltelli et al. (2008) list the most important aspects from which the most relevant for this thesis are the setting for the analysis, the computational cost needed to execute a model, and the number of inputs as well as their possible interactions.

Sensitivity analysis is used in this thesis with the goal of providing information on the possible benefits of adding missing skills to employees. The objective is thus to start from a given input configuration of employee-skill combinations and attempt to find an input with better results. Global sensitivity analysis methods are superior in finding parameter interactions and measuring model robustness. However, the underlying solution is known to be robust and only changes around the nominal input configurations are feasible. Thus, local sensitivity analysis is used where only the proximity of an initial state is explored. The number of inputs to be analyzed depends on the number of employees and the quantity of unique skills in the test scenario. The analyzed test cases have up to 1500 inputs of interest.

The desired sensitivity analysis method should be one that can be applied to any test scenario while maintaining a reasonable total execution time. Even if test scenarios with low input quantities and short execution times would allow more in-depth sensitivity analysis methods than larger more complex scenarios, a common method is chosen to achieve unified results. The method should be able to handle hundreds of input parameters while also having local sensitivity analysis characteristics. Thus, the chosen model is a Multiple One-At-a-Time (MOAT) sensitivity analysis method. It is an extension of the regular OAT method which builds a new set of inputs from the results of the OAT analysis and repeats the experiment until a certain stopping criterion is reached. Saltelli and Annoni (2010) describe this method as an alternative, more complete version of OAT.

In real-world use, employee skillsets are adjusted by the store manager according to employee wishes and their skills. Thus, it is assumed that the skillsets, as they exist in input files, are correctly set to achieve good enough results from the workforce optimization solution and satisfy the workforce need. The existing skillsets are used as a baseline from where the sensitivity analysis method starts to explore neighborhood possibilities. In this setting, an OAT iteration means adding a skill to the existing skillset of an employee while keeping all other input parameters unmodified and then executing the solution. The added skill is added as a level of 1 skill as the differences compared to adding it with level 3 are negligible and introducing a further dimension to the sensitivity analysis increases computing costs drastically. In addition, when adding a new skill, it is not desirable for it to be prioritized over existing skills. However, it can be argued that being able to complete a new task efficiently requires a lot of training or practical experience in the task.

Procedure 1 Functions

function CallLTP('in : $LTPinput$')
    Call the LTP algorithm with new input file
end function

function OAT('in : $LTPinput$')
    $S \leftarrow$ Generate LHS samples
    $R \leftarrow$ empty matrix ▷ Initiate results matrix
    for s in S **do**
        if s not in in **then**
            $in_s \leftarrow in + s$ ▷ Add sampled employee-skill to LTP input file
            $R(s) \leftarrow$ CallLTP(in_s) ▷ Collect results to matrix
        end if
    end for
    $s \leftarrow \text{argmin}_s$ idle($R[s]$) ▷ Choose sample that minimizes idle
    return (R, s)
end function

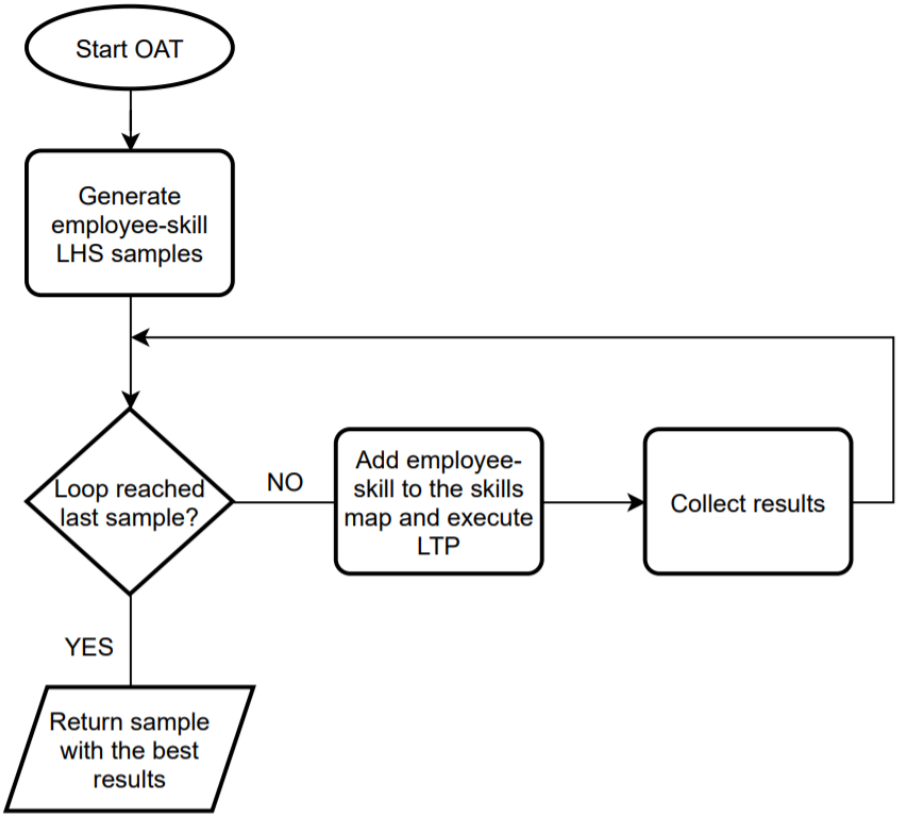

Figure 5: One-At-a-Time function from Procedure 1 as a simplified flowchart.

Procedures 1 and 2 describe the used sensitivity analysis method in simplified pseudo codes. Figures 5 and 6 depict the procedures as flowcharts. In Procedure 1, two functions that are used in the MOAT method are introduced. Lines 1-4 define a function to modify the input file of the test scenario with the desired employee skills map and uses it to call the LTP algorithm. A skills map is a collection of all employee skillsets and can be pictured as a two-dimensional matrix with employees as rows and skills as columns (see Table 3). Lines 5-16 define the OAT method, which takes a skills map as input. In the beginning of the OAT method, LHS samples are created based on the skills map provided in the function call. Each sample is checked whether it already exists in the skills map as samples are created without considering the existing employee skills. This decreases the final used sample count by approximately 25 %. If the sample is not already in the skills map it is temporarily added there as depicted in 4 and it is used to execute the CallLTP function from which resulting score and idle values are collected in a results matrix. When all samples have been iterated over, the sample that resulted in the smallest idle value when added to the skills map is chosen and it is returned as the output value for the OAT function.

Procedure 2 MOAT algorithm

1: $SM \leftarrow$ SkillsMap from input file ▷ Collect baseline SkillsMap
2: $R_b \leftarrow$ CallLTP(in) ▷ Collect baseline results
3: $AddedSkills \leftarrow \emptyset$ ▷ Initiate added skills list

4: $R,\ s \leftarrow$ OAT(SM) ▷ First iteration of MOAT
5: $SM \leftarrow SM + s$ ▷ Create the new baseline SkillsMap
6: $AddedSkills \leftarrow AddedSkills + s$ ▷ Collect added skill

7: **while** $\left[\text{diff}(\textbf{min}\ R_b(\text{idle}), \textbf{min}\ R(\text{idle}))\right] \geq 1\%$ **OR** ≤ 20 **do**
8: $R_b \leftarrow R$ ▷ Update baseline results
9: $R,\ s \leftarrow$ OAT(SM)
10: $SM \leftarrow SM + s$ ▷ Update baseline SkillsMap
11: $Add \leftarrow Add + s$ ▷ Collect added skill
12: **end while**

Procedure 2 describes the MOAT algorithm in full. Lines 1-3 set up the process by collecting the baseline skills map from the input file, calculate the baseline results with the original input file and initiating an empty list for added skills. The first MOAT iteration is done on lines 4-6, enabling the use of a while loop for further executions on lines 7-12. A single MOAT iteration is completed when the baseline skills map is updated based on the OAT approach. On line 4, the OAT function is executed with the skills map and the result is used to update the baseline skills map on line 5. Starting from line 7, the MOAT is continued while the difference between idle values in the baseline results and the results from the MOAT iteration exceed a small arbitrary number to terminate when meaningful results are no more achieved. The stopping criteria are when the differences between the results are less than 1 % or is smaller than 20 units. On line 8, the baseline result value is updated

to represent the value of the previous MOAT iteration. This way, the while loop evaluates the difference between the two latest MOAT iterations and stops if the decrease is too small.

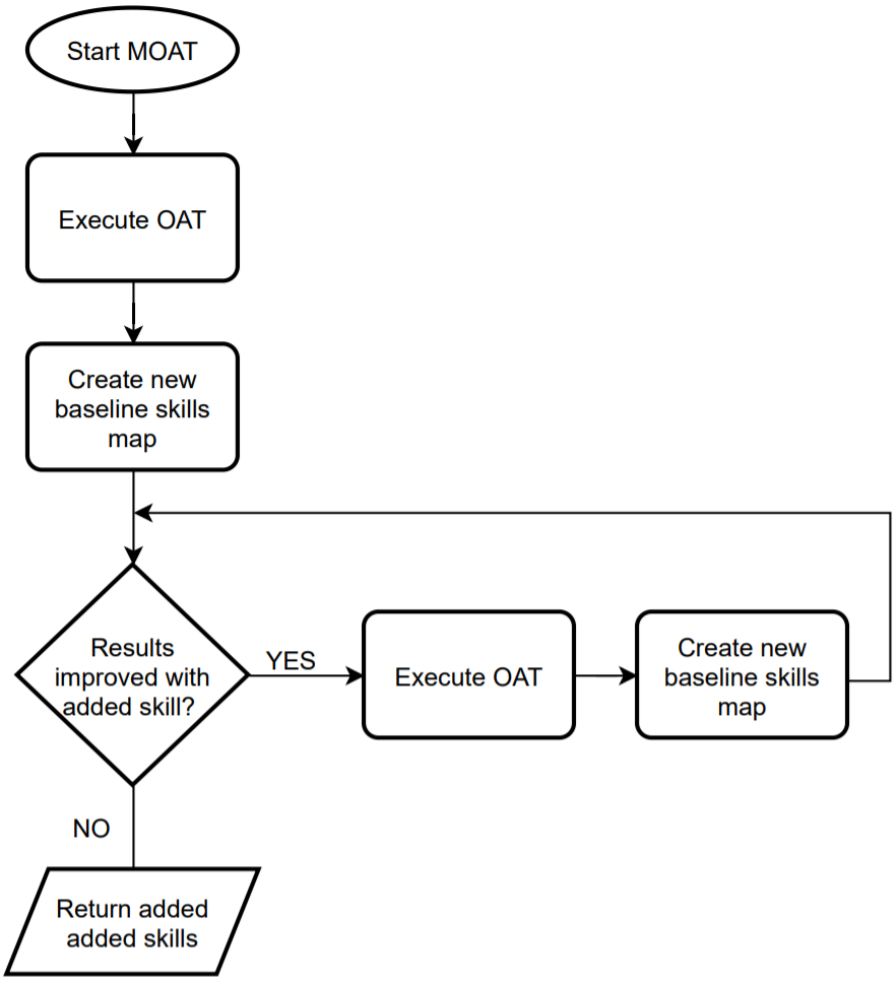

Figure 6: Multiple One-At-a-Time algorithm from Procedure 2 presented as a simplified flowchart.

In this version (A) of the MOAT algorithm, the criteria for adding a certain skill as well as stopping the algorithm is based on the largest decrease in the idle value. Another version (B), where the criteria is changed to be based on the largest improvement in score is also tested in Section 4. In this version, line 14 in Procedure 1 is changed to choose the sample with the largest resulting score and the evaluation on line 7 in Procedure 2 is changed to check whether the difference between the maximum score values is over 1 %.

The two versions use different objectives for the MOAT algorithm. In version A, the objective is to choose an addition of an employee-skill combination that minimizes the idle value. In version B, the objective is to choose an employee-skill combination that maximizes the score value. Both versions of the MOAT algorithm are tested and the better one is chosen.

4 Results

This section presents the results of executing the MOAT algorithm on test case scenarios. First, Section 4.1 discusses the technical execution of the analysis and its performance. Next, Section 4.2 compares the two different versions of the MOAT algorithm. Finally, Sections 4.3 and 4.4 present the results of adding skills and the decrease to the total idle value.

4.1 Execution of the MOAT algorithm

The sensitivity analysis is completed for a total of 20 different test scenarios containing unique data. The smallest test scenario includes 10 employees and only 2 skills for a total of 20 possible employee-skill combinations while the largest test scenario has 112 employees and 14 skills for a total of 1568 combinations. The results are obtained by calling the LTP algorithm in a Docker virtual machine with an input file that is modified using Python 3.9. The virtual machine is created on a Windows laptop with a 4-core 3.4 GHz processor and 16 GB of RAM. The Python package SciKit is used to create samples using LHS.

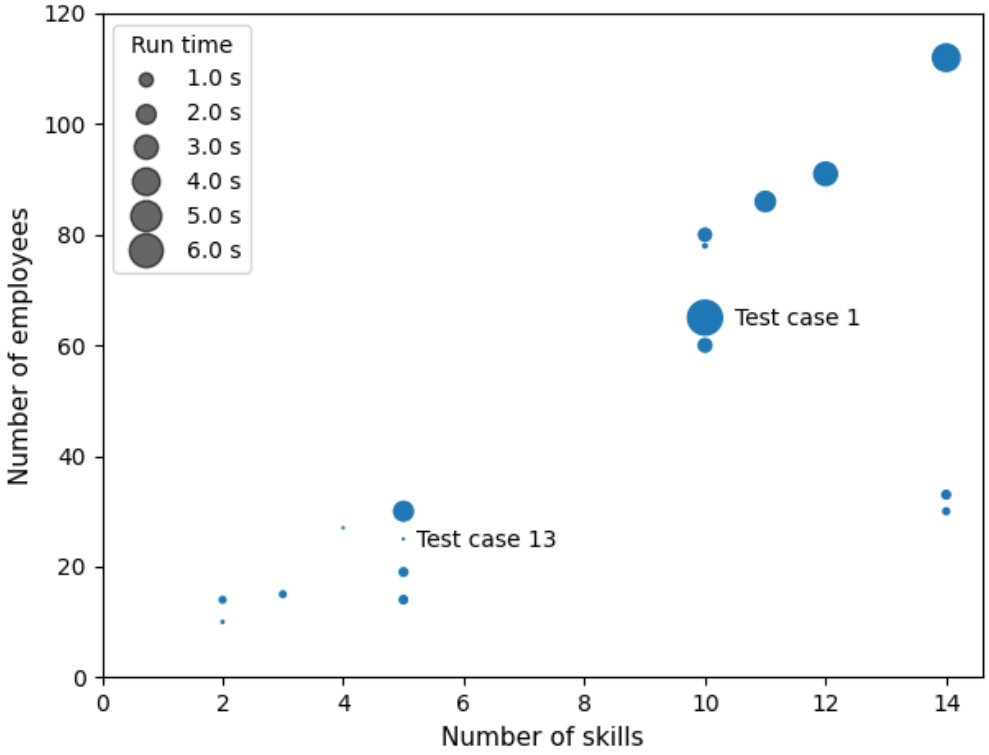

Figure 7: The LTP algorithm execution times for test scenarios with regards to their employee-skill combination count.

The computational cost of executing a single LTP algorithm run for each test case can be seen in Figure 7 in which a large variation in execution times between different test cases can be seen. The lowest execution time is 0.01 seconds for test case 13 whereas the longest execution time for test case 1 is almost three orders of magnitude longer at 8.58 seconds. For an imaginary scenario with 1000 employee-skill combinations where all combinations are experimented with would require 1000 OAT iterations. This corresponds to a total of roughly 10 seconds and 2.5 hours respectively. By using LHS in the MOAT algorithm, the OAT iteration count is decreased by up to 70 % depending on the amount of skills in the test scenario.

The Spearman correlation values of the execution time to the amount of employees and skills are 0.58 and 0.35 respectively. The moderate correlation values are supported by visually analyzing Figure 7 in which large variation in run times can in some part be explained by the difference in the amount of employees and skill between scenarios. However, the comparably high 2 second runtime of test case 3 with 30 employees and 5 skills and the low 0.06 second runtime of test case 18 with 33 employees and 14 skills are clear outliers when considering this explanation.

A more thorough exploration of run time differences is not in the scope of this thesis and based on only the number of skills and employees, one can not make an assumption on the magnitude of the execution time. Thus, baseline run times are estimated by running the LTP algorithm before starting the sensitivity analysis. The general increase in execution time when more skills and employees are added has a two-fold effect on the execution time of the sensitivity analysis. First, increased iteration execution time linearly increases the total runtime and, second, an increased number inputs implies more iterations to be executed when performing sensitivity analysis.

4.2 Comparison of MOAT stopping criteria

Both versions of the MOAT algorithm are executed for each test case. The differences in the total improvements until the algorithm stopping criteria can be seen in Table 5 where a positive value indicates a better result when using the version A and a negative value indicates a better result with version B. In version A, the skill choosing and stopping criteria are based on idle and in version B they are based on skill. As expected, version A performs better when comparing the decrease in idle value and likewise version B performs better when comparing differences in score value. For example, in test cases 5 and 18, however, version A performs better for the decreasing the score. Similarly, for test case 6, version B performs better for decreasing the idle value. This is interesting because if the objectives of the algorithm versions are to maximize the improvements of either idle or score values, one would think that the results are the best in decreasing their objectives. Since this does not hold true for some test cases, it indicates that the algorithm is not perfect in finding inputs with the largest improvements. Overall, the differences between the two versions are small but since version A is marginally better, it is chosen as the version with which further results are presented.

Table 5: Differences in KPI improvements when using both versions of the MOAT algorithm. A positive value indicates better results when using version A.

Test case	Idle improvement A vs. B		Score improvement A vs. B	
	Value	Percentage	Value	Percentage
1	2	0.0%	1	0.0%
2	0	0.0%	-151	-0.2%
3	83	3.1%	1	0.0%
4	-59	-0.3%	-1000	-0.7%
5	681	5.5%	8455	1.7%
6	-135	-1.8%	-7824	-0.1%
7	-46	-0.4%	-27122	-0.2%
8	58	0.8%	399	0.0%
9	0	0.0%	0	0.0%
10	-9	-0.2%	-1172	-0.2%
11	93	1.0%	304	0.3%
12	0	0.0%	-21	0.0%
13	0	0.0%	-2	0.0%
14	0	0.0%	11	0.0%
15	3	0.1%	-48	-0.1%
16	0	0.0%	0	0.0%
17	33	0.2%	608	0.1%
18	-12	-0.4%	364	0.5%
19	29	0.7%	-503	-1.9%
20	0	0.0%	65	0.3%

Figure 8 shows the idle and score value improvements of nine test cases where the MOAT algorithm produced the largest idle value improvements. The figure shows improvement as a percentage of the baseline values during iterations so that the two values can be compared. For either value, improving them means decreasing their value. For test cases where the MOAT algorithm is stopped before iteration 7, a dashed line is drawn to fill the plot. The total idle time is a value that is used as a component when calculating the total score value from the LTP algorithm and thus the two are correlated. In some test cases, the idle value improves slower than the score value which indicates that the idle value has a strong influence on the score value. The two other main factors affecting the score value is the task priority and deviation of working day length which in these cases have a lower presence. This can be seen especially in test case 16, where the idle value improves by only 36.1 % but the score value gets an 86.2 % increase. In other test cases, the idle value is improved drastically whereas the score is improved only slightly. For example, in test case 3, the total idle gets close to zero with a 96.6 % improvement by iteration number 6 whereas the score value has only improved by 20.1 %. This implies that the score value is largely influenced by other factors than the idle value.

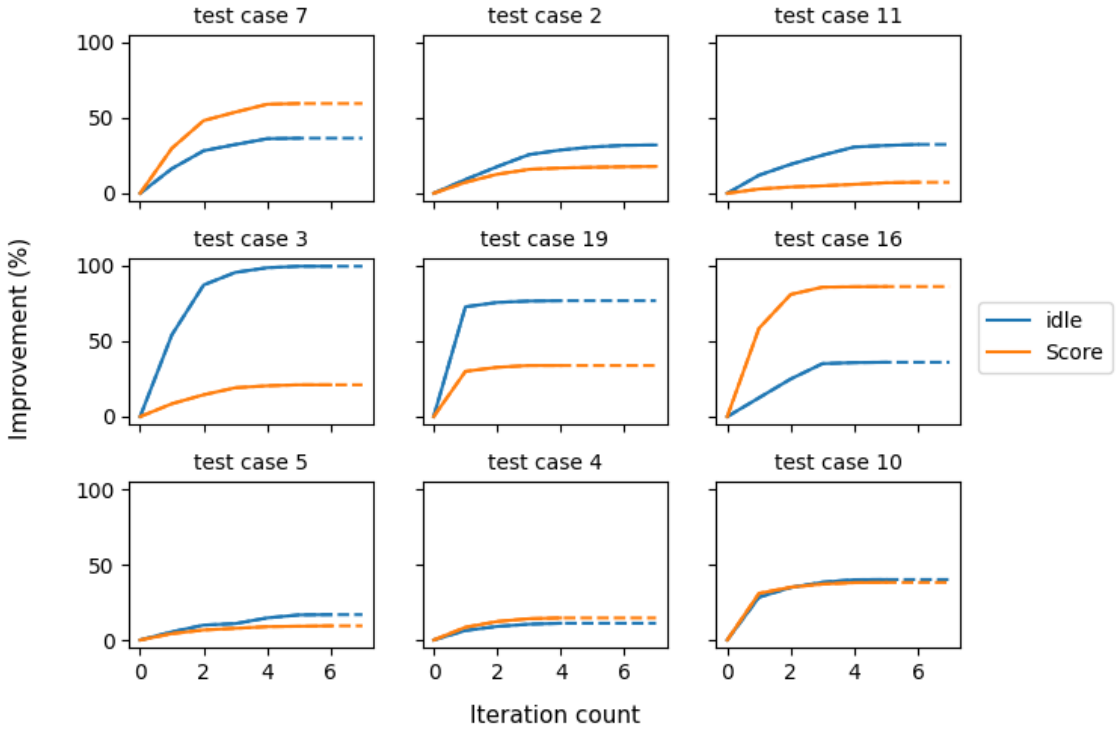

Figure 8: The percentage improvement of score and idle values from their baseline values during MOAT iterations.

4.3 Optimized addition of skills

Table 6 shows the results of the MOAT analysis for the most impacted test cases. Complete results can be found in Appendix A. For each iteration, the skill that brings the largest decrease in idle value is added to an employee with information on how much the idle value is improved. Iterations in this context indicate the number of added skills and the order in which they are added to achieve the largest decrease. A missing skill is added to an employee until the stopping criteria of MOAT is reached. Test cases where very little or no decrease could be seen reach the stopping criteria after their second added skill. In other test cases, such as number 2, up to 7 missing skills are added before the MOAT algorithm is stopped. It must be noted that skill IDs and employee numbers are not comparable between test cases.

The added skills are distributed within employees so that a single employee appears in the results only once. The only exception is in test case 10, where skills 1 and 2 are both added to employee number 1. The same missing skill, however, is added multiple times for different employees in many test cases. For example, skill 1 is added to 4 different employees in test case 3 and skill 2 is added to 3 different employees in test case 7. The addition of the same skill indicates that there is a large employee deficit for the given task. Consequently, adding multiple skills to an employee would imply

that they have significantly more idle hours than other employees.

There is also a large difference between test cases in how the idle value improves when more skills are added. In test case 19, the first skill addition improves the idle value by 2367 but further skill additions hardly improve it more. In contrast, the idle value in test case 2 does not improve as much for the first added skill but continues to improve with steadily decreasing amounts until the last added skill.

Table 6: Results for most improved test case on which skill should be added to which employee and how much it improves the idle score.

	Iteration	1	2	3	4	5	6	7
Test case 7	Skill ID	2	2	2	1	1	-	-
	Employee number	6	7	4	6	10	-	-
	Idle value decrease	1918	1392	484	452	39	-	-
Test case 2	Skill ID	4	11	10	11	10	4	1
	Employee number	34	24	43	10	3	25	2
	Idle value decrease	1175	1103	1025	396	260	141	28
Test case 3	Skill ID	1	1	1	1	2	-	-
	Employee number	17	26	24	29	21	-	-
	Idle value decrease	1445	880	222	82	26	-	-
Test case 19	Skill ID	4	5	7	-	-	-	-
	Employee number	11	13	9	-	-	-	-
	Idle value decrease	2367	94	30	-	-	-	-
Test case 10	Skill ID	2	2	1	1	-	-	-
	Employee number	9	1	4	1	-	-	-
	Idle value decrease	1252	285	153	76	-	-	-

4.4 Decreases in idle values

Decreases in idle values for each test case in each iteration are presented in Figure 9. Detailed idle improvement progress during MOAT iterations can be seen in Appendix A. If the MOAT algorithm is stopped before iteration 7, a dashed line is drawn from the last iteration to fill the plot. The test cases can roughly be categorized into two groups where in one, adding missing skills results in significant decrease in the idle value and in the other group the idle decrease is small or insignificant. The largest decrease in idle time occurs in test case 7 where a 4331 tick decrease is seen. This represents 1083 fewer hours of idle time during the planning period of 126 days which is the equivalent of 8.5 hours per day or more than a full time employee. The largest decrease in one iteration can be seen in test case 19 where adding a single missing skill to an employee decreased the idle time by 2434 ticks or 608.5 hours during the planning period of 126 days. This equals to almost 5 hours less idle per day during the whole planning period which corresponds to two thirds of the hours of a full time employee when assuming 7.5 hour working days. In test cases 12, 13 and 14, no decreases can be achieved by adding missing skills to employees.

Directly comparing different test cases is usually not feasible due to each test case representing a different planning unit with different amount of employees, skills, contracts and working hours. However, large stores might divide departments or floors into planning units where a store manager would benefit of directly comparing results within the store.

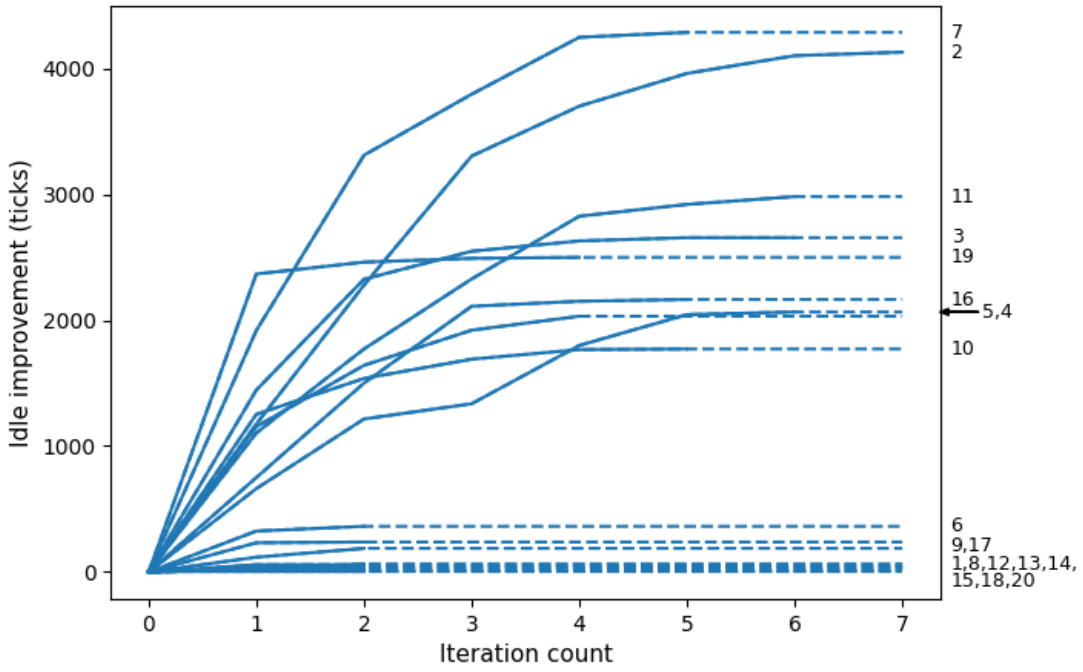

Figure 9: The idle decrease of each test case from their original value during MOAT iterations.

It is also important to look at the proportional decrease in the idle value to get a better understanding of the results. Figure 10 shows the decrease in idle value in percentage for each test case during each iteration. Test cases 3 and 19 with 99.7 % and 76.9 % decreases respectively stand out from the chart. In test case 3, with a decrease of 2655 ticks and an addition of 5 missing skills, the idle is almost completely eliminated. Adding a single missing skill in test case 19 improved the idle value by 72.8 %. While the total idle decreases in test cases 4 and 5 are above 2000 ticks and rank high in Figure 9, their proportional decreases are under 20 %. This can be explained by the size of the planning units the test cases represent. The two test cases have 86 and 60 employees respectively which ranks them in the larger end of all test cases and more employees generally results in more total idle time.

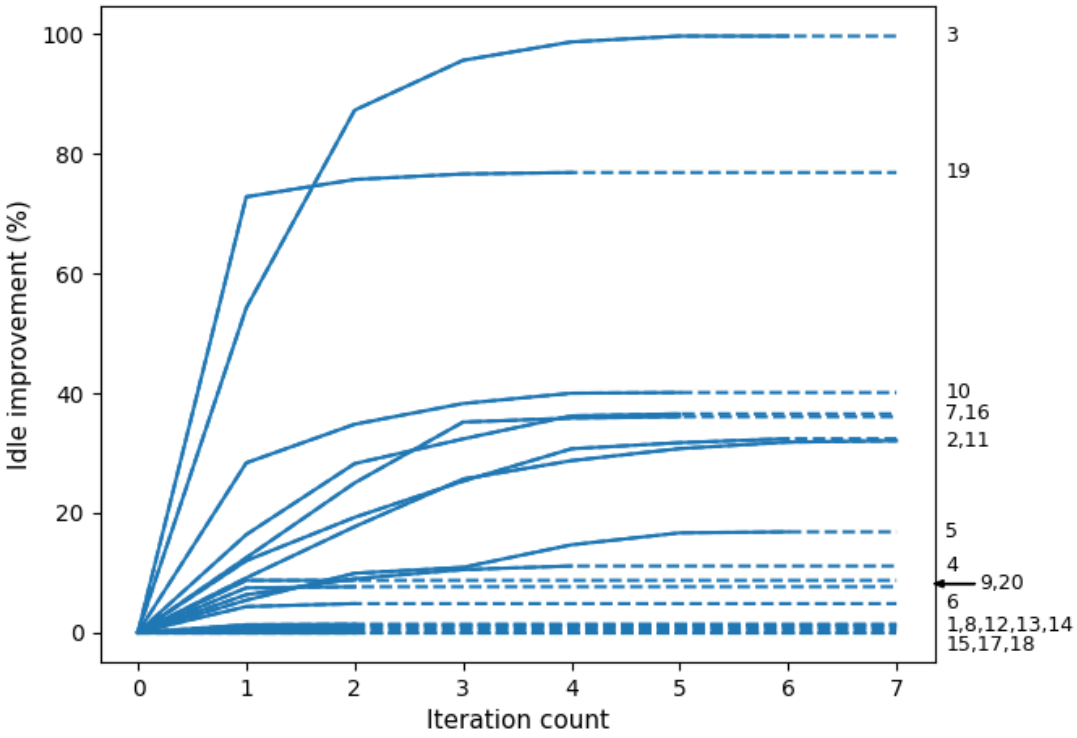

Figure 10: The idle decrease of each test case in percentage during MOAT iterations.

5 Discussion

Results show that in some test cases the idle values can be significantly decreased while in other test cases the values could not be made better. There is thus large variation in the values and the causes for that are of great interest. To determine whether it is possible to identify planning units where significant decreases can be made, an analysis is done for baseline values and planning unit variables. First, the correlations between the amount of employees, amount of skills, average amount of skills per employee, average total workload ratio and average daily workload ratio is calculated. In this context, the workload ratio indicates the amount of future workload per task per employee who has the skill to complete the task. The average is calculated over all tasks in the planning unit of the test case and a daily value is calculated by dividing the value by planning period. A high average workload ratio means that on average, there is a lot of workload per employee for a given task.

The average correlations for all test cases can be seen in Table 7 where Spearman correlation is used because the data does not follow a normal distribution. The employee count has a low correlation of 0.23 from which no conclusions can be drawn. The skill count has a correlation value of 0.04 meaning that the correlation is nonexistent. A moderate negative correlation of -0.51 can be seen in the average skill count per employee which suggest that the less skills employees have, the more the idle value could be improved. An intuitive explanation for this is that having less skills per employee makes them have to wait until there is work in the few skills they have rather than having possibilities of working on different tasks that would be more important which induces idle work. Workload ratio has a moderately strong positive correlation coefficient of 0.69 which implies that the larger the workload ratio is, the more the idle value can be decreased. Since there are differences in the lengths of the planning periods of test cases, a more reasonable value to calculate correlations with is the average workload ratio per day which is slightly lower at 0.56 but still shows moderately strong correlation. Calculating and analyzing the daily average workload ratio and the average skill count per employee can thus give indications whether there is room for improvement in employee skillsets with the goal of reducing idle hours.

Table 7: Average Spearman correlation between the total decrease in idle score and different statistics of test cases.

	Decrease in idle
Employee count	0.23
Skill count	0.04
Average skills per employee	-0.51
Average workload ratio	0.69
Average workload ratio per day	0.56

In a situation where there is a lot of daily work per employee per skill, the decrease of the total idle when adding a missing skill to an employee is not intuitive. One would

think that if there is more work than an employee can handle, there would not be any decrease if more work would be available to be done. However, there exist multiple scenarios which by their own or in tandem with other scenarios create situations when the idle score is improved. Below are listed the most common scenarios which can also be observed in the test cases:

- **Days without workload**
  If there exist working days where the workload for certain tasks is zero and if there exists an employee who only possesses skills to work on tasks that do not have workload, idle hours are accumulated for those days. Adding a skill that has workload enables the employee to be allocated to work on other tasks which will lower the idle time. However, as the workload data in this thesis is aggregated to a single value per skill for the whole planning period of the planning unit, it is not possible to inspect daily workload forecasts.
- **Skewed skillsets**
  Employee skillsets can be skewed so that there is an abundance of workforce available for some tasks while other tasks have much less available workforce. An employee with skills in the tasks with most workforce might need to wait for their turn to be allocated to the work which results in idle time. This imbalance in skillsets can be relieved by adding missing skills to these employees. This way, they can be allocated to work on other tasks rather than wait until there is work available in the overemployed task which in turn decreases the idle time.
- **Employee without skills**
  There can exist employees who for some reason do not have any skill determined yet. This means that they are not available to work on any task and their whole working time is considered as idle time. An extreme of this scenario would be that all employees have an empty skillset. Adding any skill to any employee would improve the total idle value since all work done in that scenario is idle work. In this scenario, the underlying workforce optimization solution does not work as intended and the skills should be verified by the store manager before attempting to optimize them.

It must be remembered that there can be differences in employee specific contracts which can result to different outcomes from each scenario. For example, in a situation where an employee has a contract outlining that only daily work can be assigned to them, adding skills for a task that is complete during a night shift does not decrease the idle hours even if there is employee deficit for that task. There are, however, common situations where adding missing skills does not improve results:

- **Abundance of skills**
  When every employee has a large amount of skills in their skillset, they can flexibly be allocated at any time to tasks that require workforce. If a prioritized employee is not available to do a task, almost every other employee can do the job. Adding missing skills in this situation will not improve the idle value

as the existing idle time does not originate from scenarios mentioned earlier. Instead, idle values result from an excessive amount of employees or manually fixed shifts that include idling which can not be modified by the workforce optimization.

- **Low idle to begin with**
  If a planning unit is already well optimized or there exists too much workload for employees to handle, the idle value can already be low to begin with. Adding missing skills to a well optimized solution where the employee skills and the workloads match perfectly can lead to fragmented working shifts where an employee unnecessary switches between tasks. Similarly, employees working to their full capacity will not either benefit from adding missing skills but rather increase the possibility of receiving more fragmented working shifts.

The proposed algorithm only considers which skills are missing and how much adding one of them would improve the solution. A clear weakness in this is that in real world situation every skill can not be trained to anyone and there exists conflicts between skills. For example, an employee with skills regarding keeping the store clean is specialized in cleaning tasks and would in most cases not be interested in the tasks done by a shift manager. Considering which skills are compatible with each other would require manual work for each planning unit as there are no universal rules which apply to every use case. Thus, not considering skill conflicts allows the algorithm to be used on a more general level.

Results of the proposed algorithm give suggestions to model users on which skills should be added to which employees and in what order. The best employee-skill combination in terms of improving the idle value is chosen on each iteration. Often, there are multiple employee-skill combinations per iteration that improve the idle value significantly more than others from which the best one is chosen. The algorithm could be developed to suggest other almost as good employee-skill combinations from which the model users could then choose the most suitable one. This could be done on every iteration so that the model user could see how much an additional skill would bring benefit and make decisions based on that information.

Computing time is considered in this thesis, but no time budget is set. An interesting approach would be to consider a time budget where the algorithm would have a certain time to optimize the results. This would be an important aspect as solution end-users must know and be able to limit the run time. However, restricting run time could lead to suboptimal results in larger planning units as less samples would be evaluated. The former further development suggestion could also restrict the run time by letting the model user stop the process after a sufficient amount of skills are added.

Especially more demanding skills are known to decay over a longer period of time when they are not used (Arthur Jr. et al., 1998) but for easier physical tasks the retention rate is high. In retail it can be assumed that once a skill is learned it is not lost due to the simple characteristics of the tasks. This is assumed in the

workforce optimization literature as well as in this thesis. However, the skills in the underlying workforce optimization solution indicate the tasks an employee can be assigned to and do not single-handedly indicate what the competences of the employee are. Thus, the algorithm could also be improved by adding the ability to identifying redundant skills which could be removed from the employees without negatively impacting optimization results.

6 Conclusions

This thesis proposed a simulation based sensitivity analysis algorithm to improve the results of an existing workforce optimization solution in the retail setting by recommending the training of specific missing skills to employees. The existing solution returns optimized employee schedules whose quality is measured by the amount of idle time. The goal is to reduce the idle time as much as possible to ensure employees are working efficiently since workforce is the largest expense in the retail word. The proposed algorithm outputs employee-skill suggestions to be added along with the amount of idle decrease until no more improvements can be made.

The models behind the solution are complex and the effects on the results when modifying input values are not always apparent. Thus, the proposed algorithm was based on a One-At-a-Time sensitivity analysis method, which was executed multiple times in succession to find the missing skills that improve the results the most. Latin Hypercube Sampling was used to efficiently explore the space of missing employee-skill combinations to mitigate computing times in larger scenarios. The algorithm was implemented in Python programming language, which was executed in a Docker virtual machine environment. The underlying workforce optimization solution is called in the virtual machine with modified test case input files where skills have been added to employees. In this thesis the test cases represented scenarios of stores for which the workforce is optimized. It was assumed that the unmodified test cases portray realistic situations that result in satisfactory schedules.

Results showed large variations in the amount of decreased idle time throughout test scenarios which can be explained by the differences in their properties. In best cases, the idle time was almost completely eliminated while in some cases no decreases could be made. It was more common for a skill to be added multiple times than an employee getting multiple added skills. Based on the disparity of results, different observations were gathered from the starting situations of test cases to help identify situations where adding missing skills improve the results or where they remain unchanged. Although common factors were found, definitive conclusions could not be drawn from test case characteristics alone.

Results also indicated large variations in the computing times of each test case in which almost a thousandfold difference was seen between the fastest and slowest calculations. Although, the computational time was not in the center of focus, it has a large impact on the usability of the algorithm in real life. This could be addressed in further research by adding a budget for the computation time of the algorithm instead of purely limiting the iteration count.

A large assumption made in this thesis is that every employee can be taught each skill with the sole restriction of having a skill only once. The proposed algorithm could be improved by considering relationships between skills as discussed in Section 5 although it would require additional configuration for each planning unit. Alternatively, the algorithm could be structured so that on each iteration multiple alternative choices are presented and one would have to be chosen before proceeding to the next iteration as

opposed to directly calculating until a threshold. This would remove the configuration requirement and give store managers more choices.

References

Winfred Arthur Jr., Winston Bennett Jr., Pamela L. Stanush, and Theresa L. McNelly. Factors that influence skill decay and retention: A quantitative review and analysis. *Human performance*, 11(1):57–101, 1998.

Athanassios N. Avramidis, Wyean Chan, Michel Gendreau, Pierre L'ecuyer, and Ornella Pisacane. Optimizing daily agent scheduling in a multiskill call center. *European Journal of Operational Research*, 200(3):822–832, 2010.

Kenneth R. Baker. Workforce allocation in cyclical scheduling problems: A survey. *Journal of the Operational Research Society*, 27(1):155–167, 1976.

Reinhard Bürgy, Hélène Michon-Lacaze, and Guy Desaulniers. Employee scheduling with short demand perturbations and extensible shifts. *Omega*, 89(1):177–192, 2019.

Howard Hao-Chun Chuang, Rogelio Oliva, and Olga Perdikaki. Traffic-based labor planning in retail stores. *Production and Operations Management*, 25(1):96–113, 2016.

Guillaume Damblin, Mathieu Couplet, and Bertrand Iooss. Numerical studies of space-filling designs: optimization of latin hypercube samples and subprojection properties. *Journal of Simulation*, 7(4):276–289, 2013.

Philippe De Bruecker, Jorne Van den Bergh, Jeroen Beliën, and Erik Demeulemeester. Workforce planning incorporating skills: State of the art. *European Journal of Operational Research*, 243(1):1–16, 2015.

Mieke Defraeye and Inneke Van Nieuwenhuyse. Staffing and scheduling under nonstationary demand for service: A literature review. *Omega*, 58(1):4–25, 2016.

Lisa Disselkamp. *Workforce asset management book of knowledge*. John Wiley & Sons, 1st edition, 2013.

Leslie C. Edie. Traffic delays at toll booths. *Journal of the operations research society of America*, 2(2):107–138, 1954.

Andreas T. Ernst, Houyuan Jiang, Mohan Krishnamoorthy, and David Sier. Staff scheduling and rostering: A review of applications, methods and models. *European journal of operational research*, 153(1):3–27, 2004.

Marshall Fisher, Santiago Gallino, and Serguei Netessine. Retailers are squandering their most potent weapons. *Harvard Business Review*, 97(1):72–79, 2019.

Marshall Fisher, Santiago Gallino, and Serguei Netessine. Setting retail staffing levels: A methodology validated with implementation. *Manufacturing & Service Operations Management*, 2020.

Mika Halme. Data-driven workforce optimization is fueled with accurate forecasts, 2020a. URL https://www.relexsolutions.com/resources/data-driven-workforce-optimization-is-fueled-with-accurate-forecasts/.

Mika Halme. Today's workforce optimization software: Matching tasks, talent, and timing, 2020b. URL https://www.relexsolutions.com/resources/todays-workforce-optimization-software-matching-tasks-talent-and-timing/.

César Augusto Henao, Juan Carlos Muñoz, and Juan Carlos Ferrer. The impact of multi-skilling on personnel scheduling in the service sector: a retail industry case. *Journal of the Operational Research Society*, 66(12):1949–1959, 2015.

Nicholas Hodson, Matthew Egol, and Thom Blischok. Four forces shaping competition in grocery retailing, 2012. URL https://www.strategyand.pwc.com/gx/en/insights/2011-2014/four-forces-shaping-competition-grocery.html.

Charles C. Holt, Franco Modigliani, and Herbert A. Simon. A linear decision rule for production and employment scheduling. *Management Science*, 2(1):1–30, 1955.

Jyotiranjan Hota and Debjani Ghosh. Workforce analytics approach: An emerging trend of workforce management. *AIMS International Journal*, 7(3):167–179, 2013.

C. Jones and K. Nolde. Demand driven employee scheduling for the swiss market, 2013. URL https://infoscience.epfl.ch/record/187141.

W. David Kelton and Russell R. Barton. Experimental design for simulation. In *Winter Simulation Conference*, volume 1, pages 59–65, 2003.

Saravanan Kesavan and Vidya Mani. *An overview of industry practice and empirical research in retail workforce management*, pages 113–145. Springer US, 2015.

Averill M. Law, W. David Kelton, and W. David Kelton. *Simulation modeling and analysis*, volume 5. McGraw-Hill New York, 2014.

Jason L. Loeppky, Jerome Sacks, and William J. Welch. Choosing the sample size of a computer experiment: A practical guide. *Technometrics*, 51(4):366–376, 2009.

Michael D. McKay, Richard J. Beckman, and William J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

Emanuel Melachrinoudis and Michael Olafsson. A scheduling system for supermarket cashiers. *Computers & industrial engineering*, 23(1-4):121–124, 1992.

Nordin H. Mohamad and Fatimah Said. Optimal scheduling of a full-time workforce with sensitivity analysis. *World Applied Sciences Journal*, 24(5):623–627, 2013.

M. Mohammadian, M. Babaei, M. Amin Jarrahi, and E. Anjomrouz. Scheduling nurse shifts using goal programming based on nurse preferences: a case study in an emergency department. *International Journal of Engineering*, 32(7):954–963, 2019.

Claus Adolf Moser and Alan Stuart. An experimental study of quota sampling. *Journal of the Royal Statistical Society. Series A (General)*, 116(4):349–405, 1953.

Marcelo Olivares, Daniel Yung, Victor Bucarey, and Matias Christiansen. Labor planning and shift scheduling in retail stores using customer traffic data. *SSRN Electronic journal*, 2020.

Mohammed Othman, Gerard J. Gouw, and Nadia Bhuiyan. Workforce scheduling: A new model incorporating human factors. *Journal of Industrial Engineering and Management*, 5(2):259–284, 2012.

Emir Hüseyin Özder, Evrencan Özcan, and Tamer Eren. A systematic literature review for personnel scheduling problems. *International Journal of Information Technology & Decision Making*, 19(06):1695–1735, 2020.

Alessandra Parisio and Colin Neil Jones. A two-stage stochastic programming approach to employee scheduling in retail outlets with uncertain demand. *Omega*, 53(1):97–103, 2015.

Siti Noor Asyikinbintimohd Razali, Looi Mei Fen, Norazman Arbin, and Azme Khamis. Integer linear programming on preference maximized of workforce scheduling. *Compusoft*, 7(11):2926–2930, 2018.

Saman Razavi, Anthony Jakeman, Andrea Saltelli, Clémentine Prieur, Bertrand Iooss, Emanuele Borgonovo, Elmar Plischke, Samuele Lo Piano, Takuya Iwanaga, William Becker, et al. The future of sensitivity analysis: an essential discipline for systems modeling and policy support. *Environmental Modelling & Software*, 137 (1):20–44, 2021.

Andrea Saltelli. Sensitivity analysis for importance assessment. *Risk analysis*, 22(3): 579–590, 2002.

Andrea Saltelli and Paola Annoni. How to avoid a perfunctory sensitivity analysis. *Environmental Modelling & Software*, 25(12):1508–1517, 2010.

Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 1st edition, 2008.

Joe Skorupa, Steven Strohecker, and David Spofford. How to win the battle against rising labor costs, 2019. URL https://risnews.com/how-win-battle-against-rising-labor-costs.

Luca Talarico and Pablo Andrés Maya Duque. An optimization algorithm for the workforce management in a retail chain. *Computers & Industrial Engineering*, 82 (1):65–77, 2015.

Jorne Van den Bergh, Jeroen Beliën, Philippe De Bruecker, Erik Demeulemeester, and Liesje De Boeck. Personnel scheduling: A literature review. *European journal of operational research*, 226(3):367–385, 2013.

Joan C. Williams, Saravanan Kesavan, and Lisa McCorkell. When retail workers have stable schedules, sales and productivity go up, 2018. URL https://hbr.org/2018/03/research-when-retail-workers-have-stable-schedules-sales-and-productivity-go-up.

Lishun Zeng, Mingyu Zhao, and Yangfan Liu. Airport ground workforce planning with hierarchical skills: a new formulation and branch-and-price approach. *Annals of Operations Research*, 275(1):245–258, 2019.

A Appendix

Table A1: Results of the MOAT analysis for each test case. For each iteration, the best skill-employee combination is shown with how much it improves the idle value.

Test case	Iteration	1	2	3	4	5	6	7
1	Skill id	1	-	-	-	-	-	-
	Employee number	60	-	-	-	-	-	-
	Idle value decrease	17	-	-	-	-	-	-
2	Skill id	4	11	10	11	10	4	1
	Employee number	34	24	43	10	3	25	2
	Idle value decrease	1175	1103	1025	396	260	141	28
3	Skill id	1	1	1	1	2	-	-
	Employee number	17	26	24	29	21	-	-
	Idle value decrease	1445	880	222	82	26	-	-
4	Skill id	4	4	2	6	-	-	-
	Employee number	28	57	33	36	-	-	-
	Idle value decrease	1154	487	279	111	-	-	-
5	Skill id	2	2	6	2	2	2	-
	Employee number	47	46	6	5	13	19	-
	Idle value decrease	662	553	121	463	245	21	-
6	Skill id	7	2	-	-	-	-	-
	Employee number	69	69	-	-	-	-	-
	Idle value decrease	325	38	-	-	-	-	-
7	Skill id	2	2	2	1	1	-	-
	Employee number	6	7	4	6	10	-	-
	Idle value decrease	1918	1392	484	452	39	-	-
8	Skill id	10	7	-	-	-	-	-
	Employee number	69	69	-	-	-	-	-
	Idle value decrease	23	40	-	-	-	-	-
9	Skill id	4	-	-	-	-	-	-
	Employee number	11	-	-	-	-	-	-
	Idle value decrease	232	-	-	-	-	-	-
10	Skill id	2	2	1	1	-	-	-
	Employee number	9	1	4	1	-	-	-
	Idle value decrease	1252	285	153	76	-	-	-
11	Skill id	9	12	11	4	2	8	-
	Employee number	88	74	73	46	90	53	-
	Idle value decrease	1104	667	556	498	94	62	-
12	Skill id	1	-	-	-	-	-	-
	Employee number	9	-	-	-	-	-	-
	Idle value decrease	0	-	-	-	-	-	-

Table A1 continued from previous page

Test case	Iteration	1	2	3	4	5	6	7
	Skill id	1	-	-	-	-	-	-
13	Employee number	2	-	-	-	-	-	-
	Idle value decrease	0	-	-	-	-	-	-
	Skill id	1	-	-	-	-	-	-
14	Employee number	7	-	-	-	-	-	-
	Idle value decrease	0	-	-	-	-	-	-
	Skill id	9	-	-	-	-	-	-
15	Employee number	24	-	-	-	-	-	-
	Idle value decrease	28	-	-	-	-	-	-
	Skill id	1	1	2	2	-	-	-
16	Employee number	3	4	6	7	-	-	-
	Idle value decrease	750	750	610	40	-	-	-
	Skill id	6	6	-	-	-	-	-
17	Employee number	15	10	-	-	-	-	-
	Idle value decrease	117	70	-	-	-	-	-
	Skill id	12	-	-	-	-	-	-
18	Employee number	24	-	-	-	-	-	-
	Idle value decrease	38	-	-	-	-	-	-
	Skill id	4	5	7	-	-	-	-
19	Employee number	11	13	9	-	-	-	-
	Idle value decrease	2367	94	30	-	-	-	-
	Skill id	3	-	-	-	-	-	-
20	Employee number	27	-	-	-	-	-	-
	Idle value decrease	58	-	-	-	-	-	-

Table A2: Idle value results from executing the MOAT algorithm version A.

Test case	Starting idle	End idle	Decrease	Decrease (%)
1	4419	4398	21	0.5%
2	12882	8754	4128	32.0%
3	2663	8	2655	99.7%
4	18277	16245	2032	11.1%
5	12278	10213	2065	16.8%
6	7522	7181	341	4.5%
7	11728	7443	4285	36.5%
8	7442	7379	63	0.8%
9	3114	2876	238	7.6%
10	4416	2645	1771	40.1%
11	9202	6221	2981	32.4%
12	2351	2351	0	0.0%
13	15888	15888	0	0.0%
14	2384	2384	0	0.0%
15	2961	2927	34	1.1%
16	6000	3836	2164	36.1%
17	15543	15373	170	1.1%
18	2963.4	2935	28.4	1.0%
19	3249.5	751	2498.5	76.9%
20	666	608	58	8.7%

Table A3: Score value results from executing the MOAT algorithm version A.

Test case	Starting score	End score	Improvement	Improvement (%)
1	-76257	-74886	1371	1.8%
2	-60798	-49973	10825	17.8%
3	-42167	-33712	8455	20.1%
4	-136990	-113472	23518	17.2%
5	-485797	-440134	45663	9.4%
6	-12877600	-12852831	24769	0.2%
7	-10926444	-4421091	6505353	59.5%
8	-12876492	-12874655	1837	0.0%
9	-476574	-470955	5619	1.2%
10	-566978	-350458	216520	38.2%
11	-88806	-82307	6499	7.3%
12	-136751	-136750	1	0.0%
13	-2624186	-2624185	1	0.0%
14	-159118	-159107	11	0.0%
15	-66916	-64949	1967	2.9%
16	-5415994	-744828	4671166	86.2%
17	-620912	-605355	15557	2.5%
18	-66834	-64665	2169	3.2%
19	-29637	-19590	10047	33.9%
20	-18669	-17039	1630	8.7%

Table A4: Score value results from executing the MOAT algorithm version B.

Test case	Starting score	End score	Improvement	Improvement (%)
1	-76257	-74887	1370	1.8%
2	-60798	-49822	10976	18.1%
3	-42167	-33713	8454	20.0%
4	-136990	-112472	24518	17.9%
5	-485796.7	-448589	37208	7.7%
6	-12877600.2	-12845007	32593	0.3%
7	-10926444.2	-4393969	6532475	59.8%
8	-12876491.6	-12875054	1438	0.0%
9	-476573.9	-470955	5619	1.2%
10	-566978.4	-349286	217692	38.4%
11	-88806.1	-82611	6195	7.0%
12	-136750.6	-136729	22	0.0%
13	-2624185.5	-2624183	3	0.0%
14	-159117.9	-159118	0	0.0%
15	-66915.9	-64901	2015	3.0%
16	-5415993.8	-744828	4671166	86.2%
17	-620819.1	-605870	14949	2.4%
18	-66835.5	-65031	1805	2.7%
19	-29487.2	-18937	10550	35.8%
20	-18668.9	-17104	1565	8.4%

Table A5: Idle value results from executing the MOAT algorithm version B.

Test case	Starting idle	End idle	Decrease	Decrease (%)
1	4419	4400	19	0.4%
2	12882	8754	4128	32.0%
3	2663	91	2572	96.6%
4	18277	16186	2091	11.4%
5	12278	10894	1384	11.3%
6	7522	7046	476	6.3%
7	11728	7397	4331	36.9%
8	7442	7437	5	0.1%
9	3114	2876	238	7.6%
10	4416	2636	1780	40.3%
11	9202	6314	2888	31.4%
12	2351	2351	0	0.0%
13	15888	15888	0	0.0%
14	2384	2384	0	0.0%
15	2961	2930	31	1.0%
16	6000	3836	2164	36.1%
17	15541.8	15405	136.8	0.9%
18	2964.3	2924	40.3	1.4%
19	3240.2	771	2469.2	76.2%
20	666	608	58	8.7%

Table A6: Baseline statistics for each test case.

Test case	Employee count	Skills count	Skills per employee	Planning period (Days)	Workload hours	Idle hours	Workload ratio (ticks)	Workload ratio per day (ticks)	Workload hours per idle hours
1	65	10	4.6	182	1105	44908	1193	6.6	38.7
2	91	12	0.9	182	3221	25350	1788	9.8	7.7
3	30	5	1.0	182	666	24892	3645	20.0	35.8
4	86	11	2.4	182	4569	22679	1392	7.6	4.9
5	60	10	3.9	168	3070	9833	493	2.9	3.3
6	80	10	4.8	168	1881	8560	503	3.0	4.4
7	14	2	0.00	126	2932	1071	4285	34.0	0.4
8	80	10	4.8	168	1861	8560	503	3.0	4.5
9	19	5	2.5	126	779	3216	193	1.5	4.1
10	15	3	0.6	154	1104	3419	2129	13.8	3.0
11	112	14	0.9	154	2301	28039	2049	13.3	11.4
12	14	5	3.3	119	588	3502	294	2.5	7.0
13	25	5	1.6	119	3972	4101	373	3.1	1.0
14	14	5	3.3	119	596	3502	294	2.5	6.9
15	33	14	2.0	119	740	7484	1158	9.7	9.9
16	10	2	0.2	35	1500	918	2043	58.4	0.6
17	78	10	0.9	42	3886	4784	221	5.3	1.2
18	33	14	2.9	119	741	7484	1158	9.7	9.9
19	30	14	2.1	126	812	7543	1226	9.7	8.9
20	27	4	1.0	35	167	3895	566	16.2	23.4