MODELLING AND SOLVING THE MULTI-CALENDAR NAVAL SURFACE SHIP WORK PERIOD PROBLEM UNDER ACTIVITY RELATIONSHIP CONSTRAINTS

by

Yun Yin

Submitted in partial fulfillment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
March 2022

Table of Contents

List of Tables . . . . . **iv**

List of Figures . . . . . **v**

Abstract . . . . . **vi**

List of Abbreviations Used . . . . . **vii**

Acknowledgements . . . . . **viii**

Chapter 1 Introduction **1**

1.1 Problem Description 4
1.1.1 Work period and excess works 5
1.1.2 Work package priority 6
1.1.3 Long duration activities 7
1.1.4 Multi-calendar activities 7
1.1.5 Special network structure 9
1.1.6 Precedence relations 10
1.1.7 Activity start/finish time constraints 12
1.1.8 Resource constraints 13

1.2 Research objectives and dissertation organization 13

Chapter 2 Literature Review **15**

2.1 RCPSP Classification 15
2.1.1 Constraint types for the NSWPP 15
2.1.2 Precedence relations for the NSWPP 17
2.1.3 Resource types for the NSWPP 17
2.1.4 Activity splitting types for the NSWPP 17
2.1.5 Activity execution modes for the NSWPP 18
2.1.6 Number of criteria for the NSWPP 19
2.1.7 Level of information for the NSWPP 19

2.2 Review of directly related literature 19

2.3 RCPSP formulations 21

2.4 Solution methods 26

2.5 Network Indicators 28
2.5.1 Coefficient of Network complexity 29

2.5.2 Order Strength 29
2.5.3 Resource Factor 30
2.5.4 Resource Constrainedness 30
2.5.5 Process Range 32
2.5.6 Disjunction Ratio 32

Chapter 3 Scope and methodologies 33

3.1 Duration matrices 33

3.2 Formulations for calculating ES and LS 38

Chapter 4 Mathematical model and solution method 42

4.1 Multi-calendar multi-mode discrete-time priority-duration RCPSP model 42

4.2 Heuristic Method: Adapted Listing Serial SGS 49
4.2.1 Duration Matrix 49
4.2.2 Priority Rule 50
4.2.3 Schedule Generation Scheme 52

Chapter 5 Experiments and analysis of results 57

5.1 Datasets and Instances 57

5.2 Impact of Solver Tolerance on Gap from Optimum 59

5.3 Duration-Weighted Centroid as Schedule Quality Criterion 60

5.4 Experiments and Results 61
5.4.1 Objective function comparison between BIP and Serial SGS . 61
5.4.2 DWC comparison between BIP and Serial-SGS Solutions . . . 64
5.4.3 Makespan Comparison between the BIP and Serial-SGS . . . 64
5.4.4 Results for the CRUISE-500 Dataset 65

Chapter 6 Conclusions 66

References 68

Appendix A Information of computer used for experiments 72

Appendix B Python Code of the BIP model 73

List of Tables

2.1 Classification of project scheduling problems (Schwindt et al., 2015) . . . 16

2.2 Performance of MILP formulations based on DR and PR . . . 23

3.1 Activity duration per execution mode . . . 37

3.2 ES calculation with precedence relations and activity time constraints . . . 40

3.3 LS calculation with precedence relations and activity time constraints . . . 41

4.1 ES and LS values for activity time constraints with single execution mode . . . 43

4.2 ES and LS values for activity time constraints with multiple execution mode . . . 43

4.3 Data for the Priority Rule Example . . . 51

4.4 Parameters for the Adapted Serial SGS Example . . . 55

5.1 Network Indicators of CRUISE-500 and OTC-136 instances . . . 58

5.2 Characteristics of the instances . . . 58

5.3 Computational Time as a function of Solver Tolerance . . . 59

5.4 Comparison of Schedule-Quality Measures . . . 60

5.5 Comparison between BIP and Adapted Serial SGS . . . 62

5.6 Detailed computational times (in minutes) . . . 62

5.7 Computational Time With and Without Warm-Start . . . 63

5.8 DWC Comparison between the BIP and Serial-SGS . . . 64

5.9 C_{max} Comparison between the BIP and Serial-SGS . . . 65

5.10 Results of CRUISE-500 . . . 65

5.11 DWC Comparison between the BIP and Serial-SGS for CRUISE-500 . . . 65

List of Figures

1.1 Naval ship refit work package and project generation process (Prabhu, 2021) 5

1.2 Naval surface ship maintenance lifecycle (Bertrand, 2020) . . . 6

1.3 Impact of multiple calendars and holidays on activity duration 9

1.4 Typical network structure for construction projects (Croteau, 2018) 10

1.5 Example of a NSWPP project showing the structure at the work package and activity levels 11

1.6 Precedence relationships 12

2.1 Computational complexity vs RC (Herroelen and De Reyck, 1999) 31

3.1 Sample activities 34

3.2 Calendar representation of the duration matrix $\boldsymbol{d}$ 34

3.3 Calendar representation of the duration matrix $\boldsymbol{db}$ in the backward pass 35

3.4 Example of extended duration matrix for multi modes 38

4.1 Project Network for the Priority Rule Example 50

4.2 Network for the Adapted Serial SGS Example 55

4.3 Step by Step Schedule Building based on the Adapted Serial SGS 56

5.1 Illustrating the Need of Duration-Weight 60

5.2 Detailed Computational Time Growth 63

Abstract

This thesis deals with modelling and solving the multi-calendar extensions of the naval surface ship work period problem (NSWPP). NSWPP is a variant of the resource-constrained project scheduling problem (RCPSP) with its own specific network characteristics and constraints including limited work periods, activities of varying priority, multi-calendars activities and resources, precedence relationships, and time constraints. A binary integer programming model focusing on front-loading activities based on priority and duration is developed. The NSWPP is NP-hard causing its solution time to grow exponentially with the number of activities and resources. To shorten computation time, the Serial Schedule Generation Scheme heuristic is modified to adapt to the additional features of the multi-calendar NSWPP. The modified serial SGS reduces computational time from over 40 minutes to around 10 minutes for a large size instance of the problem. Experiments are carried out using data from real large-scale naval refit operations.

List of Abbreviations Used

BIP	Binary Integer Program
CNC	Coefficient of Network Complexity
DDT	Disaggregated Discrete-Time
DR	Disjunction Ratio
DT	Discrete-Time
DWC	Duration Weighted Centroid
DWP	Dry-dock Work Period
ES	Earliest Start
ERP	Enterprise Resource Planning
FCT	Flow-based Continuous Time
FF	Finish to Finish relationship
FS	Finish to Start relationship
LP	Linear Program
LS	Latest Start
MILP	Mixed Integer Linear Program
NSWPP	Naval Surface Ship Work period Problem
OOE	On-off Event
OS	Order Strength
PR	Process Range
RCPSP	Resource-Constrained Project Scheduling Problem
RC	Resource Constrainedness
RF	Resource Factor
RS	Resource Strength
SEE	Start/End Event Based
SF	Start to Finish relationship
SGS	Schedule Generation Scheme
SS	Start to Start relationship
SWP	Short Work Period

Acknowledgements

I am deeply grateful to my supervisors Dr. Claver Diallo and Dr. Alireza Ghasemi for their support, encouragement, patience, and advice throughout this research project. My sincerest gratitude is extended to Dr. Pemberton Cyrus, a member of my supervisory committee, and to Dr. Jenny Chen, the external examiner. They have provided valuable suggestions and insightful comments.

I am very thankful to the incredible works done on the NSWP problem by LCdr Eric Bertrand and Mr. Sanjay Prabhu. Their insightful and innovative works have built a great foundation for research on this type of problem. I also appreciate all the help and support from my peer, Hyojae Kim.

Many thanks to Thales Canada for funding this "Refit Optimizer" research project, and providing me with a great opportunity of participating in this challenging and exciting research project.

Finally, I would like to give a special thanks to my parents, family and friends for their continuous support and motivation.

Chapter 1

Introduction

A project can be defined as a "one-time endeavour that consists of a set of activities, whose executions take time, require resources and incur costs or induce cash flows" (Schwindt et al., 2015). The time, resources and performance elements can further be complicated by additional factors such as due date penalties, precedence constraints and multiple execution modes. The task of effectively coordinating all these activities and requirements from start to finish is known as project management. Project managers are typically required to "perform the project within time and cost estimates at the desired performance level in accordance with the client, while utilizing the required resources effectively and efficiently" (Schwindt et al., 2015). An initial project schedule is designed to meet the intentions provided by management or project sponsor (Larson and Gray, 2021). Minimizing the project makespan (i.e., to finish as soon as possible) is the most commonly considered objective. However, other performance measures exist such as lowest cost completion by a fixed deadline, balanced or levelled resource usage over a given time horizon, maximizing net present value, and reducing the lateness of each individual activity. A number of other potential scheduling objectives have been proposed in the literature (Brucker et al., 1999; Hartmann and Briskorn, 2010).

In the naval maintenance environment, an accurate baseline schedule is needed prior to the project start date to assign specific high value or scarce resources such as heavy cranes, highly specialized technicians, generator test loads or a graving dock to specific periods during the project (Bertrand, 2020). The impact of poorly allocating scarce or high value resources is significant to overall organizational performance. A slight delay affecting these crucial resources has the potential to disrupt the current project as well as the schedule and budget of other ongoing and subsequent projects.

The critical path method (CPM) (Kelley and Walker, 1959) and the program evaluation and review technique (PERT) (Malcolm et al., 1959) are the results of early attempts to apply mathematical modelling to project scheduling problems (PSP). Both methods use known precedence relationships and duration information for each activity to generate feasible project schedules. Schedules obtained by these methods are only precedence and time feasible. They are rarely feasible once resource constraints are considered. Thus, extensions were developed to account for resource limitations giving rise to the Resource Constrained Project Scheduling Problem (RCPSP).

The RCPSP is a scheduling problem of projects containing activities with precedence constraints, and limited resource capacities. The naval surface ship work periods problem (NSWPP) is a highly complex variant of the classical RCPSP (Bertrand, 2020), with its own specific network characteristics, multi-calendars activities and resources, as well as intricate precedence relations between activities, with four types of precedence relations and lags between activities. The complex project network resulting from the NSWPP and the other requirements make it very challenging to schedule and execute, especially for the large size problems. A typical refit project in the Navy can have thousands of activities. The specific requirements of the NSWPP that distinguish it from other variants of RCPSP include activity time constraints, the necessity of completing high priority activities within given time periods, and preference on processing activities of long duration before activities of short duration to ensure that such long-duration activities have a reduced chance of not being completed before the end of the refit period.

Naval repair facilities working on these challenging projects normally use commercial enterprise resource planning (ERP) software with the auto-scheduling function disabled to prevent financial and contractual mishaps (Bertrand, 2020). Scheduling these large projects manually absorbs a great amount of work and workers, yet can hardly produce high-quality solutions.

Bertrand (2020) introduced the NSWPP and developed several binary integer

programming (BIP) models for initial scheduling and schedule recovery after disruptions. He then conducted extensive testing on maintenance data from the Royal Canadian Navy by comparing his BIP to the serial SGS heuristic. Prabhu (2021) developed matheuristic methods to quickly find high-quality solutions for the NSWPP when large problem instances are run. These BIP models, heuristic and matheuristic methods have laid an excellent foundation for the study of the NSWPP.

However, not all key characteristics of the NSWPP were considered in the BIP models and heuristic methods developed by Bertrand (2020) and Prabhu (2021). Their models can handle scheduling problems with activities following only one calendar. However, current shipyard and construction practices dictate that activities and resources can follow multiple calendars. Test datasets provided by Thales and collected from a shipyard on the West Coast of Canada reveal that several calendars are used for activities and workers. For example, some trades or unionized workers do not work on week-ends while others do. Hence, there is a minimum of two calendars to be considered: five-day and seven-day calendars. Furthermore, some calendars can have eight-hour shifts while others have 12-hour shifts. Thus, there is a need to extend the models proposed by Bertrand (2020) and Prabhu (2021) to cover multiple calendars for activities and resources.

In project management, there are four types of precedence relationships between activities: 1) Finish to Start (most commonly encountered), 2) Finish to Finish, 3) Start to Finish and 4) Start to Start (Larson and Gray, 2021). The existing NSWPP formulations only cover the usual Finish to Start relationship. The goal of this research work is to also include the other three relationships in the formulation of the NSWPP

Finally, the third extension proposed is to formulate new activity time constraints not accounted for by Bertrand (2020) and Prabhu (2021). These include constraints such as Start On, Start On or Before, Start On or After, Finish On, Finish On or Before, Finish On or After. These new constraints add significant complexity to the computation of the Earliest Start (ES) and Latest Start (LS) dates.

In summary, this thesis extends the current MILP models for the NSWPP by integrating new features in its formulation: multiple calendars for activities and resources, three new activity relationships, and six new activity time constraints. The serial SGS will also be adapted to handle these new requirements to quickly find good approximate solutions for large-scale problem instances. Experiments are performed using datasets from real large-scale refit operations and their results are analyzed and discussed.

The next section will give a detailed description of the NSWPP under consideration.

1.1 Problem Description

This thesis project is one of many projects under the "Refit Optimizer" research project aiming to develop an intelligent project scheduling application under the sponsorship of Thales Canada. Thales Canada provides service to transportation, aerospace, defence, and security sectors. Thales has been awarded a multi-million dollar in-service-support contract for six Harry-De-Wolfe Class and two Protecteur Class ships. The software being developed in the "Refit Optimizer" project is an add-on scheduling software that can receive input from an ERP system, optimize schedules, then send the optimal schedule back to the ERP system. Three research teams from Dalhousie University, École Polytechnique de Montréal, and Université Laval participate and collaborate in this "Refit Optimizer" research project. The Dalhousie team focuses on the NSWPP.

A NSWPP project typically contains several smaller projects referred to as "work packages." Each work package consists of a number of individual tasks/jobs/operations that should be finished to complete the work package. This thesis uses the term "activity" to refer to theses individual tasks/jobs/operations in work packages. NSWPP scheduling is challenging not only because of the large amount of work packages and their associated activities, but also due to the resource limitations arising from sharing the same resource pools as well as time constraints. Some other characteristics

that makes scheduling NSWPP more difficult are discussed below.

1.1.1 Work period and excess works

Notification of maintenance works are generated in three ways: preventive maintenance, repair inspection, and failures noticed by ship staff during sailing. The electronic maintenance system of the repair facility automatically generates preventive maintenance notifications. Refit notifications can also be created by the repair facility staff when they discover defects during inspection. Additionally, when ship staff find breakdowns that cannot be fixed by themselves on a naval surface ship, a refit notifications is sent to the repair facility electronic maintenance system. Then activities and work packages are created from these notifications by the repair facility planning department. The planning department collects and provides other information needed to perform corrective and preventive maintenance actions. Schedules within given time horizon of maintenance work periods are then generated using these work packages, associated activities, and all other information needed. The general process of schedule generation is depicted in Figure 1.1.

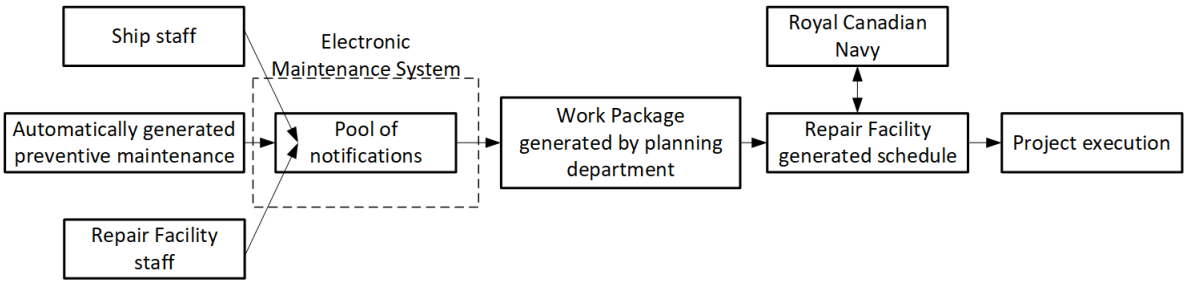

Figure 1.1: Naval ship refit work package and project generation process (Prabhu, 2021)

As depicted by Figure 1.2, a typical maintenance lifecycle of naval surface ships is composed of several short work periods (SWP) with long dry-dock work period (DWP) in between. The length of a work period varies according to the type of the work period, as well as the naval surface ship condition. A DWP may last from 20 weeks to 50 weeks depending on the condition and the age of ships (Bertrand, 2020). Before and after a DWP, naval surface ships undergo an extended work period which lasts a few months for docking preparation and sailing preparation. A naval surface

ship may experience several SWPs that last approximately 12 to 30 weeks in total per year Bertrand (2020).

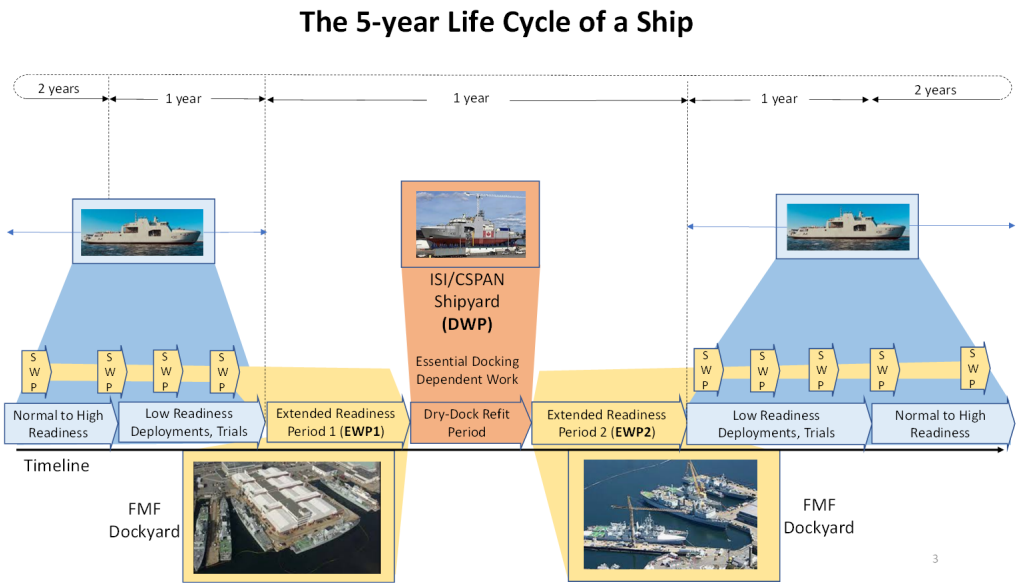

Figure 1.2: Naval surface ship maintenance lifecycle (Bertrand, 2020)

Schedulers try to fit as many activities as possible in maintenance work periods, but usually several activities remain pending at the end of work periods. The length of work periods is estimated and decided considering the number of existing refit notifications (Prabhu, 2021). However, a lot of notifications are raised during inspections and preventive maintenance work resulting in excess work. Naval surface ships may sail for their next mission as long as all essential maintenance works are done by the end of work periods (Bertrand, 2020). The excess maintenance works will then be postponed and planned for subsequent work periods.

1.1.2 Work package priority

To ensure that essential works are completed by the end of work periods and ships can sail safely, work packages and activities are prioritized to indicate their importance. There are generally three levels of priority for work packages and activities: Essential work, high opportunity work, and normal opportunity work labelled as priority levels 1, 2, and 3 respectively (Bertrand, 2020).

Work packages of priority 1 are critical to the successful completion of the next mission and must be completed by the end of work periods. Work packages level 2 and 3 are less important. It is common to see some level 3 activities not executed at the end of a work period and be postponed to subsequent work periods. Given these characteristics of work packages, schedulers in the planning department would like to generate a schedule where all priority–1 activities are completed within the current work period, and then have as many priority–2 activities as possible scheduled by the end of the work period. Priority–3 activities are the last to be considered. Given the desire to have all priority–1 activities completed during the work period and given potential unforeseen delays, it was decided to schedule these activities as early as possible.

1.1.3 Long duration activities

Some long duration activities may not fit in the short work periods (SWPs), but can only be fit in dry dock work periods (DWPs) which happen once per maintenance life cycle. If the activities with long duration are not completed in a DWP, they may have to wait for years until their next chance to be scheduled. Even if a long duration activity is scheduled in DWP, there is still a chance that it cannot be completed. If a long duration activity is scheduled near the end of the project, there is a higher probability that it will not be completed due to various delays and then have to wait for another maintenance period. Thus, it is again important to have long-duration work packages scheduled as early as possible in the DWP. Bertrand (2020) proposed a RCPSP model with the objective of front-loading jobs based on their priority and duration.

1.1.4 Multi-calendar activities

Another characteristic of large-scale projects is that activities can use different calendars while sharing the same resource pool. Calendars are classified by three attributes:

- days per week (e.g., 5 days per week or 7 days per week). Some activities can take place on the weekends other cannot because of limitations on the resources or workers needed to perform these tasks.

- hours per day (e.g., 8 hours per shift or 12 hours per shift). Here again these hours available for work are dependent on the availability of resources and workers, the trades involved, etc.

- holiday exceptions. These are calendar days other than weekends where work cannot be performed such as federal statutory holidays and provincial holidays.

Scheduling NSWPP projects becomes more challenging when multiple calendars for activities and resources are added. Scheduling activities with different calendars into a universal calendar is a challenging task. Actual durations may not be the same as the original estimated duration provided in the data, and these durations can change from one calendar to another. Durations might need to extend to skip the weekend, if weekends occur during the execution of activities. The same goes for calendars with different holiday exceptions. An example with two hypothetical activities is shown in Figure 1.3 to illustrate how durations change according to the calendar considered. In the example, Day 1 is a Monday. Days 6 and 7 are the weekend. Both activity 1 and 2 require 4 work days to be completed. Activity 1 is a predecessor of Activity 2.

In case 1, both activities follow a 7-day calendar. Work can take place during weekends. Activity 1 starts at the beginning of day 1 and ends at the end of day 4 (end date = start date + duration – 1 = 1 + 4 – 1 = 4). Activity 2 starts at the beginning of day 5 and ends on day 8 (5 + 4 –1 = 8). But since activity 2 starts on Friday (day 5), its duration will be extended for two additional days. Hence, in this case the effective duration of both activities is 4 days.

In case 2, both activities follow a 5-day calendar. Work cannot be done during weekends. Activity 1 starts at the beginning of day 1 and ends at the end of day 4 (end date = start date + duration – 1 = 1 + 4 – 1 = 4). Activity 2 starts at the beginning of day 5 and is interrupted by the weekend. Work resume on activity 2 on day 8 and ends on day 10. In this case, the effective duration of activity 2 is longer by 2 days because it spans the weekend.

Case 1: 7-day calendar

Days	Mon	Tue	Wed	Thur	Fri	Sat	Sun	Mon
t	1	2	3	4	5	6	7	8
Activity 1								
Activity 2								

Case 1: 5-day calendar

Days	Mon	Tue	Wed	Thur	Fri	Sat	Sun	Mon	Tue	Wed
t	1	2	3	4	5	6	7	8	9	10
Activity 1										
Activity 2										

Figure 1.3: Impact of multiple calendars and holidays on activity duration

1.1.5 Special network structure

Project network structures for construction projects typically have a high degree of precedence and equally important activities that are all expected to be completed with a minimum makespan. Figure 1.4 shows such a project network with a high degree of precedence. Conversely, large-scale maintenance operations typically have less precedence relations between activities as they can take place simultaneously at different locations and on different systems or subsystems.

The NSWPP exhibits less precedence relationships between work packages. However, each work package is a small project with a high degree of precedence between internal activities (Bertrand, 2020). Additionally, despite the little precedence relations between work packages, work packages have interrelations with each other by competing for the same resources (Bertrand, 2020). Figure 1.5 shows the difference of precedence degree in an NSWPP between the project level and the work package level.

Another difference between the NSWPP and typical network projects is that in

the NSWPP not all work packages must be completed. Therefore, there is no need to include a dummy end activity in the network (Bertrand, 2020).

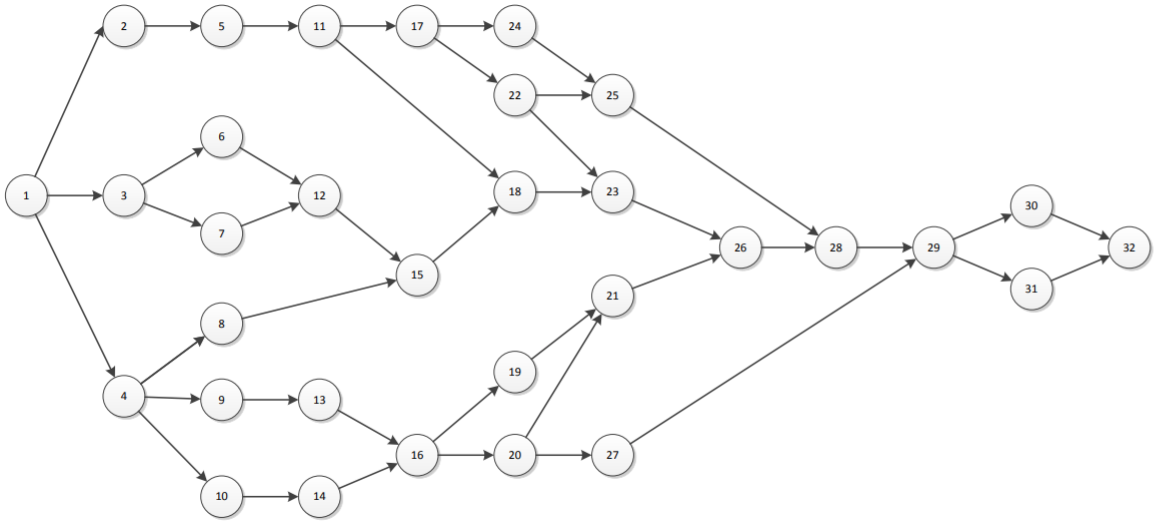

Figure 1.4: Typical network structure for construction projects (Croteau, 2018)

1.1.6 Precedence relations

When Bertrand (2020) and Prabhu (2021) introduced the NSWPP, they only considered the basic Finish to Start precedence relationship. However, project networks can have up to four types of precedence relations: Finish to Start (FS), Finish to Finish (FF), Start of Start (SS), and Start to Finish (SF). Finish to Start requires that successor cannot start until the predecessor finishes, presented in the part a) of Figure 1.6. Finish to finish requires that successor cannot finish until the predecessor finishes. This is illustrated in part b) of Figure 1.6. Start to Start requires that a successor cannot start until its predecessor starts (depicted in the part c) of Figure 1.6). Start to Finish requires that a successor cannot finish until its predecessor starts as illustrated in the part d) of Figure 1.6. With these four types of precedence relations and hundreds of activities in the NSWPP projects, the network structure on the activity level are very large and complex.

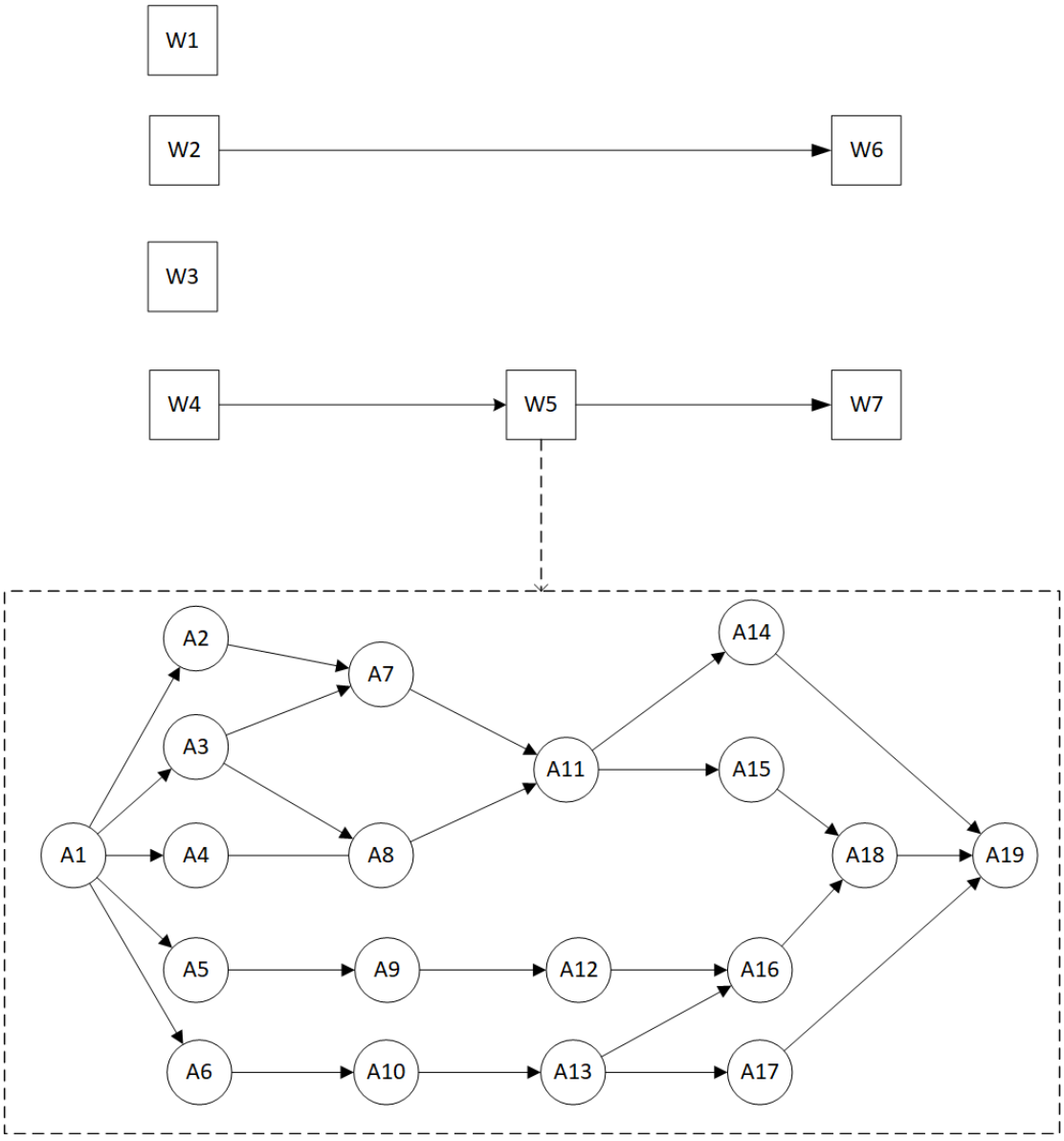

Figure 1.5: Example of a NSWPP project showing the structure at the work package and activity levels

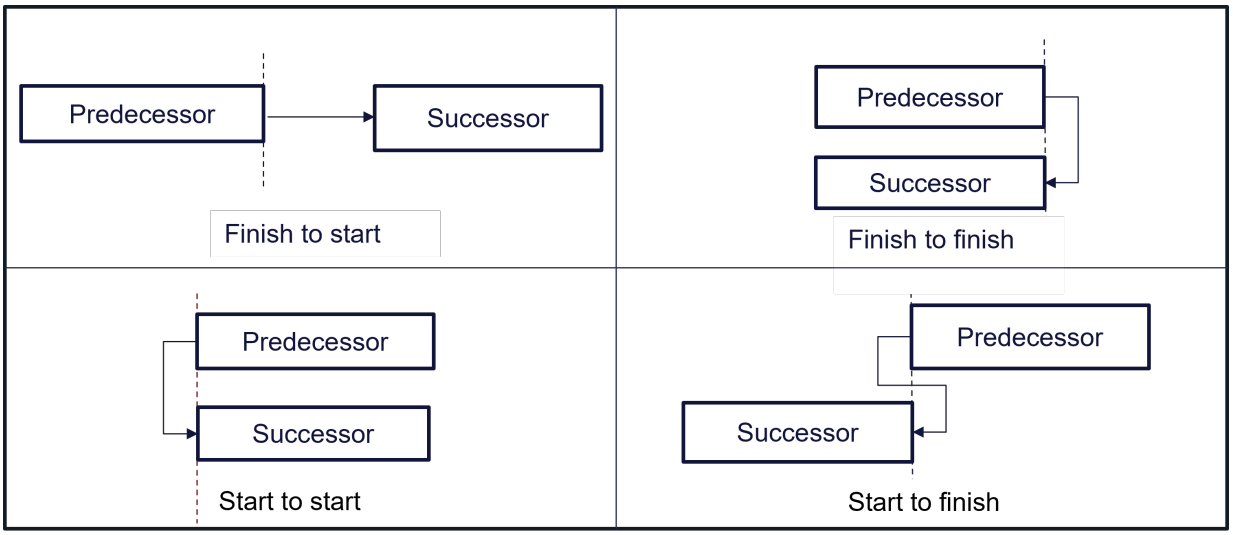

Figure 1.6: Precedence relationships

1.1.7 Activity start/finish time constraints

In typical construction projects, activities start times and finish times are dictated by the precedence and resource constraints. In NSWPPs, Program schedulers typically specify additional start and finish time constraints based on some operational requirements. For example, a special welding job may require inspection and approval from an international expert who can only travel to the shipyard in a given week of the year. The program scheduler would then require the welding job to finish on or before the expert is to arrive. Such constraints are: start on, finish on, start on or after, finish on or after, start on or before, and finish on or before.

From the datasets obtained from our partner, it was found that the ratio of the number of activity start/finish time constraints to the number of activities ranges from 0.184 to 1.956. These additional constraints add another level of complexity to the problem, thus requiring new extensions to the formulation and solving of the NSWPP. For example, adding a Finish On constraint on an activity may make the problem infeasible if the predecessor activity cannot finish early enough for the activity to start and complete on time under its normal duration. There may then be a need to resort to overtime or run the activities in multimode setting.

1.1.8 Resource constraints

Resources in RCPSP consist of labour (workers), machines, tools, space, etc. Resources in the NSWPP may follow different calendars which have various days-off and shift length arrangements. Resources may or may not be available during days-off and weekends. If resources are not available, then activity durations may extend further until the required resources are available.

Furthermore, due to activity start/finish time constraints, overtime and additional resources may be needed to reach a feasible schedule. However, some resource capacities such as space cannot be extended at all. It is also very difficult to gather information about labour resources, such as extra shift availability, during the planning phase which occurs many months ahead of the maintenance period.

1.2 Research objectives and dissertation organization

Unlike traditional construction projects, ship maintenance and/or refit projects exhibit less precedence relationships between work packages. However, despite the sparse precedence relations between work packages, work packages have interrelations with each other by competing for the same resources and spaces (Bertrand, 2020). Furthermore, given the readiness requirements and the available redundancies, ship refit projects have different performance measures than the classical makespan minimization. Bertrand (2020) and Prabhu (2021) proposed the first NSWPP models destined to address the specific characteristics of ship maintenance and/or refit. However, not all key characteristics of the NSWPP were considered in the MILP models and heuristic methods developed by Bertrand (2020) and Prabhu (2021).

The aim of this thesis is to extend the models proposed by (Bertrand, 2020) to include the new requirements and characteristics identified by our industrial partners including:

1. multicalendar activities;

2. multicalendar resources;

3. three additional activity relationships (FF, SS, and SF);

4. activity start/finish time constraints.

The targeted extensions will include building a new transfer matrix to convert multiple calendars into a universal calendar, devising new formulations to calculate Earliest Start (ES) and Latest Start (LS) values for each one of the additional relationships, developing a new binary integer programming (BIP) formulation of the NSWPP, developing a solution algorithm using Python and Gurobi, and conducting numerical experiments using different sizes of datasets provided by the industrial partner.

The remainder of this thesis is organized as follows. Chapter 2 presents a brief and condensed literature review on the NSWPP. Chapter 3 presents the multi-calendar extension under relationship constraints. It presents the transfer matrix method used to handle the multiple calendar requirements. Chapter 4 presents the new BIP formulation developed to deal with the problem. Numerical experiments conducted to test the validity of the proposed models are reported in Chapter 5. Lastly, Chapter 6 presents the conclusions along with some suggestions and areas for future study.

Chapter 2

Literature Review

Many extensions of the RCPSP have been proposed since the initial formulation developed by Kelley and Walker (1959). This section presents a brief literature review on recent developments of the Resource Constrained Project Scheduling Problem (RCPSP). We start by presenting a classification of the different variants, then relevant RCPSP formulations and solution methods are reviewed.

2.1 RCPSP Classification

According to Schwindt et al. (2015), the RCPSP consists of many variants that can be classified based on: type of constraints, type of precedence relations, type of resources, type of activity splitting, number of execution modes, number of objectives, types of objectives, level of information as depicted in Table 2.1.

2.1.1 Constraint types for the NSWPP

Time-constrained project scheduling problems (TCPSP) typically have fixed deadlines but no limit on resource amounts (Guldemond et al., 2008). The duration of activities can be shorten to meet deadlines by paying a higher cost for acquiring extra resources. On the other hand, resource availabilities in RCPSP are strictly limited. NSWPPs have both time activity constraints that specify activity start time or finish time. Resource availabilities in the NSWPP are assumed to be limited because there are very limited pools of resources shared among multiple projects. Therefore, the NSWPP can be classified as both TCPSP and RCPSP but cannot perfectly fit in neither of them.

Table 2.1: Classification of project scheduling problems (Schwindt et al., 2015)

ATTRIBUTES	**CHARACTERISTICS**
Type of constraints	Time-constrained problem
	Resource-constrained problem
Type of precedence relations	Ordinary precedence relations
	Generalized precedence relations
	Feeding precedence relations
Type of resources	Renewable resources
	Non-renewable resources
	Storage resources
	Continuous resources
	Partially renewable resources
Type of activity splitting	Non-preemptive problem
	Integer preemption problem
	Continuous preemptive problem
Number of execution modes	Single-mode problem
	Multi-mode problem
Number of objective types	Single-criterion problem
	Multi-criteria problem
Level of information	Deterministic problem
	Stochastic problem
	Problem under interval uncertainty
	Problem under vagueness

2.1.2 Precedence relations for the NSWPP

Ordinary precedence relation requires that the successor cannot start until the predecessor finishes (Larson and Gray, 2021). Generalized precedence relations express that the time gap between predecessor and successor cannot be less or more than a predefined number. As for feeding precedence relation, the successor cannot start until a certain percentage of the predecessor is completed. From observation, the NSWPP can be classified as generalized precedence relation problem because a time gap is often required between activities. For instance, no successor activity can be executed in a room that has just been painted. The time for drying is the time gap (or lag) needed between the painting activity and its successor activities.

2.1.3 Resource types for the NSWPP

Renewable resource RCPSP is a scheduling problem with resource capacities constrained during every time period that are released back to the resource pool when an activity is completed. This is usually the case for resources such as machines and spaces.

Non-renewable resources are those that are consumed once for all (total consumption) such as budget and time. Storage resources are consumed and replenished over time by activities. Continuous resources are able to be continuously and simultaneously allocated to multiple activities, such as electricity. Partially renewable resources usually refer to time interval of labour availability. Because resources in the NSWPP include space, machine, and labour resources which are occupied during activities execution but are released after the activities are completed, the NSWPP can be classified as a renewable resource RCPSP.

2.1.4 Activity splitting types for the NSWPP

Non-preemptive activities are jobs that must not be interrupted once they start. Integer preemptive activities assume that they can be interrupted and resumed, but they can only be split into integral durations. Continuous preemptive activities can stop and resume at any time. Activities in the NSWPP are assumed to be non-preemptive in this thesis, because assuming that activities are preemptive is impractical. For

most of the activities in the NSWPP, some occupied resources are not released even when the activities are interrupted. For example, when replacing corroded steel under tiles, no other activity can be executed in this space even if the replacement of corroded steel activity is stopped. Therefore, splitting activities in the NSWPP can only extend the duration of the activities with no advantage in minimizing makespan.

2.1.5 Activity execution modes for the NSWPP

Deckro and Hebert (1989) proposed the concept of project crashing in which the duration of activities can be reduced by increasing usage of resources in order to meet required deadlines. Talbot (1982) formulated the multi-mode RCPSP model which has activities with multiple processing modes resulting in various length of duration and resource usage.

Prabhu (2021) modelled the NSWPP as a single-mode RCPSP in which activities have only one execution mode with a specific resource demand. The single-mode was used due to insufficient information on how resource usage affected activity duration. However, Bertrand (2020) proposed a multi-mode NSWPP model by making assumptions on the duration of activity modes with the purpose of showing which activities had the most impact on the schedule. Different from multi-mode RCPSP model in Talbot (1982), only activity durations vary with the choice of activity execution mode while resource usage remained the same in the model proposed by Bertrand (2020). It assumed that activity duration decrease one unit of time with each incremental activity execution mode. By adding a constraint in the formulation to limit the number of non-original execution modes, Bertrand (2020) helps schedulers to make decisions on overtime decision.

With the addition of activity Start-On/Finish-On constraints, it is essential to include the multi-mode execution of activities to guarantee the existence of feasible solutions by allowing the shortening of the duration of predecessor activities. Otherwise, the Start-On/Finish-On activity may not start when required. Thus, the NSWPP is formulated as a multi-mode problem.

2.1.6 Number of criteria for the NSWPP

RCPSPs with single criterion have only one goal of problem solving such as minimizing makespan, minimizing cost, or minimizing the peak of resource utilization. On the opposite, RCPSPs with multiple criteria for decision making have several conflicting goals. In this thesis, the goal of the NSWPP is to complete as many as possible high-priority and long-duration activities within the given maintenance period. In other words, solving NSWPP is finding a solution that front-loads high-priority and long-duration activities as well as minimizes the resulting makespan. Thus, NSWPPs are multi-criteria scheduling problems.

2.1.7 Level of information for the NSWPP

Deterministic problems assume that all information is known in advance, and no change occurs during the implementation of the solution. On the other hand, stochastic problems account for uncertainties which seems more reasonable for real-world practice. However, stochastic problems require the estimates of probability distribution for all the uncertain parameters. In the case of NSWPPs, the two main uncertain parameters are activity duration and resource availability. However, information for estimating probability distribution of activity durations and resource availabilities is insufficient at the current phase of study. Therefore, although surface naval ship projects are performed in dynamic and stochastic environments with unavoidable uncertainties, NSWPP is modelled as a deterministic problem in this thesis. The front-loading of high-priority and long-duration activities permits to minimize the risk that delays may cause the non-completion of these important activities. Hence, this objective function allows for some robustness in the solution.

2.2 Review of directly related literature

In summary, the NSWPP is classified as a deterministic multi-mode multi-criteria scheduling problem with generalized precedence relations, renewable resources, and non-preemptive activities. It has some features of both RCPSP and TCPSP due to the activity time constraints. The research work done by Bertrand (2020) accounts

for many characteristics of NSWPP except for the activity time constraints, generalized precedence relations, and multi-calendar activities and resources. To the best of our knowledge, there is no literature on RCPSP with activity time constraints and the TCPSP is most often a time-resource leveling problem. The extant literature on RCPSP or TCPSP cannot achieve the goal of addressing the NSWPP.

Many papers have investigated the RCPSP with generalized precedence relations. Bianco and Caramia (2012) did a study on RCPSP with generalized precedence relations, and proposed a branch and bound algorithm aiming to minimize makespan. Li and Dong (2018) studied RCPSP with multiple activity execution modes and mode dependent generalized precedence relations, and proposed two meta-heuristics that levels mode choices and resources. de Azevedo et al. (2021) proposed an exact method for RCPSP with generalized precedence relations and criteria of satisfiability and workload. These studies convert FS, FF, and SF relationships into SS relationships for the convenience of calculation using the conversion method developed by Bartusch et al. (1988). This conversion method is for problems with fixed activity duration. Given that the NSWPP has varying activity durations due to multiple calendars, their conversion method cannot be applied.

Kong and Dou (2021) studied an RCPSP with generalized precedence relations and resource calendars with varying length of activity durations similar to the NSWPP. Kong and Dou (2021) proposed a formulation with constraints of minimum and maximum lag time between activities for four types of precedence relations without conversion. Formulations of precedence relation constraints are shown in Equation 2.1 to 2.4 with $l_{ij}^{SS_{min}}$ and $l_{ij}^{SS_{max}}$ referring to the minimum lag time between activity i and activity j in a SS relationship, and S_i and F_i referring to the start time and finish time of activity i. Although the model developed by Kong and Dou (2021) does not account for many characteristic of the NSWPP such as activity time constraints and the different solution goals, this method of formulating generalized precedence relations with varying activity duration is a good baseline reference for the NSWPP.

$$l_{ij}^{SS_{min}} \leq S_j - S_i \leq l_{ij}^{SS_{max}} \tag{2.1}$$

$$l_{ij}^{SF_{min}} \leq F_j - S_i \leq l_{ij}^{SF_{max}} \tag{2.2}$$

$$l_{ij}^{FS_{min}} \leq S_j - F_i \leq l_{ij}^{FS_{max}} \tag{2.3}$$

$$l_{ij}^{FF_{min}} \leq F_j - F_i \leq l_{ij}^{FF_{max}} \tag{2.4}$$

2.3 RCPSP formulations

There are several MILP formulations of RCPSP and its various extensions. Several studies have been conducted to compare the performance of these RCPSP formulations. These studies provide useful guidelines for choosing appropriate formulation methods for RCPSPs based on their characteristics.

There are n activities to be scheduled and k resource types. For formulation purposes a dummy Start ($i = 0$) and a dummy Finish activity ($n + 1$) are created. Each activity i has duration d_i and uses b_{ik} units of resource type k to complete. Each resource type k has B_k renewable units. Using the precedence requirements, the Earliest Start (ES_i) and Latest Start (LS_i) of each activity i are pre-calculated. Pritsker et al. (1969) developed a discrete-time formulation (DT) for this problem using binary decision variables. This binary variable, x_{it}, is indexed by i and t indicating whether activity i starts at time t.

$$x_{it} = \begin{cases} 1, & \text{if activity } i \text{ starts at time } t \\ 0, & \text{otherwise.} \end{cases} \tag{2.5}$$

The formulation for minimizing project makespan is given below where $\mathcal{P}$ is a set of pairs of predecessors (i, j), $\mathcal{H}$ is the set representing the discretized planning horizon and $\mathcal{K}$ is the set of resource types.

$$\text{Min} \sum_{m=ES_i}^{LS_i} t x_{n+1,t} \tag{2.6}$$

s.t.:

$$\sum_{m=ES_j}^{LS_j} tx_{j,t} \geq \sum_{m=ES_i}^{LS_i} tx_{i,t} + d_i \qquad \forall (i,j) \in \mathcal{P} \tag{2.7}$$

$$\sum_{i=1}^{n} b_{ik} \sum_{s=max(ES_i, t-d_i+1)}^{min(LS_i,t)} x_{is} \leq B_k \quad \forall t \in \mathcal{H}, \forall k \in \mathcal{K} \tag{2.8}$$

$$\sum_{t=ES_i}^{LS_i} x_{it} = 1 \qquad \forall i \in \mathcal{P} \cup \{n+1\} \tag{2.9}$$

$$x_{00} = 1 \tag{2.10}$$

$$x_{it} = 0 \qquad \forall i \in \mathcal{P} \cup \{n+1\}, t \in \mathcal{H} \setminus \{ES_i, LS_i\} \tag{2.11}$$

$$x_{it} \in \{0,1\} \qquad \forall i \in \mathcal{P} \cup \{n+1\}, t \in \{ES_i, LS_i\} \tag{2.12}$$

The objective is to minimize the completion time of the last activity as given by Equation 2.6. Constraints 2.7 ensure that an activity j is started only after its predecessor i is completed in an FS relationship. Constraints 2.8 ensure that the execution of activities in any time period does not use more than the amounts of each resource available. Constraints 2.9 guarantees that all activities are scheduled. Constraint 2.10 sets the start time of the dummy Start activity at time 0. Constraints 2.11 ensure that activities are not scheduled outside of their [ES, LS] time windows. Constraints 2.12 defines the decision variables as binary.

A disaggregated discrete-time formulation (DDT) for RCPSP, modified from the DT, is proposed by Christofides et al. (1987). The only difference is the precedence constraint. The DDT results in tighter relaxations than DT (Koné et al., 2011). The precedence constraint of DDT is as follows:

$$\sum_{s=t}^{LS_i} x_{is} + \sum_{s=ES_j}^{min(LS_j, t+d_i-1)} x_{is} \leq 1 \quad \forall (i,j) \in \mathcal{P} \tag{2.13}$$

A flow-based continuous-time formulation (FCT) for RCPSP was proposed by Artigues et al. (2003). This formulation contains three types of variables; start time continuous variables, sequential binary variables that indicate whether activities are processed before each other, continuous flow variables which express resource transferring between activities. According to Artigues et al. (2003), the FCT has poor

relaxations compared to DT and DDT. However, the FCT has advantages because it has a polynomial number of variables and constraints while DT and DDT have a pseudo-polynomial number of variables and constraints (Koné et al., 2011). FCT performs better than DT and DDT in problems with large time horizon.

Event-based RCPSP formulations such as start/end event-based (Zapata et al., 2008) and on/off event-based (Koné et al., 2011) are also found in the literature for different variants of the RCPSP. Event-based RCPSP formulations can take care of problems with non-integer activity processing times, and have less number of variables for problem with long horizon compared to DT, DDT, and FCT. However, event-based formulations only surpass the other three formulations when problems are complex and with very long scheduling horizon (Koné et al., 2011).

These RCPSP formulations have their own advantages and disadvantages when used to model different variants of the RCPSP. A computational study comparing performances of discrete-time, flow-based continuous time, and event-based RCPSP formulations is conducted by Koné et al. (2011). It provides a guide for choosing formulations based on the complexity of the problems measured using the Process range (PR) and Disjunction ratio (DR) indicators. These complexity indicators are presented in the Network Indicators section of this thesis (see 2.5). Formulation performances are summarized in Table 2.2 where the symbol $\succ$ means "dominate". NSWPP instances have relatively low PR and medium DR which indicates that discrete-time RCPSP formulations may be an appropriate choice. Prabhu (2021) ran many experiments for the NSWPP and found that the DDT does not outperform the DT.

Table 2.2: Performance of MILP formulations based on DR and PR

	High DR	**Low DR**
Low PR	DDT $\succ$ DT$\succ$ FCT $\succ$ OOE_x $\succ$ SEE	DDT $\succ$ DT $\succ$ OOE_x $\succ$ FCT $\succ$ SEE
High PR	FCT,OOE_x $\succ$ SEE $\succ$ DT $\succ$ DDT	OOE_x $\succ$ FCT $\succ$ SEE $\succ$ DT $\succ$ DDT

Bertrand (2020) developed a deterministic and discrete-time MILP formulation for the NSWPP that accounted for the requirements to front-load high priority and long duration activities, multi-modes activities, basic precedence constraints, and resource constraints. However, multi-calendars activities and resources, the other three types of relationship constraints, and start/finish time constraints are not included. The formulation in Bertrand (2020) is presented below.

Indices:

i index of activities

j index of activities

t index of time periods (e.g., days)

m index of execution modes

Parameters for Sets:

n integer, number of non-dummy activities

H integer, time horizon considered for scheduling

K integer, number of resource types

M integer, number of execution modes

Sets:

$\mathcal{J}$ Set of activities, $\mathcal{J} = \{0, \cdots, n\}$ with index i or j

$\mathcal{H}$ Set of time periods, $\mathcal{H} = \{0, \cdots, H\}$ with index t

$\mathcal{K}$ Set of types of resources, $\mathcal{K} = \{1, \cdots, K\}$ with index k

$\mathcal{M}$ Set of execution modes, $\mathcal{M} = \{0, \cdots, M\}$ with index m

$\mathcal{P}$ Set of immediate finish to start activity pairs (i, j)

$\mathcal{W}_j$ Set of time periods between the early start and late start of activity j, $\mathcal{W}_j = \{ES_j, \cdots, LS_j\}$

Other Parameters:

d_{im} (d_{jm}) integer, duration of activity i (j) under mode m

r_{jk} integer, activity j demand for resource type k

R_k integer, capacity of resource k

ES_j integer, earliest start time of activity j

LS_j	integer, latest start time of activity j
p_j	integer, priority of activity j
θ	exponent used to ponderate the priority of activities, $\theta \geq 0$
ε_1	very small duration value assigned to activities with no duration (i.e., milestones), $\varepsilon_1 \geq 0$
ε_2	very small weight used to penalize late scheduling of activities, $\varepsilon_2 \geq 0$
α	parameter used to give tie-breaking benefits to longer duration work (α=1.1) or making each work duration unit equal (α=0), $\alpha \geq 0$
β_{M1}	total alternative shift limit
β_{M2}	alternative mode reduction limit

Decision Variables:

$$x_{jtm} = \begin{cases} 1, & \text{if activity } j \text{ starts at time } t \text{ in mode } m \\ 0, & \text{otherwise.} \end{cases}$$

The formulation is given by:

$$\text{Max} \sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{W}_j} \sum_{m\in\mathcal{M}} \frac{x_{jtm}}{p_j^{\theta}} (\varepsilon_1 + d_{jm})^{\alpha} (1 - \varepsilon_2 t) \tag{2.14}$$

s.t.:

$$\sum_{m\in\mathcal{M}} \sum_{t\in\mathcal{W}_j} t\, x_{jtm} \leq 1 \qquad \forall j \in \mathcal{J} \tag{2.15}$$

$$\sum_{m\in\mathcal{M}} \sum_{t\in\mathcal{W}_j} t\, x_{jtm} \geq \sum_{m\in\mathcal{M}} \sum_{t\in\mathcal{W}_i} (t + d_{im})\, x_{jtm} \qquad \forall j \in \mathcal{J}, \forall (i,j) \in \mathcal{P} \tag{2.16}$$

$$\sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{J}} \sum_{b=max\{t-d_{jm}+1, ES_j\}}^{min\{LS_j, t\}} r_{jk} x_{jbm} \leq R_k \qquad \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{2.17}$$

$$\sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{W}_j} (x_{jt2} + 2x_{jt3} + ... + (M-1)\, x_{jtM}) \leq \beta_{M1} \tag{2.18}$$

$$\sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{W}_j} (x_{jt2} + x_{jt3} + ... + x_{jtM}) \leq \beta_{M2} \tag{2.19}$$

$$x_{jtm} \in \{0,1\} \qquad \forall j \in \mathcal{J}, t \in \mathcal{W}_j \qquad (2.20)$$

The objective is front-loading high priority and long duration activities in schedules as given by Equation 2.14. Constraint 2.15 ensures that all activities are scheduled no more than one time. Constraint 2.16 ensures that an activity j is started only after its predecessor i is completed in an FS relationship. Constraint 2.17 guarantees resource feasibility of all activities during their execution. Constraint 2.18 limits the total duration reduction of activities due to overtime. Constraint 2.19 limits the total number of activities executed in overtime modes.

ES-LS time window

It can be observed that discrete-time RCPSP formulations are time window sensitive. Therefore, some data pre-processing is needed to predetermine the [ES, LS] time windows for the project activities. Moreover, Kolisch et al. (1995) found that the computational time of RCPSP can benefit from ES-LS time windows by reducing the number of binary variables in the problem. However, this benefit reduces significantly if the LS is calculated from a large time horizon instead of from a time horizon relatively close to the optimal makespan (Kramer and Jenkins, 2006). Therefore, properly predetermining a tight project time horizon and [ES-LS] time windows using heuristics is helpful in reducing computational time.

2.4 Solution methods

The Branch-and-Bound, Branch-and-cut methods are the best exact methods to date for solving the RCPSP (Demeulemeester and Herroelen, 1997; Laborie, 2005; Sprecher, 1996). These methods are widely implemented by many solvers such as CPLEX, GLPK/Gusek, and Gurobi. The numerical experiments carried out in this thesis were run using Gurobi.

According to Blazewicz et al. (1983), the RCPSP belongs to the class of NP-hard problems. It is observed that NSWPPs, extensions of the RCPSP, cannot be solved in reasonable time for large-size instances. In practice, the efficiency of the scheduling method is usually preferred to the determination of the optimal schedule after a very

long resolution time. Many research studies investigated heuristic methods for the RCPSP in order to achieve feasible solutions in a reasonable amount of time. Classic heuristics for the RCPSP can be classified in two types: single-pass heuristics, and multi-pass heuristics. Single-pass and multi-pass heuristics concentrate on obtaining a feasible solution quickly but farther from optimum (Pellerin et al., 2020). Both single-pass and multi-pass heuristics are built based on schedule generation scheme (SGS), including serial and parallel SGSs.

Schedule generation scheme

Among all heuristics, priority rule based scheduling is the most popular and important technique for three reasons (Zhang and Sun, 2011): 1) it is easy to use; (2) it consumes relatively short computational time; and 3) it obtains best results compared to other heuristics, in the multi-pass method. Priority rule based scheduling scheme consists of two parts: schedule generation scheme and priority rule (Kolisch and Hartmann, 2006).

SGS, proposed by Kelley (1963), is one of the classic heuristic techniques for RCPSPs Kolisch and Hartmann (2006). SGSs build feasible schedules by step-wise extension of partial schedules, which are schedules where only subsets of activities have been scheduled Hartmann and Kolisch (2000). There are two types of SGS: serial and parallel SGSs distinguished by activity and time increment of steps.

Serial SGS schedules one activity at a time on its earliest start time constrained by resources and precedence relations (Kelley, 1963). Each iteration of serial SGS selects an eligible activity from the unscheduled activities, and schedules it at its earliest time of precedence and resource feasibility. There is a variant of serial SGS named listing serial SGS (Hartmann, 1998). This variant starts with building a list of ordered activities based on certain priority rules. This list has to be precedence feasible, meaning that each activity has its own network of predecessors as list predecessors (Hartmann, 1998). Activities are scheduled at their earliest feasible time according to this list. Geiger (2013) adapted a repair procedure in the Serial SGS heuristic to find a feasible initial solution for multi-mode RCPSP. When a mode of an

activity causes infeasibility, this repair procedure is triggered to select another mode randomly. Repair procedure stops when a feasible mode is found or the maximum number of re-selections is reached. However, feasibility is not guaranteed by this repair procedure.

On the other hand, parallel SGS schedules a set of activities every time for each discrete time unit (Kelley, 1963). For each iteration of parallel SGS, a subset of unscheduled activities are scheduled at time t. This subset of activities must be precedence feasible. It goes to next iteration that schedules activities at time $t + 1$ when there is no more activities that can be scheduled at time t feasibly. Kolisch and Hartmann (2006) concludes that parallel scheduling scheme does not generally have superior performance than serial scheduling scheme, and serial scheduling scheme is a better option for hard RCPSP problems (large size and moderate resource constraints) (Kolisch and Hartmann, 2006).
Both SGSs can be employed in X-pass methods. X-pass methods are distinguished by the number of schedules generated: single-pass ($X = 1$), multi-pass ($X > 1$) (Hartmann and Kolisch, 2000). Solution of multi-pass method is the best solution out of multiple schedules generated employing different priority rules (Hartmann and Kolisch, 2000). Schedules are created from scratch, and do not consider any knowledege from each other (Hartmann and Kolisch, 2000).

For both serial scheduling SGSs, the choice of next activity to be scheduled follows a suitable priority rule. Türkakın et al. (2021) classified heuristic priority rules as four types: network related, time values related, resource related, and combination of network, time values, and resource related. Hartmann and Kolisch (2000) found that the selection of SGS type and priority rules depend on the problem criteria.

2.5 Network Indicators

Network indicators can be used to measure the complexity of problems and help to determine the proper solution method for RCPSPs. Some indicators capture information on the size and topological structure of the project network. Other indicators

deal with the resources allocated to the project activities. These indicators are classified under four groups as follows:

- Precedence-oriented indicators: Coefficient of Network complexity (CNC), Order Strength (OS), etc.
- Resource-oriented indicators: Resource Factor (RF), Resource Constrainedness (RC), etc.
- Time-oriented indicators: Process Range (PR), etc.
- Hybrid indicators: Resource Strength (RS), Disjunction Ratio (DR), etc.

In this subsection, six commonly used instance indicators are presented: (CNC), (OS), (RF), (RC), (PR), and (DR). More indicators are described in Vanhoucke et al. (2012) and De Reyck and Herroelen (1996).

2.5.1 Coefficient of Network complexity

The Coefficient of Network complexity (CNC) measures how dense the project network is. It is the ratio of the total number of non-dummy precedence pairs (P) to the total number of non-dummy activities (n).

$$CNC = \frac{P}{n} \tag{2.21}$$

A project with many precedence pairs is said to be disjunctive in nature: many activities cannot be done in parallel. A project with a low number of precedence pairs is said to be cumulative in nature: many activities can be done in parallel. A number of studies seem to confirm that problems become easier with increasing values of the CNC (Herroelen and De Reyck, 1999).

2.5.2 Order Strength

The Order Strength (OS) is another network density indicator. It is the ratio of the number of all precedence relations (not only the direct relations but also including the transitive ones) denoted by $\hat{P}$ to the theoretical maximum number of precedence pairs.

$$OS = \frac{\hat{P}}{\frac{n^2-n}{2}} \tag{2.22}$$

Like the CNC, the OS is a precedence-oriented indicator giving a measure of the network density (De Nijs, 2013; Herroelen and De Reyck, 1999). Small values of CNC and OS indicate that a project has many activities that can be scheduled in parallel Koné et al. (2011). Small values of CNC and OS also means longer computational times for finding optimal solution resulting from wider activity [ES, LS] time windows and more decision variables (Herroelen and De Reyck, 1999).

2.5.3 Resource Factor

Resource Factor (RF) denotes the average resource demand for a resource type requested per activity.

$$RF = \frac{1}{n\,|\mathcal{K}|} \sum_{i \in \mathcal{J}} \sum_{k \in \mathcal{K}} \begin{cases} 1, & \text{if } r_{i,k} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.23}$$

where $\mathcal{J}$ is the set of activities, $\mathcal{K}$ is the set of resource types, and $r_{i,k}$ is the demand for resource type k by activity i.

The RF measures the density of the resource demand matrix r_{ik}. It simply scans for each activity/resource combination whether the resource is requested by the activity or not and calculates the average portion for all resources requested by all activities (percentage of resource use). If $RF = 1$, then all resources are demanded by all activities. If $RF = 0$, then none of the resources are demanded by any of the activities. Kolisch et al. (1995) conducted experiments on 480 RCPSP instances with 30 activities and four types of resources to show that the CPU time increases as the RF increases.

2.5.4 Resource Constrainedness

Resource Constrainedness (RC_k) defines the average usage of resource type k over the activities using that resource k.

$$RC_k = \frac{\sum_{i \in \mathcal{J}} r_{i,k}}{a_k \cdot \sum_{i \in \mathcal{J}} \begin{cases} 1, & \text{if } r_{i,k} > 0 \\ 0, & \text{otherwise} \end{cases}} \tag{2.24}$$

where a_k is the availability for resource type k.
Finally,

$$RC = \frac{\sum_{k \in \mathcal{K}} RC_k}{|\mathcal{K}|} \tag{2.25}$$

RC indicates the average usage of resources (De Nijs, 2013; Kolisch et al., 1995; Patterson, 1976). Large values of RC indicate a highly resource-constrained project. Computational time increases as RC increase at first, then drops after it reaches a peak (Herroelen and De Reyck, 1999) as depicted in Figure 2.1 below. The RC exhibits an easy-hard-easy complexity pattern that resembles a bell curve. Problem instances with very low or very high values of RC are not too difficult to solve. Instances with RC values between 0.4 and 0.75 require more computational effort to solve.

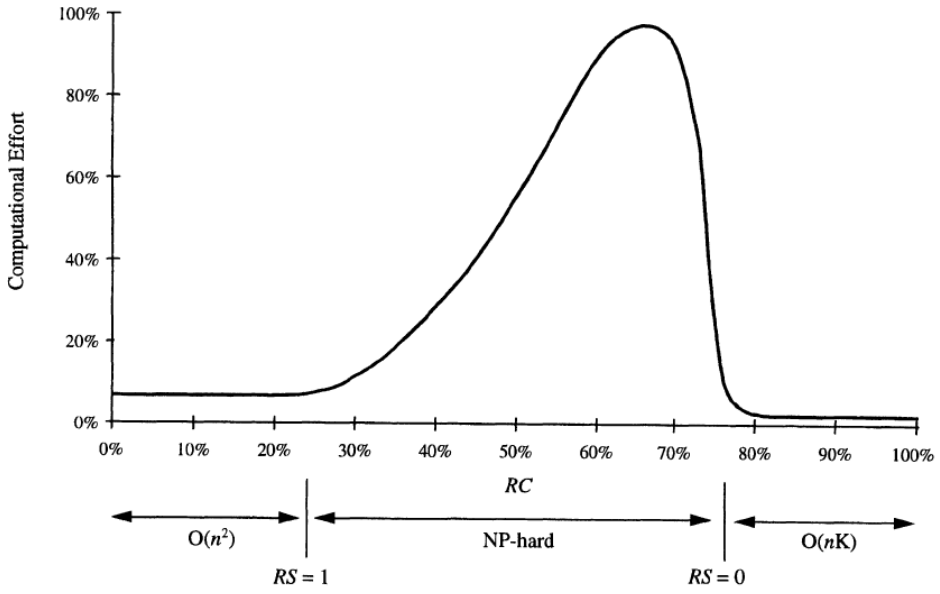

Figure 2 Computational complexity *vs* RC and RS.

Figure 2.1: Computational complexity vs RC (Herroelen and De Reyck, 1999)

2.5.5 Process Range

Process Range (PR) as shown in Equation 2.26, is the ratio between the maximum and minimum duration of project activities. A large PR means big duration difference between activities which increases the computational time as the solver must search through a larger solution space to assign an optimal start time for the longer duration activities present among the short duration ones (Herroelen and De Reyck, 1999).

$$PR = \frac{\max\limits_{i=1...n} \{D_i\}}{\min\limits_{i=1...n} \{D_i\}} \tag{2.26}$$

2.5.6 Disjunction Ratio

Disjunction Ratio (DR) is the product of the coefficient of network complexity (CNC)and the resource constrainedness (RC).

$$DR = CNC \times RC \tag{2.27}$$

Koné et al. (2011) proposed an MILP formulation type selection guide based on the DR and PR indicators as depicted in Table 2.2.

Chapter 3

Scope and methodologies

As discussed earlier the main goal of this thesis is to extend the NSWPP models proposed by Bertrand (2020) and Prabhu (2021) to include the new requirements and characteristics identified by our industrial partners including: multicalendar activities, multicalendar resources, three additional activity relationships (FF, SS, and SF), activity start/finish time relationships.

The targeted extensions will include building a new transfer matrix to convert multiple calendars into a universal calendar, devising new formulations to calculate Earliest Start (ES) and Latest Start (LS) values for each one of the additional relationships, developing a new BIP formulation of the NSWPP, developing a solution algorithm using Python and Gurobi, and conducting numerical experiments using different sizes of datasets provided by the industrial partner.

This chapter will present the transfer matrix developed to accommodate the multi-calendar requirements and the calculations of the Earliest Start (ES) and Latest Start (LS) values when all new relationships and additional constraints are incorporated in the problem. The BIP formulations will be developed in the next chapter.

3.1 Duration matrices

As discussed in the problem description section, it is challenging to formulate a mathematical model to schedule activities and resources following different calendars. To solve this problem, we develop a unique and novel approach that deals with the multi-calendar requirements outside of the mathematical formulation by creating two duration matrices that convert all calendars into one universal calendar. Once the activity durations are set in the universal calendar, the mathematical formulation can be developed as if we were dealing with one calendar.

An example is presented here to illustrate the process of building the duration matrix. Figure 3.1 shows the durations of two activities following two different calendars. Activity 1 has a duration of 3 days and follows a 7-day calendar while activity 2 has a duration of 5 days and follows a 5-day calendar (i.e., no work on weekends). It also is assumed that day 4 (Thursday) of the week under consideration is a holiday and no work can take place in both calendars.

Activity	Duration	Calendar
1	3	7 days
2	5	5 days

Figure 3.1: Sample activities

Figure 3.2 shows a calendar representation of the basic duration matrix $\boldsymbol{d}$ obtained for the example considered. Element d_{jt} represents the effective duration of activity j if it were to start on day t. Q is a large positive number. Days when works are prohibited or not allowed because of calendar requirements have duration values of $-Q$ to prevent their selection in a maximization model. The value of Q must be selected to avoid over-scaling issues. A good value would be slightly larger than the planning horizon H to be determined later with the serial SGS heuristic.

$$\boldsymbol{d} = \begin{pmatrix} 3 & 4 & 4 & -Q & 3 & 3 & 3 & 3 & 3 \\ 8 & 8 & 8 & -Q & 7 & -Q & -Q & 5 & 7 \end{pmatrix}$$

				Holiday						
Days	Mon	Tue	Wed	Thur	Fri	Sat	Sun	Mon	Tue	Wed
t	1	2	3	4	5	6	7	8	9	10
Activity 1	3	4	4	-Q	3	3	3	3	3	3
Activity 2	8	8	8	-Q	7	-Q	-Q	5	7	7

Figure 3.2: Calendar representation of the duration matrix $\boldsymbol{d}$

For example, $d_{11} = 3$ means that the effective duration of activity 1 is 3 if it is started on day 1. Indeed, if activity 1 starts at the beginning of day 1 (Monday) it will finish at the end of day 3 (Wednesday), hence its duration is 3 full days. $d_{12} = 4$ means that the effective duration of activity 1 is 4 if it is started on day 2. The duration is extend by one day because the third day of work would be Thursday which is a holiday. Thus, the third day of actual work is Friday or day 5. So that the effective duration in the universal calendar is 4 days. Similarly, we get $d_{21} = 8$ meaning that the effective duration of activity 2 is 8 if it is started on day 1 because the duration spans the holiday and the weekend, thus adding three days to the effective duration in the universal calendar.

As mentioned earlier in the literature review section, predetermining the $[ES, LS]$ windows for activities shortens the model's solve time because the optimizer does not have to consider the whole planning horizon H. In project management, the ES values are typically calculated in a forward pass while the LS values are calculated in a backward pass. The transfer matrix $\boldsymbol{d}$ presented above is constructed in the forward pass. An equivalent transfer matrix denoted $\boldsymbol{db}$ is constructed in the forward pass and will be used to calculate the LS values. Element db_{jt} represent the effective backward path duration of activity j if it were to "end" on day t. Figure 3.3 shows a calendar representation of the basic backward duration matrix obtained for the illustrative example with two activities presented in Figure 3.1

$$\boldsymbol{db} = \begin{pmatrix} -Q & -Q & 3 & 4 & 4 & 4 & 3 & 3 & 3 \\ -Q & -Q & -Q & -Q & -Q & -Q & -Q & 8 & 8 \end{pmatrix}$$

				Holiday						
Days	Mon	Tue	Wed	Thur	Fri	Sat	Sun	Mon	Tue	Wed
t	1	2	3	4	5	6	7	8	9	10
Activity 1	-Q	-Q	3	4	4	4	3	3	3	3
Activity 2	-Q	-Q	-Q	-Q	-Q	-Q	-Q	8	8	8

Figure 3.3: Calendar representation of the duration matrix $\boldsymbol{db}$ in the backward pass

For example, $db_{17} = 3$ means that the effective duration of activity 1 is 3 if it ends

on day 7. Indeed, for activity 1 to end on day 7 (Sunday) it must have started at the beginning of Day 5 (Friday). Given that activity 1 follows a 7-day calendar and there is no holiday between day 5 and day 7, its duration does not change from the original. $db_{16} = 4$ because activity 1 must start on day 3 to end on day 6 because day 4 is a holiday which extends the effective duration by 1 day.

As discussed earlier, NSWPPs have limited length of refit periods, especially for Priority-1 activities. NSWPPs also have time constraints for activities specifying their start dates and/or end dates. With these activity time constraints and complex precedence relationships, it is possible for a problem to be infeasible if some activities are not crashed (e.g., duration reduced by overtime).

Bertrand (2020) developed a multi-mode NSWPP model that can assign overtime to the most impactful activities to ensure projects can be completed on time. These processing modes include overtime and other execution modes with decreasing activity duration in the multi-mode model. For example, an activity that can normally be executed between 8am to 4pm requires 16 hours to be completed. Its duration in normal mode is 2 days, but the duration of overtime mode can be reduced to 1 day by adding an extra 8 hour shift from 4pm to 12am.

To include the multi-mode characteristic in the extended problem formulation, the forward transfer duration matrix is extended from two dimensions to three dimensions where each element of the matrix is a row vector with M elements. M is the number of execution modes available. Mode 1 is the normal duration mode while mode M is the smallest duration mode. The m^{th} element d_{jtm} of the row vector at the intersection of row j and column t of matrix $\boldsymbol{d}$ is the effective duration of activity j under execution mode m if it were to start on day t. In general, we have:

$$\boldsymbol{d} = \begin{pmatrix} [d_{111}, \cdots, d_{11M}] & [d_{121}, \cdots, d_{12M}] & \cdots & [d_{1t1}, \cdots, d_{1tM}] & \cdots & [d_{1H1}, \cdots, d_{1HM}] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & [\cdots, \boldsymbol{d_{jtm}}, \cdots] & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ [d_{n11}, \cdots, d_{n1M}] & \cdots & \cdots & [d_{nt1}, \cdots, d_{ntM}] & \cdots & [d_{nH1}, \cdots, d_{nHM}] \end{pmatrix}$$

For the illustrative example introduced at the start of this section in Figure 3.1, assuming that activity 1 can only be reduced by one day and activity 2 can be reduced by at most two days, the following duration per execution modes are defined as depicted in Table 3.1.

Table 3.1: Activity duration per execution mode

	Modes		
Activity	1	2	3
1	3	2	2
2	5	4	3

The following transfer matrix is then obtained:

$$\boldsymbol{d} = \begin{pmatrix} [3,2,2] & [4,3,3] & [4,3,3] & [4,3,3] & [3,2,2] & \cdots \\ [8,7,6] & [8,7,6] & [8,7,6] & [8,7,6] & \mathbf{[7,6,5]} & \cdots \end{pmatrix}$$

Vector $[7, 6, 5]$ at the intersection of row 2 and column 5 gives the three effective duration values for activity 2 if it where to start on day 5. Under mode 1 the duration is 7, under mode 2 the duration is 6 (reduced by one day), and under mode 3 the duration is 5 (reduced by one additional day).

Figure 3.4 shows the calendar representation of the extended duration matrix for the two-activity example. Similarly, the backward duration matrix is extended to include multi-mode execution.

				Holiday	
Days	Mon	Tue	Wed	Thur	Fri
t	1	2	3	4	5
Activity 1	[3,2,2]	[4,3,3]	[4,3,3]	[-Q,-Q,-Q]	[3,2,2]
Activity 2	[8,7,6]	[8,7,6]	[8,7,6]	[-Q,-Q,-Q]	[7,6,5]

Figure 3.4: Example of extended duration matrix for multi modes

The duration of activities in the data provided by the industrial partner are in hours. Using hours over a scheduling period of several hundreds of days would make the time horizon H too large and considerably slowdown the optimization process. Furthermore, at the initial planning and scheduling phase there is no need to be more granular than a day as unit of time. Additionally, estimates of duration are usually imprecise and subject to changes. Deviations in execution may be too large if a schedule is built in units of hours. But deviations in hours may be hidden in schedules built with units of days. For example, if an activity estimated to take 5 hours to complete ends up taking 7 hours, a schedule built in hours would see all successor activities delayed by two hours. However, a schedule in days would easily absorb this deviation because the activity would have a buffer of three hours in a 8-hours shift, thus this deviation does not affect any successor activity.

3.2 Formulations for calculating ES and LS

Many papers in the RCPSP literature use serial SGS to determine the scheduling time horizon, ES, and LS. The vast majority of these papers describe methods that only accommodate the Finish-to-Start (FS) relationship. To adapt our formulation to all types of precedence relationships that may exist in a NSWPP project, we derive new equations (3.1) to (3.8) to compute the ES and LS of project activities under one single calendar, where (i, j) denotes a pair of activities in the precedence relationships such that i is the immediate predecessor of j (i.e., j is the successor of i).

$$ES(FS)_j = ES_j + d_i \tag{3.1}$$

$$ES(FF)_j = ES_j + d_i - d_j \tag{3.2}$$

$$ES(SS)_j = ES_j \tag{3.3}$$

$$ES(SF)_j = ES_j - d_j \tag{3.4}$$

$$LS(FS)_i = LS_j - d_j \tag{3.5}$$

$$LS(FF)_i = LS_j + d_j - d_i \tag{3.6}$$

$$LS(SS)_i = LS_j \tag{3.7}$$

$$LS(SF)_i = LS_j - d_j \tag{3.8}$$

To handle the multi-calendar NSWPP, the above equations are then modified with the forward and backward duration values as follows

$$ES(FS)_j = ES_j + d_{i,ES_j} \tag{3.9}$$

$$ES(FF)_j = ES_j + d_{i,ES_j} - db_{[j],[ES_j+d_{i,ES_j}]} \tag{3.10}$$

$$ES(SS)_j = ES_j \tag{3.11}$$

$$ES(SF)_j = ES_j - db_{j,ES_j} \tag{3.12}$$

$$LS(FS)_i = LS_j - db_{i,LS_j-1} \tag{3.13}$$

$$LS(FF)_i = LS_j + d_{j,LS_j} - db_{[i],[LS_j+d_{j,LS_j}]} \tag{3.14}$$

$$LS(SS)_i = LS_j \tag{3.15}$$

$$LS(SF)_i = LS_j - db_{i,LS_j} \tag{3.16}$$

After adding execution modes and extending the duration matrices, the $[ES, LS]$ windows widen because of the options of shorter durations. If a predecessor activity is performed according to a short duration mode, its successor can start earlier. This is also true for determining the LS; if all successors of an activity use their short duration modes, the activity can start later. Thus, Equations (3.17) to (3.24) are updated to account for the execution modes. Recall that activities are assumed to have M processing/execution modes. Performing in mode M results in the shortest duration while mode 1 represents the non-overtime mode that has the longest (normal) duration.

$$ES(FS)_j = ES_j + d_{i,ES_j,M} \tag{3.17}$$

$$ES(FF)_j = ES_j + d_{i,ES_j,M} - db_{j,ES_j+d_{i,ES_j,M},1} \tag{3.18}$$

$$ES(SS)_j = ES_j \tag{3.19}$$

$$ES(SF)_j = ES_j - db_{j,ES_j,1} \tag{3.20}$$

$$LS(FS)_i = LS_j - db_{i,LS_j-1,M} \tag{3.21}$$

$$LS(FF)_i = LS_j + d_{j,LS_j,1} - db_{i,LS_j+d_{j,LS_j,1},M} \tag{3.22}$$

$$LS(SS)_i = LS_j \tag{3.23}$$

$$LS(SF)_i = LS_j - db_{i,LS_j,M} \tag{3.24}$$

In addition to precedence relationships, activity Start-On/Finish-On constraints must be considered when $[ES, LS]$ windows are determined. Tables 3.2 and 3.3 give the functions used to calculate the ES and LS values respectively. $db_{j,date,1}$ and $db_{j,date,M}$ denote the duration of the non-overtime mode and duration of maximum overtime mode respectively.

Table 3.2: ES calculation with precedence relations and activity time constraints

	ES
Start On	$date$
Start On or Before	$\max\{0, ES_{FS}, ES_{FF}, ES_{SS}, ES_{SF}\}$
Start On or After	$\max\{\text{date}, ES_{FS}, ES_{FF}, ES_{SS}, ES_{SF}\}$
Finish On	$date - db_{j,date,1}$
Finish On or Before	$\max\{0, ES_{FS}, ES_{FF}, ES_{SS}, ES_{SF}\}$
Finish On or After	$\max\{date - db_{j,date,1}, ES_{FS}, ES_{FF}, ES_{SS}, ES_{SF}\}$

At this stage, all adjustments to the duration matrix, ES and LS calculations have been made and the binary integer programming formulation of the multicalendar, multimode NSWPP with general activity relationships and activity Start/Finish On constraints can be developed. This is done in the next chapter.

Table 3.3: LS calculation with precedence relations and activity time constraints

	$\boldsymbol{LS}$
Start On	$date$
Start On or Before	$\min\{date, LS_{FS}, LS_{FF}, LS_{SS}, LS_{SF}\}$
Start On or After	$\min\{\text{H}, LS_{FS}, LS_{FF}, LS_{SS}, LS_{SF}\}$
Finish On	$date - db_{j,date,M}$
Finish On or Before	$\min\{date - db_{j,date,M}, LS_{FS}, LS_{FF}, LS_{SS}, LS_{SF}\}$
Finish On or After	$\min\{\text{H}, LS_{FS}, LS_{FF}, LS_{SS}, LS_{SF}\}$

Chapter 4

Mathematical model and solution method

This section presents the new BIP to extend the original models developed for the NSWPP by Bertrand (2020) and Prabhu (2021). The objective of the model is to build a schedule that front-loads high-priority and long-duration activities. A heuristic method, the serial schedule generation scheme, is also modified to deal with the new requirements identified by our industrial partner.

4.1 Multi-calendar multi-mode discrete-time priority-duration RCPSP model

According to Bertrand (2020) and Prabhu (2021), the schedulers of navy refit projects desire to have high-priority activities scheduled as soon as possible in the DWP to ensure that all critical jobs are done by the end of the maintenance period. Activities with long-duration should also be scheduled as early as possible to avoid any risk of not being able to complete them due to delays or other unforeseen events. Therefore, the model developed here retains the weighted-objective function introduced by Bertrand (2020) to achieve the goal of front-loading high-priority and long-duration activities.

The $[ES, LS]$ window for single-mode and multi-mode are presented in Tables 4.1 and 4.2. Table 4.1 and 4.2 show that the $[ES, LS]$ windows are the same for Start On, Star On or Before, and Start On or After, but different for the other three. For models with multi-mode, the start on/before/after constraints can still be satisfied by precalculating [ES, LS] because duration is not needed in time window calculations based on these three constraints. On the other hand, time windows for activities with finish on/before/after constraints become wider because of the shorter length of duration in mode M (maximum overtime mode). Therefore, constraints for Finish on, Finish on or before, and Finish on or after are necessary for the BIP model.

Table 4.1: ES and LS values for activity time constraints with single execution mode

Start On	ES	$date$
	LS	$date$
Start On or Before	ES	$\max\{0, ES_{FS}, ES_{FF}, ES_{SS}, ES_{SF}\}$
	LS	$\min\{date, LS_{FS}, LS_{FF}, LS_{SS}, LS_{SF}\}$
Start On or After	ES	$\max\{date, ES_{FS}, ES_{FF}, ES_{SS}, ES_{SF}\}$
	LS	$\min\{H, LS_{FS}, LS_{FF}, LS_{SS}, LS_{SF}\}$
Finish On	ES	$date - db_{j,date}$
	LS	$date - db_{j,date}$
Finish On or Before	ES	$\max\{0, ES_{FS}, ES_{FF}, ES_{SS}, ES_{SF}\}$
	LS	$\min\{ date - db_{j,date}, LS_{FS}, LS_{FF}, LS_{SS}, LS_{SF}\}$
Finish On or After	ES	$\max\{date - db_{j,date}, ES_{FS}, ES_{FF}, ES_{SS}, ES_{SF}\}$
	LS	$\min\{H, LS_{FS}, LS_{FF}, LS_{SS}, LS_{SF}\}$

Table 4.2: ES and LS values for activity time constraints with multiple execution mode

Start On	ES	$date$
	LS	$date$
Start On or Before	ES	$\max\{0, ES_{FS}, ES_{FF}, ES_{SS}, ES_{SF}\}$
	LS	$\min\{date, LS_{FS}, LS_{FF}, LS_{SS}, LS_{SF}\}$
Start On or After	ES	$\max\{date, ES_{FS}, ES_{FF}, ES_{SS}, ES_{SF}\}$
	LS	$\min\{H, LS_{FS}, LS_{FF}, LS_{SS}, LS_{SF}\}$
Finish On	ES	$date - db_{j,date,1}$
	LS	$date - db_{j,date,M}$
Finish On or Before	ES	$\max\{0, ES_{FS}, ES_{FF}, ES_{SS}, ES_{SF}\}$
	LS	$\min\{date - db_{j,date,M}, LS_{FS}, LS_{FF}, LS_{SS}, LS_{SF}\}$
Finish On or After	ES	$\max\{date - db_{j,date,1}, ES_{FS}, ES_{FF}, ES_{SS}, ES_{SF}\}$
	LS	$\min\{H, LS_{FS}, LS_{FF}, LS_{SS}, LS_{SF}\}$

Indices:

i index of activities

j index of activities

k index of resource types

t index of time periods (e.g., days)

m index of execution modes

Parameters for Sets:

n integer, number of non-dummy activities

H integer, time horizon considered for scheduling

K integer, number of resource types

M integer, number of execution modes

Sets:

$\mathcal{J}$ Set of activities, $\mathcal{J} = \{0, \cdots, n\}$ with index i or j

$\mathcal{H}$ Set of time periods, $\mathcal{H} = \{0, \cdots, h\}$ with index t

$\mathcal{K}$ Set of types of resources, $\mathcal{K} = \{1, \cdots, K\}$ with index k

$\mathcal{M}$ Set of execution modes, $\mathcal{M} = \{0, \cdots, M\}$ with index m

$\mathcal{W}_j$ Set of time periods between the early start and late start of activity j, $\mathcal{W}_j = \{[ES_1, \cdots, LS_1], \cdots, [ES_j, \cdots, LS_j], \cdots\}$

$\mathcal{P}_{FS}$ Set of immediate Finish to Start activity pairs (i, j)

$\mathcal{P}_{FF}$ Set of immediate Finish to Finish activity pairs (i, j)

$\mathcal{P}_{SS}$ Set of immediate Start to Start activity pairs (i, j)

$\mathcal{P}_{SF}$ Set of immediate Start to Finish activity pairs (i, j)

$\mathcal{CD}_o$ Set of activity and time pairs (j, s) specifying the time s when activity j must Finish On

$\mathcal{CD}_b$ Set of activity and time pairs (j, s) specifying the time s when activity j must Finish On or Before

$\mathcal{CD}_a$ Set of activity and time pairs (j, s) specifying the time s when activity j must Finish On or After

Other Parameters:

d_{itm} (d_{jtm})	integer, duration of activity i (j) when started at time period t under mode m
d_{im} (d_{jm})	integer, normal (baseline) duration of activity i (j) under mode m
r_{jkt}	integer, activity j demand for resource type k in period t
R_{tk}	integer, capacity of resource k at time period t
ES_j	integer, earliest start time of activity j
LS_j	integer, latest start time of activity j
p_j	integer, priority of activity j
θ	exponent used to ponderate the priority of activities, $\theta \geq 0$
ε_1	very small duration value assigned to activities with no duration (i.e., milestones), $\varepsilon_1 > 0$
ε_2	very small weight used to penalize late scheduling of activities, $\varepsilon_2 > 0$
α	parameter used to give tie-breaking benefits to longer duration work (α=1.1) or making each work duration unit equal (α=0), $\alpha \geq 0$
β_{M1}	total alternative shift limit
β_{M2}	alternative mode reduction limit

Decision Variables:

$$x_{jtm} = \begin{cases} 1, & \text{if activity } j \text{ starts at time } t \text{ in mode } m \\ 0, & \text{otherwise.} \end{cases}$$

The objective function and constraints of the proposed novel multicalendar multimode BIP for the NSWPP are described next.

The objective function is slightly different from the one proposed by Bertrand (2020). Its aim is still to front-load high-priority long-duration activities, but the baseline normal activity duration (d_{jm}) is used instead of d_{jtm} here to avoid artificially prioritizing activities that see their durations increased due to holidays or days-off.

$$\text{Maximize} \sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{W}_j} \sum_{m\in\mathcal{M}} \frac{x_{jtm}}{p_j^{\theta}} (\varepsilon_1 + d_{jm})^{\alpha} (1 - \varepsilon_2 t) \tag{4.1}$$

Subject to:

$$\sum_{t\in\mathcal{W}_j} \sum_{m\in\mathcal{M}} x_{jtm} \leq 1 \qquad \forall j \in J \tag{4.2}$$

Each activity must be completed between its $[ES, LS]$ time window, no more than once. Recall that for the NSWPP, it is not required to complete all activities before the ship can sail.

$$\sum_{j\in\mathcal{J}} \sum_{b=max\{t-db_{jtm}+1, ES_j\}}^{min\{LS_j, t\}} \sum_{m\in\mathcal{M}} r_{jkt} x_{jbm} \leq R_{tk} \qquad \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{4.3}$$

At each discrete time t, the total usage of resource type k by all scheduled activities must not exceed the resource capacity.

$$\sum_{t\in\mathcal{W}_i} \sum_{m\in\mathcal{M}} (t + d_{itm}) x_{itm} \leq \sum_{t\in\mathcal{W}_j} \sum_{m\in\mathcal{M}} t x_{jtm} \qquad \forall j \in \mathcal{J}, \forall (i,j) \in \mathcal{P}_{FS} \tag{4.4}$$

In a Finish-to-Start relationship, the successor j can start if and only if its predecessor i is completed.

$$\sum_{t\in\mathcal{W}_i} \sum_{m\in\mathcal{M}} (t + d_{itm}) x_{itm} \leq \sum_{t\in\mathcal{W}_j} \sum_{m\in\mathcal{M}} (t + d_{jtm}) x_{jtm} \qquad \forall j \in \mathcal{J}, \forall (i,j) \in \mathcal{P}_{FF} \tag{4.5}$$

In a Finish-to-Finish relationship, the successor j can finish if its predecessor i is completed.

$$\sum_{t\in\mathcal{W}_i} \sum_{m\in\mathcal{M}} t x_{itm} \leq \sum_{t\in\mathcal{W}_j} \sum_{m\in\mathcal{M}} t x_{jtm} \qquad \forall j \in \mathcal{J}, \forall (i,j) \in \mathcal{P}_{SS} \tag{4.6}$$

In a Start-to-Start relationship, the successor j can start if its predecessor has started.

$$\sum_{t\in\mathcal{W}_i}\sum_{m\in\mathcal{M}} t x_{itm} \leq \sum_{t\in\mathcal{W}_j}\sum_{m\in\mathcal{M}} (t + d_{jtm} - 1)x_{jtm} \qquad \forall j \in \mathcal{J}, \forall (i,j) \in \mathcal{P}_{SF} \tag{4.7}$$

In a Start-to-Finish relationship, the successor j can finish iff its predecessor i has started. The (-1) term appears here because an activity starts at the beginning of a period but finishes at the end of a period. So for an activity to be completed by the beginning of a period, it must have been completed at the end of the previous period.

$$\sum_{t\in\mathcal{W}_j}\sum_{m\in\mathcal{M}} (t + d_{jtm} - 1)x_{jtm} = s \qquad \forall (j,s) \in \mathcal{CD}_o \tag{4.8}$$

This ensures that activity j finishes exactly on time period s, when j has a Finish-On requirement.

$$\sum_{t\in\mathcal{W}_j}\sum_{m\in\mathcal{M}} (t + d_{jtm} - 1)x_{jtm} \leq s \qquad \forall (j,s) \in \mathcal{CD}_b \tag{4.9}$$

This ensures that activity j finishes on or before time period s, when j has a Finish-On or Before requirement.

$$\sum_{t\in\mathcal{W}_j}\sum_{m\in\mathcal{M}} (t + d_{jtm} - 1)x_{jtm} \geq s \qquad \forall (j,s) \in \mathcal{CD}_a \tag{4.10}$$

This ensures that activity j finishes on or after time period s, when j has a Finish-On or After requirement.

$$\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{W}_j}\sum_{\substack{m\in\mathcal{M}\\ m\neq 1}} (m-1)x_{jtm} \leq \beta_{M1} \tag{4.11}$$

The total number of overtime shifts cannot exceed the limit set up by the scheduler.

$$\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{W}_j}\sum_{\substack{m\in\mathcal{M}\\ m\neq 1}} x_{jtm} \leq \beta_{M2} \tag{4.12}$$

The total number of alternative execution modes cannot exceed the limit set up by the scheduler.

Putting the model together gives the following BIP.

$$\text{Max} \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{W}_j}\sum_{m\in\mathcal{M}} \frac{x_{jtm}}{p_j^{\theta}}(\varepsilon_1 + d_{jm})^{\alpha}\,(1-\varepsilon_2 t)$$

s.t.:

$$\sum_{t\in\mathcal{W}_j}\sum_{m\in\mathcal{M}} x_{jtm} \le 1 \qquad \forall j \in J$$

$$\sum_{j\in\mathcal{J}}\sum_{b=max\{t-db_{jtm}+1,ES_j\}}^{min\{LS_j,t\}}\sum_{m\in\mathcal{M}} r_{jkt}x_{jbm} \le R_{tk} \qquad \forall k\in\mathcal{K}, \forall t\in\mathcal{H}$$

$$\sum_{t\in\mathcal{W}_i}\sum_{m\in\mathcal{M}}(t+d_{itm})x_{itm} \le \sum_{t\in\mathcal{W}_j}\sum_{m\in\mathcal{M}} t x_{jtm} \qquad \forall (i,j)\in\mathcal{P}_{FS}$$

$$\sum_{t\in\mathcal{W}_i}\sum_{m\in\mathcal{M}}(t+d_{itm})x_{itm} \le \sum_{t\in\mathcal{W}_j}\sum_{m\in\mathcal{M}}(t+d_{jtm})x_{jtm} \qquad \forall (i,j)\in\mathcal{P}_{FF}$$

$$\sum_{t\in\mathcal{W}_i}\sum_{m\in\mathcal{M}} t x_{itm} \le \sum_{t\in\mathcal{W}_j}\sum_{m\in\mathcal{M}} t x_{jtm} \qquad \forall (i,j)\in\mathcal{P}_{SS}$$

$$\sum_{t\in\mathcal{W}_i}\sum_{m\in\mathcal{M}} t x_{itm} \le \sum_{t\in\mathcal{W}_j}\sum_{m\in\mathcal{M}}(t+d_{jtm}-1)x_{jtm} \qquad \forall (i,j)\in\mathcal{P}_{SF}$$

$$\sum_{t\in\mathcal{W}_j}\sum_{m\in\mathcal{M}}(t+d_{jtm}-1)x_{jtm} = s \qquad \forall (j,s)\in\mathcal{CD}_o$$

$$\sum_{t\in\mathcal{W}_j}\sum_{m\in\mathcal{M}}(t+d_{jtm}-1)x_{jtm} \le s \qquad \forall (j,s)\in\mathcal{CD}_b$$

$$\sum_{t\in\mathcal{W}_j}\sum_{m\in\mathcal{M}}(t+d_{jtm}-1)x_{jtm} \ge s \qquad \forall (j,s)\in\mathcal{CD}_a$$

$$\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{W}_j}\sum_{\substack{m\in\mathcal{M}\\ m\neq 1}}(m-1)x_{jtm} \le \beta_{M1}$$

$$\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{W}_j}\sum_{\substack{m\in\mathcal{M}\\ m\neq 1}} x_{jtm} \le \beta_{M2}$$

$$x_{jtm}\in\{0,1\} \qquad \forall j\in\mathcal{J}, t\in\mathcal{W}_j$$

The above obtained BIP will be modelled in Python and solved using the commercial solver Gurobi for the datasets provided by our industrial partner. Given that the current problem is a an extension of the original NSWPP, it is anticipated that it will be more difficult to solve. Therefore, a heuristic solution method is developed to quickly solve the BIP and present the scheduler with a good initial feasible solution.

4.2 Heuristic Method: Adapted Listing Serial SGS

According to Bertrand (2020), a quick, decent schedule is preferable to the schedulers in the initial planning than an optimal schedule that takes a long time to obtain. However, given that the NSWPP is NP-hard, the computation time grows exponentially with the problem size. Due to uncertainties and deviations in duration modes and labour availability, schedulers may need to adjust some parameters and re-run the model a few times before they can get a practical schedule. Therefore, a heuristic method that can generate good schedules quickly may be a better option for large-size problems. Furthermore, some experiments have shown that Gurobi can produce optimal solutions in a shorter time with a feasible initial solution (warm-start). Therefore, a heuristic method that can accommodate the new characteristics of the NSWPP would be useful. In what follows, the adapted serial SGS heuristic is presented in two parts. First the priority rule, then the actual schedule generation scheme.

4.2.1 Duration Matrix

New forward pass and backward pass duration matrixes are developed to deal with multi-calendar activities in the maximization BIP model. A large negative value, -Q, is used prevent the selection of activities starting on days when works are not allowed due to calendar requirements. However, this large negative value may cause problems in Adapted Listing Serial SGS where step by step calculation of start date and end date is needed. Here is an example to illustrate the problem due to -Q; both Activity 1 and Activity 2 have 5 days duration with only one execution mode, and follow a 5-day calendar. Activity 1 is the FS predecessor of Activity 2. Assuming Activity 1 starts on day 1 which is Monday, the end date of Activity 1 equals to $1 + d_{1,1} - 1 = 1 + 5 - 1 = 5$, which is day 5 on Friday in this example. The end date of Activity 2 equals to $1 + d_{2,6} - 1 = 1 + (-Q) - 1 = -Q$. Problem comes when calculating the end date of Activity 2. Therefore, the forward duration matrix is modified for Adapted Listing Serial SGS. Assuming activities can start/end on any day, instead of -Q, extended durations are filled in the day-offs in matrices for the calculation convenience.

4.2.2 Priority Rule

The schedule generation scheme needs an ordered list of activities to operate. Thus, the first step is to build a priority rule to order the project activities. This priority rule is built with the goal of producing a feasible schedule which can also front-load high-priority and long-duration activities. The result of applying this priority rule is a list of activities that guides the choice of the next activity to be scheduled using the serial SGS. An illustrative example of the priority rule is given below for the project network depicted in Figure 4.1 and Table 4.3.

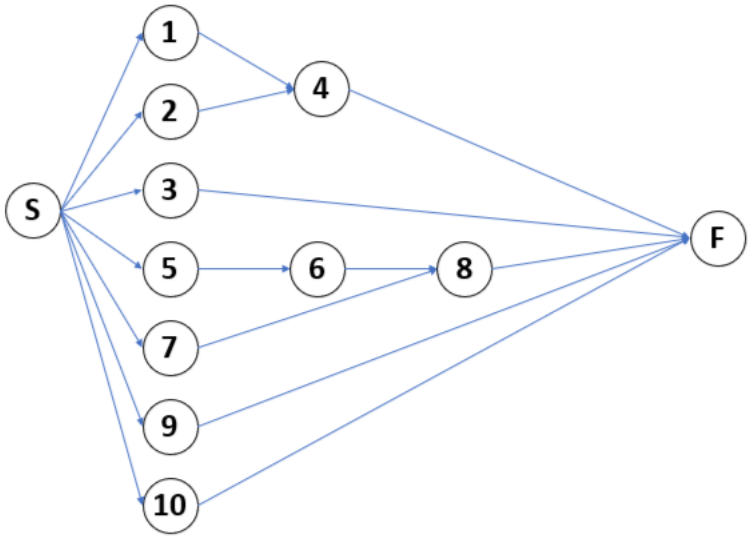

Figure 4.1: Project Network for the Priority Rule Example

For the remainder of this thesis, Start-On, Start-On or Before, Start-On or After, Finish-On, Finish-On or Before, and Finish-On or After are referred to as time-enforced constraints.

The first step of the procedure is to find all activities with time-enforced constraints and sort them by their required dates in ascending order (i.e., earliest to latest). For the illustrative example depicted in Figure 4.1 and Table 4.3, activities 4 and 8 are selected because they have Start On constraints. Then, they are sorted as [4, 8] because activity 4 must start on day 2, which is earlier than the Start On time (day 10) of activity 8.

Table 4.3: Data for the Priority Rule Example

Activity	Priority	duration	Constraint
1	1	1	
2	1	2	
3	1	1	
4	1	1	Start On day 2
5	1	1	
6	1	1	
7	1	1	
8	1	1	Start On day 10
9	2	1	
10	1	3	

The second step is to place all predecessors of each activity with time-enforced constraints before it according to their precedence relationships, priority and duration. For our example, we have activities 2 and 1 that are the predecessors of activity 4. Activities 2 and 1 have the same priority but activity 2 is longer than activity 1, thus 2 will be placed before 1 yielding the following sequence: 2, 1, 4.

Activities 5, 6, and 7 are predecessors of activity 8 and they have the same duration and priority. Given that 5 is a predecessor of 6, two sequences are possible: 7, 5, 6, 8 or 5, 6, 7, 8. Such a tie is arbitrarily broken and sequence 5, 6, 7, 8 is chosen. As a result, the output of the second step is the combined list of both chosen sequences: [2, 1, **4**, 5, 6, 7, **8**]. These two steps put the activities with time-enforced constraint and their predecessors at the top of the priority list. The reason for this is to ensure that the resources are not consumed by activities with more freedom to be scheduled, such as activities 3, 9, and 10 in the example.

The third and final step is to slot in the rest of the activities and their predecessors as sorted by ascending priority, and descending duration. Activities 10 and 3 are priority-1 activities hence activity 9 which is of priority 2 comes after them. Activity 10 has a longer duration than activity 3. Hence, the final list of activities for the example is [2, 1, 4, 5, 6, 7, 8, 10, 3, 9]. Following this order to schedule activities using the serial SGS guarantees that activities can be scheduled without violating the precedence relationships and time-enforced constraints while maintaining the goal of

front-loading high-priority, and long-duration activities.

4.2.3 Schedule Generation Scheme

In this subsection, we discuss how the classical Serial SGS is modified to handle multiple calendars, multiple execution modes, four types of precedence relationships, and time-enforced constraints. The pseudo-code of the new serial SGS is given below.

Parameter:

c_{jt} binary, 1 if activity can be executed at time t according to the calendar that activity j belongs to (i.e., $d_{jt} \geq 0$), otherwise 0 (i.e., $d_{jt} = -Q$)

Algorithm 1: Compute the start time S_j of activity j as sorted by the Priority Rule presented above

Input data: n, d_{jtm}, db_{jtm}, M, R_{kt}, r_{jkt}, $\mathcal{P}_{FS}$, $\mathcal{P}_{FF}$, $\mathcal{P}_{SS}$, $\mathcal{P}_{SF}$
Initialize: $j = 1$, $m_1 = 1$
while $j \leq n$ **do**
  Initialize $S_j = 1$
  if $\mathcal{P}_{FS} \neq \emptyset$ **then**
    Initialize: $i = 1$
    Find the cardinality P_{FS} of set $\mathcal{P}_{FS}$: $P_{FS} = |\mathcal{P}_{FS}|$
    $S_{FS} = 1$
    while $i \leq P_{FS}$ **do**
      if $S_{FS} \leq S_i + d_{i,S_i,m_i}$ **then**
        $S_{FS} = Si + d_{i,S_i,m_i}$
      end if
      $i = i + 1$
    end while
  end if
  if $\mathcal{P}_{FF} \neq \emptyset$ **then**
    Initialize: $i = 1$
    Find the cardinality P_{FF} of set $\mathcal{P}_{FF}$: $P_{FF} = |\mathcal{P}_{FF}|$
    $S_{FF} = 1$
    while $i \leq P_{FF}$ **do**

21: **if** $S_{FF} \leq S_i + d_{i,S_i,m_i} - db_{j,S_i+d_{i,S_i,m_i},m_i}$ **then**
22: $S_{FF} = S_i + d_{i,S_i,m_i} - db_{j,S_i+d_{i,S_i,m_i},m_i}$
23: **end if**
24: $i = i + 1$
25: **end while**
26: **end if**
27: **if** $\mathcal{P}_{SS} \neq \emptyset$ **then**
28: Initialize: $i = 1$
29: Find the cardinality P_{SS} of set $\mathcal{P}_{SS}$: $P_{SS} = |\mathcal{P}_{SS}|$
30: $S_{SS} = 1$
31: **while** $i \leq P_{SS}$ **do**
32: **if** $S_{SS} \leq S_i$ **then**
33: $S_{SS} = Si$
34: **end if**
35: $i = i + 1$
36: **end while**
37: **end if**
38: **if** $\mathcal{P}_{SF} \neq \emptyset$ **then**
39: Initialize: $i = 1$
40: Find the cardinality P_{SF} of set $\mathcal{P}_{SF}$: $P_{SF} = |\mathcal{P}_{SF}|$
41: $S_{SF} = 1$
42: **while** $i \leq P_{SF}$ **do**
43: **if** $S_{SF} \leq S_i - db_{jS_im_j} + 1$ **then**
44: $S_{SF} = S_i - db_{jS_im_j} + 1$
45: **end if**
46: $i = i + 1$
47: **end while**
48: **end if**
49: $S_j = \max\{S_j, S_{FS}, S_{FF}, S_{SS}, S_{SF}\}$
50: **while** $c_{jS_j} = 0$ **do**
51: $S_j = S_j + 1$
52: **end while**

53: Initialize: g = 0
54: **while** $g \leq d_{jS_jm_j}$ **do**
55: **if** $R_{kS_j+g} < r_{jkS_j+g}$ **then**
56: $S_j = S_j + g + 1$
57: $g = 0$
58: **else**
59: $g = g + 1$
60: **end if**
61: **end while**
62: **if** $S_j >$ Start-On $Date_j$ OR $S_j + D_{jS_jm_j} >$ Finish-On $Date_j$ OR $S_j >$ Start-On-or-Before $Date_j$ OR $S_j + D_{jS_jm_j} >$ Finish-On-or-Before $Date_j$ **then**
63: $j = j - 1$
64: **while** $m_j \geq M$ **do**
65: $j = j - 1$
66: **end while**
67: $m_j = m_j + 1$
68: **else**
69: Initialize: $g = 0$
70: **while** $g \leq D_{jS_jm_j}$ **do**
71: $R_{S_j+g,k} = R_{S_j+g,k} - r_{j,k,S_j+g}$
72: $g = g + 1$
73: **end while**
74: $j = j + 1$
75: $m_j = 1$
76: **end if**
77: **end while**

Although the adapted serial SGS seems complex with many steps, decision making, and formulations, its basic logic is to simply schedule activities as early as possible within resource availability. If time-enforced constraints are not satisfied for an activity, the duration of its predecessors are iteratively reduced until a feasible schedule is obtained. The decision condition and statement $S_j = S_j + 1$ if $c_{[j,S_j]} = 0$ is added to ensure that activities are not scheduled to start on a day-off. Without

this condition, if the immediate predecessor ends on a day before a holiday, the successor will be scheduled to start on the holiday.

A small example is used to demonstrate the application of this adapted serial SGS. There are Finish-to-Start (FS) relationships between activities as shown in Figure 4.2. Some other parameters are listed in Table 4.4. All activities are in a 7-days-per-week calendar with no holiday exception to eliminate the duration matrix and make the demonstration easily understood.

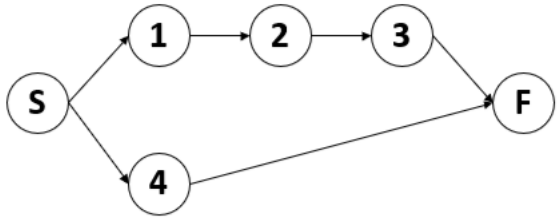

Figure 4.2: Network for the Adapted Serial SGS Example

Table 4.4: Parameters for the Adapted Serial SGS Example

	durations				
Activity j	d_1	d_2	d_3	r_j1	Start On
1	2	1	1	1	
2	4	3	2	0	
3	1	1	1	1	6
4	2	1	1	1	
R_1				1	

According to the priority rule described earlier, the order of scheduling for this example is [1, 2, 3, 4]. Figure 4.3 shows how a feasible schedule is built step by step using the adapted serial SGS.

In subfigure a), Activity 1 of normal duration 2 is scheduled and the available resource is completely used on day 1 and day 2.

Next activity 2 is to be scheduled (sub-figure b). Activity 3 cannot start until activity 2 finishes, but activity 3 has Start On constraint which cannot be satisfied if activity

2 is executed in non-overtime mode. Therefore, following the adapted serial SGS procedure, execution mode 2 is selected for activity 2 and its duration is reduced to 3 (instead of 4) so that activity 3 can start on day 6 (sub-figure c). Activity 2 does not require any resource for its execution, thus there is one unit of resource left on days 3 to 5.

Activity 4 has no predecessor but it cannot be scheduled on day 1 and day 2 because there is no resource available to perform it. The earliest that activity 4 can be performed is on day 3 as depicted on sub-figure d).

This small example shows how a Start-On constraint is dealt with. All other precedence relationships and other time-enforced constraints are similarly accounted for in the proposed adapted serial SGS with slight differences in the decision conditions and functions according to Table 4.2.

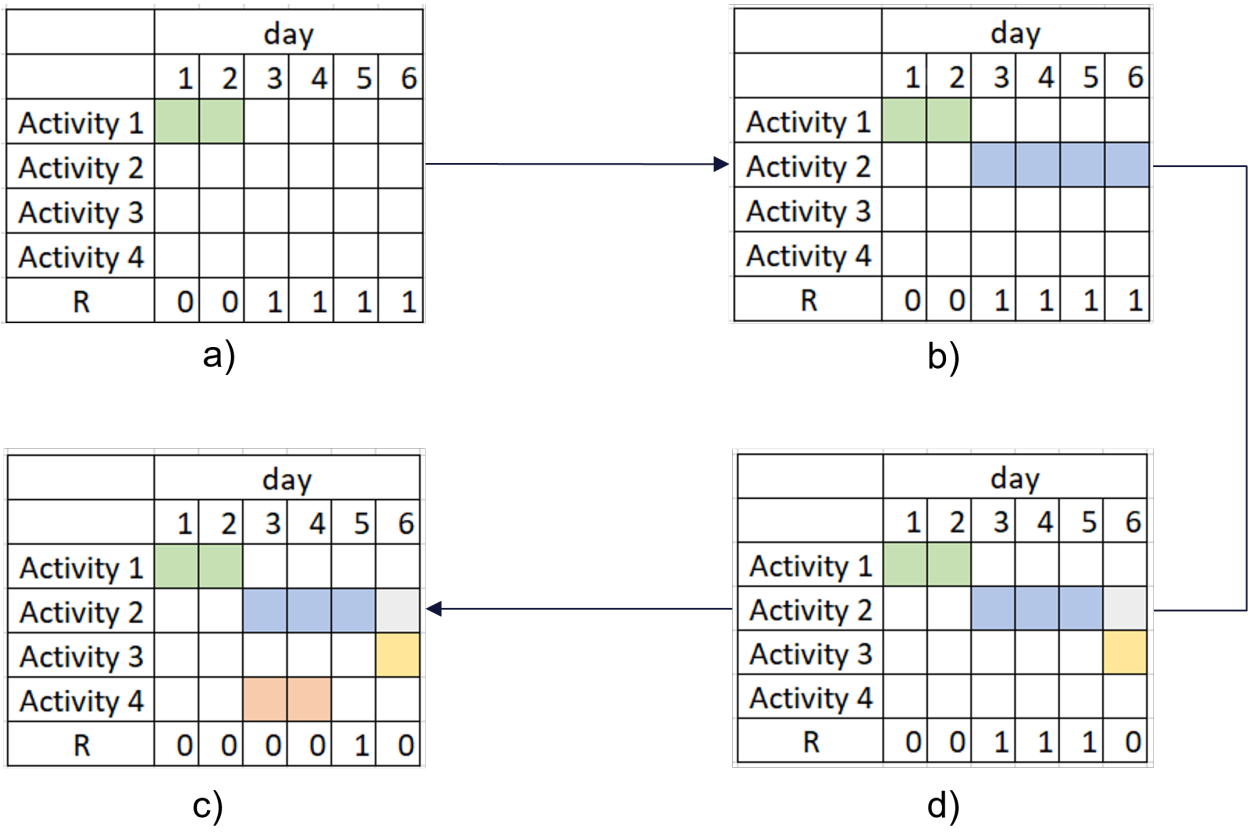

Figure 4.3: Step by Step Schedule Building based on the Adapted Serial SGS

Chapter 5

Experiments and analysis of results

This chapter presents the results of the numerical experiments run with the datasets provided by the industrial partner and built from actual maintenance operations from a Canadian shipyard. The experiments consist of solving the BIP formulation proposed with Gurobi and the adapted serial SGS. In what follows, the datasets and instances used in the experiments are presented first. Then, the average priority-1 duration-weighted centroid (DWC), a criterion used to assess the quality of a front-loaded schedule is introduced. Finally, the results of the computations are presented and discussed.

5.1 Datasets and Instances

Two real-world refit project scheduling instances form Thales are provided, OTC with 136 activities and CRUISE with 500 activities. Referring to the discussion of problem complexity indicated by network indicators in the literature review section, the CRUISE-500 instance is less complex than the OTC-136 instance, despite their problem sizes. Network indicator values of these two instances are shown in Table 5.1. Although experiments are also done on the CRUISE-500 instance, analysis on results are mainly done on the OTC-136 instance and its generated instances because of its higher complexity. Four problem instances of different sizes are created from the original OTC instance with 136 activities provided by Thales. Thus, the size of instances range from 136 to 680 activities with increments of 136. The RF, RC, PR, CNC, and DR network indicators for these instances are the same given that they are duplicates. The OS indicator decreases with size as shown in Table 5.2. These instances are used to see how problem size and OS affect the solution times and the schedule quality of the solution obtained by Gurobi and the modified serial SGS.

Table 5.1: Network Indicators of CRUISE-500 and OTC-136 instances

Value of Indicators #	OTC-136	CRUISE-500
OS	0.0227	0.0716
RF	0.2312	0.0589
RC	0.4642	0.2346
PR	20.0000	8.0000
CNC	0.7279	1.0557
DR	0.3379	0.2476

Table 5.2: Characteristics of the instances

Instance #	Number of activities	OS
1	136	0.0227
2	272	0.0113
3	408	0.0075
4	544	0.0056
5	680	0.0045

5.2 Impact of Solver Tolerance on Gap from Optimum

To find a good trade-off between computational time and solution gap from optimum, an experiment is carried out on the largest instance with 680 activities. The tolerance values of the solver are varied and the CPU time and solution obtained are recorded. Experiments starts from 0.01%, the default tolerance of Gurobi Optimizer, to 10%. The results obtained are depicted in Table 5.3.

Table 5.3: Computational Time as a function of Solver Tolerance

Tolerance (%)	CPU_t (min)	Z value	Z Gap(%)	DWC	DWC Gap(%)
0.01	548.50	1365.25	–	1271.06	–
0.05	90.82	1364.78	0.03	1268.07	-0.24
0.10	66.15	1364.35	0.07	1282.29	0.88
0.50	29.49	1362.41	0.21	1277.76	0.52
1.00	17.55	1357.26	0.58	1290.33	1.52
5.00	9.72	1316.67	3.56	1322.97	4.08
10.00	9.53	1316.67	3.56	1322.97	4.08

Gurobi Optimizer takes about 548.5 minutes (more than 9 hours) to find a solution when the tolerance is set at 0.01%, while it takes only 17.55 minutes when the tolerance is set at 1%. The gap is 0.58% from the solution obtained with a tolerance of 0.01%, which seems to be a good trade-off given the significant decrease in CPU time. Therefore, a tolerance of 1% is chosen for the remainder of the experiments.

The solution obtained with tolerance set at 0.05% has a slightly smaller (i.e., better) DWC than the solution with tolerance set at 0.01%, although the Z value of solution obtained with tolerance set at 0.05% is smaller (i.e., worse). The objective function is calculated using the baseline normal activity durations (d_{jm}) instead of d_{jtm} to avoid artificially prioritizing activities that see their durations increased due to holidays or day-offs. However, the DWC is calculated using d_{jtm}. The schedule with tolerance set at 0.05% happens to have fewer durations increased due to holidays, thus resulting in a smaller DWC.

5.3 Duration-Weighted Centroid as Schedule Quality Criterion

Bertrand (2020) proposed the average priority-1 duration-weighted centroid (DWC) as a criterion to assess the quality of a front-loaded schedule (i.e., placing the priority-1 and long-duration activities early in the schedule). This centroid is calculated by Equation 5.1, where $\mathcal{J}_1$ represents the set of priority-1 activities, S_j and F_j represent the start time and finish time of activity j respectively, and d_j represents the duration of activity j.

$$DWC_{old} = \sum_{j \in \mathcal{J}_1} \frac{S_j + F_j}{2\,|\mathcal{J}_1|}\, d_j \tag{5.1}$$

Equation 5.1 is intended to give more weight/importance to schedules that have long-duration activities starting early. However, we found that this intention is not achieved without giving more weight to the activities duration. Therefore, the DWC equation is slightly modified as shown in Equation 5.2. The reason of making this change is illustrated in Figure 5.1 where three schedules of three priority-1 activities are shown. A smaller centroid is desired as it would correspond to a better front-loaded schedule.

$$DWC = \sum_{j \in \mathcal{J}_1} \frac{S_j + F_j}{2\,|\mathcal{J}_1|}\, d_j^{1.1} \tag{5.2}$$

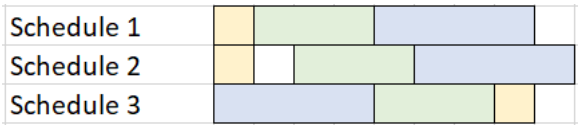

Figure 5.1: Illustrating the Need of Duration-Weight

Table 5.4: Comparison of Schedule-Quality Measures

Schedule	AvC	DWC_{old}	DWC
1	5.25	18	20.46
2	6.25	21.5	24.42
3	8.25	18	19.79

A simple visual inspection shows that Schedule 3 is superior to Schedule 1 which is superior to schedule 2 in terms of front-loading long-duration jobs. Table 5.4 shows the computed values of the Average Centroid (AvC), DWC_{old} and DWC for each of the three schedules. From these results, it is clear that DWC is a better criterion in ranking the schedule quality. The AvC yields a ranking that is the opposite of what it should be. The DWC_{old} is not capable of discriminating schedule 1 from schedule 3.

5.4 Experiments and Results

For the numerical experiments, we used the same objective function parameters as in Bertrand (2020). Hence the values of θ, α, ε_1, and ε_2 are set to 5, 1.1, 0.001, and 0.001 respectively. Thus, the objective function becomes:

$$\text{Maximize } Z = \sum_{m \in M} \sum_{j \in J} \sum_{ES_j}^{LS_j} \frac{100 x_{jtm}}{p_j^5} (0.001 + d_{jm})^{1.1} (1 - 0.001t)$$

The BIP and modified serial SGS are programmed in Python 3.8.6. The BIP is solved using Gurobi 9.5.1. Information of the computer used for experiments is given in Appendix A, and the Python code of the BIP is given in Appendix B.

5.4.1 Objective function comparison between BIP and Serial SGS

This section compares the objective function values Z and computation times CPU_t obtained by the BIP and Serial-SGS as displayed in Table 5.5. The solutions obtained by Gurobi for the BIP are all optimal. For each instance the gap between the SGS values and the optimal solution is given. Although the objective function value Z is syntactic, it is calculated for solutions obtained by Serial SGS for comparison to observe the performance of Serial SGS. The objective function values gap of Serial SGS to BIP varies from 0.39% to 4.38%, and generally increase with problem size.

As anticipated, the computation time increases as the problem size increases. Given that in the initial scheduling phase, schedulers may want to run the model several times and make resource adjustments, solution times becomes a very important factor. Table 5.6 depicts the computation times for both methods and separates the

Table 5.5: Comparison between BIP and Adapted Serial SGS

Inst.	Size	BIP		SGS		SGS Gap to BIP (%)
		Z^*	CPU_t (min)	Z	CPU_t (min)	
1	136	310.11	0.05	308.89	0.50	0.39
2	272	599.74	0.31	594.70	1.80	0.84
3	408	878.38	2.94	854.08	3.94	2.77
4	544	1128.08	7.23	1090.90	6.76	3.30
5	680	1362.41	17.54	1302.71	10.81	4.38

BIP problem generation time (.lp file creation) and the BIP actual solve time (i.e., Gurobi solver time). Figure 5.2 shows the CPU_t growth trends for the serial-SGS, BIP problem generation and BIP actual solve times.

Table 5.6: Detailed computational times (in minutes)

Inst.	Size	Serial-SGS CPU_t (min)	BIP	
			Solve time (min)	.lp file creation (min)
1	136	0.50	0.05	4.90
2	272	1.80	0.31	13.73
3	408	3.94	2.94	32.43
4	544	6.76	7.23	59.16
5	680	10.81	17.54	113.66

The above results show that the computational time of the BIP exact method grows exponentially as expected, since NSWPP is a NP-hard problem. Furthermore, the time spent on generating the .lp file for Gurobi has a similar trend as the actual Gurobi solve time. The total time needed to reach an optimal solution by the exact method makes it not very practical for large-size instances such as instances 4 and 5. On the other hand, the computational time of adapted serial-SGS grows linearly with the problem size. Besides, as mentioned earlier, the CRUISE-500 instance is less complex than OTC instances, because it has a lot more precedence relations and activity time constraints.

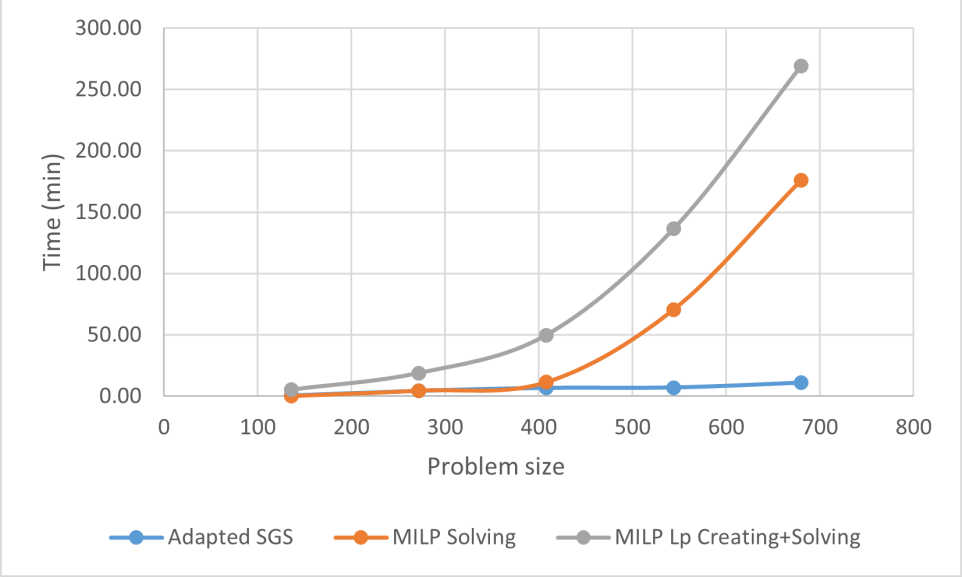

Figure 5.2: Detailed Computational Time Growth

As mentioned by Bertrand (2020), a feasible solution produced in reasonable time is more favourable than a perfectly optimal solution as it is rarely implemented in practice. To test the ability of the exact method to give a good solution within reasonable time, an experiment is carried out on instances without and with warm-start with the initial solution generated by the serial-SGS. The results obtained are in Table 5.7.

Table 5.7: Computational Time With and Without Warm-Start

		Serial SGS Solve time	With Warm-start Solve time	Without Warm-start	
				Solve time	Difference to With Warm-start
Inst.		(min)	(min)	(min)	(%)
1	136	0.50	0.05	0.12	156.16
2	272	1.80	0.31	0.98	222.28
3	408	3.94	2.94	5.73	95.28
4	544	6.76	7.23	13.00	79.83
5	680	10.81	17.55	40.65	131.62

Without warm-start, the Gurobi Optimizer takes between 79.8% to 222.3% extra

time to find a optimal solution. These results indicate that for large-size problems that cannot achieve optimal solutions in reasonable time, initial solutions obtained by the adapted serial-SGS can at least ensure a feasible solution when the time limit is reached. Using this heuristic solution to warm-start Gurobi also seems to result in very good approximate solutions.

5.4.2 DWC comparison between BIP and Serial-SGS Solutions

As discussed in Section 5.3, the Duration-Weighted Centroid (DWC) is a good schedule-quality measure. Table 5.8 summarizes the DWC values calculated for the Serial-SGS and the BIP.

Table 5.8: DWC Comparison between the BIP and Serial-SGS

		BIP	Serial-SGS	
Inst.	Size	$DWC*$	DWC	Gap (%)
1	136	290.38	320.75	10.46
2	272	534.29	567.53	6.22
3	408	762.65	854.44	12.04
4	544	1015.65	1093.96	7.71
5	680	1277.76	1340.61	4.92

Schedules generated by the exact method (BIP) have smaller DWC values than the Serial–SGS for instances of all sizes, meaning that the exact method gives better solutions for the goal of front-loading high-priority and long-duration activities.
The performance of the Serial-SGS is good given that for the instances considered it has a maximum gap of about 12% for a very short compute time.

5.4.3 Makespan Comparison between the BIP and Serial-SGS

The goal of the NSWPP is not to minimize makespan C_{max}. However, in this section we will compare the resulting calculated makespan values of the solutions obtained using the Serial-SGS and the BIP. The obtained results are presented in Table 5.9. Given that front-loading activities is part of the optimization goal, it is not surprising to see that the exact method outperforms the Serial-SGS here.

Table 5.9: C_{max} Comparison between the BIP and Serial-SGS

		C_{max}	
Inst.	Size	Serial-SGS	BIP
1	136	181	180
2	272	249	229
3	408	382	340
4	544	511	455
5	680	630	574

5.4.4 Results for the CRUISE-500 Dataset

As discussed above in the Datasets and Instances section, there are many differences of characteristics between the CRUISE-500 and OTC datasets. Therefore, direct comparisons cannot be made between their results. The results for the CRUISE-500 instance are shown in Tables 5.10 and 5.11. In this case, the Serial SGS method achieves very good results as it finds a solution 2.37% away from optimal in only 2 minutes when the BIP needs 31 minutes to get to the optimal solution. On the other hand, the BIP is capable of better front-loading the activities as its DWC is about 30% lower (lower is better for DWC).

Table 5.10: Results of CRUISE-500

		BIP		SGS		SGS Gap
Inst.	Size	Z^*	CPU_t (min)	Z	CPU_t (min)	to BIP (%)
6	CRUISE-500	512.08	30.97	499.96	2.03	2.37

Table 5.11: DWC Comparison between the BIP and Serial-SGS for CRUISE-500

		BIP	Serial-SGS	
Inst.	Size	$DWC*$	DWC	Gap (%)
6	680	187.24	245.03	30.86

Chapter 6

Conclusions

The NSWPP is a complex variant of the RCPSP with specific characteristics: limited length of maintenance period, industry-specific network structure, priority levels, multiple activity and resource calendars, general precedence relationships, multiple execution modes, and time-enforced activity constraints. These characteristics lead to the special goal of solving this problem to front-load high-priority and long-duration jobs within the same priority level as well as satisfying all constraints.

The multi-calendar, multi-mode, discrete-time priority-duration RCPSP model presented in this paper successfully accommodates all the characteristics listed above, and generates optimal schedules in terms of front-loading high-priority and long-duration jobs. Because the NSWPP is an NP-hard problem, the computational time to solve the developed BIP model grows exponentially with problem instance size. Numerical experiments showed that the proposed model can achieve optimal solutions with Gurobi in a reasonable amount of time for small-size instances (under 400 activities). However, for large-size instances the solver takes very long to find the optimal solution.

A quickly generated and approximate solution could be more useful than an optimal solution obtained after a long wait in real-world applications where schedulers must go through several iterations. Therefore, a modified serial-SGS heuristic method is proposed to quickly produce good-quality schedules. Experiments show that the solution obtained from the adapted serial-SGS can also be used to warm-start the optimal solver to produce the optimal solution faster. Additionally, the values of the ES, H, and LS parameters obtained as by-products of the serial-SGS, help to reduce the computational time of the BIP model by deceasing the number of binary decision variables in the model. In practice, the Navy scheduler can use the serial

SGS heuristic for a quick assessment of the schedule. If, the scheduler has more time then the BIP model can be solved using the SGS solution to initialize the optimization process which would decrease computation time.

Although the serial-SGS heuristic gives feasible solutions, the gap from optimum shows that there is room for improvement. The priority rule used in the serial-SGS ranks activities with time-enforced constraints on top of others even if their priority level is low. This ensures the feasibility of the obtained solutions. However, it may schedule some of these activities unnecessarily too early resulting in solutions that are far from optimum. An extension would be to develop a matheuristic method to solve small BIP models to select where to place "free-floating" high-priority activities before time-enforced constraints.

Future research may also be conducted on the rescheduling of multi-calendar NSWPPs in order help schedule recovery or reducing deviation from the initial plan. In real-world applications, over-estimated duration and delays due to various reasons are inevitable. A tool that can quickly reschedule the rest of the activities in order to catch-up to the original schedule may be very useful.

Prabhu (2021) developed a decomposition matheuristic to solve large-size instances of the original NSWPP. Similarly, decomposition matheuristics should be developed to quickly solve the multicalendar multimode NSWPP. Further studies may be done in designing guidelines for the selection of solution methods depending on the complexity of problem instances.

References

Artigues, C., Michelon, P., and Reusser, S. (2003). Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, 149(2):249–267.

Bartusch, M., Möhring, R. H., and Radermacher, F. J. (1988). Scheduling project networks with resource constraints and time windows. *Annals of operations Research*, 16(1):199–240.

Bertrand, E. (2020). Optimization of the naval surface ship resource-constrained project scheduling problem. Master's thesis, Dalhousie University.

Bianco, L. and Caramia, M. (2012). An exact algorithm to minimize the makespan in project scheduling with scarce resources and generalized precedence relations. *European Journal of Operational Research*, 219(1):73–85.

Blazewicz, J., Lenstra, J. K., and Rinnooy, A. K. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete applied mathematics*, 5(1):11–24.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., and Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European journal of operational research*, 112(1):3–41.

Christofides, N., Alvarez-Valdés, R., and Tamarit, J. M. (1987). Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*, 29(3):262–273.

Croteau, E. (2018). An on-event based model for resource constrained scheduling of aircraft heavy maintenance tasks. Master's thesis, Dalhousie University.

de Azevedo, G. H. I., Pessoa, A. A., and Subramanian, A. (2021). A satisfiability and workload-based exact method for the resource constrained project scheduling problem with generalized precedence constraints. *European Journal of Operational Research*, 289(3):809–824.

De Nijs, F. (2013). Project scheduling: The impact of instance structure on heuristic performance. Master's thesis, Technical University Delft.

De Reyck, B. and Herroelen, W. (1996). On the use of the complexity index as a measure of complexity in activity networks. *European Journal of Operational Research*, 91(2):347–366.

Deckro, R. F. and Hebert, J. E. (1989). Resource constrained project crashing. *Omega*, 17(1):69–79.

Demeulemeester, E. L. and Herroelen, W. S. (1997). New benchmark results for the resource-constrained project scheduling problem. *Management science*, 43(11):1485–1492.

Geiger, M. J. (2013). Iterated variable neighborhood search for the resource constrained multi-mode multi-project scheduling problem. *arXiv preprint arXiv:1310.0602*.

Guldemond, T., Hurink, J. L., Paulus, J. J., and Schutten, J. M. (2008). Time-constrained project scheduling. *Journal of Scheduling*, 11(2):137–148.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, 45(7):733–750.

Hartmann, S. and Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of operational research*, 207(1):1–14.

Hartmann, S. and Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 127(2):394–407.

Herroelen, W. and De Reyck, B. (1999). Phase transitions in project scheduling. *Journal of the Operational Research Society*, 50(2):148–156.

Kelley, J. E. (1963). The critical-path method: resource planning and scheduling. *Industrial scheduling*.

Kelley, J. J. E. and Walker, M. R. (1959). Critical-path planning and scheduling. In *Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference*, pages 160–173.

Kolisch, R. and Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European journal of operational research*, 174(1):23–37.

Kolisch, R., Sprecher, A., and Drexl, A. (1995). Characterization and generation of a general class of resource-constrained project scheduling problems. *Management science*, 41(10):1693–1703.

Koné, O., Artigues, C., Lopez, P., and Mongeau, M. (2011). Event-based milp models for resource-constrained project scheduling problems. *Computers & Operations Research*, 38(1):3–13.

Kong, F. and Dou, D. (2021). Resource-constrained project scheduling problem under multiple time constraints. *Journal of Construction Engineering and Management*, 147(2):04020170.

Kramer, S. and Jenkins, J. (2006). Understanding the basics of cpm calculations: What is scheduling software really telling you. In *PMI® Global Congress*.

Laborie, P. (2005). Complete mcs-based search: Application to resource constrained project scheduling. In *Ijcai*, pages 181–186.

Larson, E. W. and Gray, C. F. (2021). *Project management: The managerial process*. McGraw-Hill Irwin New York.

Li, H. and Dong, X. (2018). Multi-mode resource leveling in projects with mode-dependent generalized precedence relations. *Expert Systems with Applications*, 97:193–204.

Malcolm, D. G., Roseboom, J. H., Clark, C. E., and Fazar, W. (1959). Application of a technique for research and development program evaluation. *Operations research*, 7(5):646–669.

Patterson, J. H. (1976). Project scheduling: The effects of problem structure on heuristic performance. *Naval Research Logistics Quarterly*, 23(1):95–123.

Pellerin, R., Perrier, N., and Berthaut, F. (2020). A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 280(2):395–416.

Prabhu, S. M. (2021). Matheuristic optimization approaches for large scale resource constrained scheduling problems in refit operations. Master's thesis, Dalhousie University.

Pritsker, A. A. B., Waiters, L. J., and Wolfe, P. M. (1969). Multiproject scheduling with limited resources: A zero-one programming approach. *Management science*, 16(1):93–108.

Schwindt, C., Zimmermann, J., et al. (2015). *Handbook on project management and scheduling vol. 1*. Springer.

Sprecher, A. (1996). Solving the rcpsp efficiently at modest memory requirements. Technical report, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel.

Talbot, F. B. (1982). Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management science*, 28(10):1197–1210.

Türkakın, O. H., Arditi, D., and Manisalı, E. (2021). Comparison of heuristic priority rules in the solution of the resource-constrained project scheduling problem. *Sustainability*, 13(17):9956.

Vanhoucke, M. et al. (2012). *Project management with dynamic scheduling*. Springer.

Zapata, J. C., Hodge, B. M., and Reklaitis, G. V. (2008). The multimode resource constrained multiproject scheduling problem: Alternative formulations. *AIChE Journal*, 54(8):2101–2119.

Zhang, L. and Sun, R. (2011). An improvement of resource-constrained multi-project scheduling model based on priority-rule based heuristics. In *ICSSSM11*, pages 1–5. IEEE.

Appendix A

Information of computer used for experiments

Processor:10^{th} Generation Intel® Core™ i7-10510U Processor (1.80 GHz, up to 4.90 GHz with Turbo Boost, 4 Cores, 8 Threads, 8 MB Cache)
Operating system:Windows 10 Pro
Memory:16 GB DDR4 2667MHz
Storage:1 TB PCIe SSD

Appendix B

Python Code of the BIP model

```
# Python code goes here
# Create the problem called "problem1"
prob = LpProblem("problem1", LpMaximize)

# Create the variable x[j,t,m]
var =
    [[[pulp.LpVariable("x[%s,%s,%s" % (j + 1, t + ES.iloc[j], m + 1), None,
     None, LpBinary) for m in range(M.shape[0])]
      for t in
      range(LS.iloc[j] - ES.iloc[j] + 1)]
     for j in range(n)]

# Set the initial value
for j in range(len(var)):
    for t in range(len(var[j])):
        for m in range(len(var[j][t])):
            var[j][t][m].setInitialValue(0)

for i in range(asgs):
    var[i][SGS['ES'].iat[i] - ES.iloc[i]][0].setInitialValue(1)

# Create the objective function

prob += pulp.lpSum(
    [[[var[j][t][m] * (1 - (0.001 * (t + ES.iloc[j]))) * (100 / (PR.iloc[j] ** 5))
    *((0.001 + int(Dobj.iloc[j,m])) ** 1.1)
```

```
    for m in range(M.shape[0])]

    for t in range(LS.iloc[j] - ES.iloc[j] + 1)] for j in range(n)])

# Create constraint 1: all activities should be completed no more
 than 1 time between ES and LS
for j in range(n):
    prob += (pulp.lpSum([[var[j][t][m] for m in range(M.shape[0])]
    for t in range(LS.iloc[j] - ES.iloc[j] + 1)]) == 1)

# Create constraint 2: finish to start
for i in range(C7.shape[0]):
    # print(P["element id"].iat[i])
    j1 = int(C7["pred"].iat[i]) - 1
    j2 = int(C7["succ"].iat[i]) - 1
    lag = int(C7["lag"].iat[i])
    prob += (pulp.lpSum([[(t + ES.iat[j2]) * var[j2][t][m]
    for t in range(LS.iat[j2] - ES.iat[j2] + 1)]
    for m in range(M.shape[0])]) >=
             pulp.lpSum([[var[j1][t][m] * ((t + ES.iat[j1]) +
             D.iat[j1, t + ES.iloc[j1]][m] + lag) for t in
                          range(LS.iat[j1] - ES.iat[j1] + 1)]
                         for m in range(M.shape[0])]))

# Create constraint 3: finish to finish
for i in range(C8.shape[0]):
    # print(P["element id"].iat[i])
    j1 = int(C8["pred"].iat[i]) - 1
    j2 = int(C8["succ"].iat[i]) - 1
    lag = int(C8["lag"].iat[i])
```

```
    prob += (pulp.lpSum(
        [[(t + ES.iat[j2] + D.iat[j2, t + ES.iloc[j2]][m]) * var[j2][t][m]
          for t in range(LS.iat[j2] - ES.iat[j2] + 1)]
          for m in range(M.shape[0])]) >=
             pulp.lpSum([[var[j1][t][m] * ((t + ES.iat[j1]) +
              D.iat[j1, t + ES.iloc[j1]][m] + lag - 1) \
                          for t in range(LS.iat[j1] - ES.iat[j1] + 1)]
                         for m in range(M.shape[0])]))

# Create constraint 4: Start to start
for i in range(C9.shape[0]):
    # print(P["element id"].iat[i])
    j1 = int(C9["pred"].iat[i]) - 1
    j2 = int(C9["succ"].iat[i]) - 1
    lag = int(C9["lag"].iat[i])
    prob += (pulp.lpSum(
        [[(t + ES.iat[j2]) * var[j2][t][m]
        for t in range(LS.iat[j2] - ES.iat[j2] + 1)]
        for m in range(M.shape[0])]) >=
             pulp.lpSum([[var[j1][t][m] * (t + ES.iat[j1] + lag)
             for t in range(LS.iat[j1] - ES.iat[j1] + 1)]
                         for m in range(M.shape[0])]))

# Create constraint 5: Start to finish
for i in range(C10.shape[0]):
    # print(P["element id"].iat[i])
    j1 = int(C10["pred"].iat[i]) - 1
    j2 = int(C10["succ"].iat[i]) - 1
    lag = int(C10["lag"].iat[i])
    prob += (pulp.lpSum(
        [[(t + ES.iat[j2] + D.iat[j2, t +
         ES.iloc[j2]][m] - 1) * var[j2][t][m]
```

```
            for t in range(LS.iat[j2] - ES.iat[j2] + 1)]
            for m in range(M.shape[0])]) >=
               pulp.lpSum([[var[j1][t][m] * (t + ES.iat[j1] + lag)
               for t in range(LS.iat[j1] - ES.iat[j1] + 1)]
                           for m in range(M.shape[0])]))

# Create constraint 6: finish on
for i in range(C2.shape[0]):
    # print(P["element id"].iat[i])
    j1 = int(C2.iloc[i, 0]) - 1
    j2 = int(C2.iloc[i, 1])
    prob += (pulp.lpSum(
        [[var[j1][t][m] * (t + ES.iat[j1] + D.iat[j1, t +
         ES.iloc[j1]][m] - 1) for m in range(M.shape[0])] \
         for t in range(LS.iloc[j1] - ES.iloc[j1] + 1)]) == j2)

# Create constraint 7: finish on or after
for i in range(C4.shape[0]):
    # print(P["element id"].iat[i])
    j1 = int(C4.iloc[i, 0]) - 1
    j2 = int(C4.iloc[i, 1])
    prob += (pulp.lpSum([[var[j1][t][m] * (t + ES.iat[j1]
    + D.iat[j1, t + ES.iloc[j1]][m] - 1) \
                         for m in range(M.shape[0])]
                         for t in range(LS.iloc[j1] -
                          ES.iloc[j1] + 1)]) >= j2)

# Create constraint 8: finish on or before
for i in range(C6.shape[0]):
    j1 = int(C6.iloc[i, 0]) - 1
    j2 = int(C6.iloc[i, 1])
    prob += (pulp.lpSum([[var[j1][t][m] * (t + ES.iat[j1]
```

```
     + D.iat[j1, t + ES.iloc[j1]][m] - 1)
          for m in range(M.shape[0])]
          for t in range(LS.iloc[j1] - ES.iloc[j1] + 1)]) <= j2)

# Create constraint 9: resource availability
for k in range(int(R)):
    for s in range(int(H + 1)):
        temp = []
        for j in range(n):
            for m in range(M.shape[0]):
                for t in range(int(max(s - DB.iloc[j, s][m] + 1, ES[j])
                  - ES.iloc[j]), int(min(LS[j], s) - ES.iloc[j] + 1)):
                    form1 = RD.iloc[j, k] * var[j][t][m] *
                    AC.iloc[int(SGS["calendar"].iat[j]), s]
                    temp.append(form1)
        prob += (pulp.lpSum(temp) <= AV.iat[k, s])

# Create the .lp file
prob.writeLP("problem1.lp")

prob.solve(GUROBI_CMD(mip=MIP,options=[
("Heuristics", 0.2),
  ("Symmetry", 2)],
   warmStart=True,
   gapRel=Tol))
```