Solving a Flexible Resource-Constrained Project Scheduling Problem for an e-grocery fulfilment centre: a meta-heuristic approach

Master thesis Aerospace Engineering

J.B. van Teeffelen

TU Delft

Solving a Flexible Resource-Constrained Project Scheduling Problem for an e-grocery fulfilment centre: a meta-heuristic approach

Master thesis Aerospace Engineering

by

J.B. van Teeffelen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on December 19, 2022.

Student number:	4484290	
Project duration:	March 2022 – December 2022	
Thesis committee:	Dr. M.D. (Marilena) Pavel	TU Delft, Chair
	Dr. A. (Alessandro) Bombelli	TU Delft, Supervisor
	Dr. M. (Márcia) Baptista	TU Delft, Examiner
	Ir. F. (Floris) Boekema	Picnic, Supervisor

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Acknowledgements

Hereby, I present my final project to obtain my master's degree at the faculty of Aerospace Engineering of the Delft University of Technology. Before moving to Delft in 2016, I had never imagined what an incredible journey was coming ahead. Starting as a bachelor's student in Mechanical Engineering, I am glad that the transition to obtaining an Aerospace degree went smoothly despite events such as COVID-19. I am thrilled to explore the opportunities and possibilities that come next. During my thesis project at Picnic Technologies, I am honoured and thankful to gain experience in a young, fast-paced, professional, and inspiring environment. Yet, I am even more elevated about the people who helped me come this far. Before I encourage you to read through this report, I have to thank a couple of people.

First, I would like to express my gratitude to my supervisor Alessandro. Your optimism, enthusiasm, and critical assessments persuaded me to pursue my work every week and bring my research to a higher level. Especially the weekly meetings at the Aerospace faculty inspired me by your expertise in operations optimisation focused on exact models and meta-heuristics. Next, I want to address the contribution of Paul Roling during the kick-off, mid-term and green-light meetings. Your critical outside-in assessments were precious and resulted in relevant decisions throughout the research. I also like to thank Marilena Pavel and Márcia Baptista from my thesis committee for facilitating a graduation date before the end of the year. Last, I want to thank Neil York Smith for his intelligence in scheduling and programming at the start of my research.

At Picnic, I have to thank two people in particular who helped me every week during my project. First, Floris, your utmost precision and extensive knowledge of the subject helped me to gain new insights and to take a step back to analyse the work from time to time. Next, Max, your inexhaustible enthusiasm and knowledge of both scheduling & programming encouraged me to push forward. The unique group of people in the FC analyst team, which all three of us were part of, helped me to feel part of the Picnic Team. Therefore, I want to thank the whole team for all our experiences, meetings and fun at the "borrels" together. Furthermore, I want to highlight two other persons, Olivier and Merel, with whom I have become better acquainted during my master's through every project we did together. Olivier, for your enthusiasm, perseverance and resilience during study periods. Merel, thank you for your optimism, words of advice, critical judgements, and the fun project work together!

Finally, I want to express my gratitude to my family and Maartje for their unconditional support. I could only achieve this milestone with the many words of wisdom provided on any day matters by them. Next, I want to thank my housemates with whom I lived at the Westerstraat, the Mathenesserlaan, and those I met in Delft. Without them, my student time should not have been the same.

Jop van Teeffelen
Rotterdam, December 2022

Contents

List of Figures vii

List of Tables ix

List of Abbreviations xi

Introduction xiii

I Scientific Paper 1

II Literature Study
previously graded under AE4020 23

III Supporting work 71

1 Picnic Case Study 73
1.1 List of Tasks with Descriptions 73
1.2 Input Data Sheets 75

2 Used Software And Extensions 77
2.1 Numba 77
2.2 Tableau 78

Bibliography 81

List of Figures

2.1 Performance of machine code versus non-machine code 77

List of Tables

1.1 Overview of clocking activities 75
1.2 Precedence relationships 75
1.3 FC capacity characteristics 76

2.1 Data formatting for Tableau 78

List of Abbreviations

B.I.	Best Incumbent
B.S.	Best Solution
B2C	Business to Customer
B&B	Branch & Bound
B&C	Branch & Cut
C.T.H	Computational Time Heuristic
Confid.	Confidential
CTG	Computational Time Gurobi
DC	Distribution Centre
DV	Decision Variables
DWH	Data Warehouse
EPV	Electronic Picnic Vehicle
FC	Fulfilment Centre
FRCPSP	Flexible Resource Constrained Project Scheduling Problem
JSSP	Job Shop Scheduling Problem
KPI	Key Performance Indicator
LB	Lower Bound
MILP	Mixed Integer Linear Programming
N.A.	Not Available
NP	Non-deterministic Polynomial-time
RCPSP	Resource Constrained Project Scheduling Problem
T.T.B.	Time To Best
VNS	Variable Neighborhood Search
WMS	Warehouse Management System
WPS	Workload Planning System

Introduction

E-commerce was growing fast before COVID-19 hit. During the COVID-19 pandemic, the demand for online shopping grew tremendously. An inefficient supply chain and some inefficient operations have been the biggest obstacles to expansion in the e-grocery sector, frequently resulting in capital being expended on operating costs. E-grocery is business-to-consumer e-commerce for the primary purpose of selling groceries online. As a result, a clear and well-developed operational supply chain is required for sustainable and scalable growth. Picnic, an e-grocery retailer, has been disrupting the e-grocery market since 2015. Their efficient operations and data-driven approach cause large cost savings to be compatible with other major grocery organisations. One of the critical drivers to maintaining its competitive position in the market is efficiency.

Matching workload (anticipated work) with the workforce (employees) is a significant driver for efficient operations. Regulations commit Picnic to scheduling people to a specific shift a week before the working day starts. However, the number of workers necessary for the shift is known only one day upfront. The discrepancy between the availability of people and the forecasted amount of work converges to an ideal mix of workload and workforce defined that day. Yet, fluctuations frequently cause a mismatch between workforce and workload, resulting in inefficiencies. The current planning process heavily relies on precedence relationships, break management systems, dynamic start- and end times, and other regulations.

Picnic is not able to monitor the performance of its current planning and scheduling process. Therefore, the research focuses on solving a task allocation algorithm within Picnic's fulfilment and distribution centres. Both an exact and a meta-heuristic model have been constructed to study the task allocation process that can be solved efficiently. Historical data from Picnic is used to set up a realistic model. For the exact model, a Branch & Cut algorithm is used to solve a Mixed Integer Linear Programming formulation with lexicographical objectives. The proposed model is compared to a meta-heuristic Variable Neighbourhood Search, which uses nested neighbourhood loops to modify the solution. The research aims to implement the algorithms in a real-life FC to test the performance, increase productivity, and save costs.

The research is carried out as part of a Master's thesis in Air Transport Operations from the faculty of Aerospace Engineering at the Delft University of Technology. This thesis report has the following structure. Part I presents the scientific paper of the research. After that, Part II contains a report of the Literature Study, providing background information on e-groceries, planning and scheduling, and different exact and heuristic solution methods proposed in related work. Finally, Part III provides supporting work for the research presented in the scientific paper. All definitions of the activities are mentioned, as well as a short description of the extensions used named Numba and Tableau, respectively.

I

Scientific Paper

Solving a flexible resource-constrained project scheduling problem for an e-grocery fulfilment centre: a meta-heuristic approach

Jop van Teeffelen*

Delft University of Technology, Delft, The Netherlands

Abstract

The demand for online shopping has grown tremendously in the last couple of years. Picnic, a major player in the online grocery industry, is struggling to achieve long-term growth within its current operations. Scheduling and planning are key drivers for maintaining operational efficiency. The Fulfilment Centre (FC) and Distribution Centre (DC) costs heavily depend on efficient operations. This research focuses on improving the scheduling process in an e-grocery FC and DC. The main objective of the model is to maximise the quality of the schedule, which is achieved through two lexicographical objectives. First, the make span is minimised to improve efficiency and to calculate the number of employees required to fulfil the workload. The make span of a schedule is defined as the time between the first scheduled activity i and the last activity j in a schedule s. Next, the number of switches between activities is reduced. The second objective is to increase overall productivity since switching moments cause slack in the operations. Two solution methods are proposed to solve the Flexible Resource Constrained Project Scheduling Problem (FRCPSP). The first solution method is a Mixed Integer Linear Programming (MILP) formulation that is solved with a Branch & Cut (B&C) algorithm. Next, a meta-heuristic is proposed named Variable Neighbourhood Search (VNS). The initial solution is computed by solving the MILP for one objective, minimising the make span. Next, the VNS uses nested neighbourhoods to modify the answer resulting in fewer switches per schedule. For small instances, the exact formulation outperforms the meta-heuristic in most cases. Conversely, the meta-heuristic features a higher efficacy and efficiency when tackling more significant instances, being the only solution method capable of yielding feasible solutions for real-world scheduling problems. Despite the effectiveness of the proposed meta-heuristic, some operational adjustments are still required before implementing the proposed decision-making tool.

Keywords: *Project Scheduling, Job Shop Scheduling Problems, E-grocery, Flexible Resource Constrained Project Scheduling Problem, Meta-heuristic, Variable Neighbourhood Search*

1 Introduction

E-commerce was growing fast before COVID-19 hit. During the COVID-19 pandemic, the demand for online shopping grew even more substantial. Furthermore, changes in customer behaviour and consumer awareness of the need to find safe shopping alternatives resulted in a 55% growth in 2020, up from 10% in 2019 [McKinsey, 2022]. The largest stumbling block for growth in the e-grocery industry has been an inefficient supply chain and inefficient operations, which frequently led to capital being used up on operating expenses [Nicolo Galante et al., 2013]. E-grocery is B2C e-commerce for the primary purpose of selling groceries online. As a result, a clear and well-developed operational supply chain is required for sustainable and scalable growth.

Today, customers tend to have more requirements and demands from online retailers. Customers want delivery within 24 hours and a reliable return policy. Delivering groceries within one day comes with challenges. All products should be in stock and have to be fresh, and no damages or completeness issues should occur during the operations. As a result, enough workforce (employees) should be in place to process the workload (hours) for delivery the next day.

The online supermarket Picnic, which operates in the Netherlands, Belgium, and France, is one example of a company looking to improve its supply chain. Picnic's supply chain starts with the supplier, who delivers its products to the Distribution Centres (DC). A large warehouse for the storage of products. Next, the products are processed, subdivided, and transferred to regional Fulfilment Centres (FC). An FC replaces the physical grocery shop so online orders can be received, processed, and filled, ready for transportation to the hubs. Subsequently, the products are transferred to local hubs, where the groceries are delivered to someone's doorstep. An illustrative example of the supply chain is shown in Figure 1. Picnic offers customers the possibility to order groceries one day upfront and promises to deliver the products within a 20-minute time frame the next day. On the other hand, processing the entire operation from the supplier to a customer's doorstep takes about

*Msc Student, Sustainable Air Transport, Faculty of Aerospace Engineering, Delft University of Technology

24 hours. Their Just-in-Time (JIT) service significantly depends on a precise prognosis and comprehensive planning techniques.

As potential orders are abundant at Picnic, it is desired to schedule tasks among employees efficiently to serve more customers. The allocation of charges among employees in an FC is a very time-consuming process and is currently done by a team captain (an experienced employee with a leadership role). The quality of the timetables heavily relies on the expertise of a captain, which bases their decisions on personal preferences and experience. A job allocation algorithm will automatically resolve the time-consuming procedure. The algorithm provides the captain with optimal schedules, which can be used as a benchmark for last-moment adjustments.

In addition, higher productivity and efficient scheduling can result in significant cost savings. The increase in productivity directly decreases the number of employees required. Additionally, more clients may be serviced by processing more orders.

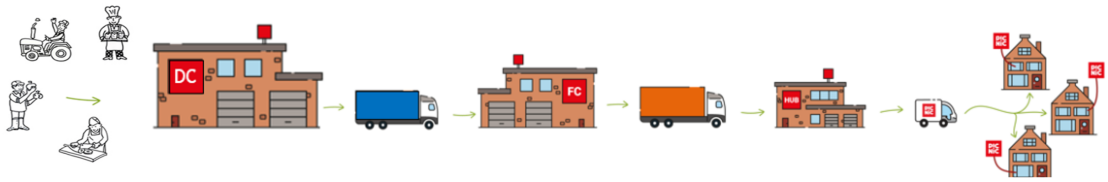


Figure 1: Illustrative example of the supply chain of Picnic

Regulations commit Picnic to schedule people a week before the working day starts. However, the number of employees required is known one day in advance. The discrepancy between the availability of people and the future amount of work converges to an ideal mix of workload and workforce defined that day. Nevertheless, fluctuations frequently cause a mismatch between workforce and workload, resulting in inefficiencies. The mismatch might therefore lead to operational constraints.

The research objective is to improve the current scheduling process by optimising a task allocation algorithm in an e-grocery fulfilment centre while maintaining productivity. Specifically, the research aims to optimise the trade-off between constrained workforce and workload. The created decision tool will be used as fundamental for allocating employees to specific tasks required that day.

The problem definition is formulated as a Flexible Resource Constrained Project Scheduling Problem (FRCPSP). Two ways of solving the FRCPSP are proposed in this research. First, the FRCPSP is mathematically formulated as a Mixed Integer Linear Problem (MILP) and solved through a Branch & Cut (B&C) algorithm to retrieve an exact solution. Next, VNS is used as a meta-heuristic to solve the FRCPSP for a real-time data instance.

The paper has the following structure. First, related work on planning and scheduling is elaborated in Section 2. Next, the problem description is discussed briefly in Section 3. Subsequently, the methodology, research setting, and mathematical model are elaborated in Section 4. Next, a proposed meta-heuristic is further elaborated in Section 5. Section 6 presents the MILP and the meta-heuristic results. Finally, the conclusion and discussion are discussed in Section 7

2 Literature Review

The topic of job scheduling and task allocation is already a common subject for many applications in published literature. Therefore, it is crucial to consider what has already been studied and where new research can add value to the academic body. First, a brief description of the Job Shop Scheduling Problem (JSSP) is provided in Subsection 2.1. Subsequently, related work of the FRCPSP is elaborated in Subsection 2.2. Subsequently, exact solution techniques are described in Subsection 2.3. Finally, more information on the VNS is provided in Subsection 2.4.

2.1 Job Shop Scheduling Problem

One of the oldest and most essential studies in scheduling is JSSP [Asadzadeh, 2015]. A JSSP is an optimisation problem mainly addressed in operations research and computer science to schedule jobs among machines or employees. JSSPs are considered NP-hard problems. In other words, among one of the most challenging combinatorial optimisation problems to solve [Christodoulos A. Floudas and Panos M. Pardalos, 2019, Asadzadeh, 2015, Leusin et al., 2018, Applegate and Cook, 1991, Ham and Cakici, 2016]. Informally, the problem can be described as follows: there are a set of jobs and a set of machines. Each job consists of a chain of operations that must be processed during an uninterrupted period of a given length per machine. Each machine can only process at most one operation at a time. A schedule allocates tasks to specific time intervals on different machines.

The objective of JSSP is to find a schedule with a minimum make span to complete the operations. JSSP has inherent intractability characteristics, resulting in a preference for solving the problem with a meta-heuristic procedure [Cheng et al., 1996].

One of the main limitations of a JSSP is based on the resource consumption of tasks where the capacity of resources is unitary. In other words, one machine can only process one task simultaneously. Additionally, one task (operation) requires only one resource (employee) to be fulfilled. In this study, one task (e.g. picking) has to be completed by many employees during the same time interval. Next, precedence constraints form chains within a JSSP algorithm, and the preemption of tasks is not incorporated within the classic JSSPs. Finally, the number of employees (resources) in an FC is constrained for which the JSSP does not hold. As a result, an alternative to the JSSP should be formulated.

2.2 Flexible Resource Constrained Project Scheduling Problem

Another way of addressing the problem description is by formulating an RCPSP [Habibi et al., 2018]. An RCPSP determines the time required to implement the tasks of a project to achieve a specific objective. Classical RCPSP have two restrictions within its formulation. First, precedence relations indicate that some tasks must end before another can start. Second, resource constraints are formulated where renewable resources are available at a constant and fixed rate throughout the project. One of the most common additional restrictions is that some tasks must end before a given due date and time [Rajeev et al., 2015]. The origin of RCPSP is referred to as a JSSP and was first formulated by J. Blazwicz & J. Lenstra in Blazewicz et al. [1983]. Hence, the JSSP and the RCPSP are also defined as NP-hard problems, indicating their difficulty in resolving the mathematical formulation.

RSPCP formulations are considered by researchers as one of the most commonly used and fundamental issues, addressing the urge and the quantity of research in the planning domain [Habibi et al., 2018, Blazewicz et al., 1983, Rajeev et al., 2015]. Many variations on the classic RCPSP are possible as many different problems can be categorised accordingly. Literature about RCPSP problems incorporating information on renewable resources, Preemption of tasks, time-based & multi-objective economic objective functions and a non-deterministic approach offers various insights to formulate the problem mathematically. Naber and Kolisch [2014] discusses the definition of a FRCPSP. Here, they simultaneously determine each activity's start time and duration while minimising the make span. The model is subject to precedence relationships, limited availability of multiple resources, and resource profile constraints.

2.3 Exact Solution Techniques

Optimisation methods can be classified into two types. First, an exact solution method ensures that an optimal solution is found, by solving a MILP for example. Second, a heuristic can be defined to approximate the optimal answer. The advantage of a well-developed heuristic is that it reduces the computational time required to obtain a high-quality approximation to the optimal answer.

Two different exact solution methods are frequently elaborated in literature for scheduling and planning problems. Respectively, Branch & Bound (B&B) and B&C. The theory on B&B is commonly used to solve JSSPs and RCPSPs. The classic B&B algorithm explores a set of candidate solutions defined within a rooted tree. The algorithm decides which branch improves the main objective function and explores every possibility to achieve the optimal solution [Morrison et al., 2016]. Christofides proposes a B&B algorithm based on the idea of using disjunctive arcs. The algorithm is used for resolving conflicts created whenever sets of tasks have to be scheduled whose total resource requirements exceed the availability in some periods [Christofides et al., 1987]. The computational time of a B&B algorithm is quite significant since all solutions need to be compiled to find an optimal solution.

Padberg also proposes a B&C algorithm applied to the B&B logic with the implementation of cutting planes within a classic Travelling Salesman Problem [Padberg and Rinaldi, 1991]. Cutting planes are constraints that can be added to an integer program to tighten the feasible region without removing integer solutions. Eventually, this technique is often used in literature to solve RSPCP definitions with remarkably better results than a B&B algorithm [Zhu et al., 2006, Chakrabortty et al., 2015]. Therefore, a B&C algorithm is proposed to solve the FRCPSP in this research.

2.4 Variable Neighbourhood Search

The typical complexity and size of the RCPSP cause the exact solution to be suitable for smaller data instances resulting in a reasonable computational time. However, because it takes much time to compute, it is frequently preferred to use meta-heuristics, as meta-heuristics tend to solve the problem with a large data instance relatively fast while still aiming for a high-quality answer.

VNS is a meta-heuristic based on specific local search algorithms made available by Mladenovir and Hansen't' [1997]. The combination of different local search algorithms as a subroutine fungate as an efficient heuristic. A set of different neighbourhoods (N_k with $k = 0, ..., 3$) presenting different local searches is defined in which every neighbourhood consists of a local search heuristic. Local Search heuristics usually uses only one neighbourhood to resolve the problem and come to an answer. The VNS iterates through a nested set of neighbourhoods until an improved solution is found. Contrary to most other local search methods, VNS does not follow a trajectory but explores increasingly distant neighbourhoods of the incumbent solution and jumps to a new incumbent solution whenever an improvement is made. After an improvement is initiated, the VNS starts at the first neighbourhood again. Various stopping criteria (e.g. maximum number of iterations, maximum CPU time, or a maximum number of iterations between two improvements) could be formulated to terminate the heuristic. The basic version of a VNS is a descent, first improvement method. The success of a VNS is because each neighbourhood has a different local optimum

2.5 Research Gap

The contributions of this work are three-fold. First, the meta-heuristic proposed to solve the FRCPSP allows for non-preemptive scheduling. Regulations require employees to take a break after a consecutive number of hours. The break management constraints directly cause the MILP and meta-heuristic to comply with non-preemptive scheduling. Non-preemptive scheduling enables employees to stop their activity before completing the workload. As a result, employees can switch to another task or break before the complete activity is fulfilled. Second, as far as we could corroborate, no existing work captures the decision-making tool for creating operational schedules in an e-grocery FC. The operation's complexity and the symmetry caused by employees permitted to perform all activities create a one-of-a-kind problem. Third, the availability and quality of the data are unique. Picnic's Data Warehouse (DWH) is enriched with data from all processes. The DWH is a cloud-based data vault with information on historical processes, HR, and other events in Picnic's daily operations. As a result, any form of stochastic modelling is not required to obtain efficient and robust timetables in an FC.

3 Problem Statement

The goal of this research is to improve the planning process in an e-grocery FC & DC. The current planning process, particular difficulties, and the future planning process are all covered in more detail within this section. First, Subsection 3.1 elaborates on the current planning process alongside its limitations. Subsequently, Subsection 3.2 elaborates on the future workload planning system. Furthermore, a deep dive into challenges around scheduling is discussed in Subsection 3.3. Finally, a brief explanation of the issues raised by precedence relations and break management systems is expanded in Subsection 3.5. Finally, the input data, the schedule matrix and the assumptions are discussed in Subsection 3.7 - 3.9.

3.1 Current Planning Process

Today, Picnic's forecasting team plays a significant role in planning. Historical data, trends in economics, and other factors influence the demand, expressed in an exact number of orders. Subsequently, orders can be translated into the number of products that should be processed to meet the required demand. The organisation is aware of each employee's productivity for all jobs thanks to clocking data. All historical data is cleaned and stored within the Data Warehouse (DWH). The number of hours required for a shift may be calculated by combining the productivity and the total number of products handled. As a result, the absolute number of necessary employees can be computed.

Allocating employees to activities is done through FC captains a day before the shift starts. The allocation is based on personal preferences and the experience of the team captain. Employees are allocated to one activity for which they become responsible during the shift. The task allocation process is done manually and it takes up to 2 hours to create a schedule for one shift. The current decentralised method of planning causes a lot of inefficiencies and discrepancies between FCs. Because of this, it is challenging to evaluate or compare the effectiveness of the scheduling process within different FCs. Additionally, the current method heavily relies on the expertise of a team captain which is therefore sensitive to human errors.

To meet all deadlines, team captains keep an eye on productivity and other factors during shifts. Based on numerous operational constraints, employees are moved from one activity to another during a shift. The time to change locations and routine adds up in spilled time and thus slack.

3.2 Future planning Process

Picnic is currently creating a new centralised method, named the Workload Planning System (WPS). The WPS is defined within four steps and illustrated in Figure 2. First, workload packages are created based on

information from the forecast data two weeks before the operations start. The growth team forecasts demand for various products and calculate the hours required per activity two weeks before the operational day.

Second, the workload packages are used as input for scheduling the required workforce to shifts. Employees can hand in their availability nine days before a day starts (2a). Next, Employees must be scheduled a week in advance (2b), but exact numbers are only available 24 hours before the operating day. The people team within Picnic operates an algorithm matching the workforce's demand and supply per shift.

Third, workforce has to be time-aligned with the workload. A task allocation algorithm should assign specific tasks to specific time intervals based on deadlines, precedence relationships and break management constraints. The task allocation is purely based on quantitative operational data and not dependent on qualitative personal preferences.

Fourth, the timetables should be assigned to the available shoppers. Capabilities per employee play a role while assigning employees to specific timetables. However, this step is not covered in the scope.

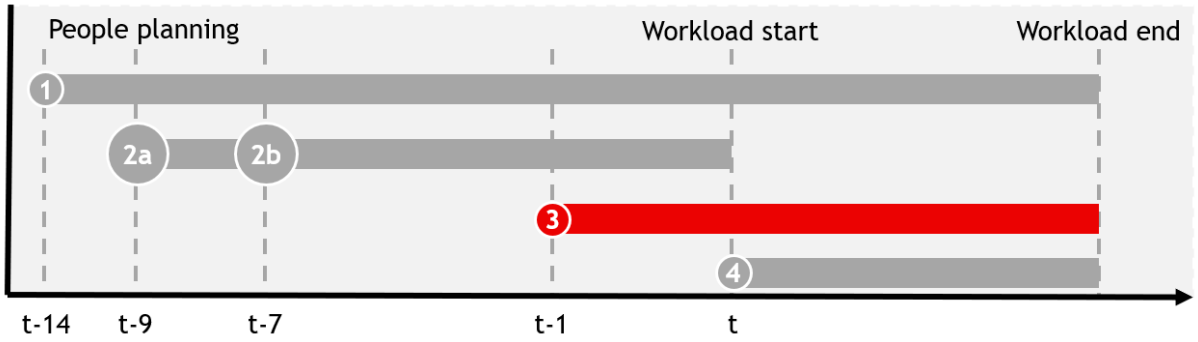


Figure 2: Timeline workload planning system in days

3.3 Scheduling

Schedules are projections or forecasts of future conditions and events for the project. They are based on information and expertise known at the time the schedules were created. Unexpected occurrences or irregularities after the schedule has been made directly impact the result. For this reason, a robust schedule should be defined to increase its flexibility of the schedule. Robust schedules can handle different inconsistencies or slack without creating a new one. The robustness problem is resolved by minimising the make span of the schedules. The workload can be scheduled to be fulfilled before the 9-hour duration of the shift. Slack or irregularities can be compensated with the bonus time made available through the extra time. In other words, bonus time addresses time that can be used for any other activity before scheduled before the shift ends.

Responsively, work can be exchanged among shifts to compensate for the availability of the workforce. An FC's primary tasks have strict time constraints. They can, therefore, only occasionally be switched during the day. The procedures only succeed if crucial tasks are completed within the suggested start- and end times. Trucks must leave at a specific time and packed fulfilled orders must be moved to the appropriate hubs. Performance indicators should clarify whether an FC is under- or overplanned so that the specific tasks to be shifted can be determined.

Another obstacle is the percentage of no-shows in an FC's current planning process. The churn rate of employees working in a DC or FC is remarkably high. The no-show rate for a shift might occasionally reach higher numbers, especially among new employees. Scheduling the workload very tightly might cause issues with the no-show rate.

In current operations, FCs are adjusting work from shifts as a reaction to the no-show rate. For instance, low-priority tasks can be completed at different times of the day. Therefore, schedules need to be reliable, adaptable to last-minute modifications, and flexible to the available workload's limitations.

3.4 Precedence Relationships

Sequential activities add complexity to the planning process of an FC. Various activities depend entirely or linearly on another predecessor. Figure 3 illustrates an example of the operations within an FC. For example, an order can contain different products from different suppliers (trucks). The trucks transporting the different goods will have different arrival times. In contrast, the picking procedure cannot begin until both shelves have been restocked with the products from the trucks. The picking can only be completed after all products from both trucks have been replenished. This is an example of full precedence relations between two successive activities; replenishment and picking. Next, the dispatching activity has to be linearly executed with picking.

When a pick round is completed, the crates from the filled pick cars have to be lifted in the dispatching frames. Dispatching can only be executed after the corresponding proportion of picking has been fulfilled. This is an example of linear precedence relations between two successive activities; picking and dispatching.

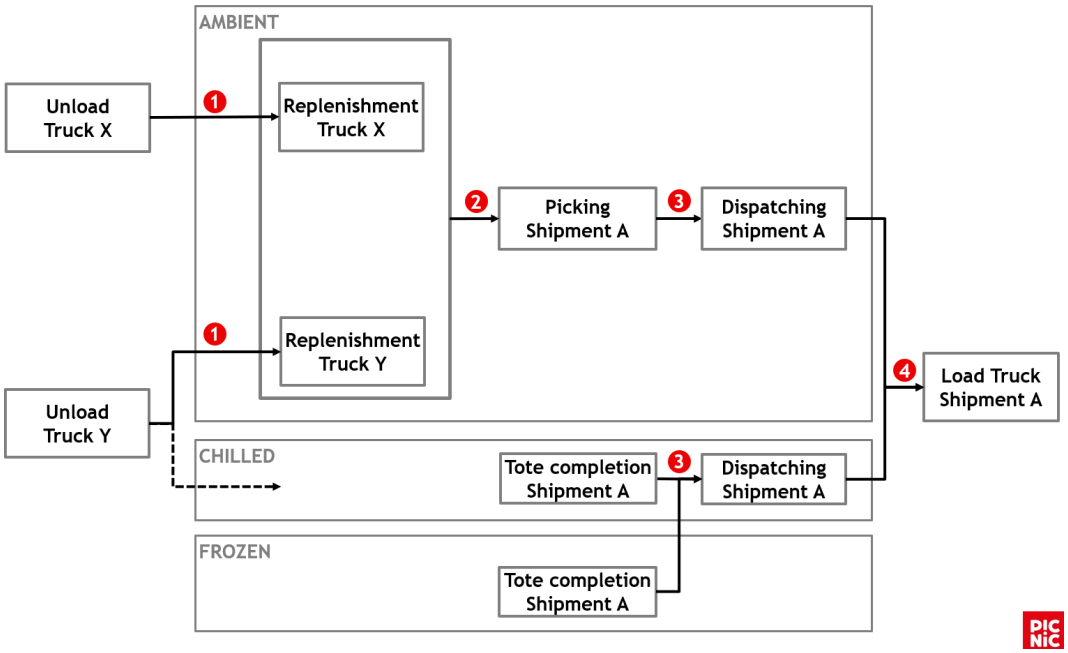


Figure 3: Sequential relations between tasks in a fulfilment centre [Picnic, 2022]

3.5 Break Management System

Employees must take a break after executing work for a specific number of hours. Government laws specify the required length of a break dependent on the length of a shift. Today, Picnic is operating within a two-shift structure. The first shift starts at 06.00 in the morning with a nine-hour duration until 3.00 in the afternoon. The second shift starts subsequently at 3.00 in the afternoon and until midnight. There are three planned breaks for each shift. After almost two hours, the first fifteen-minute break is scheduled. Employees should split break times among themselves so that the number of hours available for work remains high. Next is a 30-minute lunch break, during which the canteen can only accommodate employees to a certain extent. The last break is planned two hours before the shift's finish time.

A strictly planned timetable directly increases productivity. Employees have to be persistent to meet all daily deadlines. Productivity will decrease if there is less work that needs to be done at the end of the day. Breaks are introduced for employees to take rest and have free time. When several employees take a break, fewer hours are available to work.

Different methods of break management may result in deadlines being met or violated. Figure 4 illustrates two examples of break management: (i) employees distribute their breaks within three consecutive time intervals, (ii) all employees take a break simultaneously. The vertical blue bars illustrate the cumulative hours of work during morning and evening shifts without the implementation of breaks. The vertical red bars illustrate the cumulative number of hours that should be completed to meet specific deadlines. The red line visualises the actual hours worked for the two respective situations of break management. During the break, an employee is not able to perform work which is planned for during a shift. As a result, fewer employees are available during the break intervals. Therefore, the allocation of breaks at various intervals depicted in situation (i) leads to fulfilling all deadlines. Situation (ii) illustrates how simultaneously planned breaks may prevent deadlines from being met.

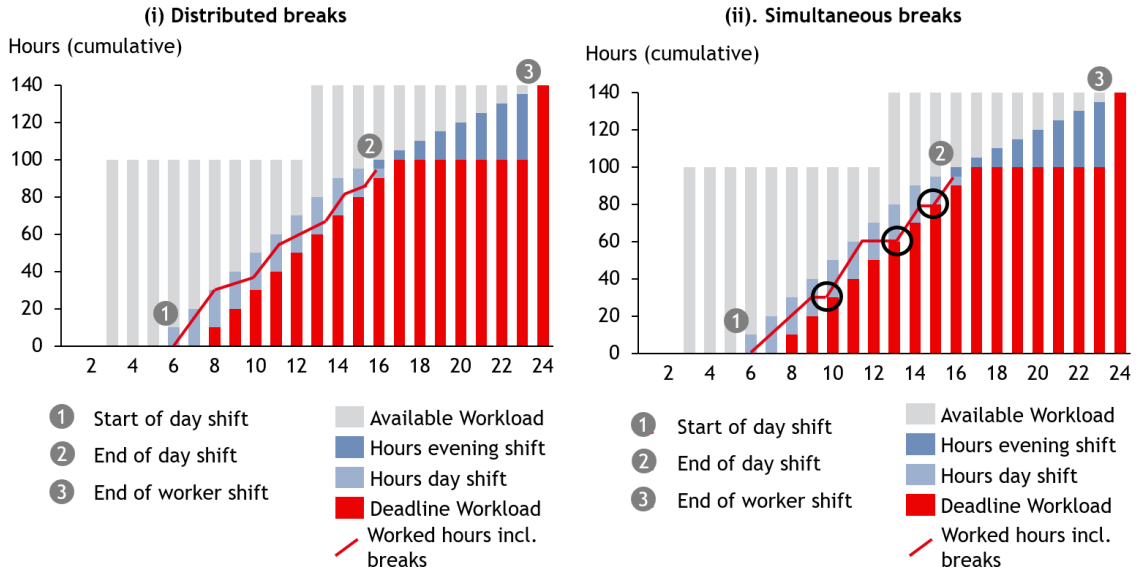


Figure 4: Distributed scheduled breaks versus simultaneously scheduled breaks

3.6 Symmetry

In optimisation models, symmetry can directly cause slow convergences as an effect of several solutions with the same equivalent outcome. For example, the exact model formulated in Subsection 4.2 has to deal with symmetry. Every schedule's tasks can be switched inside a particular time block since every employee can perform every task. Swapping workload from schedules will result in the same make span of the schedules. Symmetry is discussed in various research as a common problem in operations optimisation research Jans and Desrosiers [2013].

3.7 Input Data

The model's data is extracted from Picnic's DWH. All information on orders, productivity, performance, and other processes in the supply chain for any date and time is stored in the DWH.

An actual test can be carried out in an FC, presuming the algorithm is linked to Picnic's internal systems. The algorithm should first be transferred to the organisation's internal server. Next, the model should have a live connection with the DWH to transfer real-time data. At some point, a real-time test can be conducted to evaluate the model's effectiveness in practice. Finally, if the experiments occur, the algorithm can be tweaked based on actual practicalities.

3.8 Schedule Matrix

For simplicity, the schedules are discretised in time steps of fifteen minutes. As a result, 36-time units are required to fill a schedule based on nine hours. The schedule matrix, $M(N_s, N_t)$ is defined with N_s, corresponding to the number of schedules and N_t, corresponding to the number of time units. The schedule matrix is filled with integers corresponding to a specific activity. This way, every row can be seen as an individual schedule containing a unique activity for each time unit.

3.9 Assumptions

Summarising, the following assumptions are made to solve the FRCPSP:

- All employees can perform all clocking activities.
- The no-show percentage is already accounted for while planning the employees required for a shift. Therefore, no stochastic formulations are used in this research.
- The number of hours necessary to fulfil a task, defined in a workload package, is based on an average productivity per employee.
- Only "mature" employees, having work experience of over a week, are considered in the scheduling process.
- The time used within the model is discretised to slots of 15 minutes.
- The algorithms create schedules per shift and thus only consider 36-time units.
- Robustness of a schedule is achieved through minimising the maximum make span. Bonus hours are formulated to compensate for slack during the day.

4 Exact Solution Method

Subsection 4.1 describes the decision variables used in the model, the data sets, and the parameters for the task allocation algorithm. Subsequently, the MILP formulation used to solve the FRCPSP is described in Subsection 4.2.

4.1 Decision Variables, Sets, and Parameters

The mathematical model of the FRCPSP is formulated as a MILP with three sets mapped in Table 1, and six decision variables (DV) mapped in Table 2. The main DV is defined as $x_{i,s,t}$, which is a binary variable that denotes a value 1 if task i is assigned to schedule s at timestamp t and zero otherwise. The data sets used to define the DV $x_{i,s,t}$ are illustrated in Table 1. Next, the set of schedules $\mathcal{S}$ depends on the available number of employees during a shift. As described in Subsection 3.3, the schedules are discretised in parts of fifteen minutes, contributing to a shift duration of nine hours defined in set $\mathcal{T}$. The number of available activities and the time units are uniform for every schedule, whereas the number of schedules depends on the availability of employees. The DV c_s is an integer variable defined as the make span of a schedule. In other words, the interval from the first scheduled task to the final scheduled task per schedule. The maximum value of the make span, aggregated over all schedules, is defined as c_{max}. $z_{i,s,t}$ is a binary DV, which is 1 when the activity i at timestamp $t+1$ is different compared to timestamp t at the same schedule. The next DV, $l_{s,t}$, indicates the start of a lunch break. A lunch break consists of two consecutive time units and could be scheduled among an interval defined as set B_b. Finally, the last DV is used to linearise the full precedence relationship constraint and is shown in Equation 12. The linearisation of the second objective and the full precedence constraint is derived in Appendix A. Other parameters used to define the model are listed in Table 3.

Minimising the make span is critical for increasing the robustness of the schedules and maintaining high productivity through a tight schedule. As a result, both objectives of the mathematical model are formulated in lexicographical order, meaning that the second objective is computed after the first. The second objective can not influence the outcome of the first objective.

The first objective increases the robustness of the model by initialising bonus hours. Bonus hours are scheduled to compensate for possible slack gathered during the execution of activities. Second, the minimisation of the make span results in tight schedules. The desired productivity of employees is used for scheduling employees to given tasks. Picnic is able to monitor if employees are underperforming during the operations to know which activity is a bottleneck. Picnic observes employees being less productive when they have more time to fulfil their tasks.

The second objective (Equation 2) minimises the number of switches per schedule. As described in Section 3, switches cause drops in productivity, as Employees must stop work, move to a new location in the FC, and start a new activity. The second objective is multiplied with a cost factor Q to compensate for switches from a break to another activity. A switch from a break to another activity should not be counted as a switch. Employees should migrate to the new activity during the break to avoid wasting time. Moreover, breaks are regulated and therefore described as mandatory.

Table 1: Mathematical Model - Data Sets

Sets	
$\mathcal{I}$	Set of all activities, indexed by i
$\mathcal{I}_1$	Subset of activities with priority level 1, indexed by i
$\mathcal{I}_2$	Subset of activities with priority level 2, indexed by i
$\mathcal{S}$	Set of schedules during a day, indexed by s
$\mathcal{T}$	Set of time units of 15 min starting from 06.00 PM, indexed by t
$\mathcal{B}$	Subset of all time units that contains a break
$\mathcal{B}_b$	Subset of specific time intervals indicating a break, indexed by b*

Table 2: Mathematical Model - Decision Variables

		Decision Variables
$x_{i,s,t}$	Binary	Decision variable, equal to 1 if task i is performed in schedule s during time unit t, else 0
c_s	Continuous	Make span of schedule s
c_{max}	Continuous	Maximum make span
$z_{i,s,t}$	Binary	Decision variable for the number of switches
$l_{s,t}$	Binary	Decision variable indication of the start of a lunch break.
$y_{i,t}$	Binary	Decision variable to operate the full precedence relationship linearisation constraint

Table 3: Mathematical Model - Parameters

Parameters	
Wi	Workload of activity i in hours
K_i	Maximum capacity of employees on activity i
T_t	Parameters which is equal to the scheduling time unit t
S_i	Start time of activity i in time units t
$P^l_{i,j}$	Parameter for linear relationship between task i and task j
$P^f_{i,j}$	Parameter to indicate if task i should be fully executed before task j can start
S_i	Start time for activity i based on time unit t
D_i	Deadline for task i in time units t
V_b	Parameter to indicate the duration of specific breaks
E_{max}	Maximum difference of employees allowed to take a break
η	Parameter indicating the % hrs of tasks with priority 2 should be scheduled
Q_i	Parameter defined to penalise a switch from activity i

4.2 Mathematical Model

$$\min \quad c_{\max} \tag{1}$$

$$\min \sum_{i \in \mathcal{I}} Q_i \sum_{s \in S} \sum_{t \in \mathcal{T}} z_{i,s,t} \tag{2}$$

subject to

$$c_s \geq t \sum_{i \in \mathcal{I}} x_{i,s,t} \qquad \forall s \in \mathcal{S}, t \in \mathcal{T} \tag{3}$$

$$c_{\max} \geq c_s \qquad \forall s \in S \tag{4}$$

$$\sum_{s \in \mathcal{S}} \sum_{t \geq S_i}^{t \leq D_i} x_{i,s,t} \geq W_i \qquad \forall i \in \mathcal{I}1 \tag{5}$$

$$\sum_{s \in \mathcal{S}} \sum_{t \geq S_i}^{t \leq D_i} x_{i,s,t} \geq \eta \cdot W_i \qquad \forall i \in \mathcal{I}2 \tag{6}$$

$$\sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}} x_{i,s,t} \leq K_i \qquad \forall i \in \mathcal{I} \tag{7}$$

$$\sum_{i \in \mathcal{I}} x_{i,s,t} \leq 1 \qquad \forall s \in \mathcal{S}, t \in \mathcal{T} \tag{8}$$

$$x_{i,s,t} \cdot t \leq D_i \qquad \forall i \in \mathcal{I}, s \in \mathcal{S}, t \in \mathcal{T} \tag{9}$$

$$x_{i,s,t} \cdot t = 0 \qquad \forall i \in \mathcal{I}, s \in \mathcal{S}, t \in \{0, ..., s_i\} \tag{10}$$

$$\frac{\sum_{s \in S} \sum_{t \geq s_i}^{\min(T_{comp}, d_i)} x_{i,s,t}}{W_i} \geq \frac{\sum_{s \in \mathcal{S}} \sum_{t \geq s_j}^{\min(T_{comp}, d_j)} x_{j,s,t}}{W_j} P_{i,j}^l \qquad \forall (i,j) \in \mathcal{I} \tag{11}$$

$$\sum_{t=S_i}^{D_i} \sum_{s \in \mathcal{S}} x_{i,s,t} - \sum_{t=S_i}^{T_{comp}} \sum_{s \in \mathcal{S}} x_{i,s,t} \leq M_1^i y_{i,t} \qquad \forall i \in I, t \in \{S_i, \ldots, D_i\} \tag{12}$$

$$\sum_{s \in \mathcal{S}} x_{j,s,t} P_{i,j} \leq M_1^j \left(1 - y_{i,t}\right) \qquad \forall i, j \in I, t \in \{s_j, \ldots, d_j\} \tag{13}$$

$$\sum_{t \in \mathcal{B}} x_{0,s,t} = \mathcal{B}_b \qquad \forall s \in \mathcal{S}, t \in \mathcal{T}, b \in \mathcal{B} \tag{14}$$

$$\sum_{t \in T_{bl}} l_{s,t} = 1 \qquad s \in \mathcal{S} \tag{15}$$

$$x_{0,s,t} \geq l_{s,t} \qquad \forall s \in \mathcal{S}, t \in \mathcal{T} \tag{16}$$

$$x_{0,s,t+1} \geq l_{s,t} \qquad \forall s \in \mathcal{S}, t \in T_{bl} \tag{17}$$

$$\sum_{s \in \mathcal{S}} x_{0,s,t} \leq E_{max} \qquad \forall t \in \mathcal{B} \tag{18}$$

$$x_{i,s,t} - x_{i,s,t+1} \leq z_{i,s,t} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S}, t \in \mathcal{T} \tag{19}$$

$$x_{i,s,t+1} - x_{i,s,t} \leq z_{i,s,t} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \tag{20}$$

$$z_{i,s,t} \geq 0 \qquad \forall i \in \mathcal{I}, s \in \mathcal{S}, t \in \mathcal{T} \tag{21}$$

$$x_{i,s,t} \in \{0,1\} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S}, t \in \mathcal{T} \tag{22}$$

$$c_s \in \{0, ..., t\} \qquad \forall s \in \mathcal{S}, t \in \mathcal{T} \tag{23}$$

$$c_{\max} \in \{0, ..., t\} \qquad \forall t \in \mathcal{T} \tag{24}$$

$$y_{i,t} \in \{0,1\} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{25}$$

$$z_{i,s,t} \in \{0,1\} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S}, t \in \mathcal{T} \tag{26}$$

$$l_{s,t} \in \{0,1\} \qquad \forall s \in \mathcal{S}, t \in \mathcal{T} \tag{27}$$

The first set of constraints shown in Equations 3 and 4 describes the definition of the make span. The maximum value of all make spans c_s is the maximum make span c_{max}. Next, the constraints depicted in Equation 5 ensure that all workloads with a high priority are scheduled. Subsequently, Equation 6 describes the workload which should be scheduled for low-priority tasks. The η parameter regulates the percentage low-priority tasks that should be completed in a shift. low-priority tasks are defined as tasks that does not directly influence the operations of an FC. The capacity of FCs to execute specific activities is limited to number of employees. As a result, Equation 7 is defined to account for the capacity within FCs.

Furthermore, scheduling constraints are shown in Equations 8 and 9. First, Equation 8 ensures only one activity is scheduled at a specific time stamp. The other two Equations 9 & 10, are defined to meet all required starting times and deadlines.

Next, sequential operational processes cause precedence relationships. Three different sequential relations are formulated within this research. First, linear relationships are described in Equation 11, which defines a certain percentage of work that can be done before the same percentage of the following activity could be executed. The binary parameter $P^l_{i,j}$ is active when an activity i is related to activity j. Second, Equation 12 and 13 describe full precedence relationships between tasks. In other words, task i should be fully executed before task j can start. Parameter $P^f_{i,s}$ is active when two activities have full precedence relations. Third, tasks without any relationships to each other are not defined within the constraints. Figure 5 illustrates the relationships for the three different scenarios.

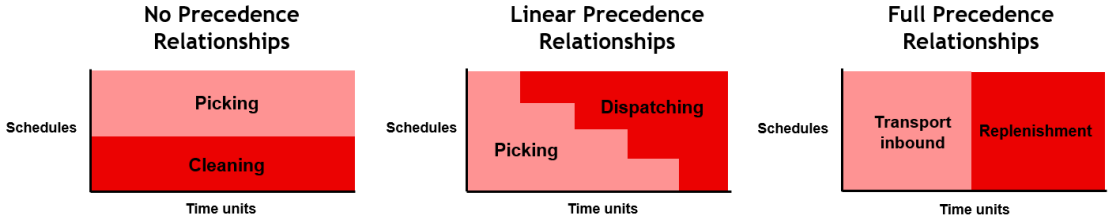


Figure 5: Different types of precedence relations

The full precedence constraint is linearised through a big-M method and derived in Appendix A. The big-M used in Equation 12 has to be carefully chosen. M^i_1 should be large enough to cover the entire feasible space but should be as small as possible to prevent unnecessary slow progress of the B&C algorithm. As a result, Equation 28 illustrates that the value for M^i_1 depends on the predecessor task the constraint applies to.

$$M^i_1 = W_i \tag{28}$$

Subsequently, the break management constraints are defined and described in Equations 14 - 18. First, Equation 14 describes the different time units t for which a break should be scheduled. The subset $\mathcal{B}$ derived from values in set $\mathcal{T}$ describes all values t for which a break can be scheduled. For instance, the lunch break consists of a 30-minute uninterrupted break and the other two coffee breaks of 15 minutes. Therefore, the following constraints shown in Equations 15 - 17 define that two consecutive time units should be scheduled for a lunch break within the subset $\mathcal{B}$. Next, the constraints formulated in Equation 18 defines the maximum number of employees allowed to take a break at the same time unit t. The maximum allowed employees going on a break can be constrained with the capacity constraint (Equation 7) and the maximum break constraint (Equation 18).

Finally, the second objective is complemented with a set of constraints to define the DV $z_{i,s,t}$. The second objective was formally defined as an absolute constraint, measuring the differences in activities between two consecutive time stamps. The linearisation of the second objective is the result of Equations 19 - 21. Absolute values produce a nonlinear constraint in a MILP, which makes them undesirable. The linearisation of the absolute constraints is derived in Appendix A.

5 Meta-Heuristic

This section elaborates on the decision of a VNS in combination with the outcome of the exact model, optimising the make span. First, an illustration of the proposed neighbourhood search is discussed in Subsection 5.1. Next, the method used to retrieve an initial solution is elaborated in Subsection 5.2. Subsequently, the VNS used to modify the initial answer to strive for fewer switches is discussed in Subsection 5.3

5.1 Overview of the Meta-Heuristic

Solving the MILP with an exact solution method is too time-consuming for real-time data instances. The exact model's limitations are due to the size of the problem and, as a result, the memory available on the used computer. Another option is to improve the computational time by solving the problem with a meta-heuristic. A definition of a meta-heuristic is provided by van Gils et al. [2018]: "A meta-heuristic is a computational procedure that determines an optimal solution by iterative attempting to improve a candidate solution to a given quality measure." Moreover, using a meta-heuristic is another technique to approximate the optimal solution. A well-performing meta-heuristic generally solves the problem with a lower computational time.

Figure 6 illustrates the intermediary steps of the meta-heuristic. First, the model's input combines workload (workload packages), workforce, and other FC characteristics. Then, an exact model is proposed to get an initial solution that complies with the requirements stated in the MILP. The number of switches within the schedule matrix is analysed in the next step. Consequently, the VNS retrieves a scheduling matrix with fewer switching moments.

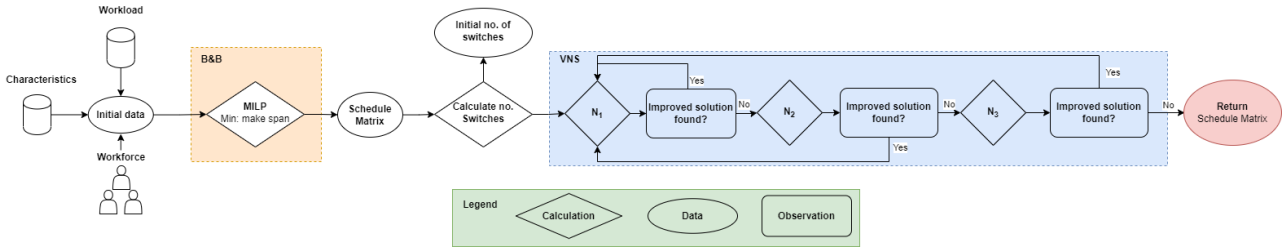


Figure 6: Overview of the variable neighbourhood search

5.2 Initial Solution for the Variable Neighbourhood Search

The B&C algorithm is able to solve the MILP for minimising the make span in a reasonable time frame for more significant data instances. However, the number of switches must be accounted for in the first step. The distribution of activities per schedule only depends on constraints regarding starting time, end time, and precedence relationships. Therefore, the exact algorithm needs to consider the various activities per schedule. Solving the minimisation of the make span resolves in a set of schedules, each with the minimal make span. Instead of using a different approach that involves starting from scratch with a fresh schedule, the MILP formulation extracts an answer bound by all constraints.

5.3 Variable Neighbourhood Search

Applying a VNS to a regular FRCPSP has proven successful in literature [Chakrabortty et al., 2020, Cui et al., 2021]. The meta-heuristic is chosen for its adaptability, flexibility, and ease of implementation. As discussed, the VNS consists of nested neighbourhoods, combinations of resource transfers that can be easily accessed within the algorithm. The neighbourhoods are explicitly defined to create the desired outcome. The meta-heuristic's adaptability and flexibility is the result of multiple neighbourhoods active in the VNS. The outcome can be adjusted by formulating another set of neighbourhoods.

Three different neighbourhoods are defined and elaborated in Table 4. The meta-heuristic uses neighbourhoods in the listed order, progressing from small to large. Neighbourhoods are established to ensure no constraints are broken when constructing the solution. Furthermore, the meta-heuristic cannot worsen the make span returned by the initial solution. The VNS only adjusts the time units are scheduled with clocking activities within the initial solution so that no bonus hours can be swapped. As a result, the make span of the schedules can not increase, which is essential because the objectives of the MILP are defined in lexicographical order.

The logic of the VNS is derived in Algorithm 1. The first neighbourhood evaluates all possible swaps within the same time unit t. If a swap causes a decrease in the number of switches compared to the old situation, the swap is performed. Suppose no swaps within the schedule matrix will result in a better solution, the meta-heuristic jumps to the second neighbourhood. The second neighbourhood consists of three consecutive swaps within the same time unit t, contributing to a better solution. If a swap is performed, the first neighbourhood will be used again. If no improvement is found, the third neighbourhood function is used to find a potential improvement. The nested neighbourhood loops will be conducted until no improvement is found. Each neighbourhood function has an unique local optimum compensated by moving to a different neighbourhood.

Table 4: Definitions of different neighbourhoods

Neighbourhood	Definition
N_1	Swapping two tasks within the same time unit t
N_2	Swapping three tasks within the same time unit t
N_3	Swapping a task from time unit t with time unit $t+1$

Algorithm 1 Variable Neighbourhood Search

```
Data: Schedule_Matrix
Result: Schedule_Matrix
k := 1;
 Opt_schedule_Matrix = Schedule_matrix;
 while k < Total_Neighbourhoods; do
    (Schedule_matrix, Δ) := N_k(Opt_Schedule_matrix);
     if Δ < 0 then
        k := 1;
         Opt_Schedule_matrix = Schedule_matrix
    else
        k += 1
    end
end
return Opt_Schedule_Matrix
```

6 Results

This section is used to describe the results. First, the result of the initial workload package is described in Subsection 6.1. Second, the computational results are evaluated and analysed in Subsection 6.2. Third, a real-life data instance is analysed and discussed in Subsection 6.3

6.1 Description of Initial Instance

An initial test instance is created to test the behaviour of the constraints and the objective functions in lexicographical order. A small data instance shown in, Table 5 illustrates the initial workload package. The workload package contains four different activities with workload for ten employees.

Table 5: Illustrative example of initial workload package

Date	**Task ID**	**Temp Zone**	**Start Time**	**End Time**	**Hours**
22/07/2022	picking	AM	09:00:00	14:00:00	10
22/07/2022	replenishment	AM	07:00:00	16:00:00	10
22/07/2022	dispatching	AM	07:00:00	16:00:00	4
22/07/2022	transport_inbound	x	07:00:00	16:00:00	2

Figure 7, illustrates the results for compiling the initial workload package. The figure clearly shows the precedence relationships and the mutual end times of all schedules at 13.15. Slack and/or irregularities can be compensated for over 2.75 hours per schedule due to the bonus hours. Next, the minimisation of switches is clear. As can be seen from Figure 7, the breaks are used as a connection to switch to another task.

The effect of other constraints are also visible in the figure. The lunch and initial coffee breaks are distributed within the right time slots. Observing the coffee breaks in the first time slot, most of the coffee breaks are scheduled at 09.30 to compensate for a switch.

Additionally, the precedence relationships are observed within the initial data set. First, full precedence relationship constraints are defined for the transport_inbound and replenishment activities. Second, the same constraints hold for the replenishment and picking activities.

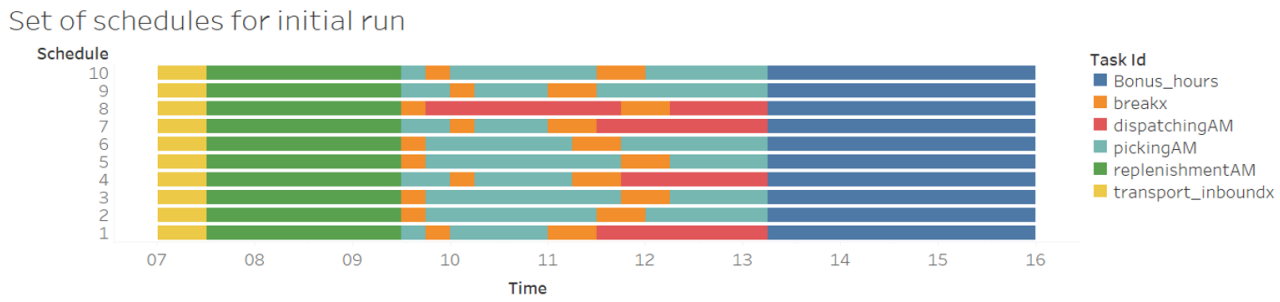


Figure 7: Initial set of schedules, workload package from Table 5

6.2 Computational Results

All models are developed in Python version 3.10 with Pycharm (Pycharm Community Edition 2021.3.3) as an integrated development environment (IDE) and compiled on a Dell 5410 I5-10310U CPU. Gurobi is used to solve the MILP with a B&C algorithm. The meta-heuristic is coded solely with the NumPy library, such that a JIT extension named Numba is able to speed up the process. A JIT compiler initialises the functions used for the meta-heuristic into machine code. Once the functions have been initialised, calculations in machine code will be executedăvery fast. Transferring the functions into machine code requires the code to be compatible with NoPython mode. NoPyhton mode ensures the code is used without any python extension. Finally, the solution will be visualised through Tableau, a commercial data visualisation tool.

The performance of the MILP solving objective 1 is shown in Table 6. The table is built in two sections; the left side of the table describes the data used, and the right side illustrates the model's performance. The B&C algorithm can solve all data instances within reasonable time limits. Picnic sets the feasible operational time to create schedules within a time horizon of 2 consecutive hours. All results are obtained within a fraction of the 2 hours defined as yhe operational maximum by Picnic. Therefore, the convergence of the best incumbent to the lower bound is obtained within a reasonable time frame, considering the computational complexity. The computer's size, and consequently available memory, is the limiting factor for getting a solution with 1000 schedules. The number of DVs for 500 schedules with 30 activities minimising the make span is already 559,581.

Table 6: Exact method solutions for the minimisation of the make span

# Schedules	# Activities	Best Incumbent [t]	Lower Bound [t]	Optimality Gap [%]	Comp. Time [s]
10	6	31	31	0	0.4
10	11	32	32	0	0.7
10	30	35	35	0	1.2
50	6	31	31	0	2.3
50	11	31	31	0	6.5
50	30	35	35	0	6.6
100	6	31	31	0	7.7
100	11	32	32	0	48.6
100	30	35	35	0	50.2
150	30	30	30	0	76.1
250	30	35	35	0	184.7
500	30	35	35	0	196.7
1000	30	n.a.	n.a.	n.a.	n.a.

The second objective, minimising the number of switches, is very time-consuming, partly due to symmetry and the increased size of the problems due to increased DVs and constraints. The second objective introduces another decision variable ($z_{i,s,t}$) with three variables, almost doubling the size of the problem compared to the optimisation of objective 1. The exact model's performance and the second objective's meta-heuristic are shown in Table 7. The table is built in three sections; first, the data type is described. Second, the exact model performance is described by illustrating the Best Incumbent solution (B.I.) for the number of switches, the Lower Bound (L.B.) of the number of switches, the Optimality Gap (O.G.), Time To Best (T.T.B.), and the Computational Time (C.T.). Third, the performance of the meta-heuristic is described. The Initial Solution (I.S.) describes the number of switches before executing the VNS. Next, the Best Solution (B.S.) describes the number of switches after the meta-heuristic is computed. Finally, the computational time of both the exact model (C.T.E) and the meta-heuristic (C.T.H.) is provided in seconds. The sum of both values is the overall computation time of the algorithm.

It is evident from the Table 7 that for smaller data instances, an exact solution strategy may address the problem more effectively. Both the computational time to converge to an answer as the quality of the solution

outperforms the meta-heuristic. The meta-heuristic can find an improvement if the best solution is between the best incumbent and the lower bound. The best solution found within the operational time of 7200s is highlighted in bold in Table 7.

Another conclusion can be drawn from Table 7 focused on computational time. The computational time constraint prevents the MILP from finding an ideal solution to any additional instances, except the first data type. The performance of the meta-heuristic and the exact solution method should be observed per data instance, thus per row. The discrepancies per initial solution result in differences when comparing the meta-heuristic and the MILP.

The meta-heuristic improves the solution for half of the data instances. Table 7 illustrated that the performance of the meta-heuristic starts improving the solution from a specific size of 50 schedules with at least 30 activities. Therefore, the VNS outperforms the B&C method for realistic data instances. On the contrary, the exact solution method is preferably used for smaller data instances.

In reality, FCs handle 30 activities and schedule around 100 employees per shift. An exception can be made on larger FCs, scheduling around 150 employees per shift compared to smaller FCs scheduling around 70 employees. Therefore, the meta-heuristic is proposed for actual data instances since the table illustrates the sweet spot for the VNS.

Currently, the limiting factor of solving 1000 schedules with 30 activities is due to the exact model. The MILP can not calculate an initial answer due to the size of the problem. If an initial answer provides a set of feasible schedules, the VNS can optimise the number of switches even for larger data instances. A peak in computational time is visible at 500 schedules with 30 activities. After that, the computational time increases from 1000s to 10000s by doubling the data instance. The VNS also fails to retrieve an answer within 7200s for 500 schedules with 30 activities. Symmetry and the size of the problem cause an increase in computational time.

Table 7: Exact model compared to the meta-heuristic for the minimisation of switches

Data Type		**MILP**					**VNS**			
# Schedules	**# Activities**	**B.I.**	**L.B.**	**O.G.** [%]	**T.T.B.** [s]	**C.T.** [s]	**I.S.**	**B.S.**	**C.T.E.** [s]	**C.T.H.** [s]
10	6	**66**	66	0	1	1	159	82	0	1.7
10	11	**76**	75	1.3	18	7200	184	105	1	1.8
10	30	**66**	52	21.5	1363	7200	281	160	1	1.8
50	6	**300**	287.3	4.2	416	7200	798	342	2	3.3
50	11	**384**	381.6	0.6	1012	7200	939	395	6	2.4
50	30	590	174	70.5	3219	7200	306	**281**	7	1.9
100	6	**684**	596	12.8	6523	7200	1828	698	8	5.4
100	11	960	702	26.2	6763	7200	2030	**760**	49	21.2
100	30	1212	390	67.8	2300	7200	2747	**1031**	50	15.3
150	30	2158	1035	52.0	5260	7200	3446	**1489**	76	222.8
250	30	4482	618.2	86.2	5349	7200	6966	**2494**	184	1005.4
500	30	16758	861.7	94.9	5415	7200	13527	**4511**	196	10656.3
1000	30	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.

6.3 Case Study

The meta-heuristic is able to generate schedules for real time data within reasonable computational time. The figure in Figure 8 illustrates a snapshot of 50 from the 150 schedules produced by the meta-heuristic using a real-time workload package as input. The input workload package is shown in Appendix B and consists of workload for 26 activities.

Figure 8 clearly illustrates the make span's performance. Each schedule ends at a maximum of 14:45 AM (31-time units). The fact that no bonus hours are planned before the shift finishes demonstrate how the first objective ensures that the activities are scheduled tight.

Nevertheless, the schedules still contain around seven or eight switches per shift. Ideally, the number of activities should be limited to four. Picnic observes A person being less productive throughout a shift if many different activities are executed. The drawback of restricting employees to four tasks every shift is that more employees are required to do the same workload. As a result, the total salary costs increases.

The 50th schedule is an example of breaks used to switch between activities. After the first 15-minute break, the employee has to start performing transport_outbound. After the lunch break, the employee picks until the next break. This pattern is a desired outcome for all of the schedules.

The number of switches can be decreased with a relaxation on the first objective. Currently, the meta-heuristic cannot modify the maximum make span calculated in the initial solution. If the VNS is able to increase the maximum make span, more space is created to pair consecutive activities in order to reduce the number of switches. In daily operations, most employees have to work until the end of their shift. Therefore, the schedule is still feasible if the maximum make span is defined within the boundaries of $\mathcal{T}$.

Next, the same observations discussed from the initial solution (Figure 7) can be concluded by observing Figure 8. The precedence relationships and the break management systems are visible in both figures. The precedent relationship constraints force the algorithm to schedule morning activities at the start of the shift. Furthermore, Figure 8 illustrates three break moments planned over all schedules within three specific time horizons.

Implementing the task allocation algorithm reduces a lot of costs for Picnic. Currently, the scheduling process takes over 2 hours per shift due to the manual allocation of tasks per employee and the verification process. The proposed work allocation technique reduces the scheduling time to a maximum of 30 minutes. Scheduling two shifts daily for 350 days adds up to 1400 hours per FC each year. Picnic currently operates 12 FCs resulting in 16800 hours each year. Based on hourly costs of € 22 per hour, the model will save Picnic approximately € 369.600 - € 92.400 = € 277.200 annually. The calculation does not include the possible increase in productivity from the new schedules.

Figure 8: Visualisation of 50/150 schedules from a real-life workload package

6.4 Discussion of the Results

The case study shows that the meta-heuristic is able to compile schedules for any operational day. The algorithm also indicates how many additional hours might be planned to complete more work. However, a few additions should integrate before the model can be used for daily operations.

The current output still requires modifications before it can be implemented in daily operations. First, the number of switches during a shift should be constrained to four or five maximum. Second, Figure 8 contains different activities which are scheduled for 15 minutes. Scheduling activities for 15 minutes is not desired during daily operations. Third, particular tasks like quality should be planned for the entire shift.

The current objectives are formulated in lexicographical order. Formulating the objectives as a multi-objective FRCPSP could increase the quality of the schedules. Subsequently, The weights for both objective

functions should be analysed according to Pareto efficiency. The number of switches can be reduced by relaxing the make span.

Picnic is planning on introducing dedicated teams specialised in one activity to increase productivity. Being able to do one task for a long time each day of the week will increase the handling speed of an employee. Although for some employees, the work can feel monotonous, indirectly decreasing productivity. Adjustments within the planning process should be taken into account while aiming for the integration of teams.

It can be seen that the VNS outperforms the exact model for real-life instances by 31% whilst reducing the computational time by 95%. The meta-heuristic is obtaining far superior results observing both objective functions. Yet, for smaller instances of around ten schedules, the exact solution method outperforms the meta-heuristic with 76%. The B&C algorithm still requires a lot of time, being stopped by the computational constraint of two hours.

Currently, the model is compiled on an i5 8GB ram desktop. The computational time will drop if the model is compiled on a computer with better computational power. Having the system set up in Picnic servers will almost immediately cause the computational time to decrease compared to the current setup.

7 Conclusion, Discussion, and Future Research

The e-grocery market is expected to continue to grow in the future. Picnic is continually seeking innovations to keep disrupting the e-grocery industry. Innovations and growth come with challenges. The importance of a centralised planning method increases with the expansion of new FCs and DCs. An overarching edge system is set up to deal with the new planning system discussed in the paper.

This paper aimed to improve the current planning process by optimising a task allocation algorithm to retrieve individual timetables on employee level automatically. The decision tool is designed to replace the current scheduling method by implementing specific adjustments. The algorithm is built upon retrieving workload packages as input. Based on the input, timetables for individual employees are created within the operational time of two hours.

This research resulted in an exact algorithm and a meta-heuristic, which matches workload with workforce to retrieve high-quality solutions. The model is specifically designed for operations within Picnic's FCs and DCs. To our knowledge, no model of this kind has been developed before this study. As far as we could corroborate, no existing work captures all the access to various (historical) data and preemptive sequential processes. However, the main problem can be subdivided into smaller pieces that have been studied before. The chosen exact method to solve the FRSPCP resulted in efficient schedules for small data instances.

The objective functions are defined in lexicographical order. Proposing a trade-off between the outcome of the different objective functions to form Pareto efficiency could improve the solution. More consecutive activities could be scheduled to decrease the number of switches per schedule when a relaxation of the make span is allowed.

Next, specific meta-heuristics are analysed to decrease the computational time for processing larger data instances. The chosen method to solve the FRSPCP resulted in significant improvements to decrease the number of switches. Namely, the meta-heuristic reduced the computational time from multiple hours to several minutes. However, The computational time of the planning model is increased by symmetry. The convergence for the minimal number of switches is computational time intensive due to the added complexity. The method is chosen for its flexibility and adaptability.

The results showed that the B&C algorithm is able to create optimal schedules for small data instances. The exact solution method outperforms the meta-heuristic for data instances up to 50 schedules with 11 tasks. The VNS outperforms the exact model for real-life instances by 31% whilst reducing the computational time by 95%. The case study shows positive results for both the make span. All timetables are finished at 2.45 in the afternoon and no bonus hours are scheduled during the operations. In addition, significant costs can be saved by implementing the proposed task allocation algorithm. However, the number of switches still requires some attention. Picnic strives for a maximum of five switches per schedule. Yet, the proposed meta-heuristic has eight switches per schedule on average. Therefore, the algorithm requires some more modifications before it can be implemented in daily operations.

Moreover, the outcome of this research results in interesting future research directions, which can improve the designed models. First, the schedules can illustrate a more precise task allocation model by introducing timetables based on continuous time. Currently, the timetables are discretised in time slots of 15 minutes. More accurate schedules can be crated by formulating the problem in continuous time. Second, the task allocation process can also be based on truck level, indicating the workload packages' dynamic start and end times. The workload packages' start times are based on incoming trucks, and the end times depend on outbound trucks. Furthermore, a tool to distribute employee schedules would contribute to the future planning system. Such a decision-making tool will immediately result in a decrease in human errors and more transparent allocation progress. Last, the quality of the meta-heuristic would improve by enhancing the initial solution. Currently,

the initial solution is created without focusing on the number of switches. Increasing the quality of the initial solution will directly increase the performance of the heuristic.

Appendices

A Linearisation Of Constraints

This section is used to substantiate different constraints used in the paper. The objective to minimise the number of switches as the full precedence constraint was formulated differently. First, the objective to calculate the number of switches defined in Equation 2, originated from the objective function described in Equation 29.

$$\text{Objective 2} : \min \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T} \setminus \{T\}} |x_{i,s,t+1} - x_{i,s,t}| \quad \forall s \in \mathcal{S} \tag{29}$$

A new decision variable $z_{i,s,t}$ is initialised to define the number of switches per activity per time unit. The following set of constraints (Equation 31 - Equation 33) and a new objective (Equation 30) is defined to replace Equation 29:

$$\min \quad \sum_{i \in \mathcal{I}} Q_i \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}} z_{i,s,t} \tag{30}$$

$$x_{i,s,t+1} - x_{i,s,t} \leq z_{i,s,t} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S} \tag{31}$$

$$x_{i,s,t} - x_{i,s,t+1} \leq z_{i,s,t} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S} \tag{32}$$

$$z_{i,s,t} \geq 0 \qquad \forall i \in \mathcal{I}, s \in \mathcal{S}, t \in \mathcal{T} \tag{33}$$

Second, the full precedence constraint ware defined as Equation 34 which clearly is not linearly defined:

$$\sum_{s \in \mathcal{S}} x_{j,s,T_{comp}} \cdot (\sum_{s \in \mathcal{S}} \sum_{t \geq S_i}^{t \leq D_i} x_{i,s,t} - \sum_{s \in \mathcal{S}} \sum_{t \geq 0}^{\min(T_{comp}, D_i)} x_{i,s,t}) \cdot P_{i,j}^{f} = 0 \quad \forall (i,j) \in \mathcal{I} \tag{34}$$

The set of constraints shown in Equation 34 is imposing a flexible deadline which relies on the starting time of a successive task. For this reason, a binary new decision variable is created and shown in Table 2
A flexible deadline that depends on the starting time of successive tasks and therefore requires an additional binary decision variable $y_{i,t}$, which is zero if no workforce will be allocated to the task i after time t and one otherwise as illustrated in Equation 35. Resulting in a linear set of full precedence constraints defined in Equation 36 - 37.

$$y_{i,t} = \begin{cases} 0 & \text{if} \quad W_i = 0 \\ 1 & \text{if} \quad \text{Otherwise} \end{cases} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{35}$$

$$\sum_{t \geq S_i}^{t \leq D_i} \sum_{s \in \mathcal{S}} x_{i,s,t} - \sum_{t \geq S_i}^{T_{comp}} \sum_{s \in \mathcal{S}} x_{i,s,t} \leq M^i y_{i,t} \quad \forall i \in I, t \in \{S_i, \ldots, D_i\} \tag{36}$$

$$\sum_{s \in \mathcal{S}} x_{j,s,t} P_{i,j} \leq M^j \left(1 - y_{i,t}\right) \quad \forall i \in I, t \in \{S_j, \ldots, D_j\} \tag{37}$$

To use the linearisation above, a careful choice for the big M is required. The big-M should be chosen large enough to cover the feasible solution space. As a result, we will therefore choose a big-M dependent on the predecessor task to which the constraint directly applies. Consequently, the big-M is defined as the workload required for the same task as seen in Equation 38.

$$M^i = W_i \tag{38}$$

B Workload Package

Table 8: Real-time workload package

date	start_time	end_time	task_id	temp_zone	units	productivity	hours
06/10/2022	07:00:00	16:00:00	picking	AM	confid.	confid.	273
06/10/2022	07:00:00	16:00:00	picking	CH	confid.	confid.	130
06/10/2022	07:00:00	16:00:00	picking	FR	confid.	confid.	9
06/10/2022	07:00:00	16:00:00	dispatching	AM	confid.	confid.	34
06/10/2022	07:00:00	16:00:00	dispatching	CH	confid.	confid.	32
06/10/2022	07:00:00	16:00:00	tote_handling	AM	confid.	confid.	35
06/10/2022	07:00:00	16:00:00	tote_handling	CH	confid.	confid.	30
06/10/2022	07:00:00	16:00:00	tote_completion	CH	confid.	confid.	45
06/10/2022	14:00:00	16:00:00	transport_outbound	x	confid.	confid.	24
06/10/2022	09:00:00	16:00:00	transport_inbound	x	confid.	confid.	5
06/10/2022	07:00:00	09:00:00	transport_inbound_morning	x	confid.	confid.	10
06/10/2022	11:00:00	16:00:00	buffer_replenishment	AM	confid.	confid.	25
06/10/2022	11:00:00	16:00:00	buffer_replenishment	CH	confid.	confid.	21
06/10/2022	11:30:00	16:00:00	return_waste	AM	confid.	confid.	33
06/10/2022	11:30:00	16:00:00	return_waste	CH	confid.	confid.	20
06/10/2022	07:00:00	09:00:00	replenishment_morning	AM	confid.	confid.	48
06/10/2022	07:00:00	09:00:00	replenishment_morning	CH	confid.	confid.	28
06/10/2022	07:00:00	09:00:00	replenishment_morning	FR	confid.	confid.	0
06/10/2022	07:00:00	16:00:00	replenishment	AM	confid.	confid.	119
06/10/2022	07:00:00	16:00:00	replenishment	CH	confid.	confid.	22
06/10/2022	07:00:00	16:00:00	replenishment	FR	confid.	confid.	0
06/10/2022	10:00:00	16:00:00	urgent_tasks	x	confid.	confid.	24
06/10/2022	07:00:00	16:00:00	counting	x	confid.	confid.	10
06/10/2022	07:00:00	16:00:00	slotting	x	confid.	confid.	0
06/10/2022	07:00:00	16:00:00	clearing	x	confid.	confid.	5
06/10/2022	07:00:00	16:00:00	cleaning	x	confid.	confid.	10
06/10/2022	07:00:00	16:00:00	quality	x	confid.	confid.	18
06/10/2022	07:00:00	16:00:00	safety	x	confid.	confid.	0
06/10/2022	07:00:00	16:00:00	projects	x	confid.	confid.	0

References

David Applegate and William Cook. Computational study of the job-shop scheduling problem. *ORSA journal on computing*, 3(2):149–156, 1991. ISSN 08991499. doi: 10.1287/ijoc.3.2.149.

Leila Asadzadeh. A local search genetic algorithm for the job shop scheduling problem with intelligent agents. *Computers & Industrial Engineering*, 85:376–383, 7 2015. ISSN 0360-8352. doi: 10.1016/J.CIE.2015.04.006.

J. Blazewicz, J. K. Lenstra, and A. H.G.Rinnooy Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1 1983. ISSN 0166-218X. doi: 10.1016/0166-218X(83)90012-4.

Ripon K. Chakrabortty, Alireza Abbasi, and Michael J. Ryan. Multi-mode resource-constrained project scheduling using modified variable neighborhood search heuristic. *International Transactions in Operational Research*, 27(1):138–167, 2020. ISSN 14753995. doi: 10.1111/itor.12644.

Ripon Kumar Chakrabortty, Ruhul Sarker, Daryl Essam, Ripon K Chakrabortty, and Daryl L Essam. Resource constrained project scheduling: A branch and cut approach. Technical report, University of South Wales, 2015. URL https://www.researchgate.net/publication/301893107.

Runwei Cheng, Mitsuo Gen, and Yasuhiro Tsujimura. A tutorial survey of job-shop scheduling problems using genetic algorithmsI. representation. *Computers & Industrial Engineering*, 30(4):983–997, 9 1996. ISSN 0360-8352. doi: 10.1016/0360-8352(96)00047-2.

Christodoulos A. Floudas and Panos M. Pardalos. Encyclopedia of Optimization Second Edition. Technical report, Sptinger, 2019.

Nicos Christofides, R Alvarez-Valdes, and J M Tamarit. Project scheduling with resource constraints: A branch and bound approach. Technical report, Imperial College & Universidad de Valencia, 1987.

Longqing Cui, Xinbao Liu, Shaojun Lu, and Zhaoli Jia. A variable neighborhood search approach for the resource-constrained multi-project collaborative scheduling problem. *Applied Soft Computing*, 107:107480, 8 2021. ISSN 1568-4946. doi: 10.1016/J.ASOC.2021.107480.

Farhad Habibi, Farnaz Barzinpour, and Seyed Jafar Sadjadi. Resource-constrained project scheduling problem: review of past and recent developments. *Journal of Project Management*, pages 55–88, 2018. ISSN 23718366. doi: 10.5267/j.jpm.2018.1.005.

Andy M. Ham and Eray Cakici. Flexible job shop scheduling problem with parallel batch processing machines: MIP and CP approaches. *Computers and Industrial Engineering*, 102:160–165, 12 2016. ISSN 03608352. doi: 10.1016/j.cie.2016.11.001.

Raf Jans and Jacques Desrosiers. Efficient symmetry breaking formulations for the job grouping problem. *Computers and Operations Research*, 40(4):1132–1142, 4 2013. ISSN 03050548. doi: 10.1016/j.cor.2012.11.017.

Matheus Leusin, Enzo Frazzon, Mauricio Uriona Maldonado, Mirko Kück, and Michael Freitag. Solving the Job-Shop Scheduling Problem in the Industry 4.0 Era. *Technologies*, 6(4):107, 11 2018. doi: 10.3390/technologies6040107.

McKinsey. Solving the paradox of growth and profitability in e-commerce. *McKinsey*, 2022.

N Mladenovir and E Hansen't'. VARIABLE NEIGHBORHOOD SEARCH. Technical Report 1, 1997.

David R. Morrison, Sheldon H. Jacobson, Jason J. Sauppe, and Edward C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2 2016. ISSN 15725286. doi: 10.1016/j.disopt.2016.01.005.

Anulark Naber and Rainer Kolisch. MIP models for resource-constrained project scheduling with flexible resource profiles. *European Journal of Operational Research*, 239(2):335–348, 12 2014. ISSN 03772217. doi: 10.1016/j.ejor.2014.05.036.

Nicolo Galante, Enrique Garcia, and Sarah Munby. The future of online grocery in Europe. *McKinsey*, 2013.

Manfred Padberg and Giovanni Rinaldi. A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. Technical Report 1, Society for industrial and applied mathematics SIAM, 1991.

Picnic. Picnic inhouse information, 2022.

Srijith Rajeev, Sabu Kurian, and Brijesh Paul. A modified serial scheduling scheme for resource constrained project scheduling weighted earliness tardiness problem. Technical Report 3, Federal Institute of Science and Technology FISAT, 2015.

Teun van Gils, Katrien Ramaekers, An Caris, and René B.M. de Koster. Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. *European Journal of Operational Research*, 267(1):1–15, 5 2018. ISSN 0377-2217. doi: 10.1016/J.EJOR.2017.09.002.

Guidong Zhu, Jonathan F. Bard, and Gang Yu. A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. *INFORMS Journal on Computing*, 18(3):377–390, 2006. ISSN 08991499. doi: 10.1287/ijoc.1040.0121.

II

Literature Study
previously graded under AE4020

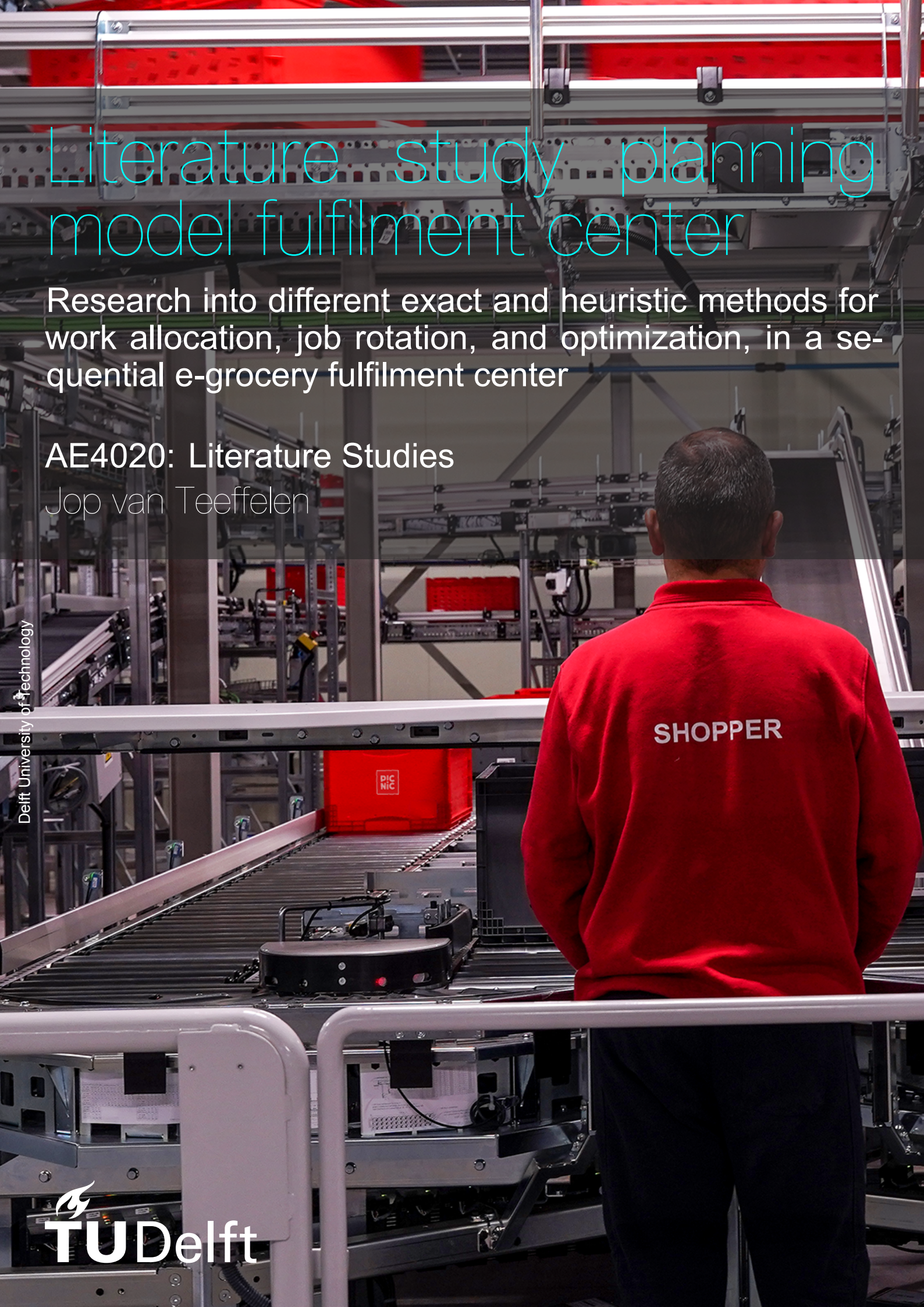
Literature study planning model fulfilment center

Research into different exact and heuristic methods for work allocation, job rotation, and optimization, in a sequential e-grocery fulfilment center

AE4020: Literature Studies

Jop van Teeffelen

Delft University of Technology



TU Delft

Literature study planning model fulfilment center

Research into different exact and heuristic methods for work allocation, job rotation, and optimization, in a sequential e-grocery fulfilment center

by

Jop van Teeffelen

Student Name	Student Number
Jop van Teeffelen	4484290

Instructor:	A. Bombelli
Picnic Supervisor:	Floris Boekema
Project Duration:	March, 2022 - December, 2023
Faculty:	Faculty of Aerospace Engineering, Delft

Cover: Automatic fulfilment center of Picnic located in Zwolle [69].

TU Delft

Summary

The literature study is focused on solving a planning and scheduling problem in an e-grocery warehouse. Today, Workers in an e grocery warehouse are allocated through supervisors, based on their expertise. A brief description on important factors to keep in mind during the planning process and a description on the current supply chain is discussed at first. After describing the practicalities in the planning process, and elaborating the steps within the supply chain of an e-grocery player, a problem description is described in words. Hence, the problem is translated into a mathematical formulation based on literature research into the subject of Resource Constrained Planning and Scheduling Problems. This literature research focuses on various exact and heuristic methods to solve the Resource Constrained Planning and Scheduling Problem. The exact solvers are based on the theory of Branch and Bound, Branch and Cut and Constraint programming. The heuristic solvers are used to give an approximation of the exact solution. The elaborated algorithms are based on the theory of Tabu Search and Adaptive Large Neighborhood Search.

Contents

Summary i

Nomenclature iii

1 Research Introduction 1
- 1.1 The history of e-groceries . . . 1
- 1.2 Picnic, a leading e-grocery company . . . 1
- 1.3 Fulfilment centers & planning . . . 2
  - 1.3.1 Activities . . . 2
- 1.4 Previous research on planning in FCs . . . 3
- 1.5 Outline of literature research . . . 4

2 Research Objective 5
- 2.1 Problem description . . . 5
- 2.2 Research objective . . . 6
- 2.3 Research question . . . 7
- 2.4 Sub questions . . . 7
- 2.5 Research planning . . . 8

3 Picnic's Operations 9
- 3.1 Problem description . . . 9
- 3.2 Tasks . . . 10
- 3.3 Activities . . . 11
- 3.4 Resource availability . . . 13
- 3.5 Scheduling . . . 13
- 3.6 Buffer analysis . . . 13
- 3.7 Key performance indicators . . . 14

4 Available Mathematical Models 15
- 4.1 Job shop scheduling problem . . . 15
- 4.2 Resource-constrained project scheduling problem . . . 16
  - 4.2.1 Resources . . . 17
  - 4.2.2 Concepts of activities . . . 17
  - 4.2.3 Objective function . . . 18
  - 4.2.4 Availability of information . . . 19
  - 4.2.5 Non-deterministic RCPSP . . . 20

5 State-Of-The-Art Solving Algorithms 22
- 5.1 Exact solving methods . . . 22
  - 5.1.1 Branch and bound . . . 22
  - 5.1.2 Branch and cut . . . 24
  - 5.1.3 Constraint Programming . . . 25
- 5.2 Online solvers . . . 26
- 5.3 Metaheuristics . . . 27
  - 5.3.1 Tabu search . . . 27
  - 5.3.2 Adaptive large neighbourhood search . . . 29
  - 5.3.3 Simulated annealing . . . 30

A Gantt Chart 37

Nomenclature

Abbreviations

Abbreviations	Definition
ALNS	Adaptive large Neigborhood Search
B2B	Business To Business
B2C	Business To Customer
B&B	Branch & Bound
B&C	Branch & Cut
BP	Binary Programming
CP	Constraint Programming
DC	Distribution Center
Epv	Electric Picnic Vehicle
FC	Fulfillment Center
IP	Integer Programming
JSSP	Job Shop Scheduling Problem
LNS	Large Neighborhood Search
MILP	Mixed Integer Linear Programming
MPP	Master Planning Process
NLP	Non Linear Programming
TS	Tabu Search
UPH	Units Per Hour
WPS	Workload Planning System
RCPSP	Resource-Constraint Planning and Scheduling Problem

List of Figures

1.1 Picnic's value chain [70] . . . 2
1.2 Core activities FC through different temperature zones [70] . . . 3

2.1 Workload Planning System [70] . . . 6
2.2 Timeline Msc. thesis Aerospace Engineering . . . 8

3.1 Example of task distribution in a fulfilment center [70] . . . 10
3.2 Tasks in a fulfilment center [70] . . . 11
3.3 Available workforce in two shifts of 8 hours [70] . . . 13
3.4 Scheduling process, dark blue the available workload, light blue scheduled workload, and grey bars indicating deadlines [70] . . . 13
3.5 Two schematic overviews of different buffers . . . 14

4.1 Different methods and applications to solve the Job Shop Scheduling Problem [48] . . . 16
4.2 Resource-constrained scheduling problem classification [36] . . . 17
4.3 Mixed Integer Linear Programming solution Space [89] . . . 19

5.1 Using branch-and-bound to solve an integer linear programming minimization [38] . . . 23
5.2 Different algorithms are within a B&B algorithm [58] . . . 23
5.3 Diagram of relationships between various algorithms [58] . . . 24
5.4 Tabu Search - Short Term Memory component [34] . . . 28
5.5 Tabu Search - Selection process of best admissible candidate [34] . . . 28
5.6 Illustrative example of a Large Neighborhood Search [98] . . . 29
5.7 Simulated annealing example of minimizing the objective function [31] . . . 30

A.1 Gantt chart indicating the planning of Msc. Thesis . . . 38

List of Tables

3.1 Key performance indicators used to measure the quality of the created set of schedules 14

5.1 Various optimization solvers for MILP formulations 26

1

Research Introduction

A research introduction is written in the first chapter. First, some background information about the history of e-grocery and its drivers are described in section 1.1. Next, a description of the operations of a leading e-grocery player named Picnic is described in section 1.2. After describing the value chain of Picnic, a deep dive into the operations and main activities of an e-grocery warehouse is described in section 1.3. Subsequently, section 1.4 illustrates the previous work done at planning within an FC (fulfilment center). Finally, the outline of the report is structured and described in section 1.5

1.1. The history of e-groceries

For over 20 years, difficulties around e-groceries have always played a significant role. Research has shown that a large number of people in Europe are interested in saving time while doing groceries online instead of pushing a cart down aisle after aisle [61]. On the contrary, this will only be the case if one can easily shop the same assortment as one is used to in the current supermarkets. As a result, an online supermarket will only be interesting whenever a full variety of products is available to buy online [26].

E-Commerce was already growing fast before COVID-19 hit. During the COVID-19 pandemic, the demand for online shopping grew tremendously. Digital commerce 360 estimates the pandemic contributed an extra $218.53 Bn. bottom line over the past two years in the U.S. only [17]. Moreover, the change in customer behavior and the propelled look of consumers to finding safe shopping alternatives lead to a growth of 55% in 2020, from 10% in 2019 [53]. As a result, a clear and well-developed operational model is required for sustainable and scalable growth. The biggest stumbling block in this growth has been logistics, and some inefficient operations which frequently led to capital being used up on operating expenses [61].

Therefore, an efficient strategy to organize and schedule employees efficiently is demanded. Within the supply chain of an e-grocery player, a lot of difficulties are faced within each step of the value chain.

1.2. Picnic, a leading e-grocery company

This research is focused on a large e-grocery player active in The Netherlands, France, and Germany named Picnic. Picnic is a unique grocery retailer with a concise data-driven decision process. The organization is solely focused on its digital presence through a mobile application where clients can buy their day-to-day groceries.

The main difference between Picnic and any other grocery retailer is their non-physical (thus devitalized) store. Picnic has zero physical shops available for its customers, which is a unique disruption in the industry. Picnic makes sure, after ordering groceries, 'a milkman' will be at a specific address within a 20-minute time window dependent on the time slots available the next day. Delivering groceries in a scheduled time window of 20 minutes is still a big challenge.

Figure 1.1 illustrates the supply chain of Picnic. The supply chain starts at its origin, the producer, who is creating and selling a product to a supermarket. The producer transports all of its quantities of products to a DC (distribution center). Here, Picnic distributes the products among more regional FCs. Hence, an FC replaces the grocery shop where workers (employees of Picnic) pick the groceries for you. After completing a fair amount of groceries for a specific region, the totes (crates to be filled with groceries) are put in a frame and shipped to more local hubs. The totes filled in the FC will be put in various small electric Picnic vehicles (epv) which will bring your groceries to your doorstep. The only point of physical contact with the client after ordering products online is during the delivery phase of the groceries and thus very important.

Clients of the online supermarket have an option to order groceries in two-time slots. If one orders groceries during a morning shift, the order has to be placed before 13.00 AM whereas the evening (or afternoon) deliveries have to be placed before 10.00 PM. As mentioned above, a time slot of 20 minutes will be given for which a milkman will deliver one's groceries at the front door. Taking in mind the value chain as described in Figure 1.1, groceries can not be processed through the whole process in 16 hours. As a result, lean and efficient operations are necessary to bring the right groceries in time.

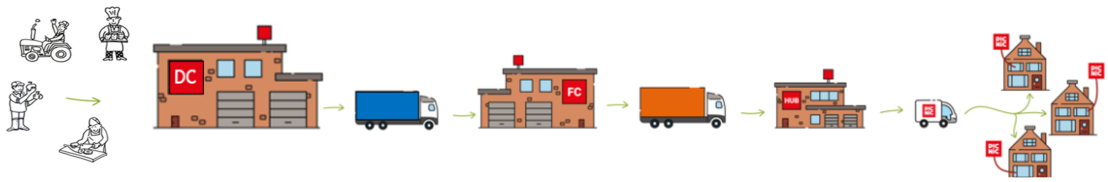


Figure 1.1: Picnic's value chain [70]

1.3. Fulfilment centers & planning

Last decade and especially during the COVID-19 pandemic, the grocery home delivery industry experienced rapid growth. Increasing competition forces grocery delivery companies to provide extraordinary service quality to their customers while operating as efficiently as possible. To achieve high service quality and maximize asset utilization, delivery services are provided throughout the whole day. Delivering from early in the morning until late in the evening implies that logistic processes in the supply chain have to be scheduled in parallel throughout the whole day.

As potential orders are abundant at Picnic, it is desired to schedule tasks among the employees optimally, such that maximum productivity at an FC can be reached while meeting a set of labor and task requirements. It is well known that solving the integrated problem of employee scheduling and job scheduling simultaneously would lead to overall better solutions. Nevertheless, solving this integrated process is generally considered too complex to solve in practice and named a NP problem [5].

1.3.1. Activities

A simplified overview of the activities at an FC is depicted in Figure 1.2. The figure illustrates the main activities based on truck level (sequential process after a truck arrives) in an FC. Trucks arriving at the FC first needs to be unloaded towards different temperature zones: ambient, chilled, and frozen. After unloading the trucks, the incoming groceries should be replenished on shelves such that every grocery is in stock in an FC.

Different products will be delivered by various trucks resulting in a parallel process before certain activities such as picking can occur. Whereas the second step shows the process of workers picking the groceries from shelves into totes intended for customers.

After completing a picking round, the totes have to be dispatched into frames. The third activity described in Figure 1.2, illustrates the process of filling empty frames with filled totes ready to be loaded into the trucks at the outbound station.

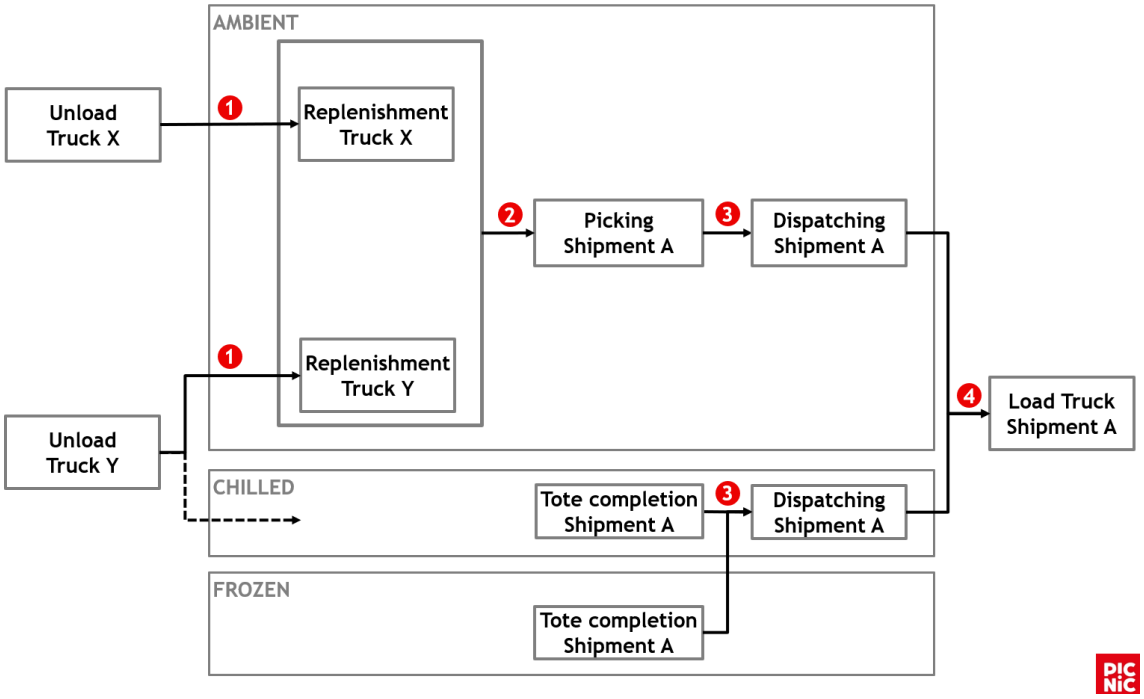


Figure 1.2: Core activities FC through different temperature zones [70]

The task allocation process is currently executed by FC captains. FC captains are experienced employees (former workers) with knowledge of every activity within an FC. The group of workers who arrives at the start of the shift is divided into various tasks based on certain deadlines each day. This research focuses on the study of creating a task allocation algorithm that will automatically divide certain tasks among workers to meet all deadlines automatically. Eventually, the created time schedules can be used as a guideline document to speed up this process.

1.4. Previous research on planning in FCs

A previous thesis intern has already worked on a part of the planning problem. During his research, an optimal task allocation algorithm was created to maximize the productivity of every worker. By solving a Resource-Constrained Project Scheduling Problem (RCPSP), the following research question is answered in his research:

To what extent can productivity at a grocery FC be maximized by solving a Multi-Objective Resource-Constrained Project Scheduling Problem? [30].

To solve the research question, a two-part heuristic approach is proposed consisting of a sophisticated serial SGS, called FTE-SGS-M in combination with a general variable neighborhood search (NVGS). Meta heuristics of which the fundamentals are similar to the heuristics used in other literature [30].

The research conducted is unfortunately not entirely feasible to use as fundament for this research. Difficulties in break management are not incorporated in the previous study. As a result, the MORCPSPs algorithm strives to optimize productivity without scheduling breaks creating unfeasible solutions. Additionally, the output of the algorithm is a graph indicating the aggregated workload of tasks to be executed on various time steps. The difficulties around breaks will be further described in chapter 2.

1.5. Outline of literature research

As described in the last section, this literature study will completely focus on how to solve the task allocation and scheduling problems in an FC. First, some more in-depth background knowledge is provided in chapter 2. Next, the main research question and other sub-questions are elaborated further in the same chapter. Subsequently, a mathematical formulation in the form of mixed-integer linear programming of the problem description is described in chapter 4. Eventually, three different exact solving methods based on the theory of Branch & Bound, Branch & Cut, and Constraint Programming are proposed and elaborated in chapter 5. Last, two Meta-heuristics based on the theory of Adaptive Large Neighborhood Search and Tabu Search are listed in section 5.3.

2

Research Objective

This chapter discusses the outline of the research. First, in section 2.1 the research problem is defined. Next, an explanation of the research objective follows in section 2.2. After which section 2.3 describes the main research question. Additionally, certain sub-questions are listed to provide a structured answer to the main research question in section 2.4. Last, the project planning is briefly elaborated upon in section 2.5.

2.1. Problem description

This year a new planning initiative has started within Picnic defined as Workload Planning System (WPS). The planning initiative consists of four sequential steps as illustrated in Figure 2.1 to sequentially schedule workers to various tasks. The WPS initiative is created to automatize the planning process to strive for a more efficient work stream.

Within the first step of the WPS, workload packages are created defining tasks that have to be fulfilled before a specific deadline. Hence, the amount of hours necessary to complete an inbound truck is forecasted by taking in mind the productivity of every worker. Outbound trucks are scheduled on a specific time slot indicating the deadlines for which a specific set of tasks have to be fulfilled.

In the second step, the availability of workers is matched with the aggregated workload packages defining the number of employees necessary to fulfill each task before every deadline. Assigning workers to a shift is done every Wednesday before a new workweek. The workload will converge to the actual number of hours and thus employees necessary, from the planning moment until the shift starts. The fluctuation of workload is caused by irregularities or other factors.

Taking in mind the four steps of the WPS, this literature research focuses on steps 3 and 4: scheduling workload and assigning workers to the right schedules. After receiving the workload packages, an efficient task allocation algorithm should provide schedules dividing the workload among specific tasks in certain time slots. As a result, a various number of unpersonalized schedules are created, necessary to fulfill every task on time. Step 4 will subsequently assign workers to the unpersonalized schedules to create personalized schedules for every employee active in the FC.

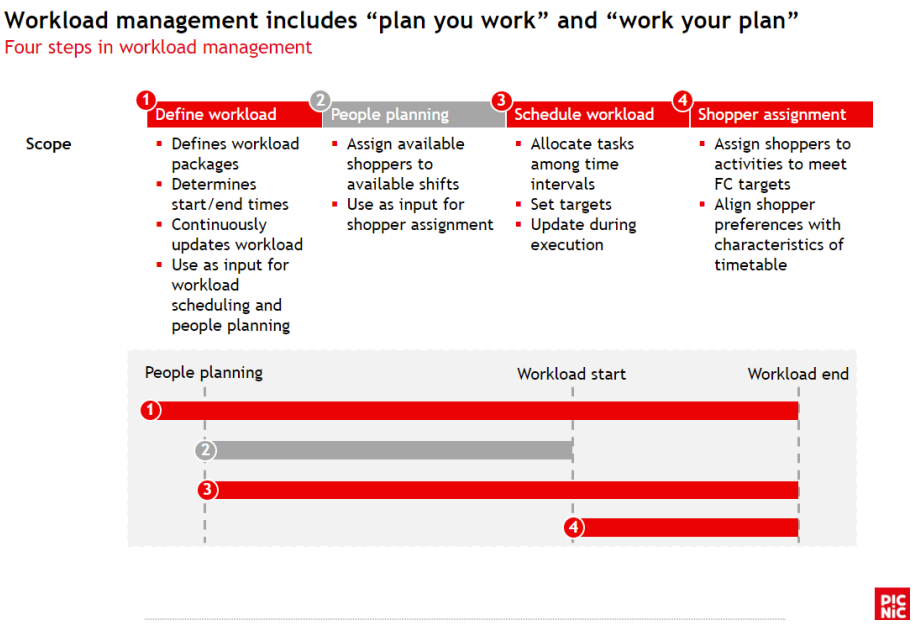


Figure 2.1: Workload Planning System [70]

2.2. Research objective

As described above, the current planning and operations of a fulfilment center heavily rely on the experience of an FC lead. Additionally, different perceptions on how to deal with efficient methods of planning are proposed depending on person to person. Taking into account the various levels of capacity per task, and the productivity of workers, the FC lead decides which worker will do what task. Picnic strives to optimize this productivity per worker with a key metric named UPH (Units Per Hour). UPH is a Key Performance Indicator that can measure each worker’s productivity per hour. A unit is a single item picked within a tote per hour. section 3.7 describes more possible key performance indicators to analyze the results of the algorithm and the overall performance.

Due to different interpretations and the urge to continuously maintain the highest efficiency possible to compete with competitors, Picnic strives for a highly efficient and automatized supply chain. One of the major components in an efficient supply chain requires an optimized planning model. The current planning model is coded within Microsoft excel which consequently results in a less scalable program and reaches a higher computation time.

The objective of this research is to optimally retrieve robust individual timetables for workers in an FC, minimize the total makespan of tasks, and by minimizing the number of switching moments of workers.

As a result, the created toolbox will be used as fundamental for allocating workers to tasks required that day. The expected outcome is to increase the UPH (Units Per Hour), decrease the number of employees required during a day, and add simplicity and computational power within the planning process [41]. By automatizing the scheduling process, a lot of time can be saved every shift setting up the planning during that day.

2.3. Research question

The main research question is split into several sub-questions to answer the main research question structured. Below each sub-question, a short explanation is provided of its relevance to answer the research question:

How can a flexible task allocation algorithm be devised to automatically retrieve timetables for individual workers in an e-grocery fulfilment center?

2.4. Sub questions

To structure the main question, several sub-questions are defined:

How does an ideal schedule for a worker in an FC looks like?

- *What tasks are scheduled daily in the FC and should be included in the time schedules?*
  Increasing the subset of activities results in the added complexity of the schedules due to sequential relations and capacity. All tasks should be scheduled during the day to retrieve a feasible solution.
- *What is the difference in urgency between all the tasks and how does this affect the planning process?*
  The core supply chain in an FC requires a higher urgency compared to other indirect tasks such as cleaning. Priorities have to be formulated to incorporate all tasks based on their level of urgency.
- *How to incorporate break teams and break management in the planning process of an FC?*
  Breaks play an important role in the planning process of Picnic and also add complexity. The availability of working hours will drop during breaks. Break teams are initiated by Picnic to create cohesiveness among the workers in an FC.
- *What are unwritten regulations that FC planners incorporate within their planning strategy?*
  Theoretical approaches often require some practical modifications to cope with difficulties encountered in real life. This sub-question is devoted to getting a better grasp on the difference between theory and practicalities defined in an FC.

How can one model the Resource-Constrained Planning and Scheduling Problem defined in an e-grocery FC?

- *What is the main objective while scheduling workers in an FC?*
  Currently, the minimization of the total makespan is the most interesting objective for Picnic. Additionally, a minimization of switching moments should increase the productivity of the workers. Next, the robustness of the schedules plays an important role in Picnic.
- *Which constraints should be implemented to create a feasible solution?*
  To create a solution space and ensure a feasible solution, various constraints have to be aligned and set up appropriately. The different constraints should be formulated up front.
- *What data is necessary to formulate the RSPCP model?*
  Picnic has a large Data Warehouse to keep track of all data available in the organization. Distracting and cleaning the right data to come to a feasible answer is required.
- *What different solution methods are available to solve the Resource-Constrained Planning Model?*
  Comparing the characteristics and outcome of an exact method with a metaheuristic solution technique is necessary to find the best use case for solving the RCPSP.
- *How can one optimally assign workers to timetables?*
  In the end, workers should be addressed to timetables based on preferences. A strategy should be defined based to link workers with schedules appropriately.
- *How to increase the robustness of the model to account for last-minute changes? (reactive, proactive, etc.)*
  A schedule is created as a short-term forecast [87] over the time a schedule is created. Last-minute changes will heavily influence the schedule and result in irregularities which causes a set of schedules to become infeasible.

How can the new task allocation algorithm be used to improve the current planning process of an FC?

- *How does the algorithm perform compared to the old planning method?*
  Way of comparing the current model versus the newly defined model. First, some performance indicators should be listed. Next, a comparison between both methods can be made.
- *How can the algorithm be implemented throughout the organization of Picnic?*
  Integration of the created model throughout the organization of Picnic and implementation in various FCs.

2.5. Research planning

The planning of the thesis for obtaining an Ir. Air Transport and Operations degree is subdivided into four phases. The first phase, the literature study is planned for 8 weeks until the kick-off meeting. In the initial phase, the literature study and a written research proposal (including the research questions, methodology, and planning) are written. After the kick-off meeting, the second phase (the initial phase) is initiated indicating the start of the research. Here, data will be gathered and a first draft of the exact and partly heuristic model will be created. After around 3 months, a mid-term meeting is scheduled to discuss the progress and also elaborate on the first results obtained with a model. The supervisory team of the Technical University of Delft will give feedback and steer the outcome of the research in the right direction. After incorporating the feedback and finalizing more results, a green light meeting is scheduled to end the Final phase. Hence, the green light meeting is a decisive moment in time to weigh up the relevance of the research and the quality of graduation. One month until the final defense is mandatory for preparation once the green light meeting is finished accordingly. Finally, a presentation is created summarizing the research and results within the last 4 weeks of the research. After incorporating the last feedback, and presenting the summary of the research, a defense is scheduled to ask questions about the research and conclusions. A broad overview of the timeline is illustrated in Figure 2.2

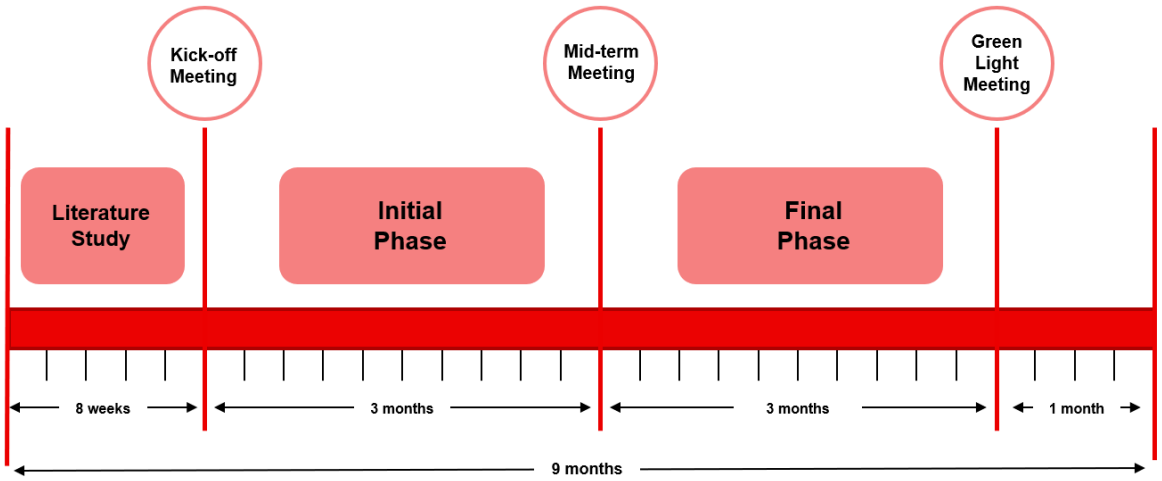


Figure 2.2: Timeline Msc. thesis Aerospace Engineering

3

Picnic's Operations

This chapter describes the operations of an FC in more depth and elaborates on the main problem description. First, a brief description of the problem statement is discussed in section 3.1. Second, the digestion of different tasks is described in section 3.2. While all different tasks available to schedule in timetables are listed in section 3.3. Next, an overview of how to deal with available employees during a shift is shown in section 3.4 and the scheduling process respectively in section 3.5. Next, the robustness of a schedule is described by theory on buffer analysis in section 3.6. Finally, some key metrics to evaluate the scheduling process is pointed out in section 3.7

3.1. Problem description

Picnic is a disruptive e-grocery player with extensive growth. Picnic's data-driven approach is unique compared to the competitors in the industry. As their organization is growing rapidly, each amortization causes simplifications and realizes a more efficient work stream. All tasks in an FC are tracked through an internal system. Therefore, the overall performance of operations and also each worker can be measured.

Today's planning of each FC heavily relies on the experience of the FC captain. The FC captain divides the work among the number of workers who are present during a shift. Its decisions are based on the number of employees available and various deadlines throughout the day. The planning principle is very sensitive to human errors and is not yet driven by any data. Additionally, the process of allocating people to specific tasks is very time demanding. around an hour at every start of a shift is used to create the planning for that day on all different FCs. Therefore, the research is focused on an algorithm to automatically schedule all employees on various tasks to meet every deadline on time and decrease the time necessary for the creation of the planning.

As described in chapter 1, the following two problem statements should be fulfilled within this research to allocate workers to time schedules:

1. **A flexible task allocation algorithm**
   Divide tasks dependent on various levels of urgency among adjacent timetables. Which is first done anonymously to optimize on a minimal makespan of all tasks.
2. **An employee satisfaction algorithm, linking workers to time tables**
   Link all workers to unpersonalized schedules based on their preferences.

Breaks

All activity schedules should include breaks. Every employee has the right to take a predefined number of pauses. Picnic strives to increase employee self-being and cohesiveness for which the organization introduces break teams. Teams are a group of workers with the same break times in which every worker can familiarize themselves with colleagues more frequently. workers within the same team do

not necessarily have to fulfill the same task at the same time and thus only have congruent breaks scheduled throughout their shift.

While planning workers, a break will directly influence the available hours of work during that time stamp. For example, within a shift at a fulfilment center, four identical teams are defined. If every team will take 15 minutes to break within the same hour, then for this hour the available hours will drop by 25%. Incorporating these phenomena while planning workers on a shift can result in not meeting every deadline in time and thus results in infeasible schedules. Additionally, the constraint of grouping employees into teams adds complexity to the planning process.

Preferences of workers

Next to teams and tasks, the preference of workers also has a direct impact on the operations. A fine line should be defined to what extent preferences are taken into account while scheduling workers. workers can theoretically do every activity but might have preferences on specific ones. For instance, if person A is not able to take part in the dispatching activity for longer than 1 hour due to disabilities or the heavy working conditions, this person should not be planning more than 1 hour on dispatching to optimize productivity. Which seems like a very logical decision. Therefore, Picnic is currently implementing specialized groups focusing on only one activity at the same time to increase productivity. For now, only operational preferences will be taken along and defined on 3 or 4 specific rules. These rules can be linked to the characteristic of a schedule.

3.2. Tasks

The introduction of the literature research already showed a shortlist of some tasks in an FC. The definition of a task is a piece of work that should be done. If an activity is scheduled the activity then transforms into a task. Figure 3.1 illustrates a subset of tasks scheduled on a day, each with a soft and hard deadline. Every task is defined from a given start time towards a deadline that has to be fulfilled. A task with a soft deadline is allowed to be delayed even though this is undesirable. On the contrary, hard deadlines can not be delayed which indicates the task has to be (partly) canceled or entirely stopped if not finished before the deadline. Every task has a fixed amount of work that has to be finished between the starting and ending time of the activity, illustrated in Figure 3.1.

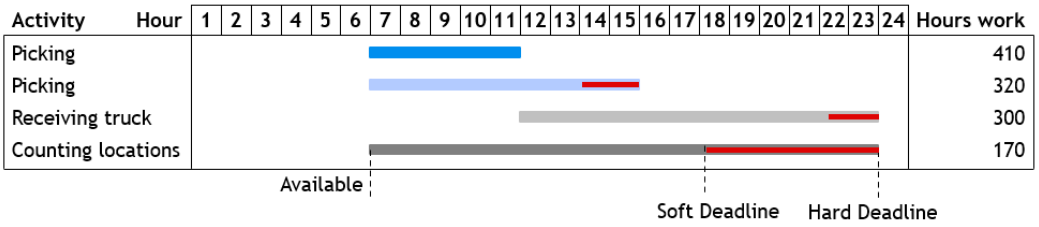
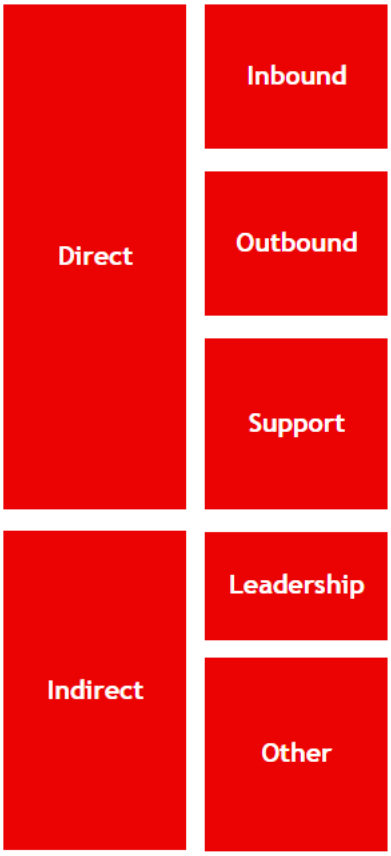


Figure 3.1: Example of task distribution in a fulfilment center [70]

Still, a distinction between tasks has to be formulated. First, certain tasks require to be executed entirely before sequential other tasks can start. For example, Trucks have to be unloaded before one can start replenishing. While everything should be replenished before a worker can start picking. All items should be in stock before the picking round can start. Additionally, various inbound trucks from different producers can result in stock differences/ emptiness.

3.3. Activities

An overview of the different activities in a fulfilment center is created in cooperation with Picnic and illustrated in Figure 3.2. A distinction between direct and indirect activities indicates different levels of contribution to the main operations at the fulfilment center. Direct activities are all executed by workers and indirect tasks are formulated for the sake of completeness to have an overview of the number of hours of work done in an FC. For this research, only the direct activities matter. Direct activities are subdivided into inbound, outbound, and supportive tasks. Both the inbound and the outbound activities will be listed with an urgency level of 1, indicating the core business activities within a fulfilment center and thus required to fulfill the grocery flow through an FC. Every supportive task is to make sure that inbound and outbound will operate more smoothly but is still mandatory to perform in a certain instance.



	Task	Diversification	Planning
Inbound	Transport inbound	AM/CH	tu/hr, %active
	Splitting	AM/CH	tu/hr, %active
	Replenishment	AM/CH	tu/hr, %active
	Buffer replenishment	AM/CH	tu/hr
	Return/waste	-	tu/hr
Outbound	Picking	AM/CH/FR	ols/hr, %active
	Dispatching	AM/CH	tts/hr, %active
	Tote handling	AM/CH	tts/hr
	Tote completion	CH	tts/hr
	Transport outbound	AM	tts/hr
Support	Counting	-	hr/day (staffel)
	Clearing	-	hr/day (staffel)
	Cleaning	-	hr/day (staffel)
	Slotting	-	hr/day (staffel)
	Urgent tasks	-	hr/day (staffel)
	Quality	-	hr/day (staffel)
Leadership	Captain	-	hr/day
	Trainer	-	hr/day
	FO-RO	-	hr/day
	Safety	-	hr/day
Other	Training	-	hr/day
	Recruiting	-	hr/day
	People admin	-	hr/day
	Canteen	-	hr/day
	Facilities	-	hr/day
	Projects	-	hr/day
		-	hr/day

Figure 3.2: Tasks in a fulfilment center [70]

Inbound

- **Transport inbound**
  Unloading carts from inbound trucks in the fulfilment center and preparing them for replenishment. This activity is only available in ambient zones.
- **Splitting**
  A special activity focused on the reallocation of goods on different carts. Necessary to optimize the replenishment. This activity can be done in ambient, chilled, and frozen zones.
- **Replenishment**
  Stocking the inbound goods on the right shelf in the fulfilment center. This activity can be done in ambient, chilled, and frozen zones.
- **Buffer Replenishment**
  Stock the items from the buffer zones on the right shelf in the fulfilment center. This activity can be done in ambient, chilled, and frozen zones.
- **Return/Waste**
  Emptying and returning the waste from the waste bins on the operational floor in the fulfilment center. This activity can be done in ambient, chilled, and frozen zones.

Outbound

- **Picking**
  Filling grocery baskets according to the orders of the customer. This activity can be done in ambient, chilled, and frozen zones.
- **Dispatching**
  Relocating the filled totes in a framework ready for transportation towards the hub. This activity can be done in ambient, chilled zones.
- **Tote handling**
  Filling empty frameworks with totes to prepare to pick cars for picking. Only for ambient and chilled zones.
- **Tote completion**
  Covering the totes and filling every bag with a cool element. This activity can be done in chilled and frozen zones.
- **Transport outbound**
  Filling the outbound trucks with the filled frames ready for transport to the hubs.

Support

- **Counting**
  Double-check if the item count in the system matches the real item count on a shelf. Necessary to calibrate the WPS accordingly.
- **Clearing**
  Emptying or clearing shelves with expired or dis-functional goods. This activity can be done in ambient, chilled, and frozen zones.
- **Cleaning**
  The act of removing dirt & other undesirable circumstances within a fulfilment center. This activity can be done in ambient, chilled, and frozen.
- **Slotting**
  Activities defined as slotting are necessary to reallocate goods or products in different locations to increase productivity. This activity can be done in ambient, chilled, and frozen.
- **Urgent tasks**
  Urgent tasks are scheduled time slots for workers to assist with difficulties that require immediate attention.
- **Quality**
  Quality is scheduled to double-check the freshness and the standards of the products picked.

3.4. Resource availability

The planning of an FC is limited by the workforce available, assuming all workers have equal skills and productivity. In other words, the resource available workforce is constrained by an X amount of available workers that day. Figure 3.3 illustrates the available workload in two shifts of 8 hours constrained up to 100 hrs. during shift 1 and 50 hrs in shift 2. The workforce is dependent on the number of workers scheduled. During this research, a single resource type is considered.

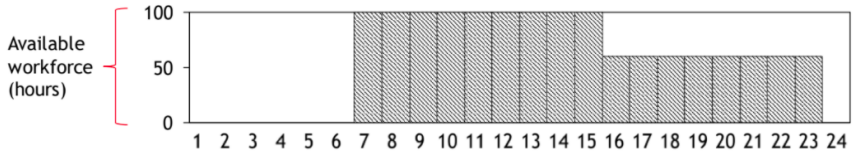


Figure 3.3: Available workforce in two shifts of 8 hours [70]

3.5. Scheduling

It is desired to generate a schedule with maximum productivity from the available workforce visible in Figure 3.3. Figure 3.4 illustrates an example of scheduling persons to a work plan and meeting every deadline accordingly. From the same figure, one can see the deadlines being accomplished in time. The deadlines are indicated with grey bars showing the work that have to be done during that moment in time.

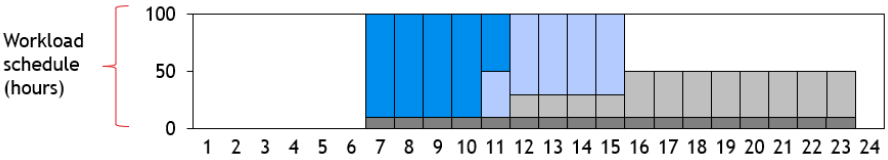


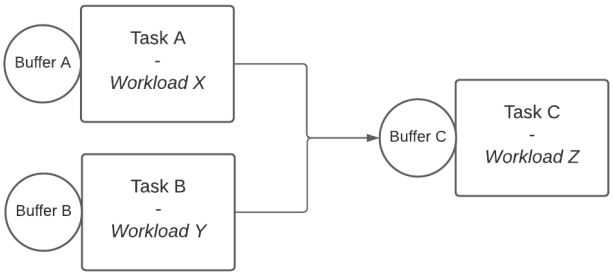
Figure 3.4: Scheduling process, dark blue the available workload, light blue scheduled workload, and grey bars indicating deadlines [70]

3.6. Buffer analysis

Buffer time between various tasks is a way of increasing the robustness of a schedule. The schedule can deal with irregularities the moment activity is delayed for various reasons and be corrected by introducing buffer zones between tasks. By maximizing the buffer between two tasks, a more robust model is created whenever undesired events occur.

Figure 3.5a and Figure 3.5b describes two scenarios for which a buffer should be optimized. Without proper scaling, buffers would receive higher whenever a light workload process is finished compared to a heavy workload process. For this reason, buffers require scaling to be incorporated appropriately within the model. If for example, a task has multiple predecessors the task can be executed, a buffer needs extra scale to prevent the model from favoring 'lighter' buffer moves over others in the network. To incorporate this a weight factor is included within the model. The buffers are normalized concerning their last task. In the illustrative example of Figure 3.5b, this would mean scaling workload A with $\frac{A}{A+B}$ and workload B with $\frac{B}{A+B}$. Similarly, the one-to-one buffers can be scaled accordingly.

(a) One-to-one buffer



(b) Multiple-to-one buffer

Figure 3.5: (a) One-to-one buffer (b) As for (a) but with two tasks which have to be fulfilled before task c can start [70]

3.7. Key performance indicators

After describing Picnic's main operations and the logic behind scheduling tasks, key performance indicators should be defined to analyze the quality of the schedules. Key performance indicators are necessary to compare different solutions with each other. The research is focused on determining an optimal set of schedules for every worker, working in a fulfilment center. An exact method will be compared to a heuristic approach

Key Performance Indicators	Units
Productivity	Units Per Hour (UPH)
Utilization rate	[%]
Buffer time	[hrs]
No. of workers required	[#]
Computational time	[sec]

Table 3.1: Key performance indicators used to measure the quality of the created set of schedules

Units per hour

Productivity is mainly measured in terms of UPH. Whenever a schedule is feasible and optimally scheduled, an improvement in UPH should be measured. In this case, UPH will only be measured in real-life experiences since it is also dependent on many other factors.

Utilization rate

The utilization rate is defined as $\frac{\text{Amount of hours scheduled}}{\text{Total workload in hours}}$.

Buffer times

The buffer times are defined as the number of hours available between the following dead

No. of workers required

The total amount of workers required in a shift to operate smoothly.

Computational time

The computational time will matter as an exact model will always give a 'better' solution compared to a heuristic algorithm. The downside of an exact solver is the computational time for large instances. If one can have a decent solution in 5 seconds compared to the exact solution in 3 hours, one can relate the relevance of the answer. Especially before a shift starts.

4

Available Mathematical Models

This chapter elaborates further on the mathematical model used to describe the problem description from chapter 3. A job shop scheduling problem is described in section 4.1 to introduce a definition of the problem statement. Next, an adaptation to the job shop scheduling problem defined as a Resource-Constrained Planning and Scheduling problem is described in section 4.2. Here, a link to the problem description at Picnic will be made. Finally, a conclusion is discussed in subsection 4.2.5.

4.1. Job shop scheduling problem

One of the oldest and most important studies in machine scheduling is around Job Shop Scheduling Problems [6]. A Job Shop Scheduling problem is an optimization problem mainly addressed in operations research and computer science to schedule jobs among machines or employees. JSSPs are considered NP-hard problems, in other words among one of the hardest combinatorial optimization problems to solve [16, 6, 48, 3, 37]. Informally, the problem can be described as follows: there are a set of jobs and a set of machines. Each job consists of a chain of operations, each of which needs to be processed during an uninterrupted period of a given length per machine. Each machine can only process at most one operation at a time. A schedule is an allocation of the operations to time intervals on different machines. The objective of JSSP is to find a schedule with a minimum makespan to complete the operations. JSSP has inherent intractability characteristics which result in a preference for solving the problem with a heuristic procedure [15].

Today, many applications on the Job Shop Scheduling problem exist. Figure 4.1 illustrates various methods to solve the JSSP. Yet, The most basic and best understandable version of a job shop scheduling problem is defined in the research of Applegate [2] and illustrated as follows:

The model is defined with a finite set J of jobs and a finite set M of machines. Furthermore, let $m = |\mathbf{M}|$ be the set of jobs and $n = |\mathbf{J}|$ the set of machines. With a sequential set of jobs, the processing order is defined as $(o_{j,1}, ..., o_{j,m})$ of **M**. Additionally, each job j and machine k is associated with a non-negative, integer valued processing time $p_{j,k}$. The different steps a job has to complete on different machines shall be further referred to as operations, whereas $o_{j,i}$ shall be read as "the machine of operation i of job j, while $p_{j,o_{j,1}}$ shall be read as 'the processing time of operation i of job j'. Furthermore, a schedule is created assigning each operation a start time $s_{j,k}$ for all $j \in \mathbf{J}$, $k \in M$. Applegate addresses the following characteristics to come to a feasible solution:

- *All start times must be positive*
  $s_{j,k} \geq 0$ for all $j \in J, k \in M$.
- *The operations of a job must be processed sequentially*
  $s_{j,o_{j,i}} + p_{j,o_{j,i}} \leq s_{j,o_{j,i+1}}$ for all $j \in J, 1 \leq i < |M|$
- *There is no overlap between operations processed on the same machine*
  $s_{j,k} \geq s_{i,k} + p_{i,k} \vee s_{i,k} \geq s_{j,k} + p_{j,k}$ for all $i, j \in J, k \in M, i \neq j$.

Job-shop Scheduling Problem	Approaches for dealing with the JSP	Heuristic Rules		
		Classical Optimization		
		Artificial Intelligence (A.I.) approaches		
	Scheduling Flexibility	Classical Approach		
		Dynamic approach	Rule used for creating the scheduling	Completely reactive scheduling
				Predictive-reactive scheduling
				Predictive-reactive robust scheduling
				Robust pro-active scheduling
			Decision modes	Simulation based approaches
				Artificial intelligence based approaches
				Agent-based approaches
	Scheduling updating and Rescheduling	Rescheduling policies	Event-based rescheduling	
			Periodic rescheduling	
			Hybrid rescheduling	
		Updating an existing scheduling	Right-shift rescheduling	
			Complete rescheduling	
			Partial rescheduling	

Figure 4.1: Different methods and applications to solve the Job Shop Scheduling Problem [48]

4.2. Resource-constrained project scheduling problem

One of the main limitations of a JSSP, applicable to the problem formulated in section 2.2, is the resource consumption of tasks and the capacities of resources are unitary. In other words, one machine can only process one task at a time. Additionally, one task (operation) requires only one resource (machine) to be fulfilled. In reality, one task (e.g. picking) has to be completed by many workers (machines) during the same time interval. Next, precedence constraints form chains within a job-shop algorithm and the preemption of activities is not incorporated within the classic Job Shop Scheduling Problems. Finally, the no. of workers (or resources) in an FC is constrained for which the JSSP can not hold. As a result, an alternative to the JSSP should be formulated.

Another way of addressing the problem description denoted in chapter 2, is through formulating the problem statement as a Resource-Constrained Project Scheduling Problem (RCPSP) [36]. RCPSPs determine the time required to implement the activities of a project to achieve a certain objective. Classical RCPSP has two restrictions within its formulation. First, precedence relations, indicating some activities must end before another can start. Second, resource constraints are formulated where renewable resources are available at a constant and fixed rate throughout the project. One of the most common additional restrictions is that some activities must end before a given due date and time. [73]. The origin of this problem is referred to as a JSSP and was first formulated by J. Blazwicz & J. Lenstra in [11]. Hence, both the JSSP and the RCPSP are defined as NP hard problems, indicating their difficulty in resolving the mathematical formulation.

RSPCP formulations are considered by researchers as one of the most commonly used and solved problems today, addressing the urge and the quantity of research in the domain [36, 11, 73]. For both the JSSP and RCPSP, many branches and versions are created for any specific problem. A classification scheme is created to categorize which version of RCPSP is applicable for the problem researched and illustrated in Figure 4.2.

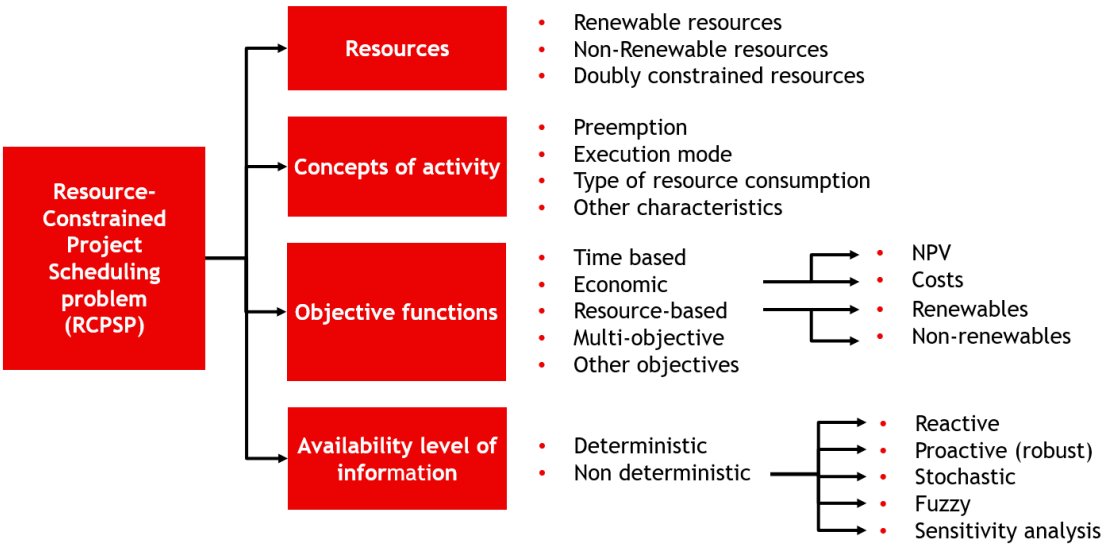


Figure 4.2: Resource-constrained scheduling problem classification [36]

4.2.1. Resources

A distinction between renewable and non-renewable resources is created in Figure 4.2. Renewable resources are periodically refilled and are exploited and returned at the end of the process. However, non-renewable resources are a type that will have limited availability within the project horizons. Within Picnic's planning system, an assumption is made based on the unlimited availability as the non-periodicity of the products. Other systems such as the Master Planning Process, will take care of this problem. MPP (Master Planning Process) is the process of planning the Picnic supply chain path for every ordered article from the FC shelves to the customer door.

4.2.2. Concepts of activities

One of the main assumptions at a JSSP (or at a classic RCPSP) is that activities can not be interrupted or discontinued. However, in reality, some of the activities have to be stopped due to dis-functionalities, equipment repair, or destruction [14]. Literature shows several ways of handling activities that can stop at any time in discrete points of time [1, 60, 7] a maximum number of preemption m [101], and Tavana [83] proposed three different features which have to be incorporated in the model. First, a minimum duration of processing activity before the first interruption is defined. Second, a maximum number of preemption during one activity is regulated. Third, a maximum duration as the limitation for the duration between the interruption and restart of the activity is defined.

Last, Cavalcante [19] elaborates on the usage of human resources on every task specifically. Moreover, he demonstrated that different No. of workers is used in every activity and the No. of workers is dependent on the demand. At Picnic, the number of workers required heavily relies on the demand per day. Within the organization, a forecasting team is very well informed on future demand & work necessary that day, which in practice makes the problem lighter. In literature, the demand is kept as a variable and additionally, the required amount of work in Hrs is not known on forehand.

Breaks

To improve the well-being of all workers, Picnics strives to create cohesiveness by creating teams of workers with the same break schedules. A team is defined as a group of workers each having a break at the same time spot as a group of colleagues. The workers do not have to perform the same task before having a break together. Creating break groups automatically increases the complexity of the schedule while more constraints have to be added. Moreover, breaks also influence the productivity of the whole operation while the number of workers taking a break will lower the available hours for that

moment in time. If 25% of all workers take a break at the same time, the total productivity will automatically be 75% of the whole process. Zhang evaluated the preemptive scheduling under break and resource-constraints (PSBRC) where the activities stopped during the break could not be immediately restarted in the next period [97]. More research into preemptive scheduling is done by M. Fallah [40] on uncertainties & access to resources accordingly.

4.2.3. Objective function

There are several key objectives to keep in mind while solving the Resource-Constrained Planning and Scheduling Problem. First, a "time-based" perspective will be addressed in subsection 4.2.3. Next, an economic and multi-objective objective function is addressed in the literature to solve the planning problem and described in subsection 4.2.3. Last, subsection 4.2.5 elaborates on a way of optimizing the buffer time to increase robustness.

Time-based objective

A time-based objective function such as a minimization of the total makespan (C_Max) is most commonly used for RCPSP formulations. While the probability of activities taking longer than expected is more common in other real-time scenarios. Creemers has analyzed the probability of activities with a certain level of delay while minimizing the expected completion time of projects. Additionally, lateness and tardiness are minimized as another time-based objective function. The lateness is defined as the difference between the completion time C_j and the specific delivery time d_j resulting in $L_j = C_j - d_j$. While tardiness (T_j) is defined as a parameter excluding negative time units on a interval ($T_j = max\{0, C_j - d_j\}$).

Slowinski described a way of minimizing the total lateness in work as an addition to minimizing the total makespan [79]. As a result, all schedules are filled and a strong focus on late work is incorporated into the algorithm. There is always plenty of work available in an FC to incorporate into schedules. The high amount of work can result in long schedules if all is scheduled. Slowinski found a way to incorporate a minimization of the lateness of all schedules such that no employee will be working too long. Drezet strives to incorporate various levels of multi-skilled humans in its solver as every worker has another level of productivity [22]. Within an FC, a distinction between trainee and shopper is made. The productivity of an inexperienced shopper, a trainee, is lower than a normal worker based on their experience.

Jing Xiao has published two different reports on optimizing both the project completion time and the total tardiness in [91, 90]. Hence, in some conditions, it is desired to minimize the duration between the completion time and the deadline of the activities. This will influence the robustness of the schedule since the buffer time will be fairly small. On the contrary, fresh products will stay fresher since they will not be in a warm atmosphere for a long duration.

A maximization of duration between the deadline and task completion time is preferred due to multiple deadlines during the day to increase the robustness of the model.

Economic objective

By analyzing objective functions, one can also address the costs involved. Minimizing costs does not necessarily have to result in scheduling the cheapest labor and getting rid of the senior employees. In contradiction, costs can also be addressed as penalties defined arbitrarily for every unwanted event. Eventually, certain penalties and costs will also add up to an economic function. The cost-based functions do not necessarily have to be defined fairly differently compared to a time-based function. In certain project scheduling problems, a trade-off between costs and time is created [36]. Berthaut for example, defined a time-cost trade-off problem while considering the acceleration of carrying out the project with overlapping sequential activities [68]. A cost function can be defined which minimizes the penalty costs in addition to the total costs of crashing an activity.

In other studies, costs are corresponding with tasks defined in a certain level of urgency. Rajeev [73] tried to schedule the activities in such a way that the total weighted penalty costs are minimized. This is done according to the precedence constraints and resource constraints. Furthermore, a couple of researchers have analyzed the costs C_{jt} for an activity j based on the initiative time t [55, 56].

Multi-objective
Cost functions can thus result in multi-objective functions while optimizing only one cost function. Other proposed options to incorporate multiple objectives are by proposing multi-objective formulations or through a MILP (Mixed Integer Linear Programming). Combining both the probability of activities taking longer than expected, Xiong et al have analyzed a way to assess so-called entitled stochastic extended resource investment project scheduling problems (SERIPSP) [94]. Hence, three different objective functions are optimized to answer the SERIPSP problem:

1. First, the maximum amount of changes in resources or workers is minimized. Hence, more switching moments will result in being less productive in reality. His objective function is also used in a study of Wang [92]
2. In the second objective function, Xiong optimizes the problem to minimize the number of resources that exceeds the permitted level. Moreover, the second objective function is also analyzed in [9]
3. Thirdly, the total changes in the consumed resource level are minimized during each period compared to the previous one. Eventually, Zhou L. and Nikoofal exploited this specific objective function [99, 62]

Mixed integer (linear) programming
All literature on JSSP indicates several constraints which are required to be met for a feasible solution. Several ways of checking the feasibility after solving the problem are mentioned in literature [4, 2]. The usage of a Mixed Integer Linear Problem will resolve the problem of validating an answer on feasibility at hand. A MILP is a formulation in which several variables are constrained to be an integer. All constraints are formulated as a linear function creating a solution space that indicates the feasibility of the solution. All answers in the solution space created by constraints will lead to a feasible solution. Allocating workers to a predefined schedule could also be done through a MILP. Hence, multiple outcomes are feasible as various time schedules can be assigned to different workers. A solution space is illustrated in Figure 4.3 where every grey dot illustrated indicates an integer solution.

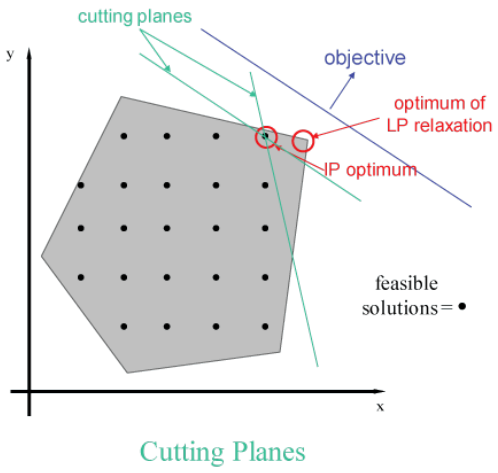


Figure 4.3: Mixed Integer Linear Programming solution Space [89]

4.2.4. Availability of information
The main goal of project scheduling is to provide a standard schedule for more accurate control. By automatizing the scheduling process, a feasible solution does not have to be the optimal solution. On the contrary, it shows a close approximation of the optimal solution. Moreover, before a shift starts, the schedules have to be ready. Indicating that a lot of things will or can change during the shift (longer time for tasks, bottlenecks, etc.) and thus heavily relies on the accuracy of predefined information. In literature, a distinction between deterministic and non-deterministic is made based on the amount of information available. Picnic has a strong forecasting team available with strong capabilities for

creating very precise information packages used as input for the model. Yet still, Irregularities occur which influences the feasibility of the schedule. A schedule can be seen as a short-term forecast before a shift starts. Hence, uncertainty always plays a role in predicting the future. A better understanding of non-deterministic tasks is further elaborated in subsection 4.2.5.

4.2.5. Non-deterministic RCPSP

RCPSPs are often faced with major uncertainties, negligence, or several other dysfunctionalities causing delays or errors in their short-term forecast. Resulting in various strategies involving probabilistic approaches to approximate a real-time solution. Other reasons a schedule can be different within Picnic's operations:

- Tasks can take shorter or longer than expected which will directly influence other deadlines due to the precedence relations in the FC process.
- A variation of starting times is unavoidable due to delayed trucks or other urgent matters. Additionally, some deadlines are easier to cope with due to the delay of outbound trucks.
- Unpredictable natural weather influences also directly impact the supply chain of Picnic directly and indirectly. Strong storms can cause extreme conditions in which the small electric vehicles can not drive while the cars will be blown over. In this case, the groceries are delayed and irregularities in the supply chain occur.
- Incidents occur which require additional tasks such as cleaning or urgent moves. Eventually, incidents will cause a delay in the main operations of an FC.

Several methods are denoted in literature to incorporate the uncertainties listed above into a model. Both reactive and proactive (Robustness) scheduling reacts directly to one of the irregularities denoted above.

Reactive scheduling

The approach of reactive scheduling is used during the project implementation phase and is based on actual information. The outcome, a basic schedule, is reviewed and revised whereas future uncertainties do not play a role. In other words, the reactive approach will create a new schedule based on new actual information (with an unexpected event) as the basis of the already created schedules.

The creation of new schedules requires a significant processing time due to the complexity of the new calculations that have to be fulfilled. Therefore, a decision on when to include a new schedule has to be answered first. This decision can be statically revised at any time by recalculating a feasible solution, keeping in mind the new actual information raised from unexpected events. Literature by Bierwirth elaborates on different methods of creating a new schedule based on occurring events [10] orVierira discusses the creation of new schedules based on predetermined intervals [88]. Strategies on how to create new schedules are incorporated by Suwa [57] or Chakrabotti [96]. Suwa uses a minimization of the tardiness of starting time in the new schedule to the previous model. Chakraborty on the other hand used a trade-off between project completion time and a weighted penalty of deflection from the previous situation to the current.

Proactive (robust) scheduling

Another method of reacting to undesired events is to make sure a buffer is implemented within the schedule. Several methods of defining an RSPCP with additions are listed within this chapter. Additionally, a short notice about robustness is already mentioned in subsection 4.2.3. Here, Lambrechts focuses on three specific topics to increase robustness in a schedule. Lambrechts uses resource buffering, time buffering, or float times in his report [46].

Resource buffering is used to ensure access to resources specifically which can also be interpreted as using the allocation of resources for every new activity. In other words, a percentage of resources is kept safe when approximating the robustness based on the proactive scheduling method.

Buffer's time insertion directly increases the robustness of a set of schedules. The time buffer method is utilized by maximizing the time between two sets of tasks in which the first task has to be completed

before the second activity can start. Within this approach, two important decisions have to be made. First, the total buffer time allocated to tasks has to be defined as a value to determine the total robustness costs. Second, a decision on which tasks are involved and what buffer times are applicable for the same tasks. Palacio proposed an exact method to maximize the robustness while minimizing the project completion time or makespan [64]. The problem denoted in Palacio is defined as a MILP used to find a solution within the solution space illustrative in Figure 4.3.

Conclusion

The JSSP method is the oldest and most frequently used method within the planning and or scheduling literature. The constraints defined with a JSSP are not yet applicable to the current situation of Picnic due to sequentially tasks and a full group of workers who can perform the same task. Additionally, the preemption of tasks is not integrated within JSSP formulations. The Resource-Constrained Project Scheduling Problem is a branched version of the JSSP incorporating the constraints necessary for an e-grocery FC. Digestion on several definitions and literature about specific RSPCP is presented in the last chapter. Conclusively, literature about RCPSP problems incorporating information on renewable resources, Preemption of tasks, Time based & multi-objective economic objective functions, and a non-deterministic approach offers various research to solve the main research question.

5

State-Of-The-Art Solving Algorithms

This chapter is devoted to discussing state-of-the-art solution methods for RCPSP formulated as a MILP. An exact solution method such as Branch & Bound, Branch & Cut, and Constrained Programming is proposed in section 5.3 to solve the resource-constrained planning and scheduling problem. Next, an approximation to solve the problem within less computational time is proposed in section 5.3, here the fundamentals of a Tabu Search algorithm, an Adaptive Large Neighborhood Search, and the theory of simulated annealing algorithms are elaborated.

5.1. Exact solving methods

Ideally, an exact solution is found by solving the RCPSP formulation to an optimal solution. The size of the problem, however, causes a high amount of computational time to solve through all the different solutions. Additionally, the similarity of the problem formulation hypothetically causes indefinite solutions which are possible to be redirected. Hence, an assumption on how to deal with a workload package of 8 hrs already has a lot of different solutions; distribute the task among 8 workers for 1 hour or leave 1 worker executing the job for 8 hours.

5.1.1. Branch and bound

A Branch and Bound (B&B) algorithm are often used in literature to solve RCPSPs. Additionally, the B&B framework is a fundamental and widely- used methodology for producing exact solutions to NP-hard optimization problems [58]. The technique is first proposed by Land and Doig [47] and is often referred to as an algorithm. However, as the name of the 'algorithm' implies, the B&B encapsulates a family of algorithms that all share a common core solution procedure. By making use of a tree search strategy to implicitly enumerate all possible solutions to a predefined problem while applying pruning rules to eliminate regions of the search space that can not lead to a better solution. Unexplored nodes in the tree generate children by partitioning the solution space into smaller regions that can be solved recursively (i.e., **branching**), and rules are used to prune off regions of the search space that are provably sub-optimal (i.e., **bounding**) [58]. By limiting the solution space to smaller regions and using rules to prune off regions, the computational time is dramatically lowered. Within the B&B process, an estimation of an optimal upper and lower bound is performed. A branch is only added to the tree if and only if the next solution is between the two bounds.

Figure 5.1 depicts the process of B&B. Hence, the blue arrows depict the node expansion order while the red nodes are depicted as the optimal nodes.

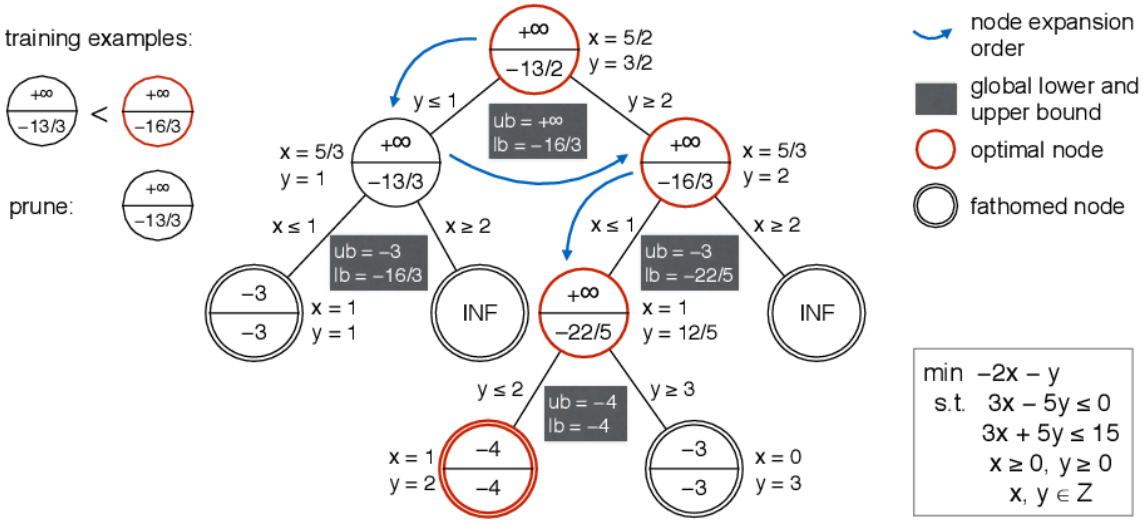


Figure 5.1: Using branch-and-bound to solve an integer linear programming minimization [38]

Figure 5.2 illustrates research conducted in B&B algorithms with its possibilities. State-of-the-art alternatives are grouped within three buckets: search strategies, Branching strategies, and pruning rules. During the first phase of any B&B algorithm, the search phase, the algorithm has not yet found an optimal solution x*. The second phase, the verification phase, is initiated whenever the incumbent solution is optimal but there are still unexplored subproblems in the tree which can not be pruned. Hence, no incumbent solution cannot be proven optimal before no explored sub-problems remain. If the algorithm manages to complete the verification phase, it can be stated as optimal. The Algorithm now has a certificate of optimality. Each of the algorithms (search strategy, branching strategy, and pruning rules) plays a distinct role in B&B algorithms. A diagram of relationships between various algorithm components is shown in Figure 5.3. The figure depicts solid lines, indicating a generalization relationship. In other words, many constraint programming techniques generalize cutting planes and dominance relations. While Column generation techniques can not be strictly defined as a generalization of other techniques (in essence, column generation adds cutting planes to the dual optimization problem to improve the computed lower bound [58]). Additionally, Figure 5.3 indicates the cohesion of all three main strategies. In particular, the choice of an algorithm within pruning rules often impacts and limits the choices of other algorithms in the other two domains.

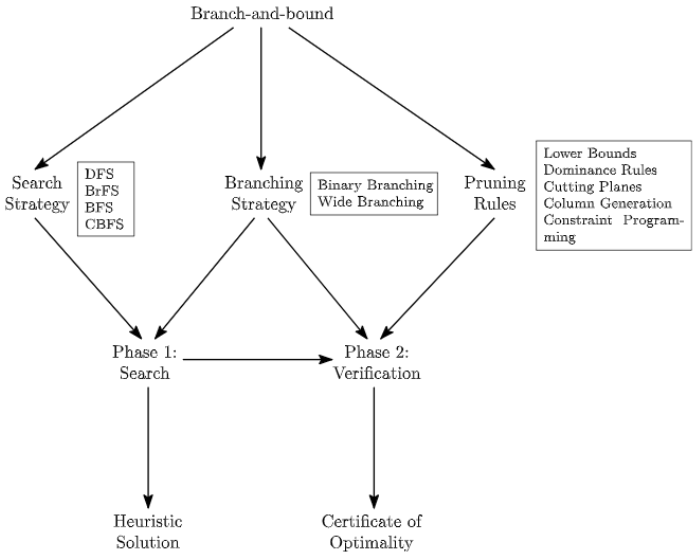


Figure 5.2: Different algorithms are within a B&B algorithm [58]

For this research on solving an RCPSP, a better understanding of pruning rules (more precise, cutting plains, dominance rules, and constraint programming) is necessary. More information about the other strategies and algorithms can be found in literature [58].

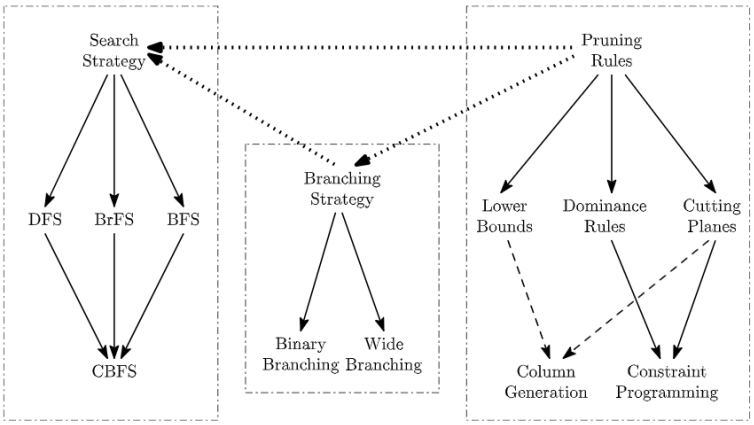


Figure 5.3: Diagram of relationships between various algorithms [58]

Dondorf initiated a time-orientated B&B algorithm with preemption constraints [20]. Here, the B&B algorithm relies to a great extent on efficient constraint propagation techniques. Constraint propagation is an elementary method for reducing the search space through repeated analyses and evaluation of variables, their domain, and interdependence between variables induced by a set of constraints. Tsang, elaborated on the principle to detect and remove inconsistent start time assignments in [84]. Additionally, their branching scheme generates at least all active schedules, so that traversing the search tree will result in an optimal solution. The lower bound is not indicated within this scheme. However, Stinson proposes a B&B algorithm of assigning feasible start times to a set of tasks making up a project statically. This is done under two constraint sets: (1) no activity may be started until all activities as its predecessor set have been completed and (2) The total resource requirements of all tasks in process at any time in the schedule must not exceed the level of availability [80]. Stinson's paper almost overlaps directly with this research without taking care of breaks or a linear formulation of tasks that have to be finished a certain percentage before the following activity can start. Moreover, Brucker also initiates a B&B algorithm for tasks to be processed without preemptions. The branching scheme starts from a graph representing a set of conjunctions (the classical finish-start precedence constraints) and disjunctions (induced by the resource constraints). Then it either introduces disjunctive constraints between pairs of tasks or places these tasks in parallel. Concepts of immediate selection are developed in connection with this branching scheme. The immediate selection allows for addition conjunctions as well as further disjunctions and parallelity relations [12]. His work originating in 1998 already has a lot of overlap with Stinson's method and the proposed method for this research. Yet, the break management and further implementation of larger data sets are still lacking.

5.1.2. Branch and cut

Branch and cut (B&C) is a generalization of B&B where LP relaxation of the MILP is used to obtain a bound at each node in the search tree [100]. With relaxation, the model illustrated in Figure 5.1 is also able to reach noninteger solutions. If a node has a fractional solution that can not be fathomed, a valid inequality that is violated by the fractional solution but still can be satisfied by all feasible solutions is proposed. The linear inequality is referred to as *cut*, whereas the name B&C originates from Zhu [100, 58]. A B&C algorithm has proven to be effective in solving some traditional optimization problem statements such as the traveling salesman.

As mentioned in the chapter 2, a common RSPCP is the MRCPSP introducing a minimum makespan objective. Zhu created a B&C algorithm to deal with the integer linear programming formulation. First, a derivation on the lower bounds and the distances between each pair of precedence-constrained tasks is derived to reduce the number of variables in the model [100]. Numerical results are reported for 20- and 30- activity benchmark problems, directly linked to this research having a maximum number of 16

tasks in total. Another study shows how to improve the planning process by using multi-valued state variables that are represented by networks and adding a control that encodes the length by progressively generalizing the notion of parallelism [86]. Eventually, the resulting integer problem is solved with a B&C algorithm. Especially the second part with the approach on B&C which have its fundamentals from Elf M. [24] can be used for this research specific.

B&C is often implemented into a standard MILP Solver. MILP solvers can be commercial or open-source software that can be used as an extension in python, C++, or Excel to come to a feasible solution. The backbone of the software relies on an algorithm that integrates a B&C algorithm to find the solution as quickly as possible. The architecture behind the solvers is customizable and can be used for several settings. A brief overview of some available solvers can be found in Table 5.1. The Gurobi solver is used for this research since the architecture is defined as the fastest and most comprehensive MILP solver [89].

5.1.3. Constraint Programming

Constraint programming (CP) is an addition to B&B and is used in various research papers to minimize the solution space by defining more constraints [54, 77, 84, 11]. CP is a methodology for solving difficult combinatorial problems by representing them as constraint satisfaction problems. CP has shown that a general-purpose search algorithm based on constrained propagation combined with an emphasis on modeling can solve large, practical scheduling problems [85]. Baptiste [8] illustrates a good example of constraint programming to solve scheduling problems. In other words, the idea of constraint programming is to solve problems by stating requirements (constraints) about the problem area and, consequently, find different solutions which satisfy all constraints.

”Constraint Programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it.” quote by E. Freuder [25]

Meng L. proposes a CP method that is able of finding a good solution for both small and large-sized instances [54]. Here, each global constraint is associated with a propagation algorithm that is used to remove the values of variables from their domains to prevent constraints from being infeasible [28]. Furthermore, Meng L. proposed a combination of CP and MP model-based logic for planning at unrelated parallel machines [29]. Meng L. Proposes two main variables as decision variables introducing an interval decision variable and a sequence decision variable. An interval variable illustrates a time interval, during which a task takes place and whose position in time is unknown. [39] A sequence decision variable makes sure to respect the sequential place in time within the process[39]. Meng L. used a CPLEX solver which is quite convenient for solving DFJSP problems and implementing a CP method [54]. Jain and Grossman proposed a hybrid MILP/CP model to compare both CPs with another MILP solver. Resulting in promising results, showing an efficient solver that outperforms other methods [23]. Gedik also proposes an effective CP model in his research for 283 benchmark instances based on logic-based bender algorithms to schedule jobs on parallel machines [27]. This is based on an unrelated parallel machine with sequence-dependent setup times on job availability intervals. Another RCPSP is encountered with a constraint programming approach. Fleszar proposes a CP model in a two-stage heuristic approach based on the hybrid of the MILP model and CP model. Also here, research and tests have shown that a heuristic model combining the MILP with CP outperformance the previously proposed methods with good solutions for larger instances.

Based on other scheduling problems, Ribeiro proposed two different CP models which are compared to a classic MILP model [63]. Additionally, Ham and Gakici used CP to solve parallel batch processing machines in [37]. Kelbel and Hanzálek also used an application of CP to solve a planning problem in production scheduling based on earliness and tardiness with a greedy algorithm [43]. Earliness and tardiness problems as discussed earlier, are also integrated within Na, H. proposing a CP model over a MILP as a comparison [59].

Symmetry

In theory, every worker can do all direct tasks activities in section 3.3. The similarity of workers causes symmetry. Symmetry causes a longer computational time for the algorithm to find the optimal solution. The symmetry can be broken by adding constraints to improve the computational time of the B&B algorithm. Raf J. [42], proposes a solution to deal with job grouping problems consisting of assigning a set of jobs with equal machines. Here, a formulation based on asymmetric representatives is addressed. The symmetry between the identical machines is broken and compared to a traditional formulation, the paper indicates an impact of 7 times faster than normal. Mao X. proposes a symmetry-breaking constraint on collaborative planning and symmetric scheduling used for shipbuilding projects [51]. This project aims to analyze a negotiation method based on combination auction (ICA) for integration of project planning and task scheduling.

5.2. Online solvers

A wide variety of online solvers is available for optimization formulations as the RSPCP. Hence, this problem will be formulated as a MILP function with specific constraints. Each of the online solvers can be used on different platforms (e.g. R, C++, Python, or MATLAB) or be applied for various optimization algorithms. One can find a brief summation of the solvers in Table 5.1. The goal of this research is to formulate an algorithm that can efficiently create and matches schedules per shopper based on different tasks and deadlines.

Gurobi and CPLEX are the most advanced and widely used software available to solve the comprehensive problem stated within this report. Next, the software offers the possibility to combine both an exact method (Through constraint programming), with a heuristic approach. Finally, Gurobi has a structured way of working which is used in various courses at the Aerospace Faculty of the Delft Technical University of Technology and therefore used in this research.

Solver	Source | Founder	Methods	Platform	Literature
CPLEX	Commercial | Robert E. Bixby	Branch & Cut and Dynamic programming	Python, MATLAB, C, C++, Java, and Excel	[39]
SYMPHONY	Open | T.K. Ralphs	Branch & Cut & Price	C, C++, Java, Python	[74, 81]
GUROBI	Commercial | Zonghao Gu	Branch & Cut, Heuristics & search techniques	C++, Java, .Net and Python	[35]
LINDO	Open source | Kevin Cunningham and Linus Schrage	Heuristics, Branch & Cut and other.	Microsoft Excel & Private	[49]
SCIP	Non commercial solver	Column generation, Markov chains & Branch & Cut	C++ and C	[77, 102]

Table 5.1: Various optimization solvers for MILP formulations

5.3. Metaheuristics

Exact solutions create a feasible solution with an optimal and feasible solution on the objective function. However, due to its large computational time, it is often desired to work with approximations, as heuristics tend to solve the problem faster. The schedules created will be used as guidance by every fulfilment lead, an approximation will already be a success compared to the current way of planning. This section is used to elaborate on three different heuristic methods for solving the RCPSP. Various heuristic solution methods are proposed in the literature. Drexl and Grunewald for example proposed a biased random sampling approach [21] while Slowinski elaborated on single-pass multi-pass and simulated annealing approach [66]. Additionally, a genetic algorithm is proposed by Özdamar based on priority rules [82] and Hartmann used a genetic algorithm with the use of a precedence feasible list of tasks and a mode assignment [44]. The subjects Tabu Search and Adaptive Large Neighbourhood Search are described in more detail within this section.

5.3.1. Tabu search

Tabu Search (TS) is a strategy for solving combinatorial optimization problems whose applications range from graph theory and matroid settings to general pure and MILPs. TS is unique in the essence of compliance with other methods, such as linear programming and other heuristics to overcome the limitations of local optimality [33]. Tabu Search has its origin in combinatorial procedures applied to nonlinear covering problems in 1977 [32], and subsequently applied to a wide range of scheduling and planning problems. TS is founded and based on three primary themes according to Fred Glover [34]:

1. The usage of flexible attribute-based memory structures designed to permit evaluated criteria and historical search information more thoroughly than rigid memory structures (Branch & Bound) or memory-less systems (Simulated Annealing).
2. An associated mechanism that evaluates the movement based on specific criteria within the TS algorithm
3. The incorporation of memory functions of different periods, to implement strategies for intensifying and diversifying the search

Short-term memory in TS can be represented as aggressive exploitation which seeks the best move possible, subject to the availability of choices to satisfy the constraints Figure 5.4. The TS algorithm provides constraints that prevent the algorithm from repeating certain moves by creating a tabu list. The tabu list contains every previous step and indicates 'forbidden' moves. The decision process on how to determine the best candidate for the next step is illustrated in Figure 5.5. First, each of the moves on the candidate's list is evaluated in turn. Second, an evaluation of what best move can be addressed through the objective function - the difference between the new and the old solution. In other cases where the objective function can not directly be measured through a new solution, the evaluation may be based on generating relaxed or approximate solutions.

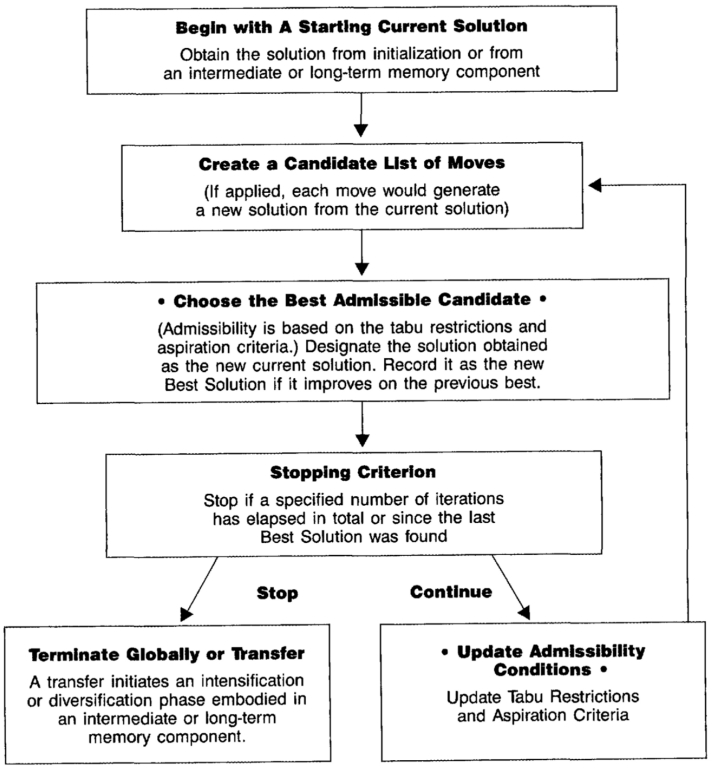


Figure 5.4: Tabu Search - Short Term Memory component [34]

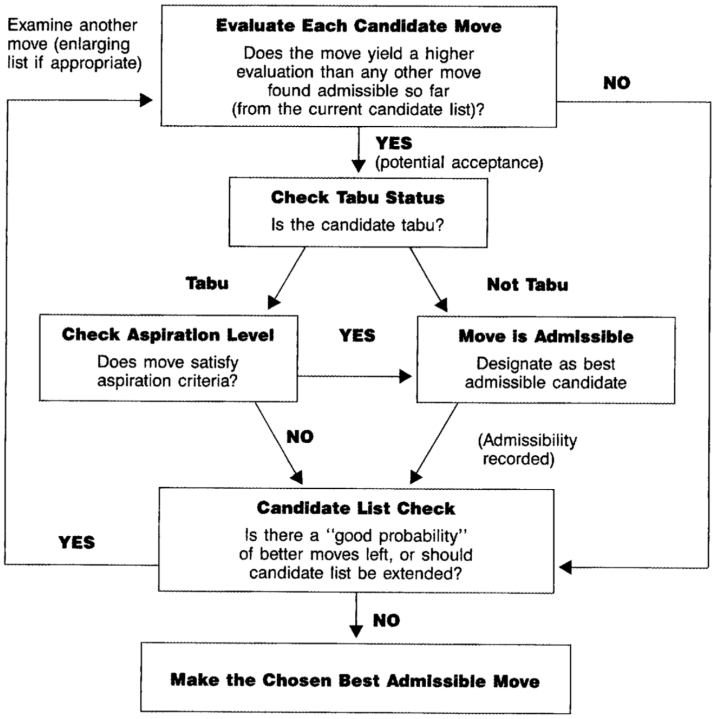


Figure 5.5: Tabu Search - Selection process of best admissible candidate [34]

As mentioned, TS is a heuristic approach used in many planning and scheduling research. Cavalcante used local search procedures to improve the solution accordingly. Additionally, by implementing a TS

algorithm, Cavalcante illustrates the performance of the algorithm creating the best upper bound for almost all instances [19]. Tom Servranckx proposes a TS algorithm to solve the integrated scheduling and decision-making for RCPSP with alternative sub graphs[78]. Hence, the TS algorithm is used to guide the search process toward high-quality solutions. Manish Kumar uses a TS algorithm for simultaneous selection and scheduling of projects. Fifteen different test instances with various levels of complexity are used to evaluate the TS algorithm. Conclusively, the TS algorithm is promising to solve every instance [45]. Additionally, Camino proposes a solution method to use a hybrid version of the TS method to optimize the solution addressed in an earlier stage [95]. Finally, A. Costa proposes a TS algorithm to strengthen the answer for a hybrid flow shop scheduling problem with limited human resource constraints [18]. After formulating a MILP model, a solution method named a discrete hybrid backtracking search optimization algorithm is used to find a preliminary solution. After implementing a multi-stage encoding scheme, the TS algorithm is used to strengthen the solution.

From the previous examples, one can draw the main trend to use the TS algorithm as an optimization tool whenever an initial solution is already defined. For this research, a Tabu Search algorithm will be used after a preliminary solution is already proposed.

5.3.2. Adaptive large neighbourhood search

Another heuristic approach analyzed in this research is a Large Neighborhood Search. Large Neighbourhood Search is first introduced by Shaw [67] in 1997 and applied the algorithm for the traditional travel salesman problems. The LNS uses a destroy and repair method to come to a solution. The destroy method is used to terminate a part of the solution while the repair method rebuilds the destroyed solution [71]. Zhang illustrates the process within his paper depicted in Figure 5.6. An example of solving an RSPCP with a Large Neighborhood Search is proposed by Mireille Palpant and Christian Artigues [65].

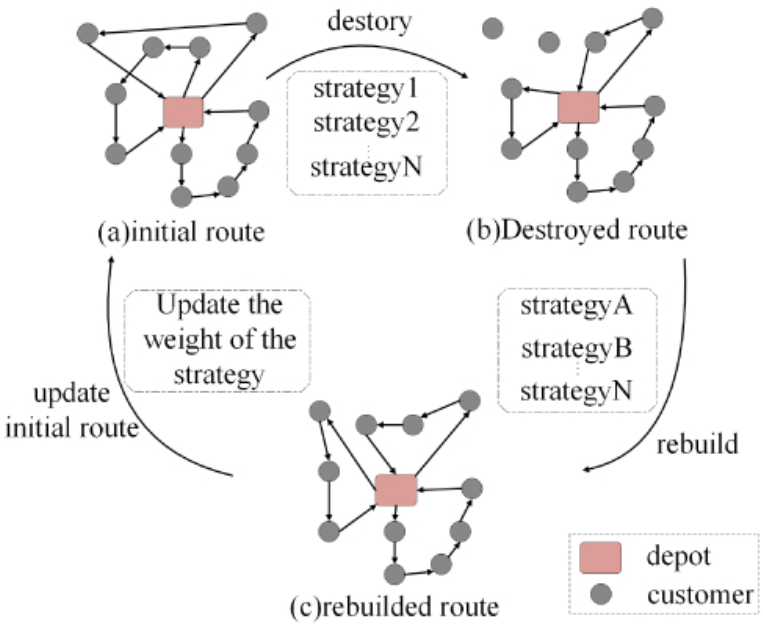


Figure 5.6: Illustrative example of a Large Neighborhood Search [98]

Additionally, The LNS is extended to an *Adaptive* Large Neighborhood Search (ALNS) first introduced by Pising and Ropke [76] to improve the solution for its application in multiple pick-ups and drop-off locations. Here, several sub heuristics with a frequency corresponding to their historic performance is used. The general framework has been given the name Adaptive Large Neighborhood Search. The computational experiments show that the proposed heuristics are very robust and adaptive to various instances, such as planning. The ALNS makes use of several removal and insertion heuristics during the same search while the LNS only makes use of one heuristic. The selection method is based on statistics and changes in the computation time for the algorithm making the algorithm more complex.

Additionally, the search algorithm is embedded within a simulated annealing algorithm (concept shortly explained in subsection 5.3.3) where the previous large neighborhood search used a simple decent approach.

Richard Martin Lusby proposed an adaptive large neighborhood search procedure to dynamically schedule patients to hospital beds in [50]. The algorithm resulted in an efficient way of solving the bed scheduling problems within hospitals and receiving an approximation of 3-14% more precise answers. Unfortunately, the computational time is a factor 12 larger than current state-of-the-art algorithms. ALNS is furthermore used in scheduling inbound logistic equipment and machinery by Rapeepan Pitakaso [72]. Which also turns out to be highly effective in tackling this problem. Within the research of Pitakaso, four new formulas are created to calculate a profitability margin to be accepted by the algorithm. ALNS also play a significant role within factory management as studied by Achmand P. Rifai [75]. He used a multi-objective large neighborhood search to simultaneously satisfy three objectives based on the Pareto front: the makespan, total costs, and average tardiness.

5.3.3. Simulated annealing

Simulated annealing is a heuristic based on the analogy between solids and vapors and is often used in combination with ALNS as described in the previous subsection [93]. The model uses the logic of heating a system and slowly lowering the temperature to decrease the number of defects and thus minimizing systems energy [13]. A new random point is chosen during each iteration in which the distance between the current solution and a new random point in time is based on a probability distribution with a scale proportional to the temperature. The algorithm allows every point to lower the objective but also accepts certain points which increases the algorithm to escape local optima [52]. Simulated annealing is also commonly used in planning and scheduling problems. The process is illustrated in Figure 5.7 where one can relate the process just described.

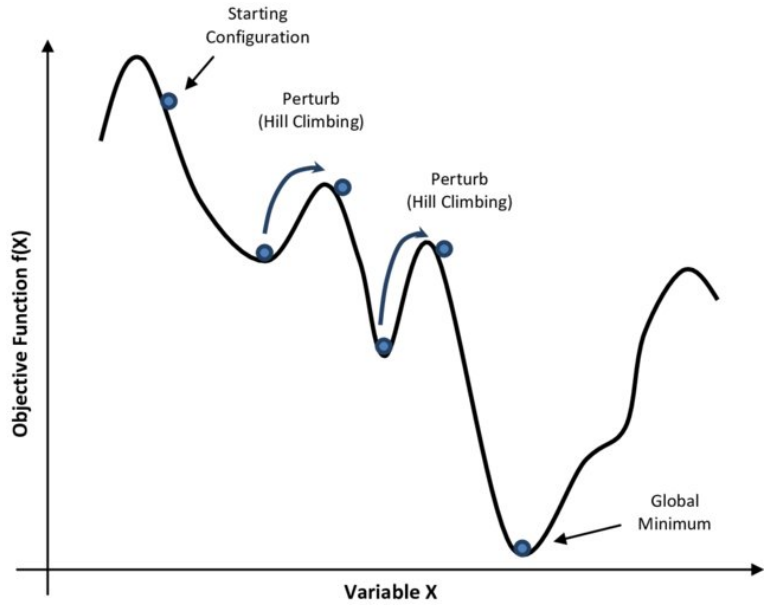


Figure 5.7: Simulated annealing example of minimizing the objective function [31]

References

[1] Behrouz Afshar-Nadjafi and Mahyar Majlesi. “Resource constrained project scheduling problem with setup times after preemptive processes”. In: *Computers and Chemical Engineering* 69 (Oct. 2014), pp. 16–25. ISSN: 00981354. DOI: 10.1016/j.compchemeng.2014.06.012.

[2] David Applegate and William Cook. “A Computational Study of the Job-Shop Scheduling Problem”. In: *ORSA Journal on Computing* 3.2 (May 1991), pp. 149–156. ISSN: 0899-1499. DOI: 10.1287/ijoc.3.2.149.

[3] David Applegate and William Cook. “Computational study of the job-shop scheduling problem”. In: *ORSA journal on computing* 3.2 (1991), pp. 149–156. ISSN: 08991499. DOI: 10.1287/ijoc.3.2.149.

[4] David Applegate and William Cook. “Computational study of the job-shop scheduling problem”. In: *ORSA journal on computing* 3.2 (1991), pp. 149–156. ISSN: 08991499. DOI: 10.1287/ijoc.3.2.149.

[5] Christian Artigues et al. “Solving an integrated employee timetabling and job-shop scheduling problem via hybrid branch-and-bound”. In: *Computers and Operations Research* 36.8 (Aug. 2009), pp. 2330–2340. ISSN: 03050548. DOI: 10.1016/j.cor.2008.08.013.

[6] Leila Asadzadeh. “A local search genetic algorithm for the job shop scheduling problem with intelligent agents”. In: *Computers & Industrial Engineering* 85 (July 2015), pp. 376–383. ISSN: 0360-8352. DOI: 10.1016/J.CIE.2015.04.006.

[7] Francisco Ballestín, Vicente Valls, and Sacramento Quintanilla. “Scheduling projects with limited number of preemptions”. In: *Computers and Operations Research* 36.11 (Nov. 2009), pp. 2913–2925. ISSN: 03050548. DOI: 10.1016/j.cor.2009.01.006.

[8] Philippe Baptiste. *A Theoretical and Experime Propagation Techniques*. Tech. rep.

[9] Lucio Bianco, Massimiliano Caramia, and Stefano Giordani. “Resource levelling in project scheduling with generalized precedence relationships and variable execution intensities”. In: *OR Spectrum* 38.2 (Mar. 2016), pp. 405–425. ISSN: 14366304. DOI: 10.1007/s00291-016-0435-1.

[10] C. Bierwirth and D. C. Mattfeld. “Production scheduling and rescheduling with genetic algorithms.” In: *Evolutionary computation* 7.1 (1999), pp. 1–17. ISSN: 10636560. DOI: 10.1162/evco.1999.7.1.1.

[11] J. Blazewicz, J. K. Lenstra, and A. H.G.Rinnooy Kan. “Scheduling subject to resource constraints: classification and complexity”. In: *Discrete Applied Mathematics* 5.1 (Jan. 1983), pp. 11–24. ISSN: 0166-218X. DOI: 10.1016/0166-218X(83)90012-4.

[12] Peter Brucker et al. “A branch and bound algorithm for the resource-constrained project scheduling problem”. In: *European Journal of Operational Research* 107.2 (June 1998), pp. 272–288. ISSN: 0377-2217. DOI: 10.1016/S0377-2217(97)00335-4.

[13] *Chapter 2 Simulated annealing 2.1 Introduction of the algorithm*. Tech. rep.

[14] Lu Chen and Zhe Zhang. “Preemption resource-constrained project scheduling problems with fuzzy random duration and resource availabilities”. In: *Journal of Industrial and Production Engineering* 33.6 (Aug. 2016), pp. 373–382. ISSN: 21681023. DOI: 10.1080/21681015.2016.1140089.

[15] Runwei Cheng, Mitsuo Gen, and Yasuhiro Tsujimura. “A tutorial survey of job-shop scheduling problems using genetic algorithms—I. representation”. In: *Computers & Industrial Engineering* 30.4 (Sept. 1996), pp. 983–997. ISSN: 0360-8352. DOI: 10.1016/0360-8352(96)00047-2.

[16] Christodoulos A. Floudas and Panos M. Pardalos. *Encyclopedia of Optimization Second Edition*. Tech. rep.

[17] *Coronavirus pandemic adds 219 bn to US ecommerce sales in 2020-2021*. Mar. 2022.

[18] A. Costa, V. Fernandez-Viagas, and J. M. Framinan. “Solving the hybrid flow shop scheduling problem with limited human resource constraint”. In: *Computers & Industrial Engineering* 146 (Aug. 2020), p. 106545. ISSN: 0360-8352. DOI: 10.1016/J.CIE.2020.106545.

[19] Carvalho De Souza et al. *”Scheduling projects with labor constraints”*. Tech. rep. 1998. URL: http://hdl.handle.net/2078.1/3961.

[20] Ulrich Dorndorf, Erwin Pesch, and Toàn Phan-Huy. “Time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalized precedence constraints”. In: *Management Science* 46.10 (2000), pp. 1365–1384. ISSN: 00251909. DOI: 10.1287/mnsc.46.10.1365.12272.

[21] Andreas Drexl and Juergen Gruenewald. “Nonpreemptive multi-mode resource-constrained project scheduling”. In: *IIE Transactions (Institute of Industrial Engineers)* 25.5 (1993), pp. 74–81. ISSN: 15458830. DOI: 10.1080/07408179308964317.

[22] L. E. Drezet and J. C. Billaut. “A project scheduling problem with labour constraints and time-dependent activities requirements”. In: *International Journal of Production Economics* 112.1 (Mar. 2008), pp. 217–225. ISSN: 09255273. DOI: 10.1016/j.ijpe.2006.08.021.

[23] Emrah B. Edis and Ceyda Oguz. “Parallel machine scheduling with flexible resources”. In: *Computers and Industrial Engineering* 63.2 (Sept. 2012), pp. 433–447. ISSN: 03608352. DOI: 10.1016/j.cie.2012.03.018.

[24] Matthias Elf et al. “Branch-and-cut algorithms for combinatorial optimization and their implementation in ABACUS”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 2241. Springer Verlag, 2001, pp. 157–222. ISBN: 3540428771. DOI: 10.1007/3-540-45586-8{_}5.

[25] Eugene C Freuder. *Xerox Palo Alto Res, Ctr, Palo AUo. Calif, 1976. 3. Knuth. D, The Art of Computer Programming. Vot. 1: Fundamental Algorithms*. Tech. rep. 4. 1973, p. 841.

[26] Nicolò Galante, Enrique García López, and Sarah Monroe. *The future of online grocery in Europe*. Tech. rep.

[27] Ridvan Gedik et al. “A constraint programming approach for solving unrelated parallel machine scheduling problem”. In: *Computers and Industrial Engineering* 121 (July 2018), pp. 139–149. ISSN: 03608352. DOI: 10.1016/j.cie.2018.05.014.

[28] Ridvan Gedik et al. “A constraint programming approach for the team orienteering problem with time windows”. In: *Computers and Industrial Engineering* 107 (May 2017), pp. 178–195. ISSN: 03608352. DOI: 10.1016/j.cie.2017.03.017.

[29] Ridvan Gedik et al. “Analysis of a parallel machine scheduling problem with sequence dependent setup times and job availability intervals”. In: *European Journal of Operational Research* 251.2 (June 2016), pp. 640–650. ISSN: 03772217. DOI: 10.1016/j.ejor.2015.11.020.

[30] Gerben Houtsma. “A Resource-Constrained Project Scheduling Problem at a Grocery Delivery Fulfillment Center”. In: *Erasmus University Rotterdam* (Feb. 2022).

[31] Omid Ghasemalizadeh, Seyedmeysam Khaleghian, and Saied Taheri. “A REVIEW OF OPTIMIZATION TECHNIQUES IN ARTIFICIAL NETWORKS.” In: *International Journal of Advanced Research* 4.9 (Sept. 2016), pp. 1668–1686. ISSN: 23205407. DOI: 10.21474/IJAR01/1627. URL: http://www.journalijar.com/article/12127/a-review-of-optimization-techniques-in-artificial-networks./.

[32] Fred Glover. “HEURISTICS FOR INTEGER PROGRAMMING USING SURROGATE CONSTRAINTS”. In: *Decision Sciences* 8.1 (1977), pp. 156–166. ISSN: 15405915. DOI: 10.1111/j.1540-5915.1977.tb01074.x.

[33] Fred Glover. “Tabu Search—Part I”. In: *ORSA Journal on Computing* 1.3 (Aug. 1989), pp. 190–206. ISSN: 0899-1499. DOI: 10.1287/ijoc.1.3.190.

[34] Fred Glover. “Tabu Search: A Tutorial”. In: *Interfaces* 20.4 (Aug. 1990), pp. 74–94. ISSN: 0092-2102. DOI: 10.1287/inte.20.4.74.

[35] LLC Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2022.

[36] Farhad Habibi, Farnaz Barzinpour, and Seyed Jafar Sadjadi. “Resource-constrained project scheduling problem: review of past and recent developments”. In: *Journal of Project Management* (2018), pp. 55–88. ISSN: 23718366. DOI: 10.5267/j.jpm.2018.1.005.

[37] Andy M. Ham and Eray Cakici. “Flexible job shop scheduling problem with parallel batch processing machines: MIP and CP approaches”. In: *Computers and Industrial Engineering* 102 (Dec. 2016), pp. 160–165. ISSN: 03608352. DOI: 10.1016/j.cie.2016.11.001.

[38] He He, Hal Daumé, and Jason Eisner. “Learning to Search in Branch and Bound Algorithms”. In: *NIPS* (Dec. 2014).

[39] IBM. *IBM ILOG CPLEX Optimization Studio*. 2022.

[40] IEEE Staff and IEEE Staff. *Preemptive Resource Constrained Project Scheduling Problem with Uncertain Resource Availabilities: Investigate Worth of Proactive Strategies*. 2010. ISBN: 9781424485031.

[41] Carlos Jahn et al. *Proceedings of the Hamburg International Conference of Logistics (HICL)/Adapting to the Future Maritime and City Logistics in the Context of Digitalization and Sustainability*. ISBN: 9783754927717.

[42] Raf Jans and Jacques Desrosiers. “Efficient symmetry breaking formulations for the job grouping problem”. In: *Computers and Operations Research* 40.4 (Apr. 2013), pp. 1132–1142. ISSN: 03050548. DOI: 10.1016/j.cor.2012.11.017.

[43] Jan Kelbel and Zdeněk Hanzálek. “Solving production scheduling with earliness/tardiness penalties by constraint programming”. In: *Journal of Intelligent Manufacturing* 22.4 (Aug. 2011), pp. 553–562. ISSN: 09565515. DOI: 10.1007/S10845-009-0318-2.

[44] Rainer Kölsch and s. *Project Scheduling*. Ed. by Jan W□glarz. Vol. 14. International Series in Operations Research & Management Science. Boston, MA: Springer US, 1999. ISBN: 978-1-4613-7529-6. DOI: 10.1007/978-1-4615-5533-9. URL: http://link.springer.com/10.1007/978-1-4615-5533-9.

[45] Manish Kumar et al. “A Tabu search algorithm for simultaneous selection and scheduling of projects”. In: *Advances in Intelligent Systems and Computing*. Vol. 741. Springer Verlag, 2019, pp. 1111–1121. ISBN: 9789811307607. DOI: 10.1007/978-981-13-0761-4{_}104.

[46] Olivier Lambrechts, Erik Demeulemeester, and Willy Herroelen. “Time slack-based techniques for robust project scheduling subject to resource uncertainty”. In: *Annals of Operations Research* 186.1 (June 2011), pp. 443–464. ISSN: 02545330. DOI: 10.1007/s10479-010-0777-z.

[47] A H Land and A G Doig. *An Automatic Method of Solving Discrete Programming Problems*. Tech. rep. 3. 1960, pp. 497–520.

[48] Matheus Leusin et al. “Solving the Job-Shop Scheduling Problem in the Industry 4.0 Era”. In: *Technologies* 6.4 (Nov. 2018), p. 107. DOI: 10.3390/technologies6040107.

[49] Lindo Systems. *Lindo*. 2022.

[50] Richard Martin Lusby et al. “An adaptive large neighborhood search procedure applied to the dynamic patient admission scheduling problem”. In: *Artificial Intelligence in Medicine* 74 (Nov. 2016), pp. 21–31. ISSN: 0933-3657. DOI: 10.1016/J.ARTMED.2016.10.002.

[51] Xuezhang Mao et al. “Research on collaborative planning and symmetric scheduling for parallel shipbuilding projects in the open distributed manufacturing environment”. In: *Symmetry* 12.1 (2020). ISSN: 20738994. DOI: 10.3390/SYM12010161.

[52] Mathworks. “Find global minima for bounded nonlinear problems”. In: *Mathworks* (2022).

[53] McKinsey. “Solving the paradox of growth and profitability in e-commerce”. In: *McKinsey* (2022).

[54] Leilei Meng et al. “Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem”. In: *Computers & Industrial Engineering* 142 (Apr. 2020), p. 106347. ISSN: 0360-8352. DOI: 10.1016/J.CIE.2020.106347.

[55] Rolf H Möhring et al. *Solving Project Scheduling Problems by Minimum Cut Computations*. Tech. rep. 2003.

[56] Rolf H. Möhring et al. “On project scheduling with irregular starting time costs”. In: *Operations Research Letters* 28.4 (May 2001), pp. 149–154. ISSN: 01676377. DOI: 10.1016/S0167-6377(01)00064-5.

[57] Daisuke Morita. “Haruhiko SUWA”. In: *Bulletin of the JSME Journal of Advanced Mechanical Design, Systems, and Manufacturing* 10.3 (2016). DOI: 10.1299/jamdsm.2016jamdsm00.

[58] David R. Morrison et al. “Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning”. In: *Discrete Optimization* 19 (Feb. 2016), pp. 79–102. ISSN: 15725286. DOI: 10.1016/j.disopt.2016.01.005.

[59] Hongbum Na and Jinwoo Park. “Multi-level job scheduling in a flexible job shop environment”. In: *International Journal of Production Research*. Vol. 52. 13. Taylor and Francis Ltd., 2014, pp. 3877–3887. DOI: 10.1080/00207543.2013.848487.

[60] B Afshar Nadjafi and S Shadrokh. *The preemptive resource-constrained project scheduling problem subject to due dates and preemption penalties: An integer programming approach*. Tech. rep. 2008, pp. 35–39.

[61] Nicolo Galante, Enrique Garcia, and Sarah Munby. “The future of online grocery in Europe”. In: *McKinsey* (2013).

[62] Niloofar Nikoofal Sahl Abadi, Mohsen Bagheri, and Mohammad Assadi. “Multiobjective model for solving resource-leveling problem with discounted cash flows”. In: *International Transactions in Operational Research* 25.6 (Nov. 2018), pp. 2009–2030. ISSN: 14753995. DOI: 10.1111/itor.12253.

[63] Renata Melo e Silva de Oliveira and Maria Sofia F. Oliveira de Castro Ribeiro. “Comparing Mixed & Integer Programming vs. Constraint Programming by solving Job-Shop Scheduling Problems”. In: *Independent Journal of Management & Production* 6.1 (Mar. 2015). DOI: 10.14807/ijmp.v6i1.262.

[64] Juan D. Palacio and Olga L. Larrea. “A lexicographic approach to the robust resource-constrained project scheduling problem”. In: *International Transactions in Operational Research* 24.1-2 (Jan. 2017), pp. 143–157. ISSN: 14753995. DOI: 10.1111/itor.12301.

[65] Mireille Palpant, Christian Artigues, and Philippe Michelon. *LSSPER: Solving the Resource-Constrained Project Scheduling Problem with Large Neighbourhood Search*. Tech. rep. 2004, pp. 237–257.

[66] J.H. PATTERSON et al. “AN ALGORITHM FOR A GENERAL CLASS OF PRECEDENCE AND RESOURCE CONSTRAINED SCHEDULING PROBLEMS”. In: *Advances in Project Scheduling* (Jan. 1989), pp. 3–28. DOI: 10.1016/B978-0-444-87358-3.50005-5.

[67] Paul Shaw. *A New Local Search Algorithm Providing High Quality Solutions to Vehicle Routing Problems*. Tech. rep. 1997.

[68] F Pellerin, R Perrier, and N Hajji. *Time-cost trade-offs in resource-constraint project scheduling problems with overlapping modes*. Tech. rep. 3. 2014, pp. 215–236.

[69] Picnic. *Automatic fulfilment center of Picnic located in Zwolle*. 2020.

[70] Picnic. *Picnic inhouse information*. Amsterdam, 2022.

[71] D ; Pisinger and S Røpke. *Large Neighborhood Search*. Tech. rep. 2010, pp. 399–420. URL: http://www.springerlink.com.globalproxy.cvt.dk/content/978-1-4419-1663-1#section=777803&page=1.

[72] Rapeepan Pitakaso and Kanchana Sethanan. “Adaptive large neighborhood search for scheduling sugarcane inbound logistics equipment and machinery under a sharing infield resource system”. In: *Computers and Electronics in Agriculture* 158 (Mar. 2019), pp. 313–325. ISSN: 0168-1699. DOI: 10.1016/J.COMPAG.2019.02.001.

[73] Srijith Rajeev, Sabu Kurian, and Brijesh Paul. *A modified serial scheduling scheme for resource constrained project scheduling weighted earliness tardiness problem*. Tech. rep. 3. 2015, pp. 241–254.

[74] Ted K. Ralphs and Menal Giizelsoy. “The symphony callable library for mixed integer programming”. In: *Operations Research/ Computer Science Interfaces Series* 29 (2005), pp. 61–76. ISSN: 1387666X. DOI: 10.1007/0-387-23529-9{_}5.

[75] Achmad P. Rifai, Huu Tho Nguyen, and Siti Zawiah Md Dawal. “Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling”. In: *Applied Soft Computing* 40 (Mar. 2016), pp. 42–57. ISSN: 1568-4946. DOI: 10.1016/J.ASOC.2015.11.034.

[76] Stefan Ropke and David Pisinger. “An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows”. In: *Transportation Science* 40.4 (2006), pp. 455–472. ISSN: 15265447. DOI: 10.1287/trsc.1050.0135.

[77] SCIP OPtimization Suite. “Solving Constraint Integer Programming”. In: ().

[78] Tom Servranckx and Mario Vanhoucke. “A tabu search procedure for the resource-constrained project scheduling problem with alternative subgraphs”. In: *European Journal of Operational Research* 273.3 (Mar. 2019), pp. 841–860. ISSN: 0377-2217. DOI: 10.1016/J.EJOR.2018.09.005.

[79] Roman Slowilqski. *Multiobjective network scheduling with efficient use of renewable and non-renewable resources*. Tech. rep. 1979.

[80] Joel P. Stinson, Edward W. Davis, and Basheer M. Khumawala. “Multiple resource–constrained scheduling using branch and bound”. In: *AIIE Transactions* 10.3 (1978), pp. 252–259. ISSN: 05695554. DOI: 10.1080/05695557808975212.

[81] “SYMPHONY-5.6.9-Manual”. In: ().

[82] Ieee Transactions On Systems, Man And, and Cybernetics-Part C. *A Genetic Algorithm Approach to a General Category Project Scheduling Problem*. Tech. rep. 1. 1999.

[83] Madjid Tavana, Amir Reza Abtahi, and Kaveh Khalili-Damghani. “A new multi-objective multi-mode model for solving preemptive time-cost-quality trade-off project scheduling problems”. In: *Expert Systems with Applications* 41.4 PART 2 (2014), pp. 1830–1846. ISSN: 09574174. DOI: 10.1016/j.eswa.2013.08.081.

[84] Edward. Tsang. *Foundations of constraint satisfaction*. Academic Press, 1993, p. 421. ISBN: 0127016104.

[85] Peter Van Beek and Xinguang Chen. *CPlan: A Constraint Programming Approach to Planning*. Tech. rep. 1999. URL: www.aaai.org.

[86] Menkes Van Den Briel, Thomas Vossen, and Subbarao Kambhampati. *Reviving Integer Programming Approaches for AI Planning: A Branch-and-Cut Framework*. Tech. rep. 2005.

[87] Camino R. Vela et al. “Evolutionary tabu search for flexible due-date satisfaction in fuzzy job shop scheduling”. In: *Computers & Operations Research* 119 (July 2020), p. 104931. ISSN: 0305-0548. DOI: 10.1016/J.COR.2020.104931.

[88] Guilherme E Vieira, Jeffrey W Herrmanny, and Edward Liny. *Analytical models to predict the performance of a single-machine system under periodic and event-driven rescheduling strategies*. Tech. rep. 8. 2000. URL: http://www.tandf.co.uk/journals.

[89] Will citrin. *The best-of-breed mathematical optimization solver*. 2022.

[90] Jing Xiao, Zhou Wu, and Jian Chao Tang. “Hybridization of electromagnetism with multi-objective evolutionary algorithms for RCPSP”. In: *GECCO 2014 - Proceedings of the 2014 Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, 2014, pp. 653–660. ISBN: 9781450326629. DOI: 10.1145/2576768.2598228.

[91] Jing Xiao et al. “Integration of electromagnetism with multi-objective evolutionary algorithms for RCPSP”. In: *European Journal of Operational Research* 251.1 (May 2016), pp. 22–35. ISSN: 03772217. DOI: 10.1016/j.ejor.2015.10.059.

[92] Linlin Xie, Yajiao Chen, and Ruidong Chang. “Scheduling optimization of prefabricated construction projects by genetic algorithm”. In: *Applied Sciences (Switzerland)* 11.12 (June 2021). ISSN: 20763417. DOI: 10.3390/app11125531.

[93] Zhao Xinchao. “Simulated annealing algorithm with adaptive neighborhood”. In: *Applied Soft Computing Journal*. Vol. 11. 2. Mar. 2011, pp. 1827–1836. DOI: 10.1016/j.asoc.2010.05.029.

[94] Jian Xiong et al. “A knowledge-based evolutionary multiobjective approach for stochastic extended resource investment project scheduling problems”. In: *IEEE Transactions on Evolutionary Computation* 18.5 (Oct. 2014), pp. 742–763. ISSN: 1089778X. DOI: 10.1109/TEVC.2013.2283916.

[95] Wenxiang Xu et al. “A multi-objective scheduling method for distributed and flexible job shop based on hybrid genetic algorithm and tabu search considering operation outsourcing and carbon emission”. In: *Computers and Industrial Engineering* 157 (July 2021). ISSN: 03608352. DOI: 10.1016/j.cie.2021.107318.

[96] Forhad Zaman et al. “Resource Constrained Project Scheduling with Dynamic Disruption Recovery”. In: *IEEE Access* 8 (Jan. 2020), pp. 144866–144879. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.3014940.

[97] Guohui Zhang et al. “An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem”. In: *Computers and Industrial Engineering* 56.4 (May 2009), pp. 1309–1318. ISSN: 03608352. DOI: 10.1016/j.cie.2008.07.021.

[98] Haiping Zhang and Wang Yang. “An Enhanced Adaptive Large Neighborhood Search Algorithm for the Capacitated Vehicle Routing Problem”. In: *ACM International Conference Proceeding Series*. Association for Computing Machinery, Feb. 2021, pp. 79–85. ISBN: 9781450389310. DOI: 10.1145/3457682.3457694.

[99] Li Zhou, Wuliang Peng, and Zhongliang Zhang. “An ant colony system for solving resource leveling problem”. In: *2010 International Conference on Intelligent Computation Technology and Automation, ICICTA 2010*. Vol. 1. 2010, pp. 489–492. ISBN: 9780769540771. DOI: 10.1109/ICICTA.2010.694.

[100] Guidong Zhu, Jonathan F. Bard, and Gang Yu. “A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem”. In: *INFORMS Journal on Computing* 18.3 (2006), pp. 377–390. ISSN: 08991499. DOI: 10.1287/ijoc.1040.0121.

[101] Jie Zhu, Xiaoping Li, and Weiming Shen. “Effective genetic algorithm for resource-constrained project scheduling with limited preemptions”. In: *International Journal of Machine Learning and Cybernetics* 2.2 (June 2011), pp. 55–65. ISSN: 18688071. DOI: 10.1007/s13042-011-0014-3.

[102] Zuse Institution Berlin. *SCIP Documentation*. Berlin, 2022.

A

Gantt Chart

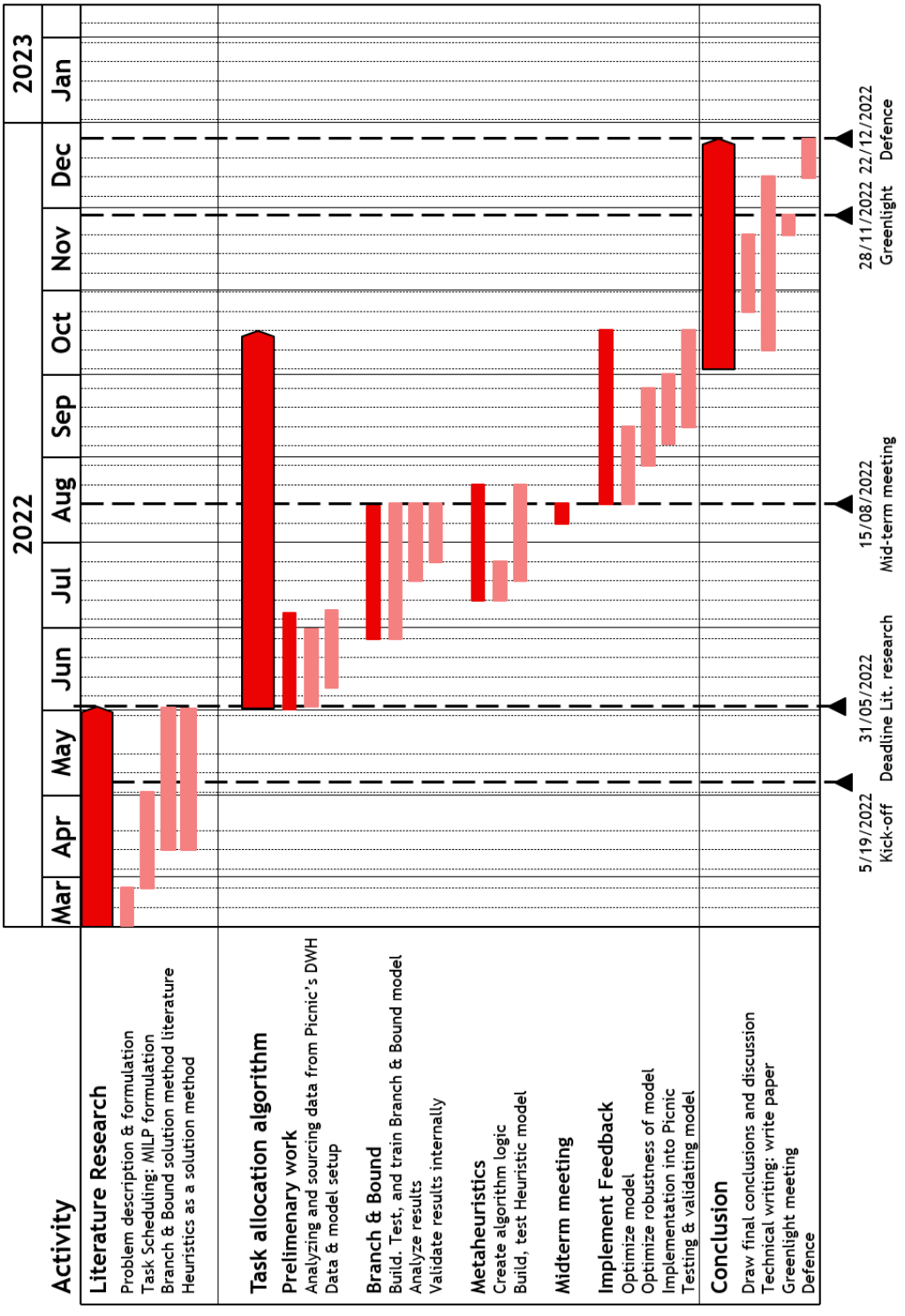


Figure A.1: Gantt chart indicating the planning of Msc. Thesis

III

Supporting work

1

Picnic Case Study

1.1. List of Tasks with Descriptions

The model's output should illustrate an overview of all schedules created by the algorithm. A matrix with on the y-axis the various schedules and on the x-axis different time units t on a fifteen-minute basis is depicted as output. Hence, every row consists of an anonymous schedule, while every column depicts a time unit. The number of schedules created is evaluated with the number of incoming workload packages. Every workload package with an urgency level of 1 should be scheduled during the day. The total workload divided by each person's maximum availability determines how many needed schedules. The activities in the matrix are listed in Table 1.1

Most activities can be executed in different temperature zones; ambient, chilled, or frozen. A description of the clocking activities is given below:

- **Bonus hours**:
  Bonus hours address extra hours which can be used to execute any activity. Bonus hours can compensate for slack within the operations and increase the schedule's robustness.
- **Break**:
  Breaks are scheduled as a resting moment for an employee.
- **Picking**:
  Order picking is the main activity of the FC. Employees are filling the order crates with goods from storage locations.
- **Dispatching**:
  The allocation of totes from a pick car into a dispatching frame. Dispatching frames are filled with clustered totes that will be transferred to the same hub and loaded into an Electronic Picnic Vehicle (EPV).
- **Tote handling**:
  Tote handling is cleaning returned totes and filling totes with plastic bags.
- **Tote completion**:
  The procedure of packing ice bags into chilled totes. Subsequently, totes are closed with a lid to keep the products inside chilled.
- **Transport Inbound**:
  Unloading roll containers from inbound trucks in the FC. All fresh articles are delivered in the morning and are defined as transport inbound morning. A discrepancy is explicitly made to address the precedence relations.
- **Buffer replenishment**:
  Products are transferred to the buffer zone if no room is available on a shelf. Replenishing products from and to the buffer zone are scheduled as buffer replenishment.

- **Return & Waste**:
  All products handled in an FC are delivered in cardboard or plastic packages. Unboxing products results in much garbage. The garbage should be returned or brought into containers. Handling the containers are defined as Return & Waste.

- **Replenishment**:
  Stocking shelves with products originating from containers from the inbound truck. A discrepancy is defined between replenishing fresh products (replenishment morning) and other products (replenishment).

- **Urgent moves**:
  Occasionally, not all products are already replenished before a pick round starts. As a result, specific orders must be completed due to missing products. Express rounds can be initialised if a product is unavailable during the initial picking round. Express rounds are set up to fill crates with missing products. An express round is defined within the clocking activity express rounds. As a result, work is scheduled as urgent moves to compensate for addressing this phenomenon.

- **Counting**:
  A centralised software is designed to track the availability of all products on the shelves. The software is defined as the Warehouse Management System (WMS). Counting ensures the data quality and the calibration of the system.

- **Slotting**:
  Products in an FC of Picnic are slotted to specific locations according to a logic based on weight, size, and demand. The activity of allocating products to other locations is defined as slotting.

- **Clearing**:
  Recalling almost overdue products based on data from the WMS.

- **Cleaning**:
  The activity of making everything clean and ordered again.

- **Quality**:
  Validation of products still holds their quality level. For example, they are checking fruits and vegetables for damages or other irregularities, damage on cardboard packages and more.

- **Safety**:
  The safety activity is scheduled to ensure all employees oblige to the safety rules during the working day. They are additionally validating if all processes are executed safely.

- **Projects**:
  Project hours are scheduled for different activities related to other projects. Projects can include but are not limited to the maintenance of the warehouses or other work necessary in an FC.

Table 1.1: Overview of clocking activities

Activity Number	Clocking Activity ID	Temp Zone	Priority Level
-1	bonus hour	-	-
0	break	-	1
1	picking	AM	1
2	picking	CH	1
3	picking	FR	1
4	dispatching	AM	1
5	dispatching	CH	1
6	tote_handling	AM	1
7	tote_handling	CH	1
8	tote_completion	CH	1
9	transport_outbound	x	1
10	transport_inbound	x	1
11	transport_inbound_morning	x	1
12	buffer_replenishment	AM	1
13	buffer_replenishment	CH	1
14	return_waste	x	1
15	return_waste	x	1
16	replenishment_morning	x	1
17	replenishment_morning	x	1
18	replenishment_morning	x	1
19	replenishment	AM	1
20	replenishment	CH	1
21	replenishment	FR	1
22	urgent_moves	x	1
23	counting	x	2
24	slotting	x	2
25	clearing	x	2
26	cleaning	x	2
27	quality	x	2
28	safety	x	2
29	projects	x	2

1.2. Input Data Sheets

The following tables illustrates the input tables used for the algorithm. First, Table 1.2 illustrates all precedence relationships within an FC. The parameter $P^L_{i,j}$ is based on this table. Next, Table 1.3 illustrated an example of the capacity per FC. Note that all values of the FCs are arbitrary due to confidentiality agreements. The output of the real-time data set used in the paper is illustrated in the last two pages of this report.

Table 1.2: Precedence relationships

task_id	temp_zone	break	transport_inbound	replenishment	replenishment	replenishment	picking	picking	picking
		x	x	AM	CH	FR	AM	CH	FR
break	x	0	0	0	0	0	0	0	0
transport_inbound	x	0	0	1	1	1	0	0	0
replenishment	AM	0	0	0	0	0	1	0	0
replenishment	CH	0	0	0	0	0	0	1	0
replenishment	FR	0	0	0	0	0	0	0	1
picking	AM	0	0	0	0	0	0	0	0
picking	CH	0	0	0	0	0	0	0	0
picking	FR	0	0	0	0	0	0	0	0

Table 1.3: FC capacity characteristics

task_id	temp_zone	FC0	FC1	FC2	FC3	FC4	FC5	FC6	FC7	FC8
picking	AM	18	16	18	18	30	18	18	18	18
picking	CH	16	16	16	16	16	16	16	16	16
picking	FR	2	2	2	2	2	2	2	2	2
dispatching	AM	10	11	12	11	12	10	10	10	10
dispatching	CH	5	5	5	5	5	5	5	5	5
tote_handling	AM	4	4	4	4	4	4	4	4	4
tote_handling	CH	5	5	5	5	5	5	5	5	5
tote_completion	CH	5	5	5	5	5	5	5	5	5
transport_outbound	x	40	40	40	40	40	40	40	40	40

2

Used Software And Extensions

2.1. Numba

Numba is a Just-In-Time (JIT) compiler for Python that works best on code that uses NumPy array and loops. The most common way to use Numba is through its collection of decorators that can be applied to one's functions to instruct Numba to compile them. When a call is made to a Numba-decorated function, it is compiled to machine code "just in time" for execution, and all or part of one's code can subsequently run at native machine code speed. The heuristic is therefore coded solely with NumPy values.

The Numba JIT decorator fundamentally operates in nopython mode. The behaviour of the nopython compilation mode is to compile the decorated function so that it will run entirely without the involvement of the Python interpreter. This is the recommended and best-practice way to use the Numba JIT decorator, as it leads to the best performance. Numba reads the Python byte code for a decorated function and combines this with information about the types of input arguments to the function. It analyses and optimises one's code and uses the low level virtual machine compiler (front-end software library) to generate a machine code version of your function tailored to your CPU capabilities. This compiled version is then used every time one's function is called. [1]

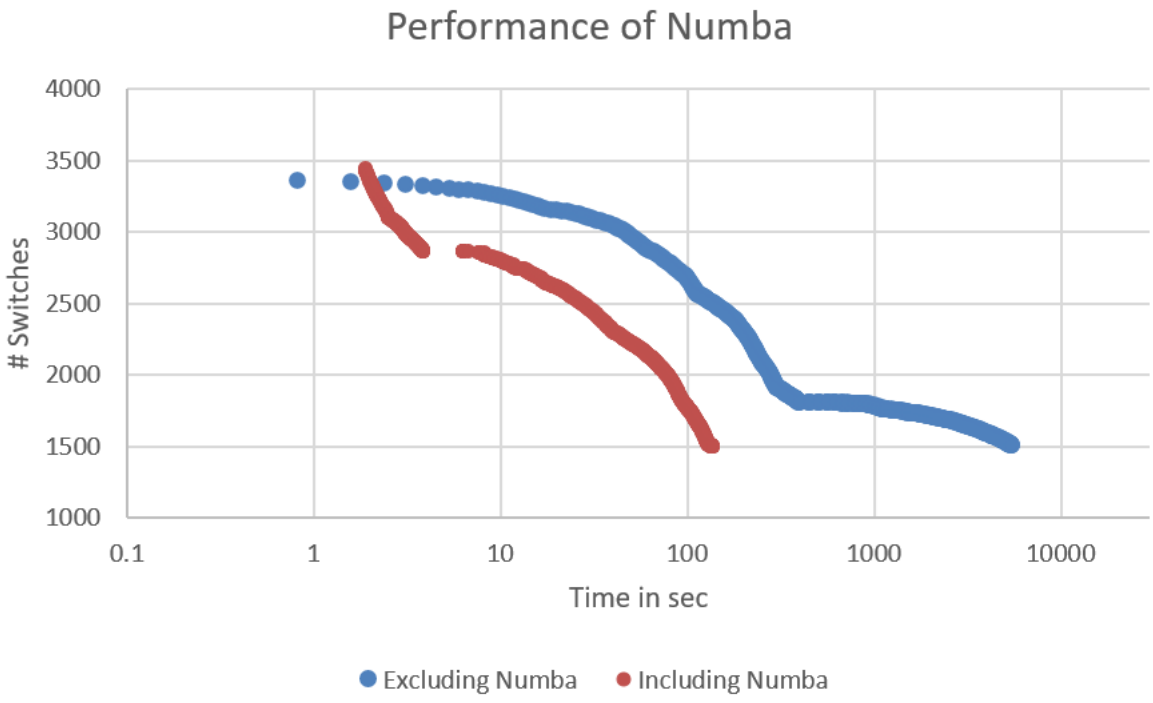


Figure 2.1: Performance of machine code versus non-machine code

2.2. Tableau

The output values are visualised through Tableau, a data visualisation software frequently used by Picnic [2]. The data is sorted to a new data format visualised in Table 2.1 Tableau is used to visualise the output of the real time data set, which is illustrated in the final 2 pages of this report.

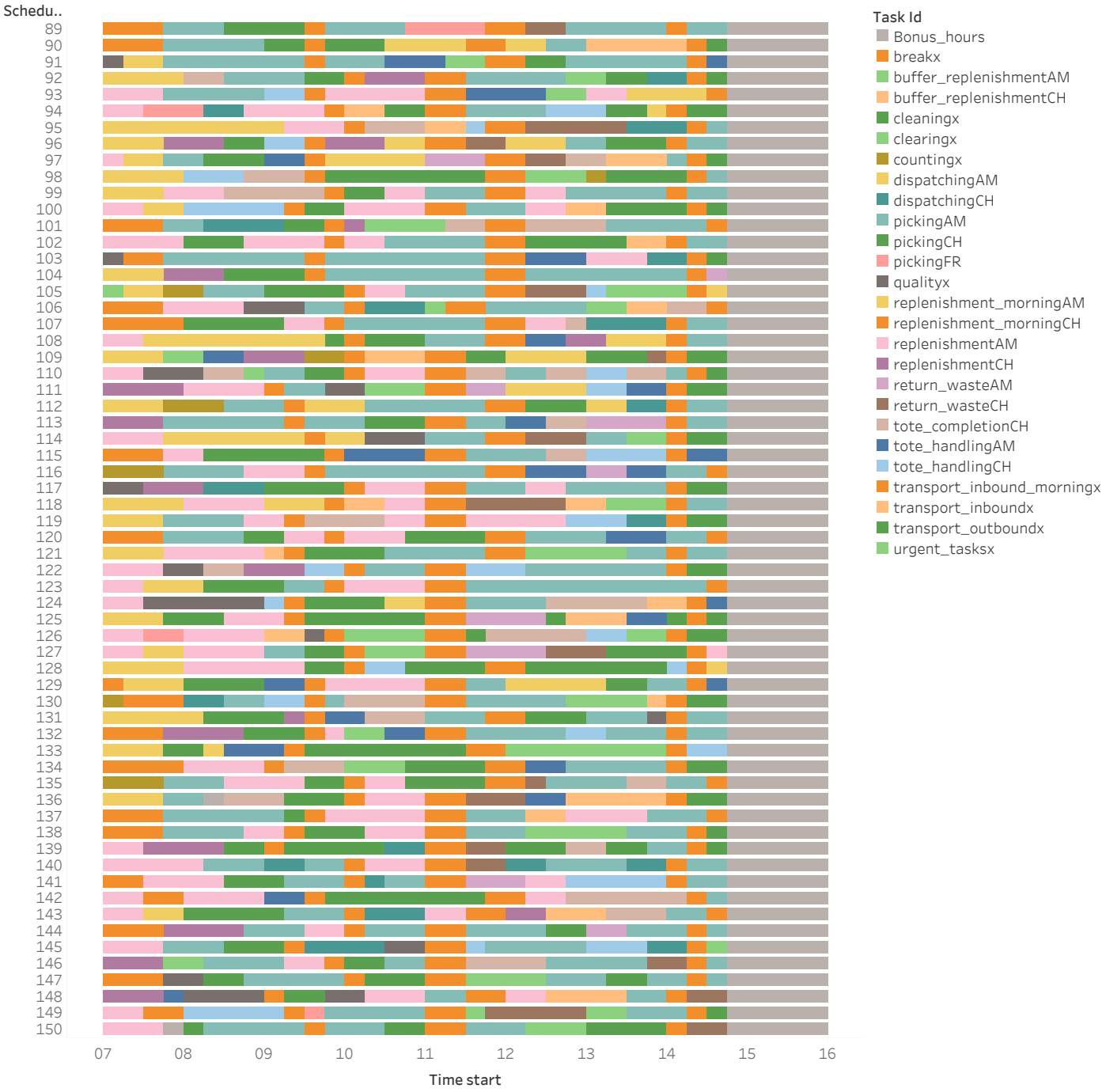
Table 2.1: Data formatting for Tableau

FC	Shift	Schedule	Time_start	Time_end	Activity	Temp_zone	Duration
FC1	1	1	07.00	07.15	Picking	AM	15
FC1	1	1	07.15	07.30	Picking	AM	15
FC1	1	1	07.30	07.45	Picking	AM	15
FC1	1	1	07.45	08.00	Picking	AM	15

Set of schedules from a real-time data source

Time start for each Schedule. Colour shows details about Task Id. Size shows details about Duration.self.The view is filtered on Task Id, which keeps 26 of 26 members.

Set of schedules from a real-time data source



Time start for each Schedule. Colour shows details about Task Id. Size shows details about Duration.self.The view is filtered on Task Id, which keeps 26 of 26 members.

Bibliography

[1] Community project. Numba, 8 2012. URL https://numba.pydata.org/.

[2] Pat Hanrahan, Christian Chabot, Chris Stolte. Tableau, 2013. URL https://www.tableau.com/.