MATHEURISTIC OPTIMIZATION APPROACHES FOR LARGE SCALE RESOURCE CONSTRAINED SCHEDULING PROBLEMS IN REFIT OPERATIONS

by

Sanjay Manjunath Prabhu

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

at

Dalhousie University
Halifax, Nova Scotia
July 2020

Dedication

I dedicate this thesis to Devi Nagayakshi.

Table of Contents

List of Tables

List of Figures

Abstract

This thesis will discuss the development of matheuristic methods for scheduling the refit activities in the naval surface ship work period problem (NSWPP). The NSWPP is a variant of the classical resource-constrained project scheduling problem (RCPSP) with specific network structure and constraints. The NSWPP is a non-deterministic polynomial-time hard problem whose solution becomes computationally demanding as the problem size increases. Therefore, a matheuristic method called multi-step optimization (MSO) is developed in this paper, which combines heuristics and mixed-integer linear programming (MILP) to schedule the NSWPP activities efficiently. MSO uses heuristic priority rules to decompose the activities from the main project into subgroups. The subgroups are then iteratively optimized using a time-indexed discrete-time MILP formulation. The results from the comparative tests conducted show that MSO is computationally efficient and produces near-optimal solutions. Also, a comparative study of the solutions generated by MSO proves that the proposed method outperforms state-of-the-art heuristics.

List of Abbreviations Used

BnB	Branch-and-Bound
CI	Confidence Interval
CPM	Critical Path Method
Dal-R	Dal - Randomizer
DDT	Disaggregated Discrete-Time
DR	Disjunction Ratio
DT	Discrete-Time
DWC	Duration Weighted Centroid
ERP	Enterprise Resource Planning
ES	Early Start Time
FCT	Flow-based Continuous Time
FMF	Fleet Maintenance Facility
HDR	High Disjunction Ratio
HPR	High Process Range
ISSP	In-Service Support Provider
LDR	Low Disjunction Ratio
LP	Linear Program
LPR	Low Process Range
LS	Late Start Time
MILP	Mixed Integer Linear Program
MP	Mathematical Programming

MSO	Multi-Step Optimization
NC	Network Complexity
NSWPP	Naval Surface Ship Work Period Problem
OOE	On-off Event
OS	Order Strength
PD	Priority-Duration
PERT	Project Evaluation and Review Technique
PM	Preventive Maintenance
PR	Process Range
PSPLIB	Project Scheduling Problem Library
RC	Resource Constrainedness
RCN	Royal Canadian Navy
RCPSP	Resource-Constrained Project Scheduling Problem
RF	Resource Factor
RFc	Repair Facility
RP	Refit Period
RS	Resource Strength
SEE	Start/End Event Based
SG	Subgroup
SSGS	Serial Schedule Generation Scheme
WP	Work Package

Glossary

Dummy: Placeholder activity, that is used to represent the start and finish milestones. This activity has zero resource demand and duration.

Makespan: Actual time required to finish all activities in the project.

Robust Schedule: A schedule with all priority-1 activities scheduled as early as possible

Schedule quality: The proximity of the generated schedule in comparison to the optimal schedule.

Time windows: Time frame within which a project gets executed

Instance: A project data frame with information on activity duration, resource demand, resource capacity, priority numbers, etc

High Priority Activity: Priority-1 activity

Low Priority Activity: Priority-3 activity

Acknowledgments

I am incredibly thankful to my parents, brother, and friends for their love, support, encouragement, and motivation, which helped me pursue this opportunity.

I am appreciative of the Industrial Engineering Department, Dalhousie University, NS for supporting my academic endeavours.

I am thankful to Shahin Hameed K. for introducing me to the Mitacs Accelerate internship Program.

I am deeply grateful to my supervisors Dr. Claver Diallo, Dr. Alireza Ghasemi, for giving me the opportunity to work with them on a challenging project. A special thanks to Dr. Pemberton Cyrus and Dr. M. Ali Ülkü for offering guidance to improve the quality of the thesis. The process would have been very arduous without their patience, time, guidance, and support.

I extend my admiration and gratitude to LCdr. Eric Bertrand, for supporting me and always being ready to answer my questions and provide assistance.

A special thanks to Thales Group, Canada, and Mitacs for funding this internship and research project under the Mitacs Accelerate program.

Chapter 1: Introduction

This thesis is a part of the "Refit Optimizer" research project undertaken by Thales Group, Canada, to develop a decision support tool to optimize the process of scheduling refit operations. Thales Group is a French multinational company reputed for providing service to the aerospace, defence, transportation, and security industries. It has been awarded a multi-million-dollar contract to conduct maintenance operations on arctic/offshore patrol ships (Harry DeWolfe Class) and joint support ships (Protecteur Class). Three universities namely, Dalhousie University, Polytechnique Montreal and Université de Laval are working in a consortium to help Thales Group, Canada in its endeavour along with the help of other organizations such as Fleet Maintenance Facility (FMF) on both coasts, and Seaspan from Victoria, BC.

The naval surface ship work period problem (NSWPP) is a highly complex variant of the classical resource-constrained project scheduling problem (RCPSP) with hundreds of activities to be scheduled using a limited resource pool. The NSWPPs are subjected to a variety of contingencies which makes their planning, scheduling and execution very challenging. In addition, due to the high variability of the processing duration of the activities, a large number of planned activities go unattended by the end of the project [1].

The repair facilities (RFc) working on the NSWPP generally use commercial enterprise resource planning (ERP) software to schedule the activities and to manage the resources [1]. The usefulness of the ERP software is limited due to the disabling of the auto-scheduling option, which is done to prevent the mass-modification of information in the centralized system [2]. Therefore, the schedulers end up planning all the projects manually.

This thesis focuses on developing efficient scheduling techniques using mathematical formulations which can be used alongside RFc's ERP system to schedule the NSWPPs optimally.

1.1 The Resource-Constrained Project Scheduling Problem (RCPSP)

The resource-constrained project scheduling problem (RCPSP) is a combinatorial optimization problem wherein the objective is to minimize the makespan of the project by determining the start time for the project activities while making sure that the precedence constraints and resource constraints are satisfied [3]. The resource constraints can be budget, human resources, materials, or machines [4]. RCPSP belongs to the class of NP-hard problems, which means that as the size of the problem increases, the computational time grows exponentially, thus making it hard to solve the problem in polynomial time using exact solution methods [5]. Even with improvements to exact solution methods, software, and hardware technologies, NP-hard problems are still difficult to optimize for large instances in a reasonable amount of time [6]. Therefore, large RCPSP instances are generally solved using heuristic algorithms because of their speed and flexibility. The downside to this is the compromise with the optimality of the obtained solution. A heuristic algorithm generally gives a solution that is far away from the optimum [7]. The gap between the heuristic solution and the optimal solution becomes significant with an increase in the size of the problem, and this will result in issues such as overestimation of the project makespan, overestimation of the budget, and inefficient resource management that are detrimental to the progression of the project [8]. For the past couple of decades, researchers have been working on efficient hybrid algorithms and solution methods to solve NP-hard problems. One such method known as *matheuristic* designates

(meta)heuristic procedures that are incorporated into mathematical programming models or algorithms that integrate the mathematical programming concepts into the (meta)heuristic framework to optimize complex real-world problems [9].

This thesis contributes to the body of knowledge in the area of RCPSP by proposing a new MILP formulation called the priority-duration (PD) formulation that better captures the operational objective and schedules the project in a way that complies with the practical protocol followed in the naval industry. In addition, a new matheuristic method called multi-step optimization (MSO) is developed that uses a unique three-level heuristic priority rule to decompose the large NSWPP instances into subproblems that are later iteratively optimized using the PD formulation to generate near-optimal solutions. The proposed matheuristic proved to be very efficient in solving large NSWPP instances and also capable of producing superior quality solutions when compared to state-of-the-art heuristic methods like the serial schedule generation scheme (SSGS).

Chapter 2: Problem Description

This chapter is devoted to a detailed description of the naval surface ship work period problem (NSWPP). It includes an overview of project management, a description of the implementation of project management in NSWPP, an outline of the challenges in the NSWPP, the nature of the network in the NSWPP, a description of the sources of uncertainty in naval projects, and a discussion of the issues with the use of commercial scheduling software.

2.1 Project Management Overview

A project can be defined as an undertaking of a unique idea to meticulously create a product or service through efficient management of resources [10]**.** The primary elements of project management, regardless of the type of project (construction, manufacturing, product development, etc.) are the 4Ms, i.e., Manpower, Machine, Materials, and Money. Project management is the process of efficiently managing all these elements in a coordinated manner throughout the project's lifecycle [10].

According to the Project Management Institute (PMI), project management processes can be classified into five phases: 1) Initiating, 2) Planning, 3) Execution, 4) Monitoring and Controlling, 5) Closing [10], [11].

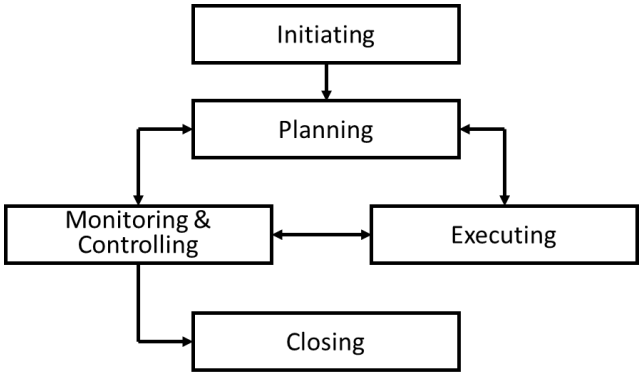


Figure 1. Phases of project management (source: [11])

In this thesis paper, the focus is directed towards the scheduling operation, which is a part of the planning phase of project management. Scheduling is a highly detailed process and is commonly dealt with a proactive approach [11]. According to the Project Management Body of Knowledge (PMBOK), a book on standard terminologies and guidelines for project management, the planning phase is composed of 24 distinct steps such as; determining the planning approach, finalizing the requirements, creating activities, determining the activity duration, creating the work breakdown structure, assigning resources to the activities, and constructing a schedule [11], [12]. The resultant of the planning operations is the generation of the baseline schedule. The baseline schedule is constructed as per the objectives set by the management. The most commonly used objective is to minimize the makespan for the given project [13]. There are also possibilities for the administration to use other objectives depending upon the nature of the project such as the following: 1) Maximizing the net present value, 2) Minimizing the idle time, 3) Balancing resource consumption levels, 4) Minimizing the cost of executing the project, etc. [13]. A detailed overview of different possible objectives can be found in [14], [15], [16].

A proactive baseline schedule is paramount before the actual start of the project because it acts as a point of reference to monitor the performance of the project and it enables the management to vigilantly track metrics such as resource usage, time adherence, and precedence adherence [17]. A delay in the execution of critical tasks can have a snowball effect directed towards the succeeding activities of the project resulting in disruption of the schedule, budget, and other following endeavours.

In the NSWPP, there are a few high-value resources such as specialized technicians, material handling machines, and dry docks that need to be planned as accurately as possible in the baseline schedule [18]. These resources always tend to be critical for the project's performance. Disruptions caused by these resources can affect the performance of the existing and subsequent projects as well [18].

2.2 Naval Surface Ship Work Period Problem (NSWPP)

The NSWPP is the summation of all the work packages (WP) and their associated activities. The term "work period" refers to the length of time during which the naval repair facility performs the maintenance operations on a ship by executing the majority of the WPs. A work package consists of a number of "activities" that represent individual tasks required to complete the WP. The activities can execute in sequence one after the other or execute in parallel where multiple activities can be performed simultaneously.

The NSWPP consists of a large list of prioritized WPs that must be completed using limited resources from a shared resource pool [18]. During NSWPP, many ships can undergo maintenance simultaneously and the ships compete for majority of similar resources, which makes the scheduling operation very challenging.

NSWPP differs from other RCPSP seen in the literature for a number of reasons. Firstly, not all activities in a NSWPP need to be completed [1]. The goal in a NSWPP is to finish all the high priority WPs within the makespan such that even if there are uncompleted low priority WPs, they can be reasonably mitigated so that the project is not compromised [1]. Apart from human and machine resources, NSWPP includes spatial resources while planning to prevent over-crowding a particular space/compartment [1]. Since NSWPP occurs in a relaxed environment with numerous spaces, the WPs are relatively independent of each other. Therefore, the number of precedence constraints is less, which in return makes the scheduling problem challenging because the problem will have a larger solution space and the WPs can fit anywhere within the time window. NSWPP does not comprise only of maintenance operations but also includes equipment upgrades, steel renewal, engineering changes, testing and trials etc. [19]. All these operations are important because the ship must remain relevant with the state-of-the-art technologies throughout its lifetime that spans over many decades [19].

2.3 The Maintenance Lifecycle of a Ship

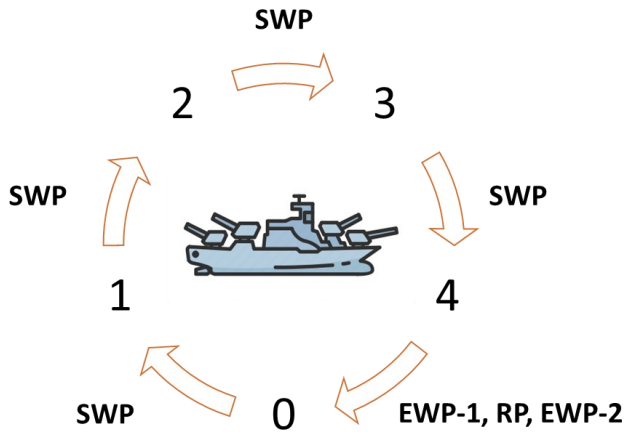


Figure 2. The 5-year maintenance life cycle of a ship

Figure 3 is a representation of the typical maintenance life cycle of a navy ship. A naval ship undergoes refit period (RP)/docking work period once every 5 years during which it undergoes extensive maintenance operations such as welding, painting, deck floor replacement, and equipment upgrades [18], [1]. The RP can last for about 53 weeks, but the time horizon of the work period can be different depending on the type of ship under consideration [1]. In-between the successive RPs, the ship undergoes a brief maintenance period called the short work period (SWP) that generally lasts for three to six weeks, and a ship can have 2-4 SWPs in a year [18], [1]. The maintenance operations during the SWP are performed at the Fleet Maintenance Facility Cape Scott (FMFCS), or the Fleet Maintenance Facility Cape Breton (FMFCB) [18]. Before the start and after the completion of the RP, the ship undergoes an extended work period (EWP) that lasts for several months. During EWP-1 the ship is made as light as possible by removing the fuel, drinking water, ammunition, etc. to rest it on the dry dock platform comfortably [1]. At the end of the RP, the EWP-2 commences during which the ship is taken back into the ocean where it gets re-fuelled, re-filled with drinking water, and reinstalled with the necessary equipment and set to sail [1].

2.4 NSWPP Planning Process

When the ship is on a mission or alongside a port city, equipment can breakdown either under the harsh conditions at sea, or on standby while rotating machinery is laying idle and corrosion progresses. As shown in figure 2, when a breakdown occurs, and the ship staff cannot fix it by themselves, they generate notifications in the primary repair facility's (RFc) electronic maintenance system, for planning and later for conducting corrective maintenance on the faulty equipment. The RFc staff also creates additional notifications

when defects are discovered during the inspection of the ship at the dockyard [1]. Preventive maintenance (PM) notifications are also automatically generated from the RFc's electronic maintenance system. Prior to the start of the refit period, the Royal Canadian Navy (RCN) gives a time horizon to complete as many work packages as possible, for that period, to the repair facility (e.g., FMF, Thales, or SNC Lavalin). The RFc's planning department will then use the notifications to create activities and work packages that include all the information needed to correct the defect or perform the PM. These WPs will be then used by the project leader and scheduler to generate a schedule and propose an estimated number of WPs that can be completed within the given time horizon [1].

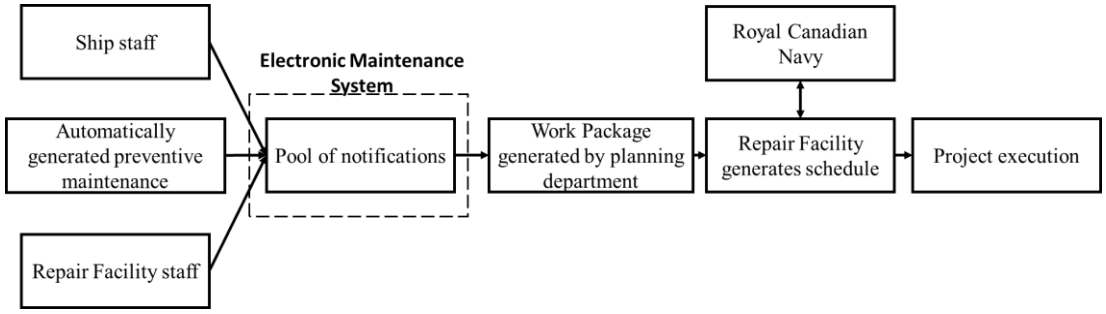


Figure 3. Process of work package generation in NSWPP

In the NSWPP, it is common to see more pending WPs to be completed within the given time horizon and with limited resource availability. Unlike the maintenance projects for aircraft and submarines, the NSWPP does not require all the outstanding maintenance operations to be completed during the work period [1]. From table 1, which represents the data from one of the work periods at FMFCS (Source: [1]), it can be seen that although a WP was first created in 2015, it was scheduled to execute in 2019. To ensure that the essential WPs are finished during the active work period, priorities are assigned to the WPs

[18] [1]. It is acceptable to have low-priority WPs unattended as the high redundancy in naval ships allow defect tolerance; plus, even if a ship loses power from a catastrophic failure, it still floats and lives are not in immediate danger [1]. Also, there can be back and forth negotiations with the RCN throughout the life cycle of a ship for deferring the execution of WPs into the successive work periods, thus ensuring that mission-critical WPs are completed. [1].

WP	Sched. start	Sched. finish	Created on
16058239	07-01-2019	16-01-2019	06-03-2015
16059428	10-09-2018	24-09-2018	08-12-2015
16042178	07-08-2018	23-08-2018	04-01-2016
4521901	05-06-2017	13-06-2017	07-11-2016
4553264	06-02-2019	08-02-2019	02-06-2017

Table 1. Deferment of work package into the successive work period (source: [1])

As discussed earlier in section 2.1, the processes in a project can be divided into five groups. Similarly, the NSWPP can also be divided into these groups. The following section gives an overview of each phase.

2.4.1 Initiation and Planning

The initiation and planning phase begins as early as 16 weeks priors to the start of the work period during which the vital data regarding operational goals, engineering changes, material requirements, activity duration, etc. are collected and reviewed [18]. Once the activity details are available, the experts from The Fleet Engineering Readiness (FER) and Naval Engineering Operations Department (N-37) assign priority numbers to the work

packages and the activities within the work package inherit the same priority number [18], [20].

2.4.2 Scheduling

In this phase, the data available is used to schedule the WPs starting with the high priority ones [18]. The initial draft is shared with the ship's command team for a review, and any potential issues are brought up during the subsequent meetings before finalizing the baseline schedule [18], [1].

2.4.3 Execution, Monitoring, and Control

The resources are expected to conform to the schedule and finish the work within the determined time window [18]. Any resource conflicts arising during the execution of the project are discussed during the weekly meetings [18], [1]. If a planned high priority WP is getting delayed due to a shortage of resources, a lower priority WP gets cancelled or deferred into the next work period to expedite the high priority WP by satisfying its resource demands [18], [1].

2.4.4 Closing

At the end of the project, a meeting is held to discuss the project's performance. Comparisons such as actual hours vs. planned hours, number of WPs completed vs. the number of WPs planned are reviewed [18]. The completed WPs are broken down by priority to review the number of high priority WPs finished [18]. Other information, such as labor hours and the number of WPs cancelled or deferred to the next work period are also presented [18].

The performance of the In-Service Support Provider ISSP (i.e., Thales) will be quantified using a contractor scoring scheme, which compares the number of WPs accepted at the

start of the work period and the number of WPs completed by the end of the work period [1]. Different scores are assigned to WPs based on their priority status [1]. Since the priority-1 WPs are valued highly, the ISSP must ensure that it completes all the high priority WPs to get a good score.

2.5 Challenges in Scheduling Naval Maintenance Operations

Scheduling naval maintenance operations is subject to various constraints and high levels of uncertainty. Apart from precedence, time and resource constraints, which are commonly witnessed during project scheduling, there is a multitude of additional constraints that are to be considered, such as – the spatial constraints, operational constraints and priority constraints. Some of these constraints are unique to the NSWPP and adds to the complexity of generating a proactive schedule. The following section briefly presents the unique constraints that make the scheduling operation challenging.

2.5.1 Spatial Constraint

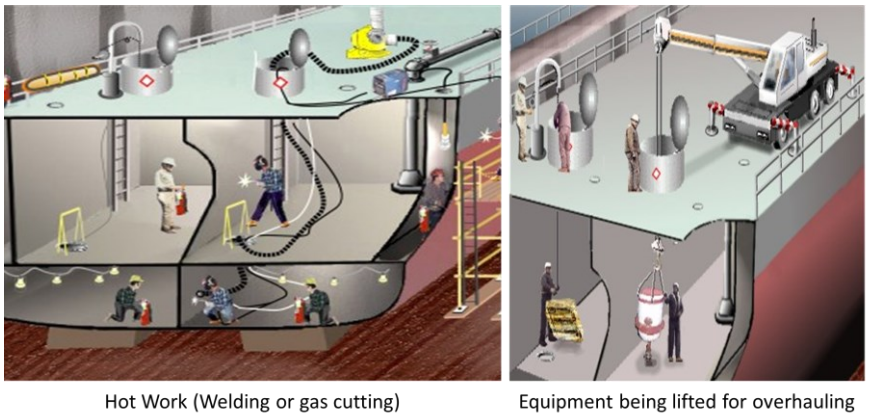


Figure 4. Representation of spatial constraint in a ship (source: [21])

Consider a ship like the Halifax class ship, which has 5 decks and 353 spaces [1]. Though a ship has lots of spaces, there can be situations where work cannot be scheduled even when the space is available. For example, every deck (below the upper deck) has a narrow passageway connecting multiple adjacent spaces. If WPs are scheduled to occur in all the spaces at once, conflicts can occur over the use of the passageway. Furthermore, when the scheduled WP is "hot-work" such as welding or gas-cutting a patch of metal, as shown in Figure 4, no WPs can be scheduled to occur in the adjacent spaces [21]. A Firewatch is stationed in the adjacent space to prevent any mishaps [21]. Another example is when heavy equipment from the lower deck needs to be removed from the ship. The soft patches (floorboard) present on top of the equipment needs to be removed so that it can be lifted using a crane [1], [21].

2.5.2 Operational Constraints: Radiation and Emissions

The ship contains radars and other equipment that use strong electromagnetic waves for operating its combat and communication systems. During calibration and testing of these equipment, no WP can be scheduled on the upper deck of the ship as it can have severe consequences, such as radiation burn on the people working in close quarters with these equipment [1], [22]. In addition, upper deck operations involving jetty cranes are deferred during the testing of the ship's gas/diesel-powered equipment because these equipment emit large volumes of exhaust gases into the atmosphere creating an inhospitable environment for the operator sitting in the cabin on top of the crane to control it [1].

2.5.3 Priority Constraints

The naval ship maintenance operations are priority dominant. The FMFCS uses three priority levels to categorize work based on its importance [18], [1]. They are essential work

(high priority), high opportunity work (medium priority), normal opportunity work (low priority) [18], [1]. The primary objective during any work period is to complete all the high priority WPs first since they are considered mission-critical.

2.6 Topology/Structure of the Project Network

The work packages in NSWPP do not have many precedence relationships with each other, as in the network structure shown in Figure 5 which is very typical of projects in the construction industry, submarine industry, and aircraft industry. Since the ship is very big with multi-levelled decks and lots of spaces, the WPs do not cross-interact and exhibit a network structure, as shown in Figure 6 [1]. Since it is not mandatory to finish all the WPs, dummy start and end nodes are not required. Most project scheduling problems require that all activities be completed while minimizing the makespan of the project. For this purpose, a fictitious dummy end node is used where all the open-ended activities are given as the predecessor to the dummy end node. Then the objective function seeks to minimize the start time of the end node to get an optimal makespan, as shown in equation (1). One of the issues with this approach is that the slack activities tend to get scheduled close as late as possible [19] and if there are open-ended activities that are not connected to the end node, they might, sometimes, get poorly optimized. In NSWPP, the goal is the minimize the start time of all the activities starting with the high priority ones and long duration ones within the same priority, as shown in equation (29). Therefore, having the dummy nodes serves no purpose to the objective function, but it can be included, if needed, to specify important milestones.

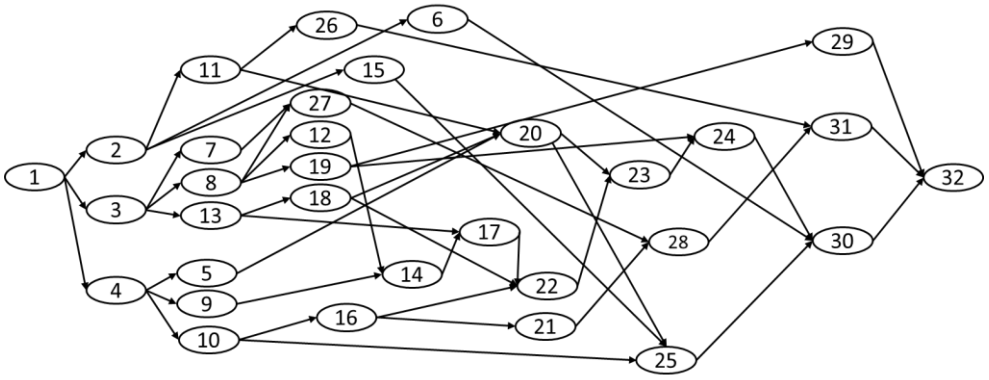


Figure 5. A typical project network structure

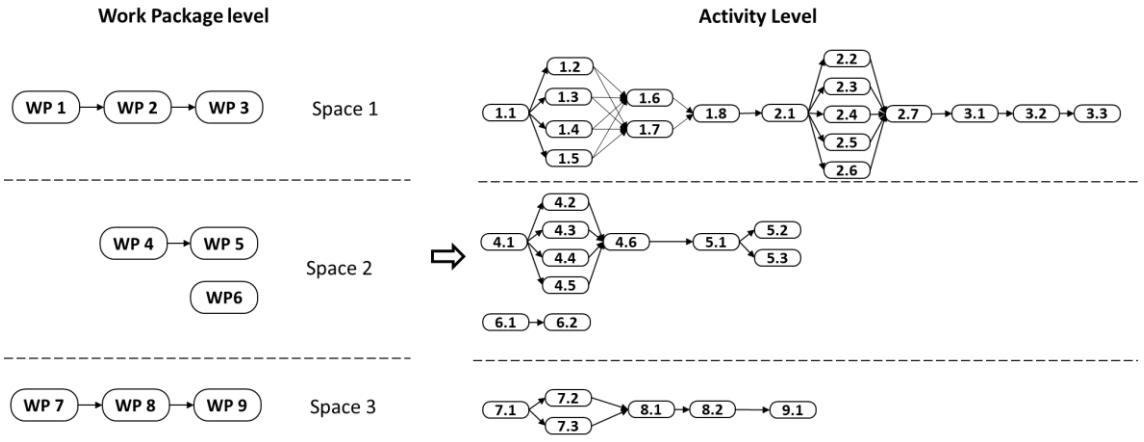


Figure 6. A typical NSWPP network structure

2.7 Uncertainty in NSWPP

The NSWPP is very uncertain and full of contingencies which make the activity duration stochastic in nature. This means the activities can finish earlier than planned, finish later than planned, or may not get executed at all. This uncertainty is evident in the data obtained from the FMFCS for an extended work period (2018-2019) that lasted for four months (Source: [19]). According to the data, 636 WPs were initially planned, out of which only 399 were executed, as shown in Figure 7. Thus, 37.26% of WP never got executed. Of the

399 WPs that got executed, 16.29% of WPs were underestimated (took longer to complete than estimated), 72% of WPs were overestimated (took less time to complete than estimated), and only 11.03% of WPs were completed exactly on time. The major reasons for this uncertainty are scope growth of the planned WPs, resources required to address high priority work on another ship or the submarine, unavailability of the essential parts or equipment, unforeseen engineering changes, inclement weather, procrastination, absenteeism, and other contingencies [18], [19]

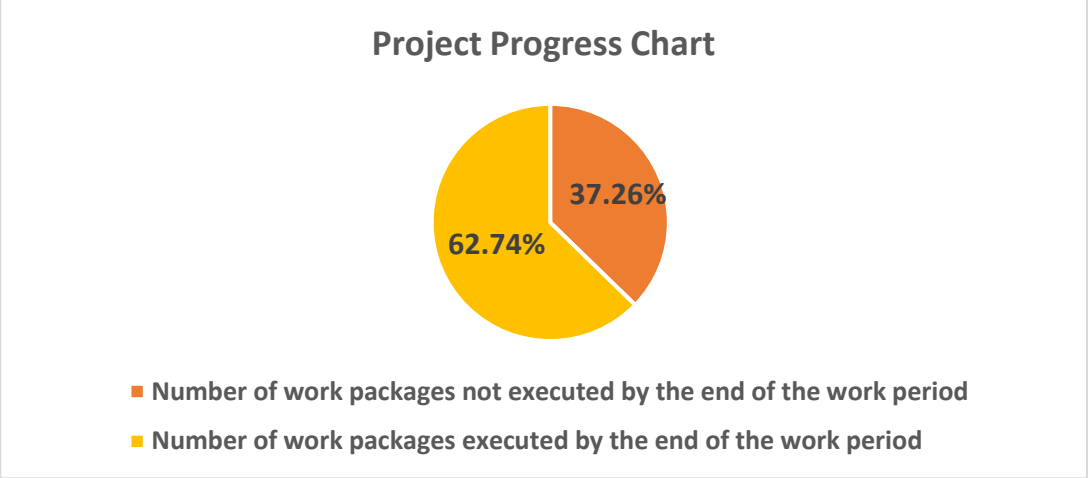


Figure 7. Project progress chart for an extended work period (source: [19])

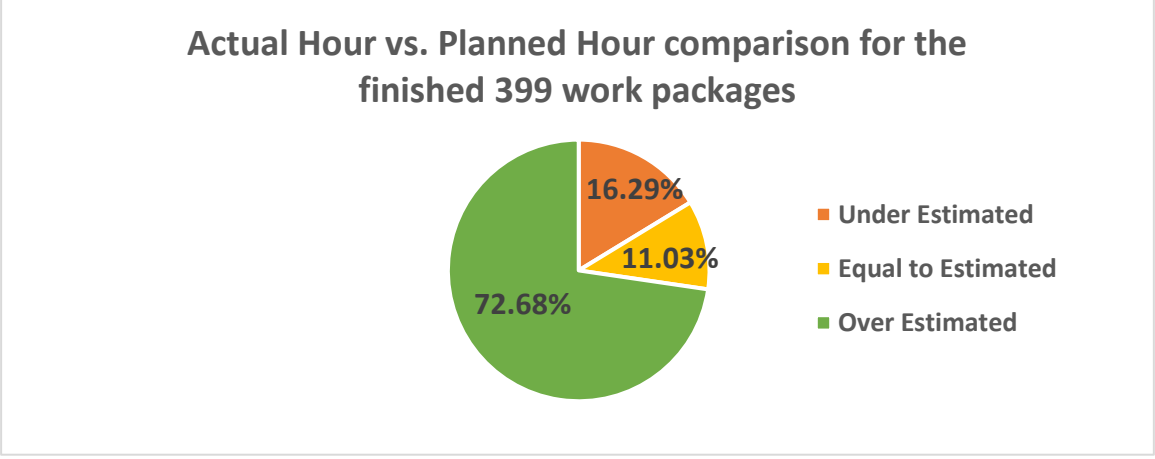


Figure 8. Actual hours vs. Planned hours comparison for an extended work period (source: [19])

Currently, the FMF schedules all the projects manually, which is an arduous and time-consuming process [1]. In order to cope with the changes and uncertainties, the project is frequently rescheduled (usually every week) to make sure that it adheres to the baseline schedule [1], [2]. Though ERP software such as Primavera P6 is used at FMFCB, the auto-scheduling privileges remain locked to prevent accidental modifications to financial and contractual information [1], [2].

2.8 Issues with Commercial Software

Kolisch (1999) conducted a study to analyze the gap in makespan between the optimal solution and that obtained using commercial software such as MS Project and Primavera project planner on 160 instances with the number of activities ranging between 10-30 and 2-4 resource types. The results from the study concluded that the solution deteriorates with an increase in the number of activities. Similar results were also reported by [8], [23]. Table 2 represents the mean and standard deviation of the makespan percentage deviation obtained by the commercial software packages [24].

Commercial software	μ	σ
MS Project	5.35	6.53
Primavera Project Planner	4.39	6.04

Table 2. Mean and standard deviation of the makespan percentage deviation for commercial software (source: [24])

Baumann and Trautmann (2015) evaluated the resource allocation capability of 8 popular commercial scheduling software such as Primavera P6, MS project 2010, MS project 2013, and CS project professional. They conducted experiments on a total of 1560 instances from PSPLIB with the number of activities ranging between 30-120 and 4 resource types. The results concluded that commercial software gives a makespan that is longer when

compared to the optimum. In the worst-case scenario (highly resource-constrained), Primavera P6 could take 113% longer to generate a schedule with mere 60 activities when compared to the branch and bound algorithm [8]. Table 3 shows the mean and maximum of makespan percentage deviation relative to the optimal value given by [8].

	% Mean				% Maximum			
No. of Activities	**30**	**60**	**120**	**All**	**30**	**60**	**120**	**All**
Primavera P6	2.38	3.75	9.9	5.69	18.64	24.24	48.84	48.84
CS Poject Professional	3.32	5.46	14.7	8.35	20.59	23.89	32.89	32.89
MS Project 2010	5.03	6.25	15.77	9.54	32.76	38	47.4	47.4
MS Project 2013	9.6	11.56	25.73	16.4	58.82	75	86.05	86.05

Table 3. The mean and maximum of makespan percentage deviation relative to the optimal value (source: [8])

Despite evidences showing the drawback of commercial software, organizations still insist on using them over the optimization software because of its speed and its capability to store a large amount of information such as multi-project portfolio, resource usage and availability, project's daily progress details and project's expenditure at every level, all within a single software [1]. In addition, since commercial software schedules the project activities very quickly, it enables the planners to perform what-if analyses and compare multiple schedules [1]. Although the commercial software packages are fast, their solutions are poor because of their poor resource allocation capabilities. Therefore, the project activities would be scheduled inefficiently, and the resource utilization would also be poor. This results in a schedule with a large makespan right from the start when the baseline is generated. Further, following a similar approach while monitoring the project's regular performance and for re-scheduling the activities whenever the project starts to deviate from the baseline could extend the project's makespan much more. A practical way to overcome the adversities would be to manually schedule the most important activities first (the project

leaders usually have good knowledge about the most demanding activities) and then use optimization procedure to automatically schedule the remaining activities around the important activities within the given time horizon [1]. This approach helps to achieve a good balance between heuristic and optimization techniques and prevents the case of overestimating the project makespan or making the makespan too tight, which can cause the baseline schedule to breakdown rapidly at the occurrence of uncertainty.

2.9 Research objectives and thesis outline

The research objective for this thesis is to propose effective formulations and solution methodologies that can be used to generate high-quality schedules for large NP-hard priority dominant NSWPP.

This thesis will investigate the following research questions:

1. How and on what basis to select the most suitable MILP formulation to schedule the large priority dominant NSWPP efficiently?
2. Is there a solution technique that will help to combine positive attributes of both pure optimization techniques and heuristic approaches to solve the large NSWPP efficiently in order to overcome the exponential growth in computational time witnessed with the use of pure optimization technique and, at the same time, obtain a solution that is close to the optimum?

The thesis is structured such that a literature review of the popular scheduling techniques, RCPSP development, variants of RCPSP, RCPSP solution methods, RCPSP formulations and network topology of the project networks is performed to study the existing models that could apply to NSWPP. After the literature review, preliminary experimentation with

different MILP formulations is conducted on the real-world inspired instances to analyze their performance. It is followed by a description of the new proposed priority-duration (PD) objective function that helps to schedule the NSWPP activities in a way that complies with the practical protocol followed in the naval industry. To further narrow down on the most suitable formulation, the objective function of the original MILP formulations (minimize makespan) is replaced with the PD objective function while still retaining the constraints. Upon further experimentation with NSWPP instances, the constraints from the best performing formulation are used to develop the new PD formulation.

Following the tests conducted with the PD formulation using pure optimization techniques, a new matheuristic method called multi-step optimization (MSO) is developed in this thesis to overcome the computational complexity of the pure optimization technique used to solve the NSWPP. Later, experiments are conducted with MSO on NSWPP instances to analyze its performance and solution quality in comparison to pure optimization technique and heuristic methodology like the serial schedule generation scheme (SSGS). Furthermore, the instance characteristics of the NSWPP are studied to understand their influence over the computational time when using MSO.

Chapter 3: Literature Review

This chapter is structured to provide an overview of the popular scheduling approaches, the classification of RCPSP and its variants in section 3.1 and section 3.2, respectively. Section 3.3 discusses the different MILP formulations available for RCPSP with major focus placed on time-indexed and event-based MILP formulations. Section 3.4 discusses about the different RCPSP solution methods, section 3.5 discusses about the computational complexity and finally section 3.6 delineates the instance characteristics and how it dictates the complexity of an RCPSP.

3.1 Classical Project Scheduling Techniques

Scheduling is one of the areas that has seen developments throughout centuries [25]. It can be traced back to the time of Sun Tzu, the famous Chinese general of the Zhou Dynasty (722-481 BCE), who wrote about scheduling and strategies from a military perspective [25]. Scheduling became very popular in the 1950s with the invention of the Critical Path Method (CPM) [26] and Project Evaluation and Review Technique (PERT) [27]. The Dupont Corporation first used CPM in 1957 for the construction of its new chemical plant [28], and PERT was used to control and measure the progress of the Polaris Fleet Ballistic Missile project by the U.S. Navy in 1958 [28].

CPM is deterministic in nature and uses predetermined times to calculate the earliest possible project finish times through an iterative calculation of individual activity duration while respecting all the precedence constraints. The solution gives information about critical and non-critical activities. The non-critical activities carry a slack or buffer that gets used in case of delay without affecting the original schedule.

PERT shares similarities with CPM with the major distinguishing factor being the probabilistic nature of PERT, wherein it uses the mean value of three duration estimates, namely the optimistic, most likely, and pessimistic durations to determine the expected activity duration.

Both PERT and CPM produce only precedence and time feasible schedule and not consider the resource feasibility [17], [29]. A sound scheduling algorithm must mandatorily incorporate the resource availability and resource demands along with the precedence constraints to generate precedence, time & resource feasible schedule [30]. CPM and PERT algorithms are not designed to achieve this objective [29], [30]. This technique cannot be used to schedule complex NSWPP that deal with hundreds of activities sharing the available resources and subject to additional constraints. These inherent limitations of CPM and PERT paved the way for RCPSP.

3.2 Classification of RCPSP

The RCPSP is one of the most researched problems because of its practical importance and relevance with all industries like construction, manufacturing, logistics and pharmaceutical. Every industry poses different kinds of constraints and objective functions, with the underlying aim being to determine the optimal start time of project activities, which has led to the development of many variants of the RCPSP. RCPSPs can be classified based on the following attributes:

- Type of constraints
- Type of precedence relationships
- Type of resources
- Type of activity splitting

- Number of execution modes
- Number of objectives
- Types of objectives
- Level of information

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-pre-emptive problem
	Integer pre-emption problem
	Continuous pre-emption problem
Number of execution modes	Single-modal problem
	Multi-modal problem
Number of objectives	Single-criterion problem
	Multi-criteria problem
Type of objective function	Regular function
	Non-regular function
Level of information	Deterministic problem
	Stochastic problem
	Problem under interval uncertainty
	Problem under vagueness
Distribution of information	Centralized problem (symmetric distribution)
	Decentralized problem (asymmetric distribution)

Table 4. Classification of RCPSP and its variants adapted from [31]

Table 4, adapted from Schwindt and Zimmermann (2015), classifies the RCPSP variants according to their attributes. This section gives a brief description of some of the RCPSP variants to explain the nature of the NSWPP project better.

3.2.1 Time Constrained Problem vs. Resource-Constrained Problem

In the Time Constrained problem, the deadline for completing the activities of the project is fixed, and the schedule is generated by varying the resource demand and resource

capacity [32]. In the Resource-Constrained problem, the project activities are to be scheduled with an aim to minimize the makespan of the project while ensuring that the resources are not allocated over their maximum capacity [32]. The NSWPP is a resource-constrained problem since the objective is to finish the project in the least possible time by strictly respecting the resource and precedence constraints.

3.2.2 Renewable Resources vs. Non-Renewable Resources

Renewable resources are the ones for which only the total usage during every time period is constrained [33]. During the execution of an activity, the renewable resource is used for a pre-determined time period [14]. On completion of the activity, the resource is released back into the resource pool, thus making it available for the next activity in line. Machines, tools, and human resources are regarded as renewable resources [34]. The non-renewable resources are the ones for which only the total consumption over the project duration is constrained [33]. Money, energy, and raw materials are some of the best examples for non-renewable resources [17]. An additional resource type, namely the doubly constrained resource, was introduced by Slowinski in 1980. According to him, these resources share the characteristics of both renewable and non-renewable resources [16]. The energy utilization and the project budget are good examples of doubly constrained resources because their total usage during every time period and their total consumption for over the entire project duration can both be constrained [33]. In this thesis, the NSWPP project is modelled with renewable resources. The information on the non-renewable resources like the project budget and personnel salaries are kept confidential during NSWPP and thus, they are not considered [1].

3.2.3 Pre-emptive Model vs. Non-Preemptive Model

In general, RCPSP assumes non pre-emption of activities, which means that once an activity is started, it cannot be interrupted. This assumption is practically sound and conforms with the naval project practices where the activities are completed once they are started. If a project supports pre-emption of activities, it means that activity under execution can be interrupted at any point of time to start working on another activity. For a review on RCPSP with pre-emption, see [35], [36], [37] where they support pre-emption of activity at discrete time units [15].

3.2.4 Single-mode vs. multi-mode

A problem that allows activities to execute in one way within a fixed processing time and fixed resource demands is called a single-mode problem. The concept of multi-mode was introduced by [38] in which the activities were given an option to execute in a different mode with each mode having a unique activity duration and resource demands [15]. Multi-mode project scheduling is useful in job-shop scheduling and determining the cost involved in crashing a project [15].

3.2.5 Single-Criterion Problem vs. Multi-Criteria Problem

If the objective function has only a single criterion to either minimize or maximize the intended goal, it is called a single criterion problem. There are instances where multiple criteria need to be satisfied for a given problem. They can be either similar (every criterion needs to be maximized or minimized) or contradicting to understand the important trade-off between criteria such as makespan, resource level, and project cost as done by [15], [39]. In a problem with the multi-criteria objective function, the impact of each criterion on the overall objective function is controlled by assigning weights [15]. The larger the

weight, the higher its impact on the objective function. In this paper, the NSWPP project is modeled with a multi-criteria objective function that aims at scheduling the high priority activities before the low priority ones. Also, for activities within the same priority, the ones with longer duration are scheduled first.

3.2.6 Deterministic Problem vs. Stochastic Problem

In a deterministic problem, the activity duration is assumed to be fixed. Most of the research conducted on RCPSP considers the activity durations to be fixed [40]. In practice, every project is subject to uncertainties and it is common for activity durations to fluctuate. This makes the need for a stochastic planning approach indispensable in the field of project management. Stochastic scheduling incorporates the variability in activity duration. Using stochastic scheduling techniques for scheduling naval maintenance operations is a complicated endeavour because of the lack of historical data that is needed to understand activity duration distributions [18]. Table 5 represents a snapshot of a work period data. It can be seen that the calendar days represented by scheduled start and scheduled finish does not relate to the planned hours. The calendar days are generally exaggerated so that if the staff or the contractors is required to work on some other important job, they can get back and still finish work within the allotted time [20]. In addition, the FMF records the billed hours as opposed to actual total hours between the start date and finish date of a WP [1]. That means if a WP gets finished in 5 hours, it could be on one calendar day or spread over many calendar days [1]. Also, any delays that occur due to contingencies like bad weather, unavailability of a part, etc. are not included in the schedule [1]. Therefore, it is very hard to determine the actual execution time for activities [1].

Sched. start	Sched. finish	Revision	Priority	PM actual hours	PM planned hours
15-10-2018	19-10-2018	FNA181FR	07	1.000	3.000
16-11-2018	19-11-2018	FNB181FS	43	4.330	5.000
15-01-2019	28-01-2019	FNC181FT	07	18.000	16.000

Table 5. Data showing that the actual calendar start-to-finish time are much larger than the planned preventive maintenance hours (source: [1])

This, in turn, hinders the application of buffer insertion techniques such as project buffer, feeding buffer, and resource buffer to protect the project against uncertainties [41]. For further review on stochastic RCPSP with buffer insertion techniques, refer [15], [18], [40], [41].

3.2.7 Proactive Scheduling and Reactive Scheduling

The technique where a schedule is generated by integrating enough protection to the baseline schedule by anticipating future contingencies is called proactive scheduling [42]**.** If the project starts to go astray despite proactive scheduling, the reactive scheduling technique is incorporated to bring the project back in line. The reactive scheduling is an indispensable part of project management. There are different methods to perform reactive scheduling. Irrespective of the method chosen, the primary objective is to make sure that the revised schedule is in proximity to the baseline schedule [42]. The inability to bear resemblance can have catastrophic ramifications such as unavailability of scarce and special skilled resources, frequent changes in agreement with subcontractors, and violation of project deadline. [42]. For a review on reactive scheduling policies for the NSWPP project, readers can refer to [19], wherein a rescheduling policy using MILP formulations is developed to minimize the number of scheduling changes when re-scheduling activities.

In this thesis, the NSWPP is modelled with a multi-criteria objective function that aims to minimize the total weighted start time of the project while scheduling all the high priority activities before the low priority ones and for activities within the same priority, the ones with a longer duration are scheduled before the shorter ones. In addition, the following assumptions are made:

- Non pre-emption – once an activity starts it cannot be interrupted till it finishes.
- Single-mode – the activities have only one mode with a specific resource demand and processing duration to execute in.
- Deterministic duration – the activity durations are fixed.

3.3 MILP Formulations

MILP is a variant of integer linear programming where some variables are restricted to be integers, while others can be continuous variables [43]. There are three main categories of MILP formulation for solving RCPSP [44]. They are:

- Time-Indexed Formulations: This formulation makes use of the binary variable $x_{i,t}$, where *i* is the activity index and *t* is the time index. Here, $x_{i,t} = 1$ when activity *i* starts at time period *t*; else it is 0 [44].
- Sequence-Based Formulation: This formulation makes use of the binary variable $y_{i,j}$ and continuous-time variable S_i, where *i* and *j* are activity indices. Here, $y_{i,j} = 1$ if $S_i + D_i \leq S_j$ where D_i is the duration of activity *i*. This formulation was originally intended to solve machine scheduling problems, but with the addition of constraints to satisfy resource demands, it was extended to solve RCPSP [44].

- Event-Based Formulation: This formulation makes use of the binary variable $z_{i,e}$, where i is the activity index and e is the event index. Here, $z_{i,e} = 1$ if activity i starts at event e or still in process at event e. It also makes use of the continuous-time variable t_e to determine the start time of events [44].

3.3.1 Description of Notations

Constants	
c	Positive constant
H	Planning Horizon
n	Number of activities to be scheduled, including dummy activities (if present)
Sets	
A	Set of project activities
E	Set of events
M	Machines
P/prec	Set of precedence pairs
R/res	Set of resource types
Indices	
e	Index used to denote event in set E
i and j	Indices used to denote activity in set A
k	Index used to denote resource in set R
Parameters	
B_k	Represents maximum availability of resource type k
$b_{i,k}$	Represents number resources of each type k demanded by activity i
D_i / d_i	Represents the duration of activity i
ES_i	Represents the earliest possible start time of activity i
LS_i	Represents latest possible start time for activity i
Decision Variable	
S_i	Continuous-time variable
$x_{i,t}$	Binary, 1 if activity i starts at time t
$y_{i,j}$	Binary, 1 if $S_i + D_i \leq S_j$
$z_{i,e}$	Binary, 1 if activity I starts at event e or still being executed after event e
C_i	Completion time
Objective Function	
C_{max}	Makespan, max$\{ C_1, \ldots, C_n \}$
$\sum C_i$	Total completion time ($C_1 + \ldots + C_n$)

3.3.2 The Discrete-Time Formulation

The basic discrete-time (DT) formulation introduced by [45] is given below. It uses a single activity and time indexed binary decision variable $x_{i,t}$ such that

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

The formulation obtained is:

$$Minimize = \sum_{t=ES_i}^{LS_i} t x_{n,t} \quad (1)$$

$$\sum_{t=ES_j}^{LS_j} t x_{j,t} \geq \sum_{t=ES_i}^{LS_i} t x_{i,t} + D_i \qquad \forall (i,j) \in P \quad (2)$$

$$\sum_{i=1}^{n} b_{i,k} \sum_{\tau=\max(ES_i, t-D_i+1)}^{\min(LS_i, t)} x_{i,\tau} \leq B_k \qquad \forall t \in H, \forall i \in A, \forall k \in R \quad (3)$$

$$\sum_{t=ES_i}^{LS_i} x_{i,t} = 1 \qquad \forall\, i \in A \quad (4)$$

$$x_{i,t} \in \{0,1\} \qquad \forall\, i \in A, \forall t \in H \quad (5)$$

Constraint (2) is a precedence constraint that makes sure that the successor does not start before the completion of the predecessor. Constraint (3) makes sure that the resources are not over-utilized beyond their maximum availability. Constraint (4) ensures that all activities are scheduled.

3.3.3 Disaggregated Discrete-Time Formulation

The disaggregated discrete-time formulation (DDT) is similar to the DT formulation with an exception in the way the precedence constraint is formulated, which is shown in equation (6). It ensures that the precedence constraint is satisfied by preventing the predecessor (i) and successor (j) from executing simultaneously. This approach was introduced by Christofides et al. (1987) [46].

$$\sum_{\tau=ES_i}^{LS_i} x_{i,\tau} \; + \sum_{\tau=ES_j}^{\min(LS_j, t+D_i-1)} x_{j,\tau} \; \leq \; 1 \qquad \forall (i,j) \in P \quad (6)$$

3.3.4 On/Off Event-Based Formulation

The On/Off Event-based formulation (OOE) was introduced by Kone et al. in 2011 to solve RCPSP instances. Unlike the time-indexed formulations, where the number of variables grows rapidly with a larger time window, the variable used in the event-based formulation is not a function of time, and it does not exhibit similar growth [46].

$$Minimize = \; C_{max} \qquad (7)$$

$$\sum_{e \in E} z_{i,e} \geq 1 \qquad \forall i \in A \; (8)$$

$$C_{max} \; \geq \; t_e + \left(z_{i,e} - z_{i,e-1}\right) D_i \qquad \forall e \neq 1 \in \; E, \forall i \in A \; (9)$$

$$t_1 = \; 0 \qquad (10)$$

$$t_{e+1} \geq \; t_e \qquad \forall e \neq n \in E \; (11)$$

$$t_{e2} \geq t_{e1} + \left(\left(z_{i,e1} - \; z_{i,e1-1}\right) - \left(z_{i,e2} - \; z_{i,e2-1}\right) - 1\right) D_i \quad \forall (e1, e2, i) \in E^2 \times A, e2 > e1 \; (12)$$

$$\sum_{f=1}^{e-1} z_{i,f} \leq e\left(1-\left(z_{i,e}-z_{i,e-1}\right)\right) \qquad \forall e \neq 1 \in E \quad (13)$$

$$\sum_{f=e}^{n} z_{i,f} \leq (n-e)\left(1+\left(z_{i,e}-z_{i,e-1}\right)\right) \qquad \forall e \neq 1 \in E \quad (14)$$

$$\sum_{f=1}^{e} z_{j,f} + z_{i.e} \leq 1+\left(1-z_{i,e}\right) e \qquad \forall e \in E, \forall (i,j) \in P \quad (15)$$

$$\sum_{i=1}^{n} b_{i,k}.z_{i,e} \leq B_k \qquad \forall e \in E, \forall k \in R \quad (16)$$

$$ES_i.z_{i,e} \leq t_e \qquad \forall e \in E, \forall i \in A \quad (17)$$

$$t_e \leq LS_i\left(z_{i,e}-z_{i,e-1}\right)+LS_n\left(1-z_{i,e}+z_{i,e-1}\right) \qquad \forall e \in E, \forall i \in A \quad (18)$$

$$ES_n \leq C_{max} \leq LS_n \qquad \forall e \in E, \forall i \in A \quad (19)$$

$$t_e \geq 0 \qquad \forall e \in E \quad (20)$$

$$z_{i,e} \in \{0,1\} \qquad \forall e \in E, \forall i \in A \quad (21)$$

Constraint (8) assigns each activity at least one event, ensuring it is scheduled. Constraint (9) defines the makespan, C_{max}, as the end of the last activity. Constraint (10) initializes the first event to the start time of zero. Constraint (11) ensures that events happen in order. Constraint (12) links the binary variable *z* to the corresponding *t* variable. Constraint (13) ensures non-preemption from predecessor events. Constraint (14) ensures non-pre-emption for successor events. Constraints (15) imposes precedence constraints. Constraint (16) imposes resource constraints. Constraints (17) and (18) impose earliest and latest start

times, respectively. Constraints (19) ensures that makespan is within the time window of the last activity. Constraints (20) and (21) describe the continuous variable *t* and binary variable *z,* respectively.

Kone et al. (2011) conducted experiments on a variety of standard RCPSP instances imported from the PSPLIB to test the performance of different MILP formulations. They suggested that there is no single class of MILP formulation that will dominate over the other types of formulations and added that appropriate formulation has to be selected based on the characteristics of the problem.

3.4 RCPSP Solution Methods

The most prominent solution methods for the Mixed Integer Linear Programming (MILP) formulations of the RCPSP are simplex and branch-and-bound solvers, heuristics, metaheuristics, constraint programming, satisfiability test, and matheuristics.

3.4.1 Simplex and Branch-and-Bound based Solvers

A "solver" is a mathematical software that incorporates multiple algorithms that can be used to optimally solve one or more class of problems [47], [48]. For solving combinatorial problems like an RCPSP, solvers such as IBM ILOG CPLEX, Gurobi, CBC and Gusek/GLPK are used which runs on simplex and branch-and-bound algorithms. Although the solvers give an optimal solution for an RCPSP, it can be computationally intensive and it can take several hours to solve a moderate-sized problem with moderate resource constrainedness. However, the advantage of a solver is that it determines the upper bound, lower bound and the optimality gap for a given problem. Therefore, even if the problem does not solve completely in a reasonable amount of time, the value of the optimality gap

can be very useful to analyze the quality of the solution obtained using faster techniques such as heuristics and metaheuristics [49].

3.4.2 Heuristic and Metaheuristic Approach

Blazewicz et al. (1983) showed that the RCSPSP is an NP-hard problem and thus, it would be computationally demanding to solve complex RCPSP using MILP formulations and exact algorithms. Therefore, heuristics and metaheuristic algorithms have become the go-to methods for researchers to solve complex RCPSP [50]. A heuristic is a technique wherein computationally difficult problems are solved quickly at the cost of optimality, accuracy, and completeness [51]. A heuristic algorithm is dependent on the nature of the problem on which it is used, and it is generally greedy in its approach to finding a solution. This often results in entrapment within a local optimum [52]. One of the fastest heuristic algorithms for scheduling an RCPSP is priority rule based scheduling. The application of priority rules helps to reduce the solution space and fastens the computation ([53], [54], [55], [56], [57], [58], [59], [60]). The schedule generation scheme is the core tool on which different priority rules are used to schedule the RCPSP instances [41], [61]. Diana et al. (2012) conducted experiments on the standard RSPSP instances from PSPLIB to compare different priority rules. After experimenting on 23 different priority rules, it was found that the Latest Start Time rule yielded results that were close to the optimal solution. The metaheuristic algorithms are high-level strategies that help a heuristic to efficiently search the solution space to find a good solution for an optimization problem [62]. Unlike the heuristic algorithm, the metaheuristic algorithm is less dependant on the nature of the problem on which it is used [52]. It explores the solution space much more thoroughly, even at the cost of temporary deterioration of the solution, in the hope of finding the optimal

solution [7]. For further reading on heuristic and metaheuristic approaches, refer [50], [61], [63].

3.4.3 Constraint Programming and Satisfiability Test

Constraint programming is a method that solves combinatorial problems such as RCPSP using variables and logical constraints [64]. It derives its solution techniques from various fields like operations research, computer science, and artificial intelligence [65]. Similarly, Satisfiability test (SAT) is another unique solving technique which uses Boolean expressions to solve the RCPSP [64]

3.4.4 Matheuristics

A matheuristic is a hybrid solution technique obtained by combining (meta)heuristic techniques and mathematical programming (MP) techniques [9]. This method integrates the positive characteristics of both techniques to enhance the performance of the algorithms used on complex real-world problems [7]. These hybrid algorithms are extensively used to solve NP-hard problems such as vehicle routing problems, travelling salesman problems, job shop scheduling, and RCPSP [7]. According to Archetti and Speranza (2014), matheuristic methods can be broadly grouped into three classes: decomposition method, improvement heuristics, and relaxation-based method. The decomposition method involves breaking the original problem into smaller subproblems that can then be solved using MP models to optimality [6]. A time limit is generally specified to prevent the optimization process from being limited by the complexity of the subproblem [6]. The improvement heuristics class of matheuristics integrates heuristics and MP models. In this method, the initial solution obtained from a heuristic is improved by using the MP model. This is a popular class, and many ways of incorporating the MP model into (meta)heuristics

have been proposed [6]. [66] proposed a local search with subproblem exact resolution method based on large neighborhood search for solving the single-mode RCPSP. In this method, the initial solution for the given problem is generated by using forward & backward improvement algorithm on serial schedule generation scheme with an aim to minimize the makespan [63]. The obtained solution is further improved by iteratively rescheduling randomly selected subproblems, under a predetermined time limit, using exact solving approach while freezing the start times of activities which are not part of the subproblem [63]. The resulting solution from optimizing the subproblem is fed into the existing initial solution and further post-optimized using the forward-backward improvement algorithm [63]. [67] proposed a matheuristic method integrating integer programming models and a local search based on forward-backward improvement algorithm to solve multi-mode RCPSP. In this method, the initial solution detailing the start times of activities and its execution mode is obtained using a constructive heuristic algorithm [67]. The obtained solution is further improved by iteratively solving subproblems, which are generated based on the earliest start time of the activities within the range of the predetermined time window, using an integer programming model [67], [68]. The range of the time window keeps on increasing after every iteration to generate new subproblems [67]. The process is repeated until the entire problem is solved and a feasible solution is obtained. This feasible solution is further post-optimized using a local search based on the forward-backward improvement algorithm hybridized with an integer programming model to control the execution modes and changes to the solution [67], [68]. [69] proposed the use of the variable neighborhood search algorithm, modified to be used as an advanced optimization method, for solving multi-mode RCPSP. The relaxation-based

method produces an approximate solution for the given problem by relaxing certain attributes in the MP formulation that increases the complexity and computational effort [6]. The obtained solution is further improved using heuristics or metaheuristics. For further review on matheuristics, the reader is referred to [70], [71], [72], [73].

The proposed multi-step optimization matheuristic is of the decomposition class. The large list of activities is decomposed into subgroups using heuristic priority rules. The subgroups are later optimized iteratively using a MILP formulation that generates a near-optimal schedule. The multi-step optimization is discussed in detail in Chapter 5.

3.5 Computational Complexity

The concept of computational complexity was born when the Turing machine, a theoretical computation model developed by Turing, helped to prove the major barrier to the computational speed of the computer [74]. Since then, to understand the computational difficulty, the problems are categorized into different classes such as P-class (Polynomial) and NP-class (Non-deterministic polynomial) [74]. The P-class problems are the ones that can be solved and have their solutions verified in polynomial time. A good example of P-class problem would be multiplication, where irrespective of the size of the problem, the computational time varies as a polynomial function, i.e., O(n^c) where n is the input size and c is a positive constant. NP-class problems are those which are hard to solve in polynomial time, but if the solutions are given, its feasibility/correctness can be verified in polynomial time. A good example of NP-class problem would be the Sudoku game, where solving the entire sudoku can be difficult, but if the solutions are already given, it is easy to check for its correctness. The computational time for NP-class problems increase exponentially with input size, i.e., O(c^n).

The NP problems can be further classified into NP-hard and NP-complete problems. To prove that a problem L is NP-complete, the following two conditions must be satisfied [75]:

1. L is in NP
2. Every problem in NP is reducible to L in polynomial time

Further, if a problem only satisfies condition 2, it is said to be NP-hard [75]. In 1971, Cook showed that the Boolean Satisfiability problem is NP-complete and proved the existence of NP-complete problems [76]. Later in 1972, Karp used Cook's theorem and proved nearly 21 problems to be NP-complete [75]. Blazewicz et al. (1983) extended the machine scheduling problem to RCPSP by considering additional scarce resources that would be required to complete a job. They considered models with parallel machines, unit time duration and minimization of the maximum completion time (makespan) as the optimality criterion (*M2*|res; prec ; d_i=1|C_{max}) for their experimentation and showed that RCPSP problems are NP-hard. They also added that NP-hardness for RCPSP with other optimality criteria such as minimizing total completion time ($\sum C_i$) can also be shown by replacing C_{max} in the same problem [5]. [77] showed that the problem of minimizing the total weighted completion time of the jobs on a single machine with unit time duration and finish to start precedence constraint (1|prec; d_i=1| $\sum w_i C_i$) to be NP-hard. The NSWPP is a variant of the RCPSP, the objective function in the priority-duration formulation for NSWPP, as shown in equation (29), aims to minimize the total weighted start time of the project and has similar characteristics as RCPSP with regards to resources, nature of precedence constraint and processing duration. Thus, by analogy, we can say that NSWPP is also NP-hard.

During experimentation, the NSWPP instances are solved using the Gurobi optimizer which uses the Branch-and-Bound algorithm (BnB) to solve complex NP-hard optimization problems. When optimizing a given problem, the BnB algorithm generates subsets of the search tree and then iteratively explores every subset [78]. When exploring, every candidate solution is checked against the lower bound and upper bound value estimated on the optimum [79]. If the subset being explored does not give a better value than the incumbent, it gets pruned (Bounding) [78]. If a better value is obtained, the subset branches further and the algorithm continues to explore it [78]. In the end, once the entire search tree is explored, the best-found solution is returned as an optimal value [78]. The time complexity of the BnB algorithm can be given as $O(Vv^w)$ [78] where V is a bound on time needed to explore a subset, v is the branching factor of the tree which is the maximum number of children generated at any node in the tree, and w is the search depth, which is the length of the longest path from the root to a leaf. It depends on the size of the input.

An upper bound on the maximum completion time to solve a problem can be estimated by determining the number of activities in the input data [74]. Greater the number of activities in the input, greater would be the computation time required to solve the problem.

3.6 Instance/Network Indicators

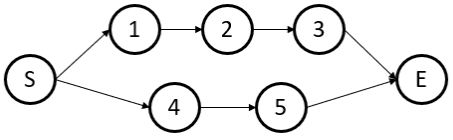
Project networks exhibit unique characteristics that tend to affect the efficiency of the solution methods used [80]. The study of network characteristics is important because it can help to determine the solution method that can solve the hardest problem efficiently and later, the same method can be used to solve other relatively easier problems [81]. These

unique characteristics can be measured using instance indicators. There are four types of instance indicators:

1) Precedence-oriented indicators: Order Strength (OS), Network complexity (NC),
2) Resource-oriented: Resource factor (RF) and Resource Constrainedness (RC),
3) Time-oriented: Process Range (PR), and
4) Hybrid indicators: Resource Strength (RS) [46].

Kolisch et al. (1995) worked extensively on this classification and created the benchmark PSPLIB instances. These instances are regarded as the standard to test any new solution methods to solve RCPSP.

This section aims to present the most promising instance indicators for RCPSP and their effect on computational difficulty. The calculations for determining the network indicator values are explained using an arbitrary instance shown in Figure 9 where A – activity, B – maximum availability of resource type *k*, D – processing duration of the activities and *k* – resource demands.



A	D	k1	k2	k3
1	2	2	1	3
2	5	1	1	2
3	3	2	3	2
4	1	3	0	1
5	4	1	3	2

B	4	3	5

Figure 9. Arbitrary instance with 5 activities and 3 resource types

3.6.1 Network Complexity

This measure indicates how dense the network is. It is given by the ratio of the total number of non-dummy precedence pairs (P) to the total number of non-dummy activities (n) [81]:

$$NC = \frac{P}{n} \tag{22}$$

The network complexity for instance shown in Figure 9 can be calculated as follows:

$$NC = \frac{3}{5} = 0.6$$

A project with many precedence pairs is said to be disjunctive in nature, and thus many activities cannot be executed in parallel [46]. A project having a low number of precedence pairs is said to be cumulative in nature, wherein many activities can be executed in parallel [46].

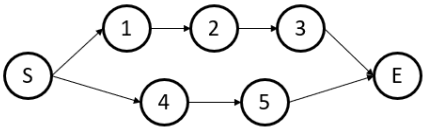
3.6.2 Order Strength

This measure is the ratio between the number of precedence pairs (transitive once included) of non-dummy activities and the theoretical maximum number of precedence pairs [82].

$$OS = \frac{|\,P\,|}{\frac{n^2 - n}{2}} \tag{23}$$

Where n is the number of non-dummy activities. The numerator is the sum of the number of activities that need to be completed for a given activity to start. The order strength for instance shown in Figure 9 can be calculated as follows:

$$OS = \frac{4}{\frac{5^2 - 5}{2}} = 0.4$$



A	|P|
1	0
2	1
3	2
4	0
5	1
Sum	4

De Reyck et al. (1999) conducted experiments on 7200 RCPSP instances using the branch-and-bound algorithm and showed that OS follows a continuous hard-easy phase transition pattern, as shown in Figure 10 [82]. This is because, as the number of precedence relationships within a network increases, the number of feasible solutions decreases due to the reduced possibility of the enumeration tree [83]. Similar results can also be observed with network complexity value [83].

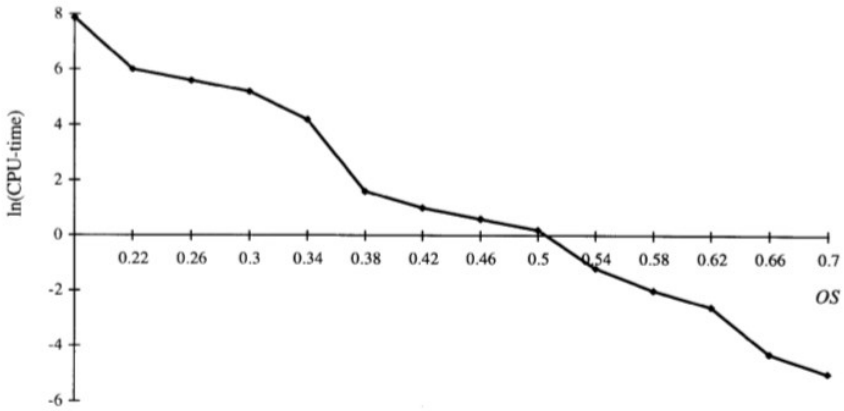


Figure 10. Logarithm of Computation Time vs Order Strength (source: [82])

3.6.3 Resource Factor

This measure was introduced by Pascoe (1966). It denotes the average resource demand for a resource type requested per activity [81], [83].

$$RF = \frac{1}{nk}\sum_{i=1}^{n}\sum_{k=1}^{K}\begin{cases}1, if\ r_{i,k} > 0\\ 0, otherwise\end{cases} \quad (24)$$

$r_{i,k}$ = Demand for resource type k by activity i

The resource factor for instance shown in Figure 9 can be calculated as follows:

$$RF = \frac{14}{5 \times 3} = 0.933$$

If RF = 1, then the all the resources are demanded by all the activities. If RF = 0, then none of the resources are demanded by any of the activities [83]. [83] conducted experiments on 480 RCPSP instances with 30 activities and 4 types of resources using the branch-and-bound algorithm and showed that the computational effort increases as the RF value increases, as shown in Table 6.

RF_R	0.25	0.50	0.75	1.0
μ_{CPU}	0.54	128.35	787.98	928.30
σ_{CPU}	1.81	526.69	1432.74	1498.02

Table 6. Effect of the Resource Factor on the Computation Time (source: [83])

3.6.4 Resource Strength

This measure was coined by Cooper (1976) to measure the size of resource conflict [84].

$$RS_k = \frac{a_k - r_k^{min}}{r_k^{max} - r_k^{min}} \quad (25)$$

Where a_k is the maximum availability of renewable resource type *k*

$$r_k^{min} = max_{i=1,..n}\, r_{i,k}$$

$$r_k^{max} = max_{t=1...H} \left(\sum_{i=1,..n} r_{i,k} \right)$$

r_k^{min} – Maximum demand for resource type *k* by the activities.

r_k^{max} – Peak demand for resource type *k* in the schedule generated on early start time.

The resource strength for instance shown in Figure 9 can be calculated as follows:

$$RS_1 = \frac{4-3}{5-3} = 0.5$$

$$RS_2 = \frac{3-3}{3-3} = 0$$

To get the resource strength for the instance, RS_k values must be averaged over all the resource types *k* [81], [82].

De Reyck and Herroelen (1996) showed that instances with RS ≥ 1 are trivial problems that are no longer resource-constrained, and instance with RS ≈ 0 are extremely computationally demanding, as shown in Figure 11 [82].

The use of RS as an instance indicator has been under a disagreement. While [83] argue that the RS is advantageous as it uses the precedence information along with the resources, [82] and [85] argue that incorporation of precedence information to determine the resource conflicts makes it an impure measure [81], [84]. The fact that RS relies on the peak value of resource demands makes it forget to consider the frequency and magnitude of other

relatively smaller values, which can induce serious bias into the measure, especially when the problem size grows big [81], [84]. To overcome this, many researchers use Resource Constrainedness (RC) as a resource measure [82].

3.6.5 Resource Constrainedness

This measure was introduced by Patterson (1976). It is defined as the average usage of a resource *k* over the activities using resource *k*.

$$RC_k = \frac{\frac{\sum_{i=1}^{n} r_{i,k}}{\sum_{i=1}^{n} \begin{cases} 1, if\ r_{i,k} > 0 \\ 0, otherwise \end{cases}}}{a_k} \tag{26}$$

Where $r_{i,k}$ is the demand for resource *k* by activity *i* and a_k is the maximum availability of renewable resource *k*. The resource constrainedness for instance shown in Figure 9 can be calculated as follows:

$$RC_1 = \frac{\frac{9}{5}}{4} = 0.45$$

$$RC_2 = \frac{\frac{8}{4}}{3} = 0.6667$$

To get the resource constrainedness for the instance, RC_k values must be averaged over all the resource types *k* [81], [82]. [85] showed that RS and RC exhibit an easy-hard-easy complexity pattern that resembles a bell curve that supported a similar conjecture made by [86]. This helped to reject the negative correlation between RS and computational effort shown by [83], [84]. They also emphasized the fact that, along with the average computational effort, the variance is also high at the phase transition region [82]. Therefore,

instances need not be computationally demanding even though their RC and RS values exist in the 'NP-hard region' [82].

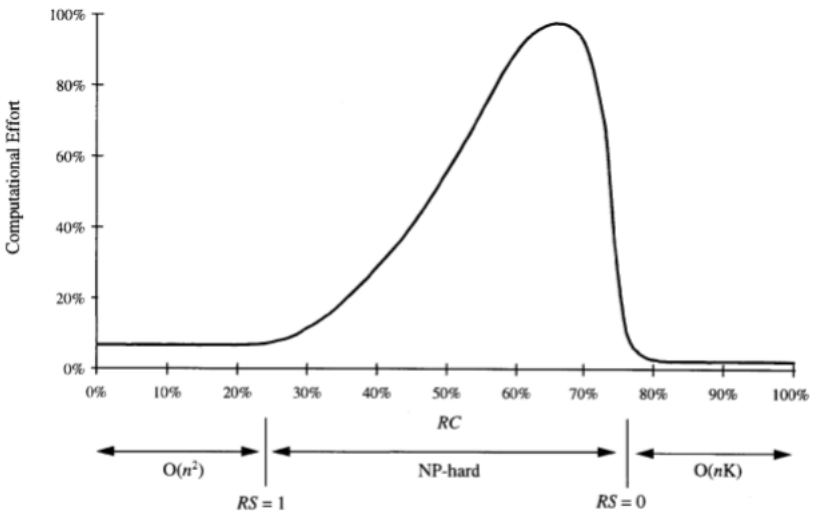


Figure 11. Computational effort vs. RC and RS values (source: [82])

RS and RC can be used as strong indicators to determine network complexity when there is only one type of resource (K=1) or when RC_k and RS_k are similar over all resources type k [82]. Since RC_k and RS_k values are averaged over all the resource types k, in cases where there is more than one type of resource with dissimilar demands, it induces serious bias into the measure [82]. Presently, there are no stable and consistent measures identified to analyze the impact of resource usage on the computational difficulty [82].

3.6.6 Process Range

This measure is the ratio between the maximum and minimum processing duration of the project activities.

$$PR = \frac{max_{i=1..n}\, D_i}{min_{i=1..n}\, D_i} \tag{27}$$

The process range for instance shown in figure 9 can be calculated as follows:

$$PR = \frac{5}{1} = 5$$

A larger PR will significantly increase computation time when minimizing the makespan of a project because the algorithm has to search through a larger solution space to assign an optimal start time for the longer duration activities present among the short duration ones while ensuring that the precedence and resource constraints are satisfied.

3.6.7 Disjunction Ratio

This measure distinguishes between disjunctive instances (with a high number of precedence relations) to cumulative instances (with a low number of precedence pairs) [46]. It integrates resource and precedence measures [46]. It can be defined as a product of resource constrainedness and network complexity [4]

$$DR = NC \times RC \tag{28}$$

[46] observed a trend when comparing the performance of DT, DDT, OOE, Flow-based continuous time (FCT) and Start/End event based (SEE) MILP formulations respectively based on the process range and disjunction ratio. After analyzing the result, they gave a table depicting the performance of different MILP formulations based on these two instance indicators, as shown in table 7.

	High DR	Low DR
Low PR	DDT ≻ DT ≻ FCT ≻ OOE_x ≻ SEE	DDT ≻ DT ≻ OOE_x ≻ FCT ≻ SEE
High PR	FCT,OOE_x ≻ SEE ≻ DT ≻ DDT	OOE_x ≻ FCT ≻ SEE ≻ DT ≻ DDT

Table 7. Synthesis of experiments from Kone et al. (2011)

[46] concluded that when a MILP is used to solve an RCPSP the following formulation should be preferred:

1) DDT formulation if the PR is low.
2) Flow-based continuous-time formulation (FCT) or OOE formulation if the PR and DR are high
3) OOE formulation if PR is high and DR is low

3.7 Characteristics of NSWPP instances

The NSWPP instances are low/medium in network complexity and medium/high in resource constrainedness. Therefore, the DR can vary between low and high based on the instance under consideration. Upon analysis of the NSWPP data, it was observed that the maximum processing time for an WP during the NSWPP is 20 days, which can be considered as low in terms of process range. But the process range will be high if the same were to be considered in hours. Thus, it is necessary to perform tests on instances resembling the NSWPP for all combinations of DR and PR to determine the best suited MILP formulation. The following section discusses the test designed to select the most suited MILP formulation for scheduling activities in the NSWPP.

Chapter 4: Research Methodology and Preliminary Experiments

This chapter details the research procedure followed for developing the matheuristic model. It includes information about the Dal-Randomizer [19] that helps to generate instances resembling the NSWPP project, selection of the best MILP formulation type for solving the NSWPP instances, development of the objective function, and development of the solution scoring criteria to evaluate the solutions.

4.1 Dal-Randomizer

When any new solution method is developed for RCPSP, the general practice is to test it on benchmark instances from the Project Scheduling Problem Library (PSPLIB). The PSPLIB instances were not used in this study for the following reasons:

1) PSPLIB instances have a highly interconnected network structure, as seen in Figure 5, which makes it difficult to distinguish between work packages and activities.

2) PSPLIB instances do not have priorities for activities. Even if we were to assign priorities to the activities for the purpose of experimentation, the connectedness of the network would still make the instance precedence-dominant rather than priority-dominant.

3) PSPLIB instances use a maximum of 4 types of resources in all instances, while the NSWPP consists of 40-60 resource types.

4) PSPLIB instances have low resource constrainedness, while the NSWPP instances have medium-high resource constrainedness.

In addition, the schedule is prepared on the work package level rather than on activity level to reduce the time required to input the data into the system [20]. Since the contractor and RFc staff working on the NSWPP projects are well experienced, they do not need the schedule to guide them on which activity to perform and in which order. Therefore, the project leader does not have to schedule on activity level with all the precedence information [1], [20].

To overcome the limitations of PSPLIB instances, the Dal-Randomizer (created by [19]) was used to generate instances that closely resemble the actual NSWPP. The randomizer is based on a fictitious ship with 51 types of resources, of which 40 are spatial resources, and the remaining 11 represent resources like the crane, jetty spaces, FMF crew, Thales crew, etc.

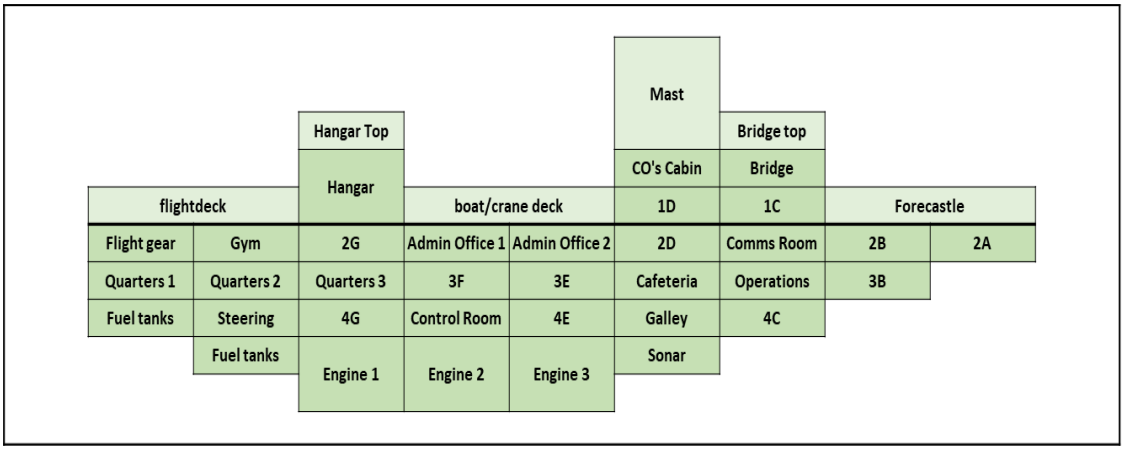


Figure 12. Representation of fictitious ship used in Dal-Randomizer

It uses a random number generator to create unique NSWPP types instances whose network structure is similar to that seen on the work package level, as shown in Figure 53 in appendix 1. Every node generated by the Dal-randomizer is regarded as an "activity" in this paper. The following gives a brief description of the design of the Dal-Randomizer (Dal-R):

- Instances with the number of activities ranging between 0-500 are randomly generated.
- 1 to 4 activities are consolidated into a set with a finish to start precedence relationship, as shown in figure 6. For every set generated: 40% have one activity in them, 30% have two activities in them, 20% have three activities in them, and 10% have four activities.
- Every set is assigned with a priority number ranging between 1 to 3, with 1 representing a high priority activity and 3 representing a low priority activity. The activities within the set inherit the same priority number. 50% of the sets are assigned priority – 1, 30% of the sets are assigned priority – 2, and 20% of the sets are assigned priority – 3.
- Similar percentage metrics are used to assign resources for the activities.

The developer decided these percentage metrics through his experience working in NSWPP projects and analyzing the data provided by the FMFCS. The Dal-R helps to build instances that capture the important aspects of the NSWPP projects and helps to select/develop an efficient solution method to tackle the real-world problem.

4.2 Selection of the formulation type for the NSWPP

To select the most efficient MILP formulation type for scheduling activities in the NSWPP with makespan minimization, tests were performed on multiple formulation types. The time-indexed formulations (DT and DDT) and event-based formulation (OOE) were used, and their computational times were compared to select the best performing formulation. Based on the results in [46] (See Table 7), the event-based formulation is, in theory, capable of performing well on classical RCPSP. Thus, it is included in this test to find out how it

performs on the NSWPP. 10 instances were considered for the initial experimentation, including the preliminary instance shared by Ecole Polytechnique (Figure 52 in appendix 1) and instances generated by the Dal-Randomizer (Figure 53 in appendix 1), to analyze the performance of these formulations for all possible combinations of DR and PR pertaining to NSWPP. All the experiments were conducted on Acer Aspire S7 Ultrabook powered by Intel®Core™ i5-4200 CPU @ 1.60GHz and 8GB DDR3 RAM. The MILP formulation was coded in Gusek/GLPK. The instance data was converted into a readable file using Microsoft Excel VBA and fed into Gusek to generate the LP file, which was later solved using Gurobi 8.1. The maximum time limit was set to 600 seconds (10 minutes), and the MIP Gap was set to 0%.

From Table 8, it is evident that the performance of DT and DDT formulations are in accordance with the table proposed by [46]. The OOE, however, did not perform as expected. Though [46] suggested using the OOE formulation for solving highly cumulative instances and instances with high PR, it performed in accordance with the results in Table 7 only when the instance size was small. The formulation did not solve to optimality even after 600s when the number of activities was increased to 30. [4] showed that OOE fails to solve an instance with 32 activities even after 3600s. The last line in Table 8 ranks the formulations based on the computation speed and the MIP gap from best to worst. Since the results were promising in the case of time-indexed formulation, it was considered for further testing and analysis.

Test Instances	1	2	3	4	5	6	7	8	9	10
|A|	16	16	16	27	27	31	31	32	102	102
|R|	51	51	4	4	4	51	51	4	12	12
NC	0.66	0	1.5	1.93	0	0.67	0	1.5	1.57	1.57
PR	144	144	6	8.33	116	133	133	97	8	116
RC	0.2	0.2	0.54	0.26	0.26	0.39	0.39	0.52	0.55	0.55
DR	0.13	0.00	0.81	0.50	0.00	0.26	0.00	0.78	0.86	0.86
Characteristics	LDR - HPR	LDR - HPR	HDR - LPR	HDR - LPR	LDR - HPR	LDR - HPR	LDR - HPR	HDR - HPR	HDR - LPR	HDR - HPR
DT - Time(s) (MIP Gap(%))	2.48 (0)	2.11 (0)	2.98 (0)	205.34 (0)	600 (0.1)	30.76 (0)	7.66 (0)	187.65 (0)	188.71 (0)	600 (0.05)
DDT - Time(s) (MIP Gap(%))	13.97 (0)	2.12 (0)	3.01 (0)	298.5 (0)	600 (0.02)	161.66 (0)	7.83 (0)	230.28 (0)	66.28 (0)	600 (32.12)
OOE - Time(s) (MIP Gap(%))	0.12 (0)	0.86 (0)	600 (4)	600 (60.37)	600 (19.4)	600 (35.92)	600 (8.43)	600 (82.17)	600 (No sol)	600 (28)
MILP Performance	OOE>DT>DDT	OOE>DT,DDT	DDT,DT>OOE	DT>DDT>OOE	DDT>DT>OOE	DT>DDT>OOE	DT>DDT>OOE	DT>DDT>OOE	DDT>DT>OOE	DT>OOE>DDT

Table 8. Computation time comparison for different MILP formulations tested on NSWPP instances

4.3 The Priority-Duration formulation (PD)

As discussed earlier in section 2.5, the NSWPP is a priority dominant project. Through interviews with the project leaders and schedulers at Thales and FMFCS, it was found that it is desirable to schedule all the high priority activities first and as early as possible [1], [20]. They mentioned the contingencies in the NSWPP and emphasized that by scheduling the high priority activities at its earliest possible time, it will allow them to finish all the mission-critical activities without having to go overboard with the project makespan. Therefore, rather than minimizing the makespan, the objective function from the time-indexed formulation was modified to minimize the total weighted start time of the project while scheduling all the high priority activities before the low priority ones and for activities within the same priority, the ones with a longer duration are scheduled before the shorter ones

$$\text{Minimize } Z = \sum_{i \in A} \sum_{t=ES_i}^{LS_i} \frac{x_{i,t}}{p_i^{\theta}} (\varepsilon + D_i)^{\alpha} \, t \tag{29}$$

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

p_i: Priority of activity i

D_i: Duration of activity i

θ: Weight parameter used to schedule higher priority activities earlier than lower priority activities, respectively in the objective function ($\theta > 0$).

α: Parameter used to ensure that the longer duration activities are scheduled before the shorter ones for activities within the same priority ($\alpha > 1$).

ε: Parameter of very small positive value to account for milestones with no duration ($\varepsilon \geq 0$).

The weight θ increases the objective function coefficient value for priority-1 activities more than that of priority 2 and 3 activities starting at the same time period and ensures that they are scheduled first. In the remainder of this thesis, θ is set to 5 by default. The reason for θ being equal to 5 has to do with Gurobi's objective function co-efficient ratio precision limitation which is discussed in the next section.

The weight α acts as a tiebreaker and ensures that the longer duration activities get scheduled before the shorter duration ones for activities with the same priority. When the activities under consideration are of the same priority, if α is set to 1, there are high possibilities for different sequences of activities to yield the same objective function. This makes the process of distinguishing between a good and a lousy schedule difficult. For example, let's consider the two alternative schedules with 3 priority-1 activities, as shown in Figure 13. If α = 1, then both schedules give the same objective function value while it is evident that schedule 1 is better than schedule 2. Without loss of generality and for ease of argument, we use ε=0 to calculate the objective function in both cases.

Figure 13. Representation of a good and a bad schedule

$$\text{Schedule 1:} \quad Z = (6 \times 0) + (3 \times 6) + (1 \times 9) = 27$$

$$\text{Schedule 2:} \quad Z = (1 \times 0) + (3 \times 1) + (6 \times 4) = 27$$

If $\alpha > 1$, say 1.1, schedule 1 gets a lower objective function value than schedule 2

$$\text{Schedule 1:} \quad Z = (6^{1.1} \times 0) + (3^{1.1} \times 6) + (1^{1.1} \times 9) = 29.09$$

$$\text{Schedule 2:} \quad Z = (1^{1.1} \times 0) + (3^{1.1} \times 1) + (6^{1.1} \times 4) = 32.05$$

Therefore, when $\alpha > 1$, it helps to break the tie and obtain a better schedule where the longer duration activities start before the shorter duration ones. In the remainder of this thesis, α is set to 1.1 by default.

4.4 Gurobi Numerical Issues

The values of θ and α plays an important role while solving the model using Gurobi. The algorithms used in Gurobi are sensitive to the numerical values encountered during optimization, and the software can produce erroneous results in the presence of numerical issues [87]. To prevent this, certain parameters are to be dealt with care. The issue that had a prominent effect on the solution quality was the objective function coefficient ratio. During the initial phase of testing, the value of θ was set to 10 and that of α was set to 2. Although these values produced a large difference in the objective function coefficient

value between activities of different priority levels which in turn helps to solve the instance quickly, it caused the objective function coefficient ratio to be very large.

For example, let's consider a project with a time horizon of 200 days with a priority-3 activity of duration 1 day, and a priority-1 activity with a duration of 20 days. Furthermore, the priority-3 activity has a time window (i.e., range of start time ES–LS) of (1-199) days. The priority-1 activity has a time window of (1-180) days. Then, the objective function coefficients (i.e., $\frac{1}{p_i^{10}}(\varepsilon + D_i)^2 t$) will be as follows for both cases:

$$c_1 = \frac{1}{1^{10}} \times 20^2 \times 180 = 72000 \quad \text{(maximum value)}$$

$$c_3 = \frac{1}{3^{10}} \times 1^2 \times 1 = 1.694 \times 10^{-5} \quad \text{(minimum value)}$$

$$\text{Objective function Coefficient ratio} = \frac{72000}{1.694 \times 10^{-5}} = 4.25 \times 10^9$$

Because of such a large ratio, Gurobi scheduled the activities very poorly. In some cases, the priority-3 activities were not at all optimized. They were scheduled at a time period equal to their LS time.

Given that the objective function coefficient ratio is recommended to be less than 10^9, the lesser the better, to eliminate the possibility of erroneous solution [87], the values of θ and α were set to 5 and 1.1 respectively, which resulted in the improvement of the objective function coefficient ratio ($1.1804 \times 10^6 \ll 10^9$).

In addition to the above-mentioned issue, Gurobi can sometimes return non-integer values for integer variables, especially for a MILP model. Thus, 0.9999934543 can be returned

instead of 1. This happens because the integer feasibility tolerance limit is set to $1E^{-5}$ by default [88]. Since a value like 0.9999934543 falls within default tolerance value (1 - 0.9999934543= $6.5E^{-6}$), Gurobi considers it to be feasible [88]. This can be prevented by appropriately tightening the tolerance [88]. The integer feasibility tolerance in Gurobi ranges between $1E^{-9}$ to $1E^{-1}$ and during experimentation the value was set to $1E^{-7}$ to prevent erroneous values. However, the user must be careful not to tighten the value too much as it can significantly increase the solution time [88].

4.5 Solution Comparison Criteria

The objective function in the PD formulation does not aim to minimize the makespan. It minimizes the total weighted start time of the project while scheduling all the high priority activities before the low priority ones and for activities within the same priority, the ones with a longer duration are scheduled before the shorter ones. Since makespan cannot be used as an effective metric to differentiate between a good schedule and a bad schedule, a new metric called the average priority-1 duration weighted centroid (DWC) is used. This measure was developed by [19] and is defined as follows.

$$\text{Average Priority} - 1 \text{ Duration Weighted Centroid} = \frac{\Sigma_{i \in \mathbb{P}_1} \left[\frac{(x_{start} + x_{finish}) \times D_i}{2} \right]}{n_{\mathbb{P}_1}} \quad (30)$$

Where $\mathbb{P}_1$ is the set containing priority-1 activities and $n_{\mathbb{P}_1}$ is the number of priority-1 activities.

Figure 14. Differentiating a good schedule from a bad schedule using Average Priority-1 Duration Weighted Centroid

The schedules shown in Figure 14 are considered. If makespan is used as a performance measure, all three schedules will end up having the same value of 16, although it is evident that schedule 3 is undesirable. The average priority-1 DWC clearly distinguishes between a good and a bad schedule by considering only the priority-1 activities in its formula, thus eliminating any bias introduced by lower priority activities. Smaller the value of average priority-1 DWC, the better the schedule because a smaller value represents a front-loaded schedule with all the priority-1 activities scheduled at the earliest.

4.6 Comparative test between Discrete-Time and Disaggregated Discrete-Time formulations

There are two variants of time-indexed formulation: Discrete-Time and Disaggregated Discrete-Time. They differ only in the way the precedence constraint is formulated as discussed in section 3.3. To decide between DT and DDT, tests were conducted on a total of 50 random instances generated on the Dal-Randomizer to measure the computational speed of both formulations. Table 9 shows the characteristics of the instances.

	Dal-R 100
|A|	100
|R|	51
T	200
NC	0.45 - 0.68
RC	0.61 - 0.81
DR	0.32 - 0.48
PR	20
Instances	50

Table 9. Characteristics of the Dal-R 100 test instances

The computation times for solving each instance using DT and DDT formulations were recorded, and a paired *t*-test was used to determine the best by measuring the mean difference between the two formulations with the assumption that the relative difference between the solutions (solve time, makespan etc.) for the same instance is normally distributed.

A *t*-test was used since the sample size selected was not too large (i.e., <100) [89]. A confidence interval of 90%, i.e., an alpha value of 0.10 was used for all the *t*-tests. A 90% CI was selected instead of 95% CI or 99% CI as the chosen sample size was just 50. To be 95% or 99% definite about the results, the number of test samples need to be high (more than 500) so that a strong result can be obtained with a very small margin of error [90].

4.6.1 Computation Time Comparison

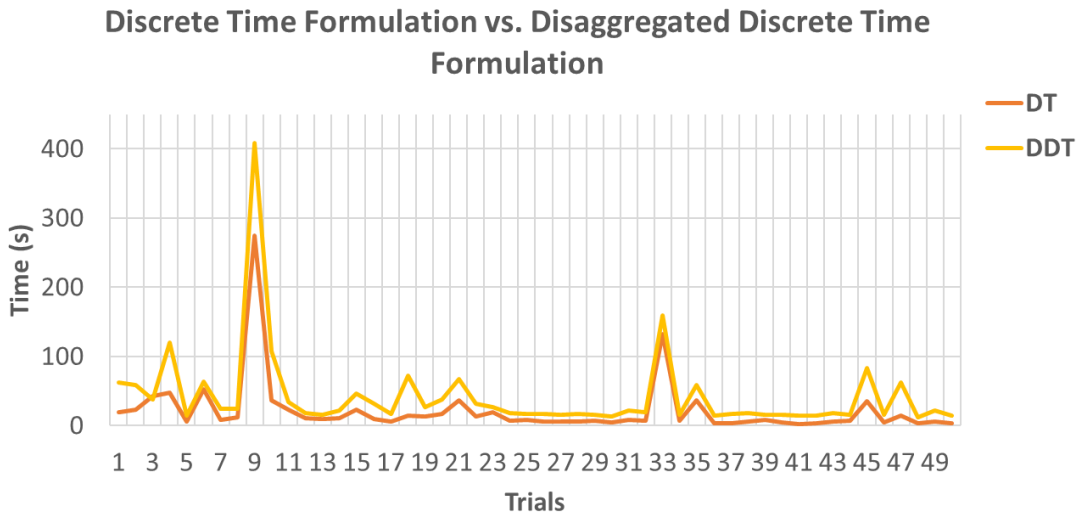


Figure 15. Computation time comparison between DT and DDT formulation

The test result shows that DT formulation performs 57% $\pm$ 4.4% faster than the DDT formulation at 90% CI.

The testing concluded that the DT formulation exhibits superior computational performance over DDT for solving NSWPP instances. Thus, the constraints from the DT formulation were incorporated into the priority-duration formulation, as shown in the following.

$$\text{Minimize} \quad Z = \sum_{i \in A} \sum_{t=ES_i}^{LS_i} \frac{x_{i,t}}{p_i^{\theta}} (\varepsilon + D_i)^{\alpha} t \tag{31}$$

$$\sum_{t=ES_j}^{LS_j} t x_{j,t} \geq \sum_{t=ES_i}^{LS_i} t x_{i,t} + D_i \qquad \forall (i,j) \in P \tag{32}$$

$$\sum_{i=1}^{n} b_{i,k} \sum_{\tau=\max(ES_i, t-D_i+1)}^{\min(LS_i, t)} x_{i,\tau} \leq B_k \qquad \forall t \in H, \forall i \in A, \forall k \in R \tag{33}$$

$$\sum_{t=ES_i}^{LS_i} x_{i,t} = 1 \qquad \forall\, i \in A \quad (34)$$

$$x_{i,t} \in \{0,1\} \qquad \forall\, i \in A, \forall t \in H \quad (35)$$

4.7 Comparison between minimization and maximization models

In this thesis, the developed MILP is a minimization model. A maximization formulation with similar constraints was proposed by [19], as shown in equation (36), to deal with the NSWPP.

$$\text{Maximize} \quad Z = \sum_{i \in A} \sum_{t=ES_i}^{LS_i} \frac{100\, x_{i,t}}{p_i^{\theta}} (\varepsilon_1 + D_i)^{\alpha} (1 - \varepsilon_2 t) \qquad (36)$$

The validity of both models was analyzed through experimentation on instances shown in Table 9 with the condition that both models should give similar solutions, and if the solution from the minimization model were to be plugged into the maximization model, the objective function value should equal that of the maximization model.

The experimental results showed that both the formulations produce similar solutions, as depicted in Figures 16 and 17. Thus, proving that both formulations are suitable for scheduling activities in NSWPP.

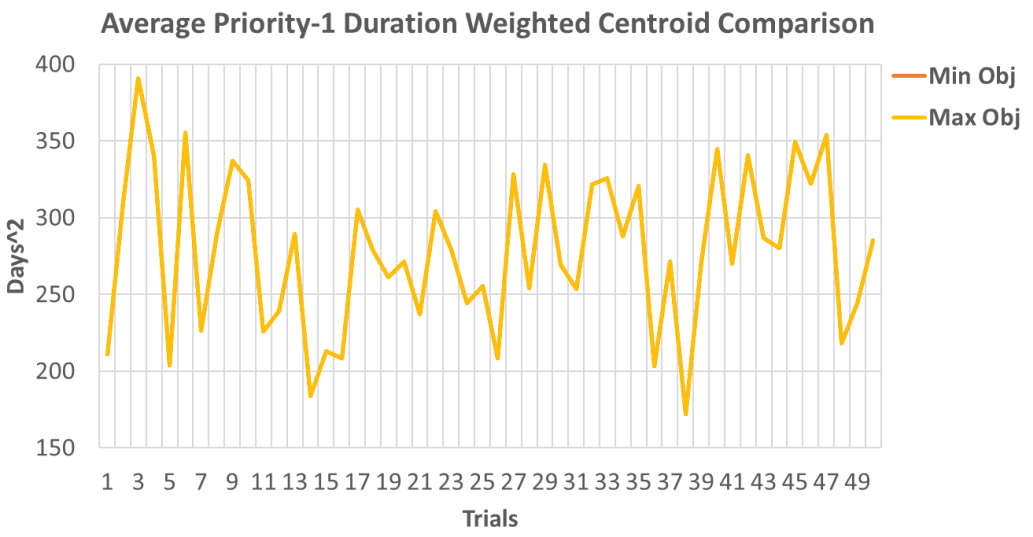


Figure 16. Average priority-1 DWC comparison between minimization and maximization model

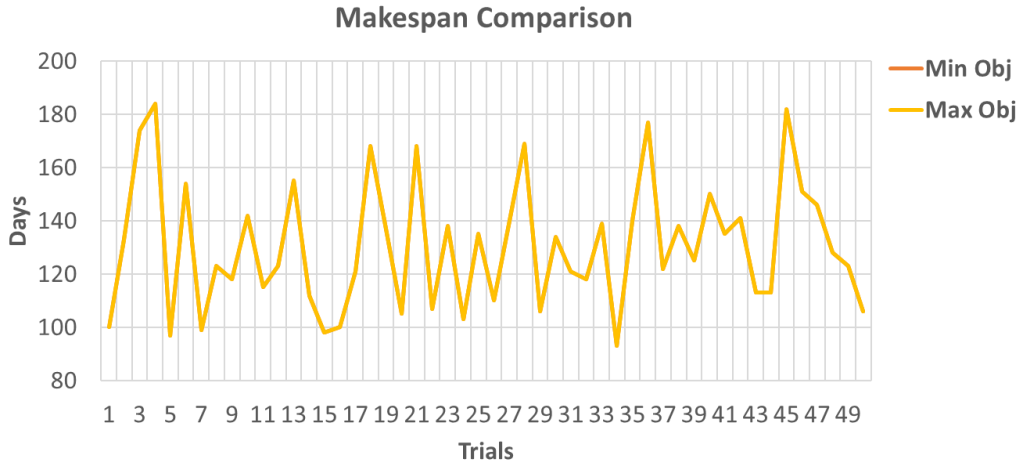


Figure 17. Makespan comparison between minimization and maximization model

During the tests, it was observed that the minimization model performed faster than the maximization model. To analyze this further, the *t*-test was used on the solutions times for both models. From Figure 18, the test showed that the minimization model performs 12.9% $\pm$ 7.1% faster than the maximization model at 90% CI. Since the minimization model proved to be computationally faster, it was used in the remaining experimentations.

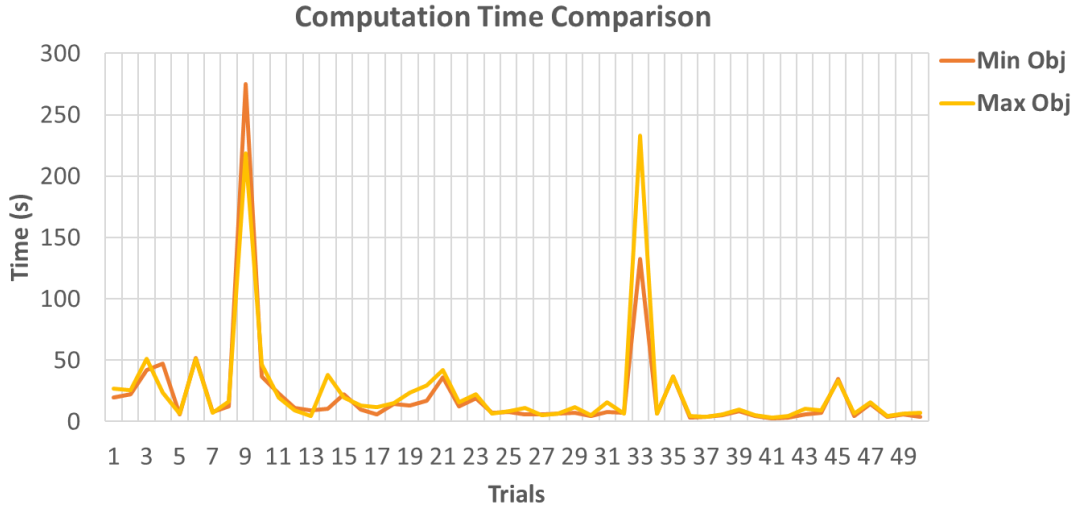


Figure 18. Computation time comparison between minimization and maximization model

4.8 Computation time growth for NSWPP

It has been established that RCPSP is an NP-hard problem, and thus it exhibits an exponential increase in computation time with an increase in the instance size. NSWPP being a variant of RCPSP also exhibits similar behaviour. To show this, an experiment was conducted using the PD formulation on seven NSWPP instances generated on the Dal-Randomizer with the number of activities ranging between 50 to 200 and increasing in increments of 25. Table 10 shows the characteristics of the instances.

Dal-Randomizer Instances							
|A|	50	75	100	125	150	175	200
|R|	51	51	51	51	51	51	51
T	100	125	125	150	150	175	175
NC	0.46	0.47	0.46	0.48	0.49	0.49	0.48
RC	0.61	0.71	0.77	0.81	0.83	0.83	0.83
DR	0.28	0.33	0.35	0.39	0.41	0.40	0.39
PR	20	20	20	20	20	20	20

Table 10. Characteristics of the Dal-R test instances used to show the growth in computation time for NSWPP

From Figure 19, the computation time grows exponentially as the size of the problem increases. The instance with 200 activities ran for more than 7 hours without solving to optimality when solved using the regular optimization method wherein the entire instance is solved at once.

To reap the benefits of the exact solution method to solve large NSWPP projects, it is important to make sure that the instances are solved within a short time frame irrespective of its size.

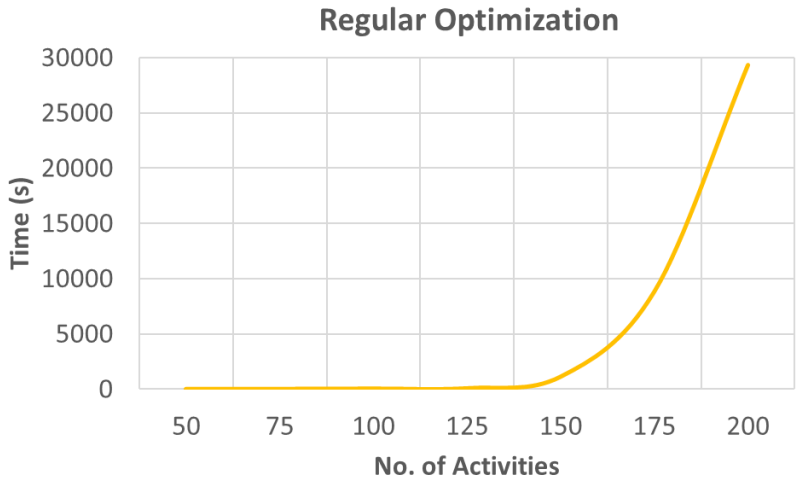


Figure 19. Computation time growth with an increase in instance size for regular optimization

The following chapter discusses the development of matheuristic methods to schedule large NSWPP problems efficiently.

Chapter 5: Development of an efficient matheuristic approach and numerical experiments

In the course of developing an efficient matheuristic approach, different decomposition strategies were examined sequentially. The results from the primary experimentation were analyzed to understand the merits and demerits of the opted strategy, and this knowledge was used to set-up subsequent experiments to improve the matheuristic. This chapter details different decomposition strategies used for the matheuristic approach and their results

5.1 Multi-step Optimization Version 1

The developed multi-step optimization (MSO-1) is a matheuristic technique where a large instance is solved in multiple iterations by decomposing its list of activities into subgroups and then iteratively optimizing each subgroup using the priority-duration formulation.

$$\text{Iteration 1}: \qquad \text{Min } Z = \sum_{i \in \mathbb{P}_1} \sum_{t=ES_i}^{LS_i} \frac{x_{i,t}}{p_i^{\theta}} . (\varepsilon + D_i)^{\alpha} . t \tag{37}$$

$$\text{Iteration 2}: \qquad \text{Min } Z = \sum_{i \in \mathbb{P}_2} \sum_{t=ES_i}^{LS_i} \frac{x_{i,t}}{p_i^{\theta}} . (\varepsilon + D_i)^{\alpha} . t \tag{38}$$

$$\sum_{t=ES_j}^{LS_j} t x_{j,t} \geq \sum_{t=ES_i}^{LS_i} t x_{i,t} + D_i \qquad \forall (i,j) \in P \tag{39}$$

$$\sum_{i=1}^{n} b_{i,k} \sum_{\tau=\max(ES_i, t-D_i+1)}^{\min(LS_i, t)} x_{i,\tau} \leq B_k \qquad \forall t \in H, \forall i \in A, \forall k \in R \quad (40)$$

$$\sum_{t=ES_i}^{LS_i} x_{i,t} = 1 \qquad \forall\, i \in A \quad (41)$$

$$x_{i,t} \in \{0,1\} \qquad \forall\, i \in A, \forall t \in H \quad (42)$$

Where, A $\supset \mathbb{P}_1, \mathbb{P}_2$

$\mathbb{P}_1$ is the subgroup consisting of only priority-1 activities.

$\mathbb{P}_2$ is the subgroup consisting of only priority 2 and 3 activities.

Unlike existing matheuristic approaches for RCPSP, the proposed approach does not create subproblems. It operates on the entire problem during every iteration but only considers the activities of the "in-process" subgroup in the objective function. Therefore, only the activities considered in the objective function will be optimized. The remaining activities get scheduled in any timeslot within their time window that satisfies the precedence and resource constraints. The following describes the steps required to implement MSO-1 with the help of the arbitrary instance generated on the Dal-Randomizer as shown in Figure 20.

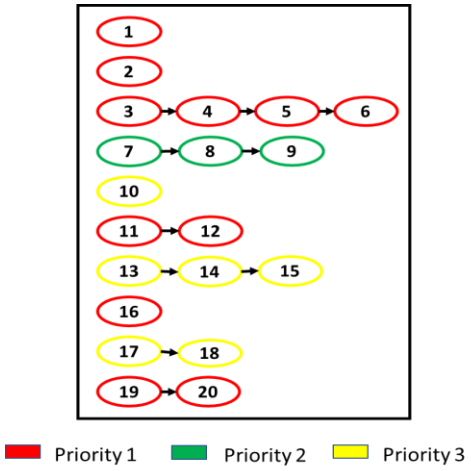


Figure 20. The network structure of the arbitrary instance with 20 activities

Activities before sorting

Sl No.	D	ES	LS	p
1	10	0	50	1
2	19	0	41	1
3	3	0	37	1
4	2	3	40	1
5	7	5	42	1
6	11	12	49	1
7	19	0	10	2
8	14	19	29	2
9	17	33	43	2
10	6	0	54	3
11	3	0	50	1
12	7	3	53	1
13	6	0	36	3
14	16	6	42	3
15	2	22	58	3
16	15	0	45	1
17	7	0	49	3
18	4	7	56	3
19	20	0	29	1
20	11	20	49	1

Activities after sorting

Sl No.	D	ES	LS	p
1	10	0	50	1
2	19	0	41	1
3	3	0	37	1
4	2	3	40	1
5	7	5	42	1
6	11	12	49	1
11	3	0	50	1
12	7	3	53	1
16	15	0	45	1
19	20	0	29	1
20	11	20	49	1
7	19	0	10	2
8	14	19	29	2
9	17	33	43	2
10	6	0	54	3
13	6	0	36	3
14	16	6	42	3
15	2	22	58	3
17	7	0	49	3
18	4	7	56	3

$\mathbb{P}_1$:= 1,2,3,4,5,6,11,12,16,19,20

$\mathbb{P}_2$:= 7,8,9,10,13,14,15,17,18

Figure 21. Activity decomposition after sorting the arbitrary instance with 20 activities

Data for 1st iteration

Sl No.	D	ES	LS	p
1	10	0	50	1
2	19	0	41	1
3	3	0	37	1
4	2	3	40	1
5	7	5	42	1
6	11	12	49	1
7	19	0	10	2
8	14	19	29	2
9	17	33	43	2
10	6	0	54	3
11	3	0	50	1
12	7	3	53	1
13	6	0	36	3
14	16	6	42	3
15	2	22	58	3
16	15	0	45	1
17	7	0	49	3
18	4	7	56	3
19	20	0	29	1
20	11	20	49	1

⇨

Data for 2nd iteration

Sl No.	D	ES	LS	p
1	10	0	0	1
2	19	0	0	1
3	3	0	0	1
4	2	3	3	1
5	7	5	5	1
6	11	12	12	1
7	19	0	10	2
8	14	19	29	2
9	17	33	43	2
10	6	0	54	3
11	3	0	0	1
12	7	3	3	1
13	6	0	36	3
14	16	6	42	3
15	2	22	58	3
16	15	0	0	1
17	7	0	49	3
18	4	7	56	3
19	20	0	0	1
20	11	20	20	1

Figure 22. Representation of the change in data after the first iterations in the MSO-1

Step 1: Sort the activities in an ascending order based on their priority numbers and create two subgroups with the 1st subgroup ($\mathbb{P}_1$) consisting of only priority-1 activities and the 2nd subgroup ($\mathbb{P}_2$) consisting of priority-2 and priority-3 activities, as shown in Figure 21.

Step 2: For the first iteration, consider only the activities of the 1st subgroup ($\mathbb{P}_1$) in the objective function and optimize it. At the end of the first iteration, fix the time window (i.e., ES time and LS time) of the activities in $\mathbb{P}_1$ to their start times from the solution, as shown in Figure 22. Once the time window of the optimized activities gets fixed, it will remain fixed until the entire instance is solved.

Step 3: For the second iteration, consider only the activities from the 2nd subgroup ($\mathbb{P}_2$) in the objective function and optimize it.

Decomposing the activities helps to reduce the computational difficulty and solve the given problem quickly. Fixing the time-window of the activities in $\mathbb{P}_1$ to their start time values from the solution prevents any changes to the schedule during the subsequent iterations.

Figure 23 shows the improvement in computational time for using MSO-1 over the regular optimization on the instances shown in Table 10. While the regular optimization ran for over 7 hours without generating an optimal solution for an instance with 200 activities, the MSO-1 solved the same instance in under 10 minutes.

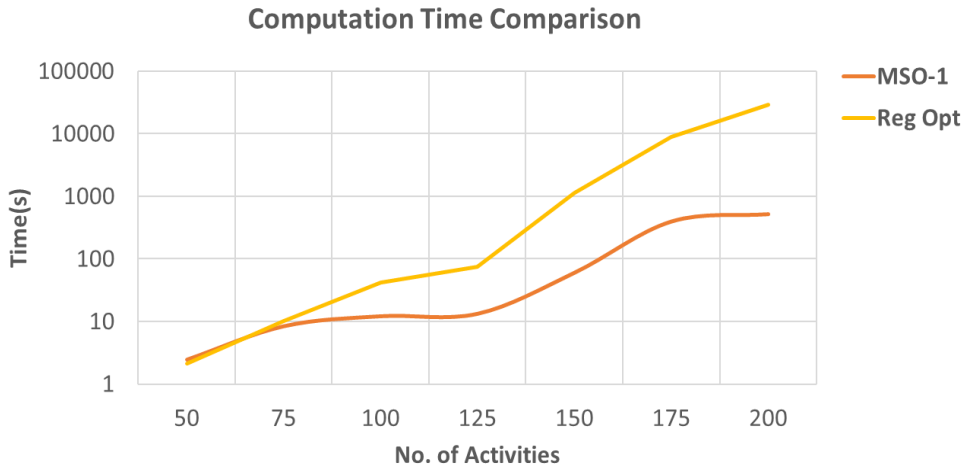


Figure 23. Computation time comparison between Regular Optimization and MSO-1 with an increase in instance size

It should be noted that for computation time comparisons, only the time taken for Gurobi to solve the instance was considered. The pre-processing time taken in-between iterations to update the datasheet and to generate the new LP file was not considered because the pre-processing time can vary depending on the software being used. The author used MS Excel VBA to interact with multiple software such as – notepad, excel, Gusek, and Gurobi. While data updating in-between iterations was done in 5s – 15s, the generation of the LP file took the longest and it grew with an increase in the instance size (between 20 seconds to 5

minutes for the number of activities ranging between 100 to 500). The interaction between multiple software and the pre-processing time can be reduced considerably or eliminated with the use of efficient software having a powerful in-built solver to optimize the problem directly without having to generate an LP file.

5.1.1 Multi-step Optimization-1 vs. Regular Optimization

To analyze the efficiency and solution quality, MSO-1 was compared with the regular optimization technique. Both approaches were tested on 50 instances generated on the Dal-Randomizer. The details of the test instances are shown in Table 9.

5.1.1.1 Computation time comparison

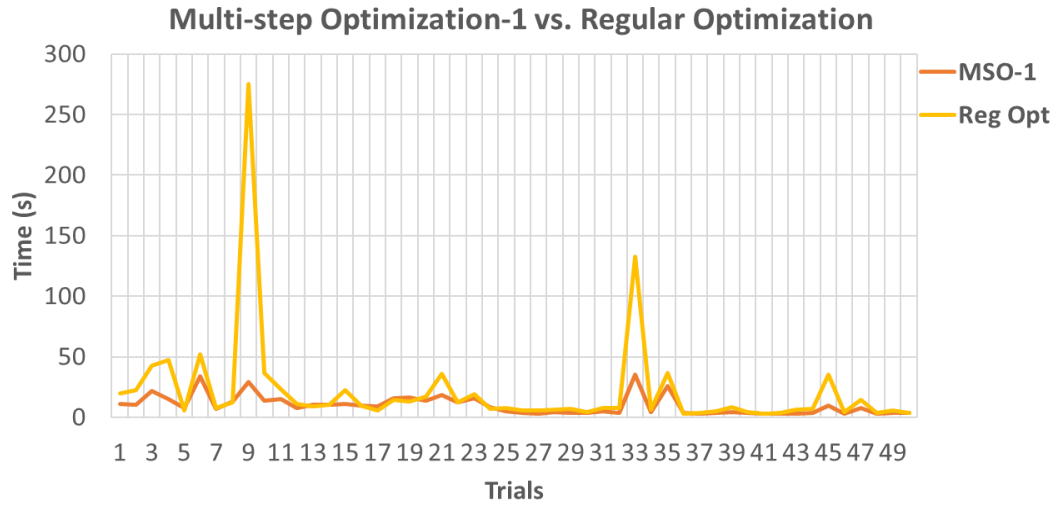


Figure 24. Computation time comparison between MSO-1 and Regular Optimization

The tests showed that MSO-1 is 25.7% $\pm$ 7.2% faster than the regular optimization at 90% CI, which is a significant improvement.

5.1.1.2 Average Priority-1 DWC comparison

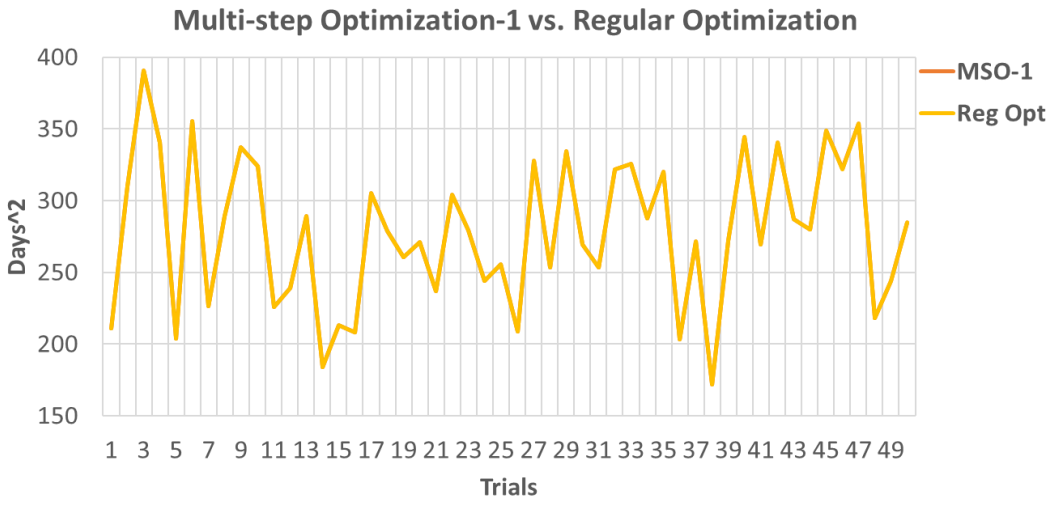


Figure 25. Average priority-1 DWC comparison between MSO-1 and Regular Optimization

The MSO-1 gives identical average priority-1 DWC value to that of regular optimization at 90% CI. Since the priority-1 activities are weighted so greatly in the PD formulation, MSO-1 gives similar DWC values as regular optimization.

5.1.1.3 Makespan comparison

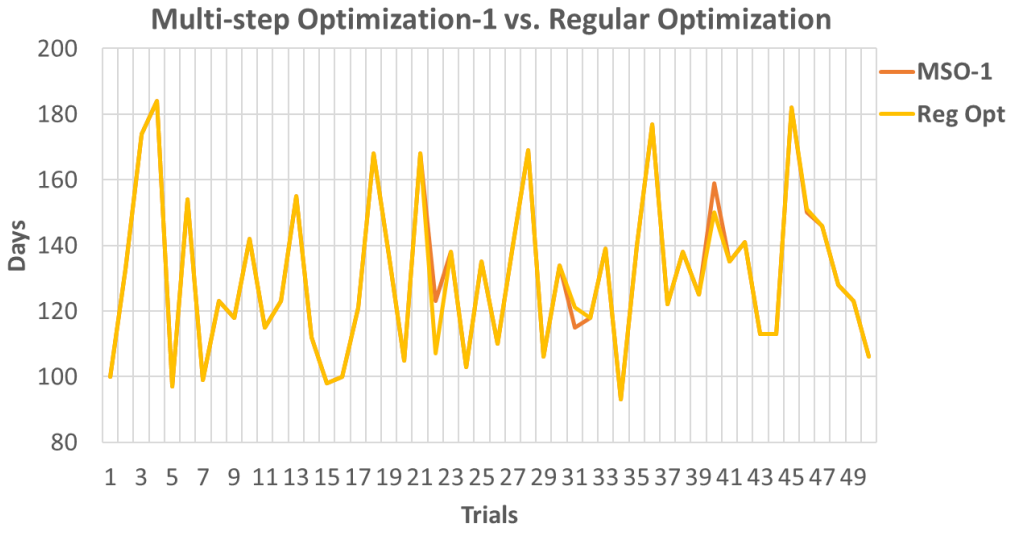


Figure 26. Makespan comparison between MSO-1 and Regular Optimization

MSO-1 gives 0.31% $\pm$ 0.6% longer makespan than regular optimization at 90% CI which is not statistically significant. Therefore, MSO-1 gives solutions identical to that of regular optimization while performing 25% faster.

Although there are advantages of using MSO-1, it comes with certain drawbacks. This technique is not beneficial if the instance under consideration consists of a large number of activities belonging to the same priority. From the experiment discussed in this section, if all the 100 activities were of the same priority number, say priority-1, then creating the activity subgroups from the large instances would not be possible using the MSO-1 decomposition technique. As a result, MSO-1 would perform similarly to the regular optimization and exhibit similarly large computational time. Thus, Multi-step optimization version 2 presented in the next section was developed to overcome this drawback.

5.2 Multi-step Optimization Version 2

This version divides the activities into *n* subgroups, A $\supset S_1, S_2,\ldots, S_n$, that are iteratively optimized using the priority-duration formulation. Thus, the objective functions for each iteration are:

$$\text{Iteration 1}: \qquad \text{Min } Z = \sum_{i \in S_1} \sum_{t=ES_i}^{LS_i} \frac{x_{i,t}}{p_i^{\theta}} . (\varepsilon + D_i)^{\alpha} . t \tag{43}$$

$$\text{Iteration 2}: \qquad \text{Min } Z = \sum_{i \in S_2} \sum_{t=ES_i}^{LS_i} \frac{x_{i,t}}{p_i^{\theta}} . (\varepsilon + D_i)^{\alpha} . t \tag{44}$$

……..

$$\text{Iteration } n: \qquad \text{Min } Z = \sum_{i \in S_n} \sum_{t=ES_i}^{LS_i} \frac{x_{i,t}}{p_i^{\theta}} . (\varepsilon + D_i)^{\alpha} . t \qquad (45)$$

The constraints remain the same as before.

$$\sum_{t=ES_j}^{LS_j} t x_{j,t} \geq \sum_{t=ES_i}^{LS_i} t x_{i,t} + D_i \qquad \forall (i,j) \in P \quad (46)$$

$$\sum_{i=1}^{n} b_{i,k} \sum_{\tau=\max(ES_i, t-D_i+1)}^{\min(LS_i, t)} x_{i,\tau} \leq B_k \qquad \forall t \in H, \forall i \in A, \forall k \in R \quad (47)$$

$$\sum_{t=ES_i}^{LS_i} x_{i,t} = 1 \qquad \forall\, i \in A \quad (48)$$

$$x_{i,t} \in \{0,1\} \qquad \forall\, i \in A, \forall t \in H \quad (49)$$

Similar to MSO-1, MSO-2 also operates on the entire problem during every iteration by only optimizing the activities of the "in-process" subgroup. The only difference between the two versions is the sorting technique used to create the subgroups. The following describes the steps required to implement MSO-2 with the help of the arbitrary instance shown in Figure 20.

Activities before sorting

Sl No.	D	ES	LS	p
1	10	0	50	1
2	19	0	41	1
3	3	0	37	1
4	2	3	40	1
5	7	5	42	1
6	11	12	49	1
7	19	0	10	2
8	14	19	29	2
9	17	33	43	2
10	6	0	54	3
11	3	0	50	1
12	7	3	53	1
13	6	0	36	3
14	16	6	42	3
15	2	22	58	3
16	15	0	45	1
17	7	0	49	3
18	4	7	56	3
19	20	0	29	1
20	11	20	49	1

Activities after sorting

Sl No.	D	ES	LS	p
19	20	0	29	1
2	19	0	41	1
16	15	0	45	1
1	10	0	50	1
3	3	0	37	1
11	3	0	50	1
12	7	3	53	1
4	2	3	40	1
5	7	5	42	1
6	11	12	49	1
20	11	20	49	1
7	19	0	10	2
8	14	19	29	2
9	17	33	43	2
17	7	0	49	3
10	6	0	54	3
13	6	0	36	3
14	16	6	42	3
18	4	7	56	3
15	2	22	58	3

S_1 := 19,2,16,1,3
S_2 := 11,12,4,5,6
S_3 := 20,7,8,9,17
S_4 := 10,13,14,18,15

Figure 27. Activity decomposition after applying three-level sorting to the arbitrary instance with 20 activities

Data for 1st iteration

Sl No.	D	ES	LS	p
1	10	0	50	1
2	19	0	41	1
3	3	0	37	1
4	2	3	40	1
5	7	5	42	1
6	11	12	49	1
7	19	0	10	2
8	14	19	29	2
9	17	33	43	2
10	6	0	54	3
11	3	0	50	1
12	7	3	53	1
13	6	0	36	3
14	16	6	42	3
15	2	22	58	3
16	15	0	45	1
17	7	0	49	3
18	4	7	56	3
19	20	0	29	1
20	11	20	49	1

⇨

Data for 2nd iteration

Sl No.	D	ES	LS	p
1	10	0	0	1
2	19	0	0	1
3	3	0	0	1
4	2	3	40	1
5	7	5	42	1
6	11	12	49	1
7	19	0	10	2
8	14	19	29	2
9	17	33	43	2
10	6	0	54	3
11	3	0	50	1
12	7	3	53	1
13	6	0	36	3
14	16	6	42	3
15	2	22	58	3
16	15	0	0	1
17	7	0	49	3
18	4	7	56	3
19	20	0	0	1
20	11	20	49	1

Figure 28. Representation of the change in data after the first iterations in the MSO-2

Step 1: Sort the activities on three levels. The first level by priority number (smallest to largest), the second level by their ES time (shortest to longest), and the third level by their duration (longest to shortest). After sorting, create subgroups of the desired size, as shown in Figure 27.

Step 2: Start the first iteration by considering only the activities of the 1st subgroup (S_1) in the objective function and optimize it. At the end of the first iteration, fix the time window (i.e., ES time and LS time) of the activities in S_1 to their start times from the solution, as shown in Figure 28. Once the time window of the optimized activities gets fixed, it will remain fixed until the entire instance is solved.

Step 3: Start the second iteration by considering only the activities of the 2nd subgroup (S_2) in the objective function and optimize it. At the end of the second iteration, fix the time window (i.e., ES time and LS time) of the activities in S_2 to their start times from the solution. Once the time window of the optimized activities gets fixed, it will remain fixed until the entire instance is solved.

Step 4: Repeat the steps by considering the next subgroup and follow the same until all the subgroups are optimized.

Decomposing the activities using the three-level sorting technique is advantageous because it helps to create ideal subgroups that mimic the objective function of the PD formulation by ensuring that the priority-1 activities with the shortest ES time and the longest duration are scheduled first. Also, since the time window of the activities in the previously optimized subgroup gets fixed at the end of each iteration, as shown in Figure 28, it is important to use three-level sorting as it ensures that the activities with lower priority or shorter duration are not present in the earlier subgroups. If not, the resulting schedule could be far away from the optimum.

Determining the size of the subgroup is an open-problem. For testing MSO-2 on NSWPP instances, the activities were arbitrarily broken into subgroups of size 50. The size of the

subgroup is determined such that it is small enough to be solved easily in a reasonable amount of time using a MILP model and, at the same time, large enough to be able to explore relevant resource allocations.

No time limit was imposed for the execution of the MILP model, as interrupting the optimization process would decrease the probability of getting near the optimum solution.

Figure 29 shows the computation times of MSO-1, MSO-2 and the regular optimization tested on the instances shown in Table 10. While the regular optimization ran for over 7 hours without generating an optimal solution for an instance with 200 activities, MSO-2 solved the same instance in under 4 minutes while MSO-1 took a little under 10 minutes.

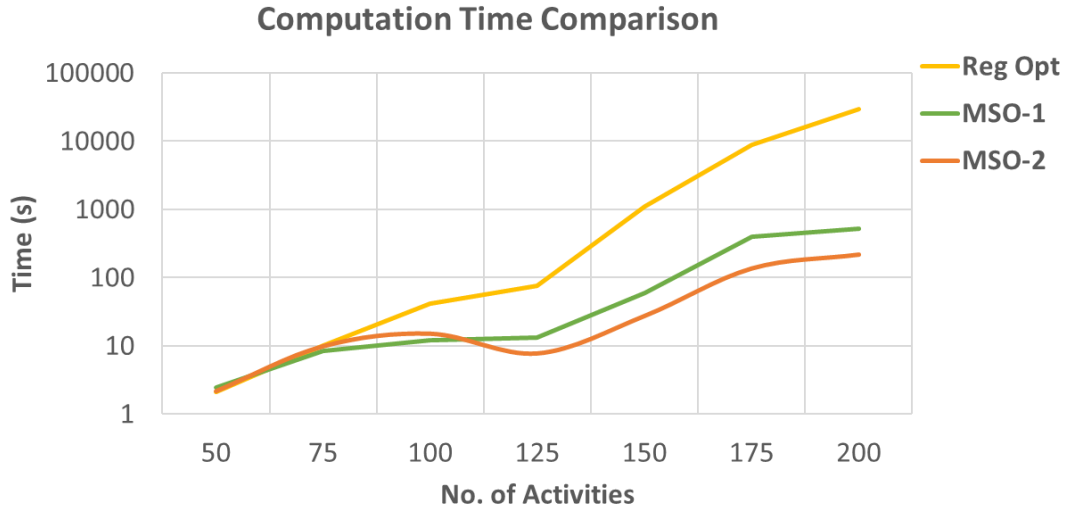


Figure 29. Computation times comparison between Regular Optimization, MSO-1 and MSO-2

5.2.1 Multi-step Optimization-2 vs. Regular Optimization

To analyze the efficiency and solution quality, MSO-2 was compared with the regular optimization technique. Both approaches were tested on 50 instances generated on the Dal-Randomizer. The details of the test instances are shown in Table 9.

5.2.1.1 Computation time comparison

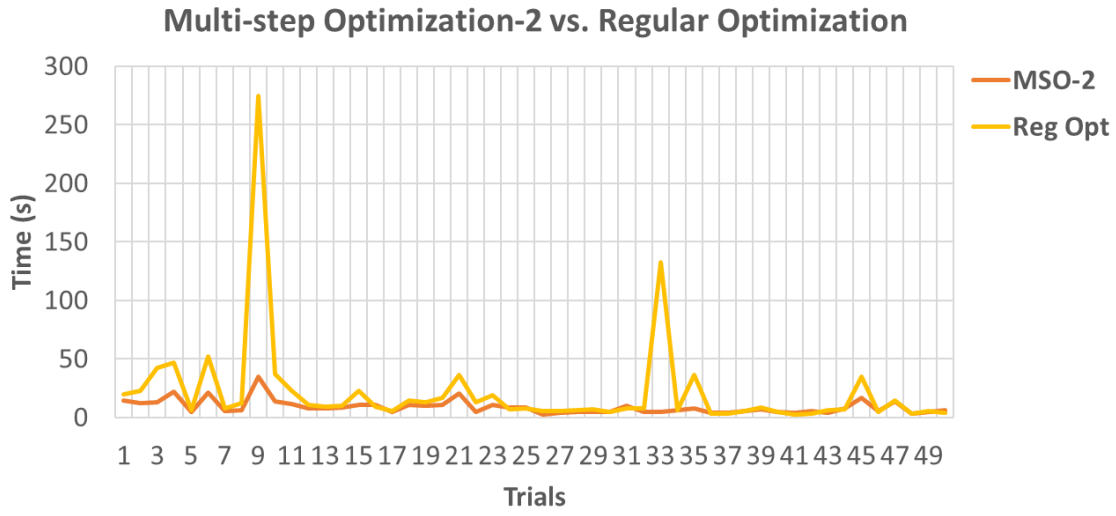


Figure 30. Computation time comparison between MSO-2 and Regular Optimization

The tests showed that MSO-2 is 21.8% $\pm$ 8.5% faster than the regular optimization at 90% CI, which is a significant improvement.

5.2.1.2 Average Priority-1 DWC comparison

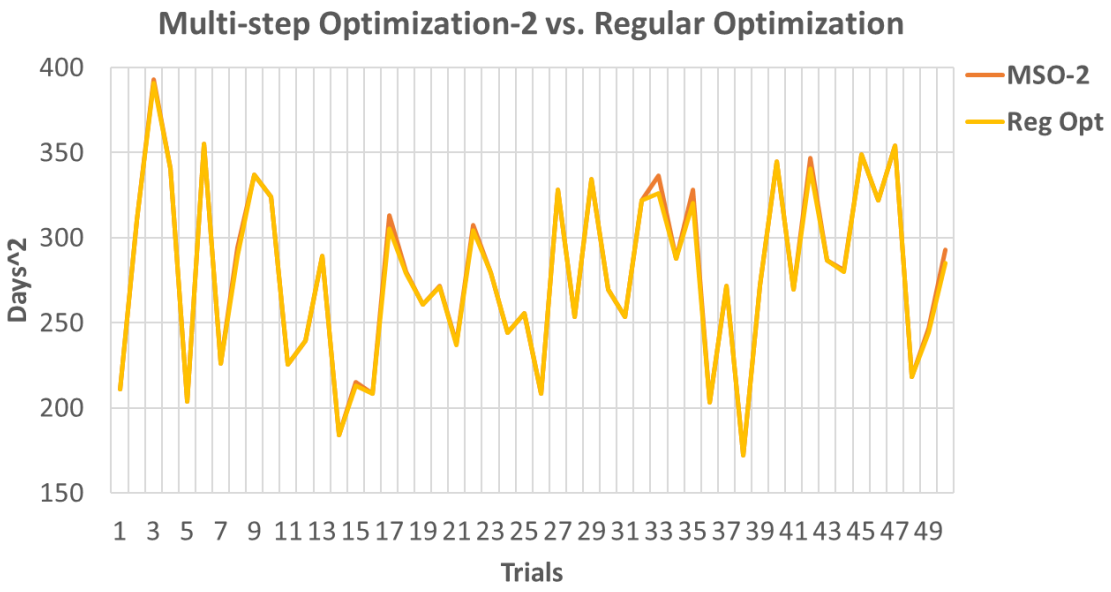


Figure 31. Average Priority-1 DWC comparison between MSO-2 and Regular Optimization

The MSO-2 gives 0.39% ± 0.2% larger average priority-1 DWC value than regular optimization at 90% CI. The result shows that the solution obtained from the regular optimization is better when compared to the matheuristic solution and this is expected because a solution obtained from a pure optimization technique is always optimum. However, the interesting fact is that MSO-2 gives a solution that is just 0.39% away from the optimum while performing 21.8% faster. Therefore, a deviation of such a small magnitude would be considered negligible and "forgivable" in a practical setting.

5.2.1.3 Makespan comparison

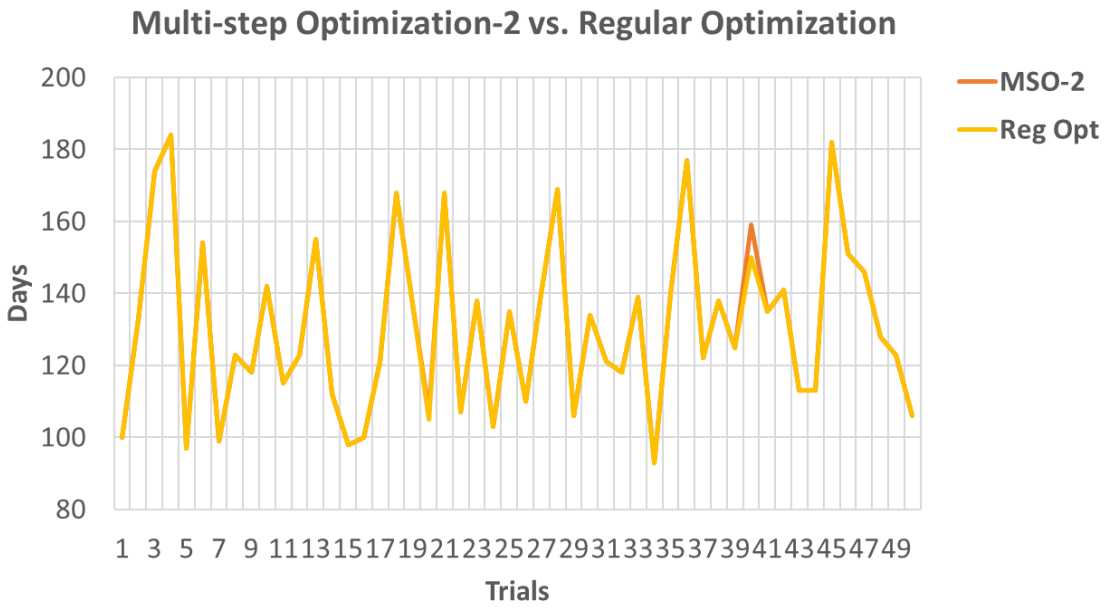


Figure 32. Makespan comparison between MSO-2 and Regular Optimization

MSO-2 gives 0.15% ± 0.2% longer makespan than regular optimization at 90% CI. Again, MSO-2, on average, gives a makespan that is just 0.15% away from the optimal value while obtaining the result 21.8% faster. This small deviation would again be negligible in a practical scenario.

From the results discussed above, it is evident that the multi-step optimization v2 can solve a large NSWPP in a much shorter time while generating results that are extremely close to the optimum.

5.3 Multi-step Optimization Version 3

A third version of the proposed multi-step optimization matheuristic MSO-3 was developed by combining the subproblem approach and MSO-2 with the aim of improving solution time for solving large NSWPP instances. Instead of considering the entire instance during every iteration, as in MSO-2, subproblems were created out of the decomposed activities and then they are solved iteratively using the priority-duration formulation.

In MSO-3, the size of the subproblem increases with every iteration as shown in Figure 33 (i.e., Iteration *i* optimizes subproblem SP_i made of *i* subgroups S_1,..,S_n).

Here, A $\supset S_1, S_2,\ldots, S_n$

$$\text{Iteration } 1: \quad \text{Min } Z = \sum_{i \in SP_1} \sum_{t=ES_i}^{LS_i} \frac{x_{i,t}}{p_i^{\theta}} . (\varepsilon + D_i)^{\alpha} . t \tag{50}$$

$$\text{Iteration } 2: \quad \text{Min } Z = \sum_{i \in SP_2} \sum_{t=ES_i}^{LS_i} \frac{x_{i,t}}{p_i^{\theta}} . (\varepsilon + D_i)^{\alpha} . t \tag{51}$$

……..

$$\text{Iteration } n: \quad \text{Min } Z = \sum_{i \in SP_n} \sum_{t=ES_i}^{LS_i} \frac{x_{i,t}}{p_i^{\theta}} . (\varepsilon + D_i)^{\alpha} . t \tag{52}$$

The constraints remain the same as before.

$$\sum_{t=ES_j}^{LS_j} tx_{j,t} \geq \sum_{t=ES_i}^{LS_i} tx_{i,t} + D_i \qquad \forall (i,j) \in P = SP_n \quad (53)$$

$$\sum_{i=1}^{n} b_{i,k} \sum_{\tau=\max(ES_i, t-D_i+1)}^{\min(LS_i, t)} x_{i,\tau} \leq B_k \qquad \forall t \in H, \forall i \in A, \forall k \in R \quad (54)$$

$$\sum_{t=ES_i}^{LS_i} x_{i,t} = 1 \qquad \forall\, i \in A \quad (55)$$

$$x_{i,t} \in \{0,1\} \qquad \forall\, i \in A, \forall t \in H \quad (56)$$

The following describes the steps required to implement MSO-3 with the help of the arbitrary instance shown in Figure 20.

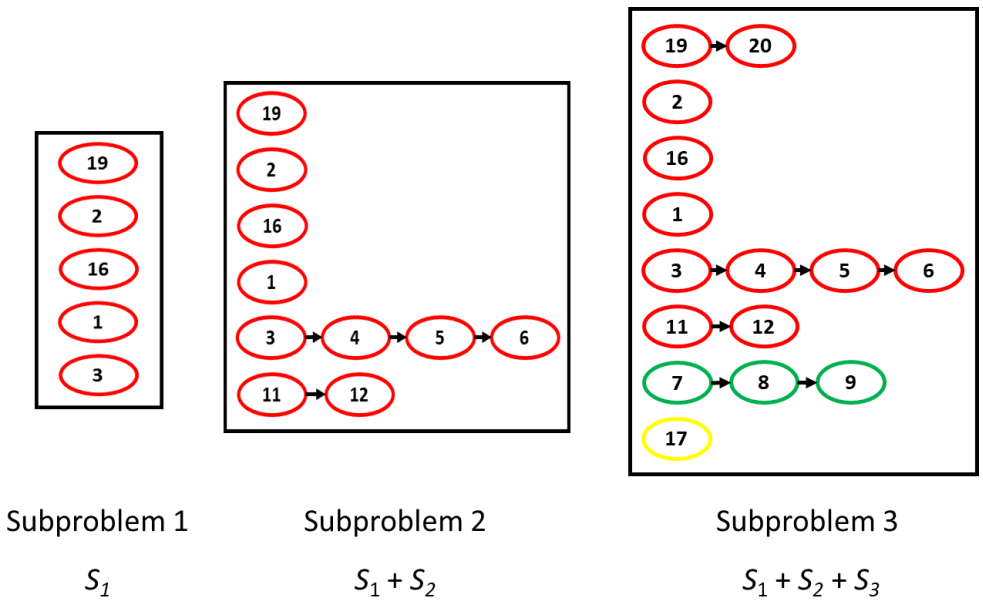


Figure 33. Representation of MSO-3

Step 1: Sort the activities on three levels (priority, ES, & duration) as in MSO-2 and create subgroups of the desired size. For experimentations with MSO-3, the size of the subgroup is maintained at 50, but other sizes can be used.

Step 2: Create subproblem-1 with information on the precedence pairs, processing duration, resource availability, and resource demand for the activities belonging to subgroup-1.

Step 3: Optimize subproblem-1 and at the end of the first iteration, fix the time window of the activities in subproblem-1 to their start times from the solution.

Step 4: Create the next subproblem (i.e., subproblem-2) with information on the precedence pairs, processing duration, resource availability, and resource demand for the activities belonging to subgroup-1 and subgroup-2. Since the time window of the activities in subgroup-1 gets fixed, it will not add to the computational time of subproblem-2. Also, the presence of subgroup-1 ensures that when subgroup-2 is being optimized, the precedence relationships are not violated.

Step 5: Optimize subproblem-2 and, at the end of the second iteration, fix the time window of the activities in subproblem-2 to their start times from the solution.

Step 6: Repeat Steps 4–5 above for each of the remaining subgroups adding one subgroup at a time.

It is essential to add the activities from the previous subproblem when a new subproblem is created in order to satisfy the precedence constraints. When the activities are sorted on three levels and decomposed, as shown in Figure 27, the predecessor of an activity need

not be present in the same subgroup. If we create subproblems by just using every single subgroup, then we get the subproblems, as shown in Figure 34, where it is evident that the activity-3 is in the subproblem-1 and its successors, i.e., activities 4,5 and 6 are present in subproblem-2. Similarly, activity-19 is present in subproblem-1 and its successor, activity-20, is in subproblem-3. In such a situation, the precedence relationship can be violated, and the succeeding activity can start before the start/completion of its predecessor if the resources are available.

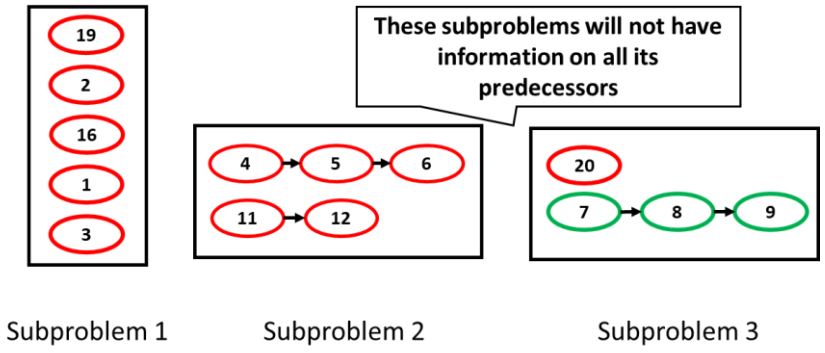
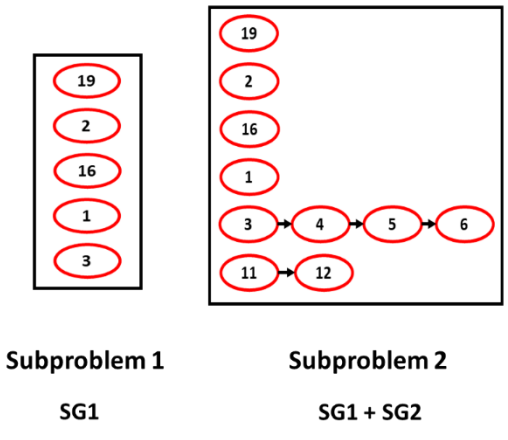


Figure 34. The network structure of the subproblems when created using only individual subgroups

The main difference between MSO-2 and MSO-3 is that while MSO-2 operates on the entire instance during every iteration by only optimizing the activities of the "in-process" subgroup, MSO-3 optimizes individual subgroups during every iteration by including the activities from the previously optimized subgroup.

Consider the arbitrary instance with twenty activities, as shown in Figure 20. The activities are sorted on three-level and decomposed into 4 subgroups with 5 activities each, as shown in Figure 27.

When using MSO-2, only the 5 activities from subgroup-1 will be considered in the objective function and optimized during the first iteration. Although the remaining 15 activities are not considered in the objective function, the solver still roughly schedules the remaining activities in any timeslot within their time window that satisfies the precedence and resource constraints. This rough scheduling of the remaining activities adds to the computation time during initial iterations. As a result, the total solve time increases. MSO-3 eliminates this process by only optimizing individual subgroups at a time. This means, only the 5 activities from subgroup-1 will be considered in the subproblem-1 and it gets optimized during the first iteration. For the second iteration, although the activities from the previously optimized subgroup (subgroup -1) is included while solving the subproblem-2, the computation time to solve the subproblem-2 will not be affected since the time window of the first 5 activities will be fixed to their start time values from the solution as shown in Figure 35. Therefore, no matter how large the problem, MSO-3 solves individual subgroups during every iteration and, in turn, solves the entire instance quickly.



Data for 1st iteration

A	D	ES	LS	p
19	20	0	29	1
2	19	0	41	1
16	15	0	45	1
1	10	0	50	1
3	3	0	37	1

Data for 2nd iteration

A	D	ES	LS	p
19	20	0	0	1
2	19	0	0	1
16	15	0	0	1
1	10	0	0	1
3	3	0	0	1
11	3	0	50	1
12	7	3	53	1
4	2	3	40	1
5	7	5	42	1
6	11	12	49	1

Figure 35. Representation of the change in data after the first iterations in the MSO-3

Figure 36 shows the improvement in computational time for using MSO-3 over MSO-2 on the instances shown in Table 10. While MSO-2 took around 4 minutes to solve an instance with 200 activities, MSO-3 solved the same instance in under 0.5 minutes.

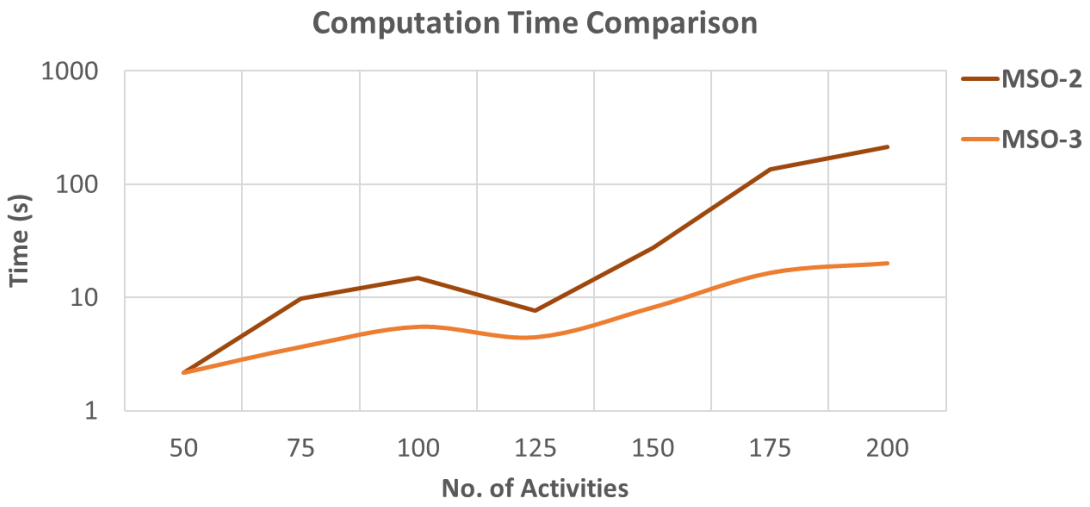


Figure 36. Computation time comparison between MSO-3 and MSO-2

To further analyze the speed and solution quality, MSO-3 was compared with regular optimization and MSO-2. All three approaches were tested on 50 instances generated on the Dal-Randomizer. The details of the test instances are shown in Table 9.

5.3.1 Computation time comparison

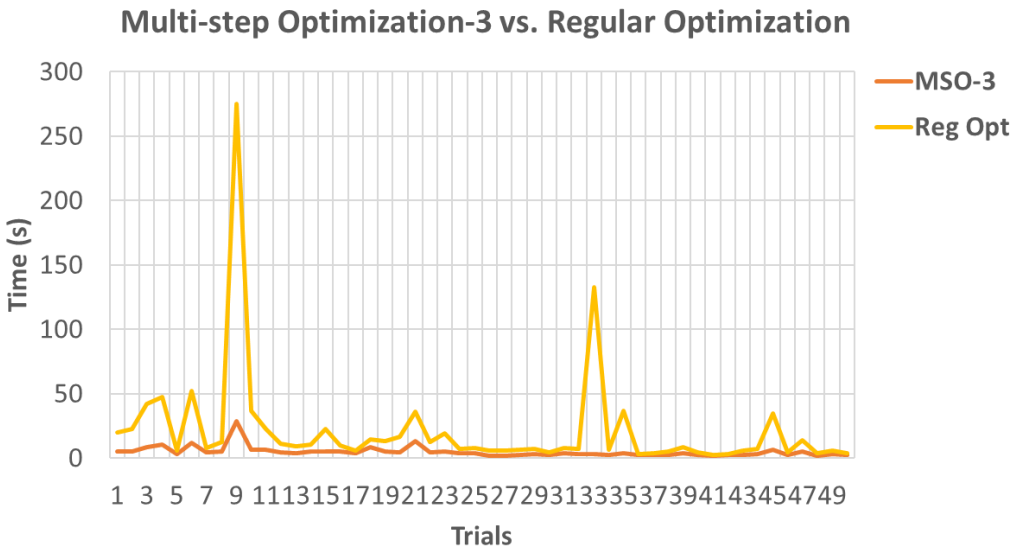


Figure 37. Computation time comparison between MSO-3 and Regular Optimization

The test showed that MSO-3 is 59% $\pm$ 4% faster than the regular optimization at 90% CI, which is a significant improvement.

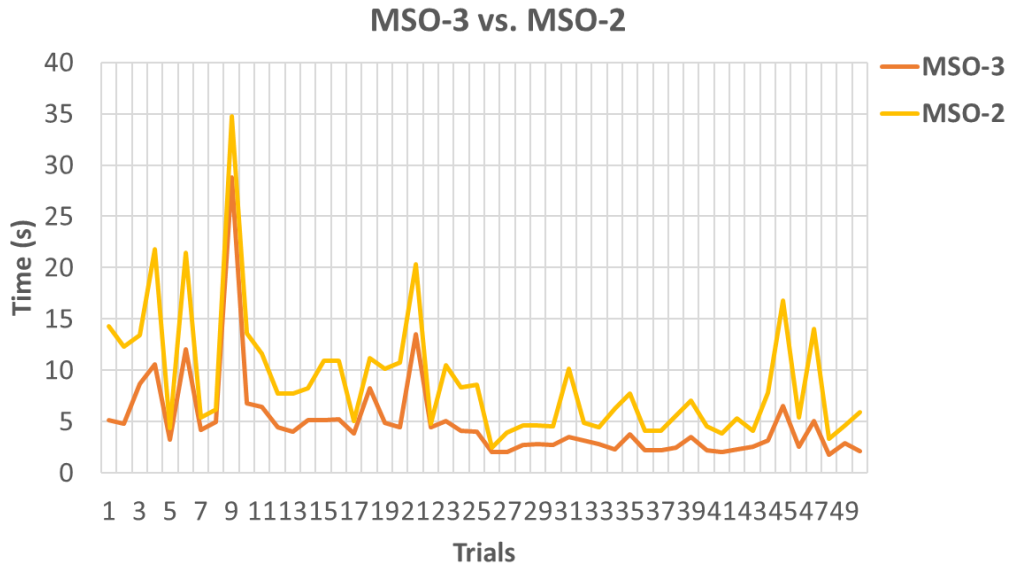


Figure 38. Computation time comparison between MSO-3 and MSO-2

MSO-3 is 45% $\pm$ 3.3% faster than MSO-2 at 90% CI, which is a significant improvement.

5.3.2 Average Priority-1 DWC and Makespan comparison

The average priority-1 DWC and makespan values produced by MSO-3 are similar to those obtained for MSO-2, as presented in Figures 31 and 32. Therefore, MSO-3 gives the average priority-1 DWC that is just 0.39% away from the optimal value and the makespan that is only 0.15% away from the optimal value while performing 59% faster than the regular optimization. Thus, by combining the benefits of the subproblem approach and MSO-2, the proposed matheuristic was made much more efficient for tackling large NSWPP instances.

5.4 Influence of network indicator on the computation time of MSO-3

This section discusses the influence of network indicators on the computational effort required to solve an instance using MSO-3. Table 9 represents the network characteristics of the Dal-Randomizer instances used for experimentation. When using MSO-3, the network indicator value changes during every iteration because the main problem is decomposed into subproblems and these subproblems have different network indicators, as shown in Figure 39.

For example, consider an instance with 100 activities being solved using MSO-3. Let's assume that the size of the subgroup is 50. When the main problem is sorted and decomposed, it gives 2 subgroups with 50 activities each. The 1st subproblem will optimize the activities in subgroup-1. The 2nd subproblem consists on activities from subgroups 1 and 2. Though, we include the activities from the 1st subproblem when solving the 2nd subproblem, it does not have any influence over the computational effort to solve the 2nd subproblem as the time window of the 1st subproblem gets fixed at the end of the iteration. The only reason why we include the activities from the 1st subproblem is to ensure that the

precedence constraints are satisfied. So technically, only 50 activities are optimized during every iteration.

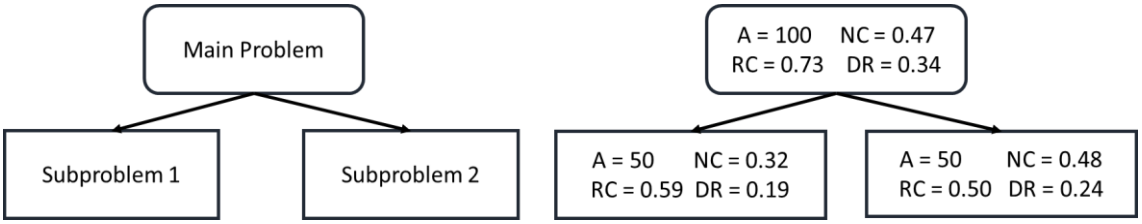


Figure 39. Changes in network characteristics when the main problem is decomposed into subproblems in MSO-3

For studying the influence of the network indicators on the computational speed, the values of NC, RC and DR were calculated for every subproblem solved when experimenting on the fifty Dal-R100 instances shown in Table 9 and later analyzed to understand the trend.

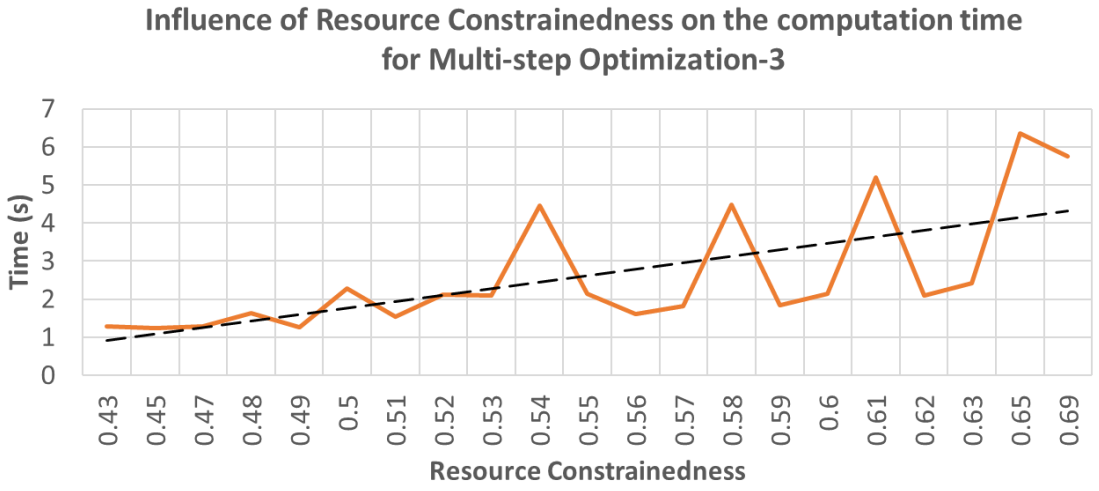


Figure 40. Influence of Resource Constrainedness on the computation time for MSO-3

From Figure 40, the average computation time shows an increasing trend with an increase in the resource constrainedness value. The results are in accordance with the chart shared by [82]. From Figure 11, the area between RC values ranging from 0.25 to 0.78 is called the "NP-hard region" or the "phase transition region" where the complexity of an instance increases and then drops which resembles a bell-curve pattern [82]. The computational

effort is the greatest when RC value is between 0.5 - 0.73. [82] also mentioned that though the computational effort is greatest in the phase transition region (denoted as NP-hard region in Figure 11), the instance need not be computationally difficult since the variance is also high in that region [82]. This explains the rise and fall in computational time even when RC value is within the "NP-hard region". In addition, the network complexity (NC) value also keeps changing during every iteration and it also has an influence over the computational time.

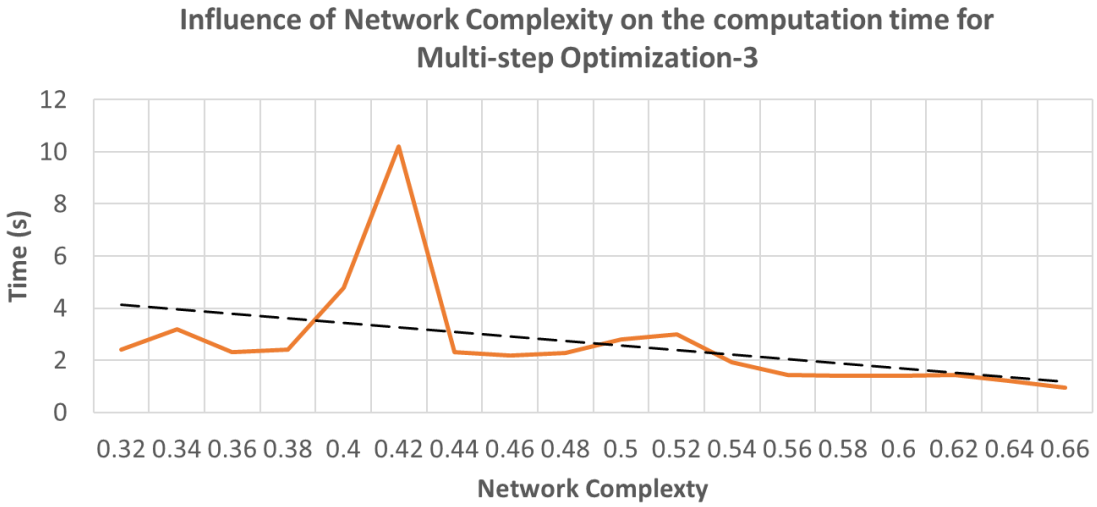


Figure 41. Influence of Network Complexity on the computation time for MSO-3

From Figure 41, the average computation time shows a decreasing trend with an increase in the network complexity value. A high network complexity value denotes a high number of precedence pairs. If the number of precedence pair is high, then the computational effort required would be less because of the smaller solution space. Since other contributing factors, such as the RC value, keeps changing with every iteration, a rise in computational time could be seen even when the NC value is high.

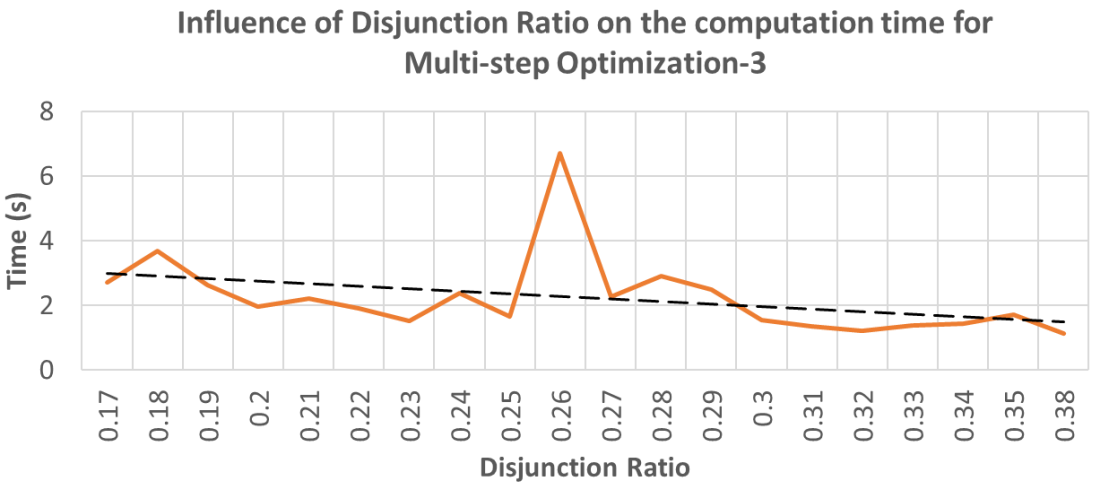


Figure 42. Influence of Disjunction Ratio on the computation time for MSO-3

From Figure 42, the average computational time shows a decreasing trend with an increase in the DR value. Since DR is the product of RC and NC and because of high variance of RC on its influence over the solution time, a rise in computational time could be seen even when the DR value is high.

5.5 SSGS vs. MSO-3

The results from experimentation on MSO-3 shows that it is capable of producing near-optimal solution for NSWPP instances. Since it is still a heuristic, it is important to compare MSO-3 with existing state-of-the-art heuristics to study its benefits.

The most commonly used heuristic for solving RCPSP is the serial schedule generation scheme (SSGS). The SSGS is a scheduling algorithm that makes use of the activity increment approach to schedule each project activity at the earliest possible time within their precedence and resource constraints [91]. The quality of the schedule generated by the SSGS depends on the sequence of activity in the given data because it moves along the list from the top to the bottom while selecting and scheduling the next activity in the list at every step [91].

5.5.1 Experiment 1

In this experiment, a comparative test for average priority-1 DWC and makespan was conducted between SSGS and MSO-3 on instances from the Dal-Randomizer. The test was conducted on 5 sets of 10 instances each, with the number of activities in each set increasing in increments of 100 (set 1 consists of 10 instances with 100 activities, set 2 consists of 10 instances with 200 activities, etc.) The details of the test instances are shown in Table 11.

	Dal-R 100	Dal-R 200	Dal-R 300	Dal-R 400	Dal-R 500
|A|	100	200	300	400	500
|R|	51	51	51	51	51
T	200	275	400	450	475
NC	0.44-0.56	0.45-0.54	0.47-0.52	0.47-0.51	0.46-0.54
RC	0.67-0.81	0.79-0.83	0.81-0.83	0.83	0.83
DR	0.30-0.41	0.36-0.44	0.39-0.43	0.39-0.43	0.38-0.45
PR	20	20	20	20	20
Instances	10	10	10	10	10

Table 11. Characteristics of test instances used for the SSGS and MSO-3 comparison

Three-level sorting, as in the case of MSO-3, was employed on the activities before solving each instance in SSGS. The same instances were also solved using MSO-3 and resulting solutions were compared using a *t*-test. For solving the instances with MSO-3, the activities were broken into subgroups of 50.

5.5.1.1 Average Priority-1 DWC comparison

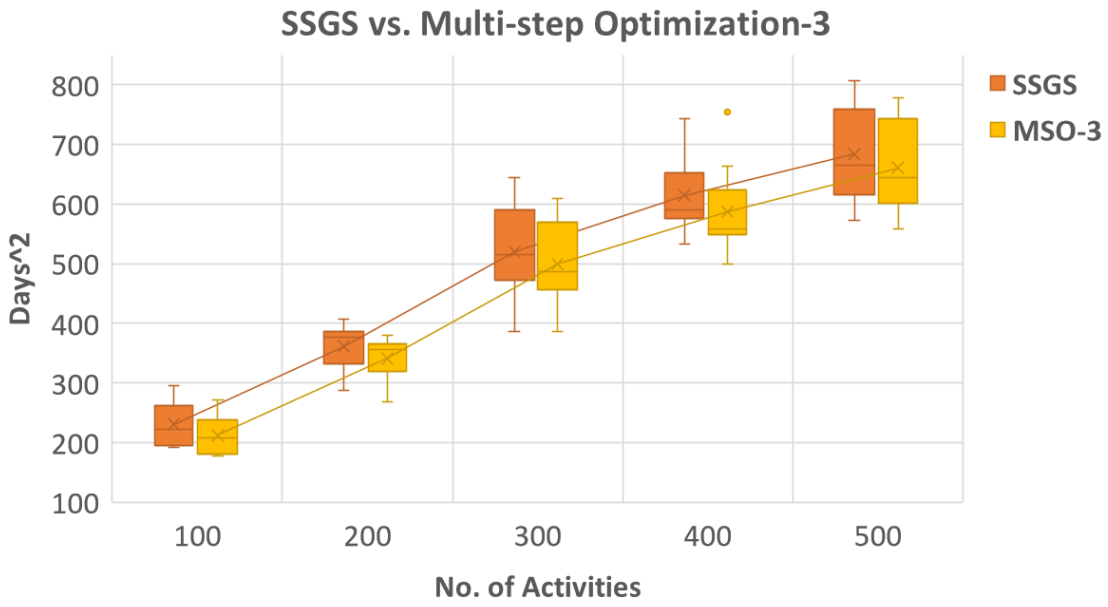


Figure 43. Average Priority-1 DWC comparison between SSGS and MSO-3

Avg Pri-1 DWC	Mean(%)	Margin Of Error(%)	Better	Significant
100 Activities	8.6	2	MSO-3	Yes
200 Activities	6	1.6	MSO-3	Yes
300 Activities	4.2	1.4	MSO-3	Yes
400 Activities	4.9	1.6	MSO-3	Yes
500 Activities	3.5	0.9	MSO-3	Yes

Table 12. Percentage mean difference and margin of error for Average Priority-1 DWC between SSGS and MSO-3

From Figure 43, it is evident that the average priority-1 DWC value given by the MSO-3 is always statistically better when compared to SSGS irrespective of the instance size. This is because the MILP model searches the solution space in-depth when compared to the heuristics.

Table 12 reads as follows: For instances with 100 activities, the SSGS gives 8.6% $\pm$ 2% larger average priority-1 DWC value when compared to MSO-3 at 90% CI.

5.5.1.2 Makespan comparison

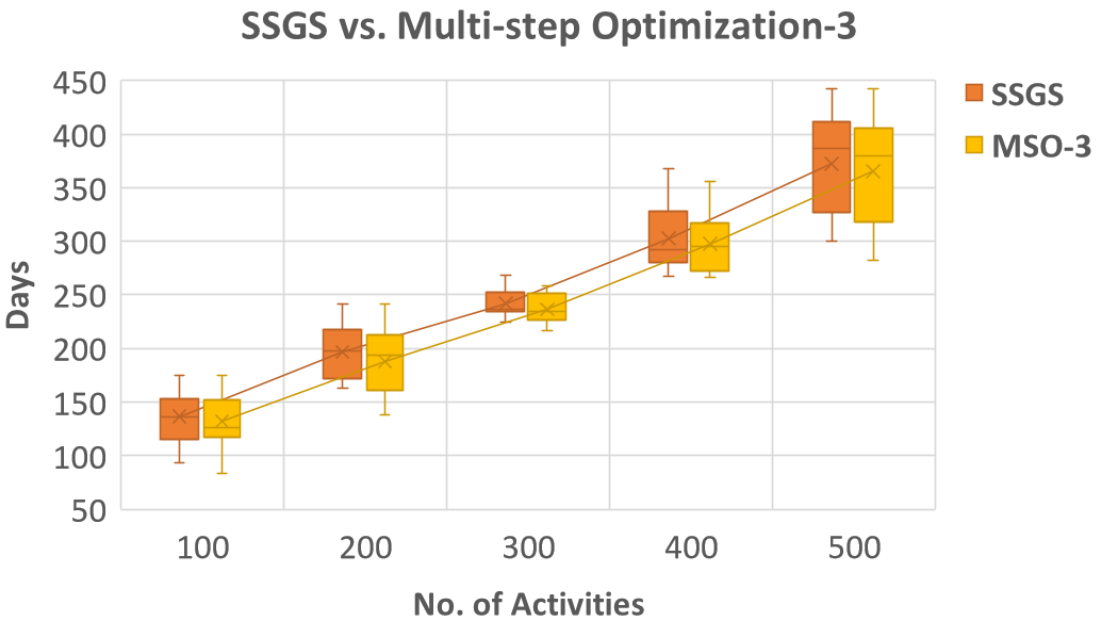


Figure 44. Makespan comparison between SSGS and MSO-3

Makespan	Mean(%)	Margin Of Error(%)	Better	Significant
100 Activities	4	3.5	MSO-3	Yes
200 Activities	5.4	3.5	MSO-3	Yes
300 Activities	2.4	1.8	MSO-3	Yes
400 Activities	1.7	2.1	MSO-3	No
500 Activities	2.1	1.4	MSO-3	Yes

Table 13. Percentage mean difference and margin of error for makespan between SSGS and MSO-3

Makespan	Mean (Days)	Max (Days)
100 Activities	4.5	15
200 Activities	8.9	27
300 Activities	5.4	14
400 Activities	5	18
500 Activities	6.8	18

Table 14. Mean and Maximum difference in makespan between SSGS and MSO-3

Table 13 shows the percentage mean difference and margin of error for makespan between SSGS and MSO-3. Though the results are not statistically significant in all cases, the SSGS, on average, gives a longer makespan in comparison to MSO-3 irrespective of the instance size, as shown in Table 14.

5.5.1.3 Computational Time for Multi-Step Optimization Version 3

The computational speed of a heuristic like SSGS is not surprising. The instances with 500 activities were solved in under 3 seconds with the SSGS. The interesting part was the speed at which the MSO-3 was able to solve the same instances while generating better quality solutions. From Figure 45, the instances with 500 activities were solved in under 35 seconds on average. This shows that the multi-step optimization can be used to schedule large projects quickly.

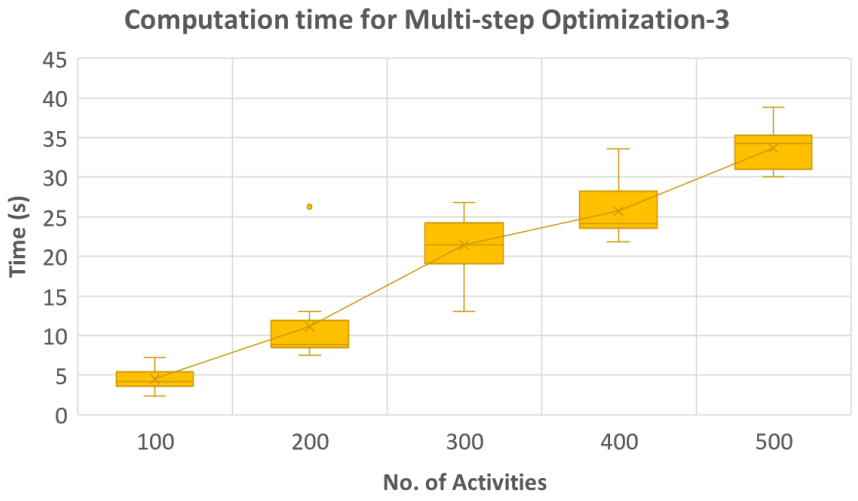


Figure 45. The computation time for MSO-3

The project leaders and schedulers consider it acceptable to wait for 1-2 minutes for an automatically generated schedule for a large NSWPP project [1], [20]. Therefore, even though MSO-3 is not as fast as the SSGS, it is still beneficial as it will produce a solution of better quality than the SSGS within the permissible time.

5.5.2 Experiment 2

This section explains the influence of the size of the subgroup on the computational speed and solution quality for MSO-3. When using MSO-3, if the size of the subgroup is small, then it leads to a poor quality solution where the higher priority activities will not be

effectively scheduled. If the subgroup is very large, then the quality of the solution will improve, but the computational effort required to solve the problem increases significantly.

Figures 46 and 47 show that, as the size of the subgroup increases, the average priority-1 DWC value improves but at the cost of computational efficiency. The improvement is very much prominent when the size of the problem is large.

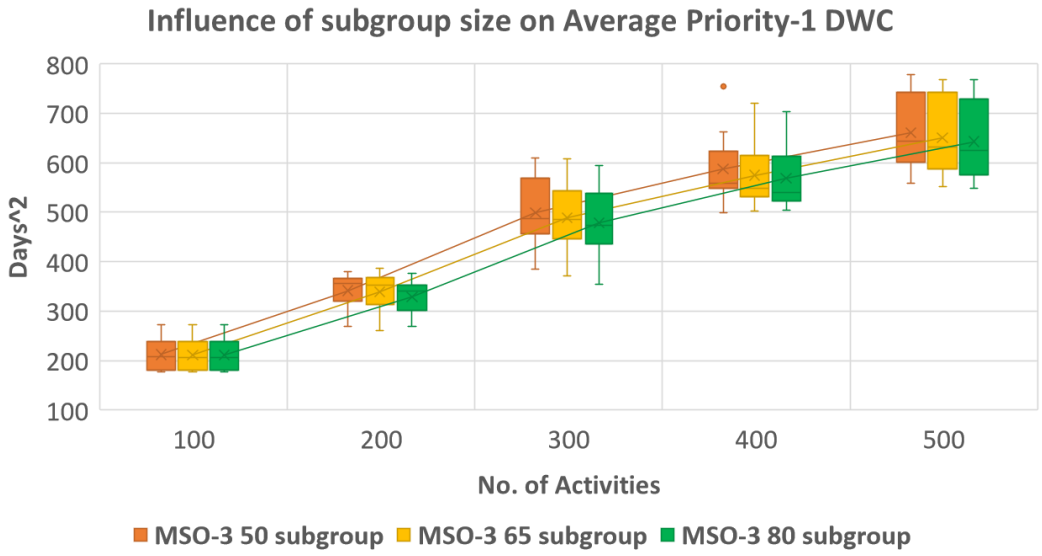


Figure 46. Influence of subgroup size on Average Priority-1 DWC for MSO-3

Average priority 1 DWC comparison between 50 & 65 subgroup size				
	Mean(%)	Margin Of Error(%)	better	Significant
100 Activities	0.40	0.70	65 SG	No
200 Activities	0.74	1.13	65 SG	No
300 Activities	2.15	1.14	65 SG	Yes
400 Activities	2.14	1.09	65 SG	Yes
500 Activities	1.75	0.77	65 SG	Yes

Average priority 1 DWC comparison between 50 & 80 subgroup size				
	Mean(%)	Margin Of Error(%)	better	Significant
100 Activities	0.40	0.70	80 SG	No
200 Activities	3.79	1.42	80 SG	Yes
300 Activities	4.39	1.51	80 SG	Yes
400 Activities	3.19	1.57	80 SG	Yes
500 Activities	2.95	0.78	80 SG	Yes

Average priority 1 DWC comparison between 65 & 80 subgroup size				
	Mean(%)	Margin Of Error(%)	better	Significant
100 Activities	0.00	0.00	Equal	No
200 Activities	3.05	1.62	80 SG	Yes
300 Activities	2.19	0.81	80 SG	Yes
400 Activities	1.02	0.76	80 SG	Yes
500 Activities	1.19	0.60	80 SG	Yes

Table 15. Percentage mean difference and margin of error for Average Priority-1 DWC for different subgroup sizes

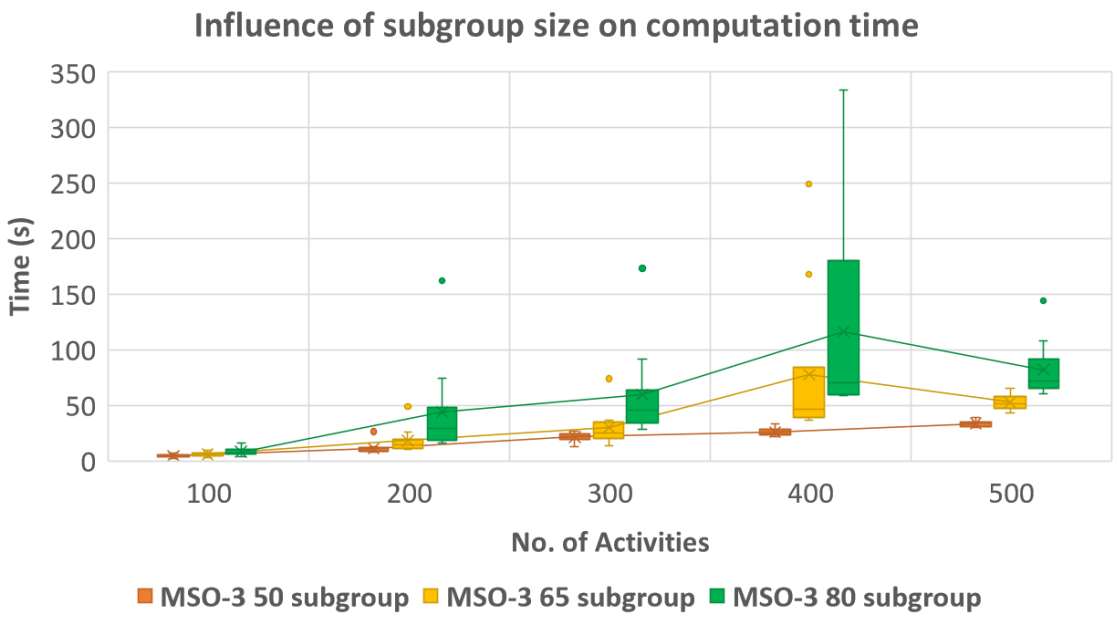


Figure 47. Influence of the subgroup size on the computation time for MSO-3

Average percentage increase in computation time for MSO-3 with an increase in the subgroup size			
	50 to 65 subgroup	**65 to 80 subgroup**	**50 to 80 subgroup**
100 Activities	38.56	37.06	88.72
200 Activities	58.23	131.89	252.23
300 Activities	35.60	111.45	174.12
400 Activities	187.43	64.83	328.38
500 Activities	55.94	53.60	140.40

Table 16. Average percentage increase in computation time for MSO-3 with an increase in subgroup size

The computational time for instances with 400 activities shows a more significant increase than any other instances, even higher than the 500 activity instances when the subgroup size is increased. The reason for this is firstly, due to small sample size, a high variance in computation time is seen for 400 activity instances and secondly, the increase is because of the characteristics of the subproblems. As mentioned in Section 5.4, the resource constrainedness, the number of precedence pairs and other factors keep changing with every iteration. So, even though we are optimizing only 50, 65 or 80 activities at a time, the computational time can vary greatly.

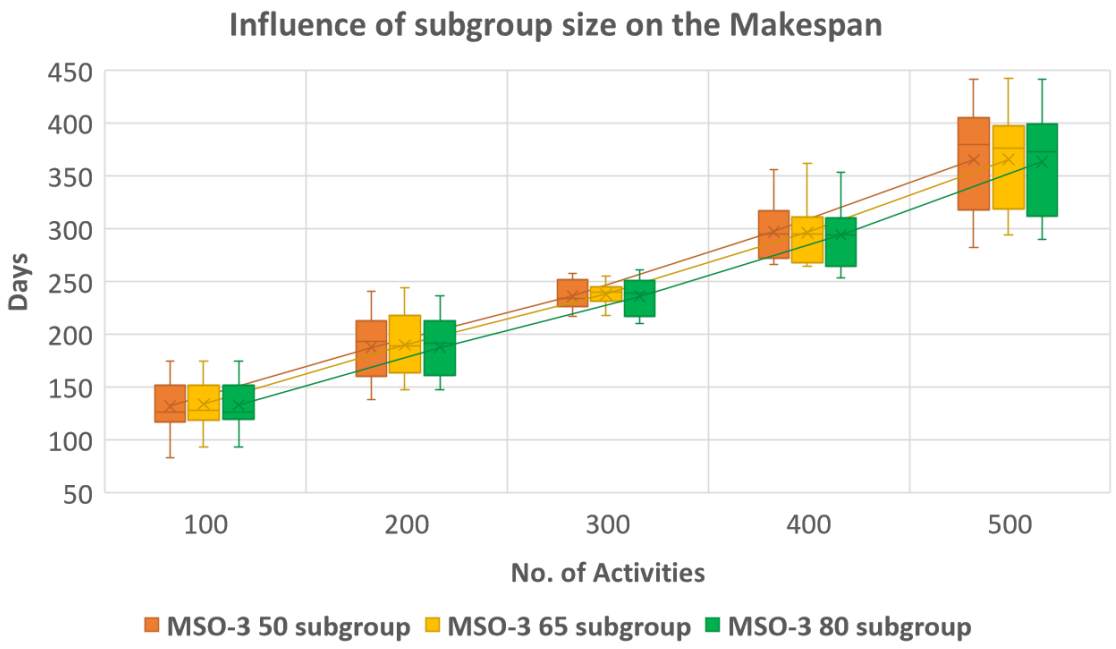


Figure 48. Influence of subgroup size on the Makespan for MSO-3

Makespan comparison between 50 & 65 subgroup size				
	Mean(%)	Margin Of Error(%)	better	Significant
100 Activities	1.76	2.44	50 SG	No
200 Activities	1.09	2.59	50 SG	No
300 Activities	0.68	2.06	50 SG	No
400 Activities	0.35	1.19	65 SG	No
500 Activities	0.22	1.09	50 SG	No

Makespan comparison between 50 & 80 subgroup size				
	Mean(%)	Margin Of Error(%)	better	Significant
100 Activities	1.35	2.16	50 SG	No
200 Activities	0.14	1.98	50 SG	No
300 Activities	0.25	2.03	80 SG	No
400 Activities	1.29	2.89	80 SG	No
500 Activities	0.53	1.06	80 SG	No

Makespan comparison between 65 & 80 subgroup size				
	Mean(%)	Margin Of Error(%)	better	Significant
100 Activities	0.50	2.02	80 SG	No
200 Activities	1.04	1.93	80 SG	No
300 Activities	1.02	2.26	80 SG	No
400 Activities	0.95	2.75	80 SG	No
500 Activities	0.75	0.56	80 SG	Yes

Table 17. Percentage mean difference and margin of error for Makespan between different sized subgroups with an increase in instance size

For the makespan, however, as the size of the subproblem increases, the makespan can extend, reduce, or remain the same. Since reducing makespan is not our primary objective, when the higher priority activities get scheduled at the earliest, the lower priority activities get pushed a little further. From Table 17, though the makespan can get extended, the changes are not significant because of high variability.

Since the goal in an NSWPP is to obtain a good quality solution quickly and be able to do frequent rescheduling when required due to the high uncertainty in the naval environment, subgroup size 50 was used for the experimentation as it offered a better speed/quality trade-off.

5.5.3 Experiment 3

The last experiment uses Gurobi's capability of accepting an initial solution to generate a feasible incumbent objective function within a couple of seconds and then improve the solution further to reach the optimum eventually. In this case, an initial solution for the given instance generated using the SSGS is sent to Gurobi along with the LP file to be solved. This method is labelled SSGS+optimizer.

In this experiment, the solution quality of SSGS+optimizer and MSO-3 were compared after a time limit of 600 seconds. For testing, 18 Dal-Randomizer instances with an average of 512 activities and 53 resource types were considered. The details of the test instances are shown in Table 18. The network structure of instances used for this experiment was similar to that shared by Thales, as shown in the Figure 49. Since these instances had a higher network complexity, a larger subgroup of 150 activities was selected to experiment with MSO-3, as it would be beneficial to get a near-optimal solution without overly increasing the computational time. In addition, while testing MSO-3, a maximum time limit of 300 seconds was enforced for every iteration.

	Dal-R
|A|	502 - 526
|R|	53
T	200
NC	1.84 - 2.19
RC	0.21 - 0.22
DR	0.40 - 0.47
PR	10
Instances	18

Table 18. Characteristics of test instances used for comparing SSGS+Optimizer and MSO-3

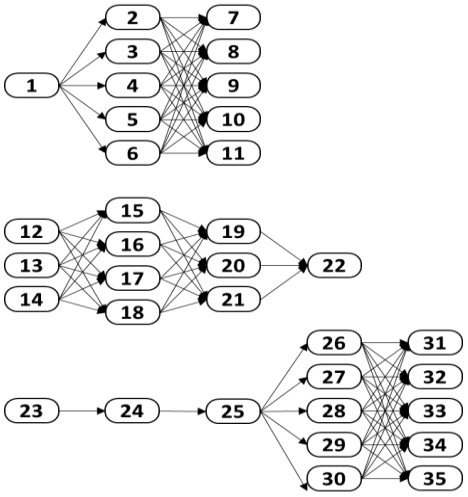


Figure 49. The network structure of the Thales shared test instances

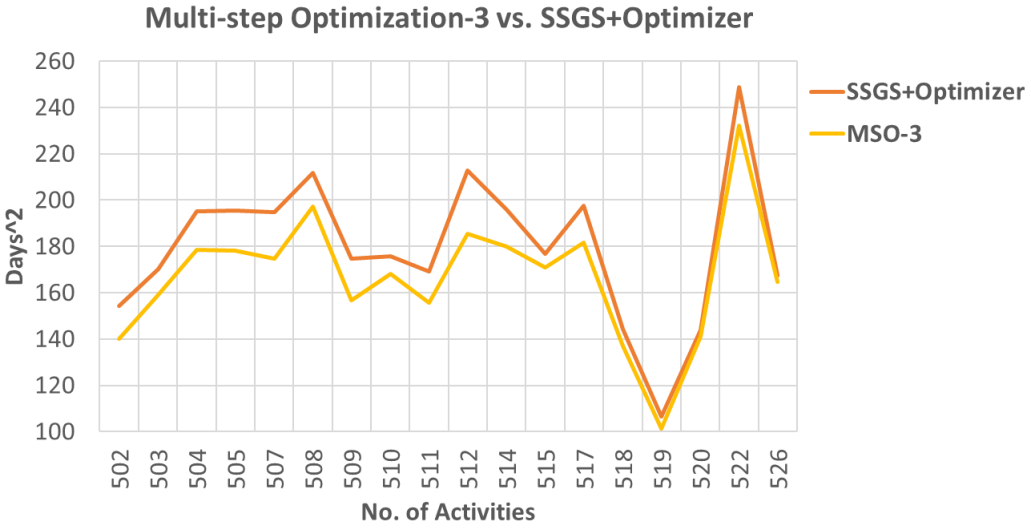


Figure 50. Average Priority-1 DWC comparison between MSO-3 and SSGS+Optimizer

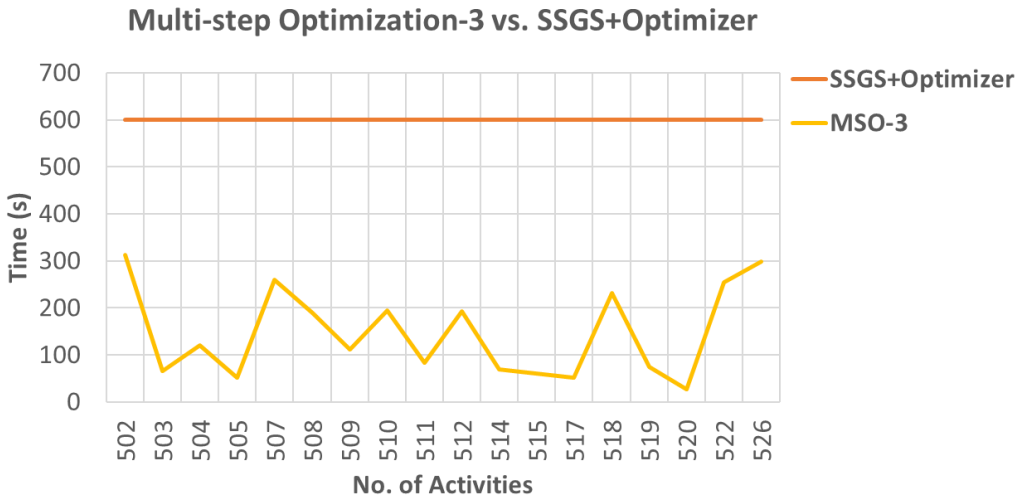


Figure 51. Computation time comparison between MSO-3 and SSGS+Optimizer

The experimental results show that MSO-3 gives 7% $\pm$ 1.2% smaller priority-1 DWC value when compared to SSGS+optimizer while performing 76% $\pm$ 6.5% faster at 90% CI. The results show that even without an initial solution, the MSO-3 can perform very efficiently and deliver a much better solution than the state-of-the-art heuristics.

Chapter 6: Conclusion and Future Research

Generating the baseline schedule, managing the activities, and controlling the risks arising from the NSWPP activities is a very onerous yet interesting venture. The industry-specific network structure, priority constraints, space constraints, and operational constraints witnessed in the NSWPP are an addition to the classic resource-constrained project scheduling problem. Using heuristic algorithms and commercial software running on heuristics to schedule such projects results in solutions that are generally far from the optimal value. In a challenging environment such as the NSWPP, a heuristic would make the project last longer by giving an overdrawn makespan.

The matheuristic method proposed in this paper helps to intelligently schedule the NSWPP activities in a practical way that is mandated by the naval industry. The experimental results show that the multi-step optimization method can schedule large RCPSP instances in a short time frame while generating a near-optimal solution.

Finally, the comparative test between serial schedule generation scheme (SSGS) and multi-step optimization shows that the latter provides a better-quality solution. For the makespan, while the SSGS can produce shorter makespan than multi-step optimization at times, on average, the multi-step optimization produces a shorter makespan than the SSGS.

Many potential areas for research were identified during the project, but only a few could be explored due to time constraints. The following are potential areas where the matheuristic can be extended.

The multi-step optimization could be extended to solve the multi-mode resource-constrained NSWPP project where activities can be given an option to execute in a different mode with each mode having a unique activity duration and resource demands.

Parallel to this research, [19] conducted research on rescheduling NSWPP projects using a multi-mode RCPSP model to reduce the deviation days of delayed activities with regards to the baseline schedule. It would be interesting to use multi-step optimization for rescheduling the NSWPP. The single-mode multi-step optimization can be used to generate an aggregated schedule for the entire project, which will act as a baseline. Later during the project monitoring and control phase, the multi-mode multi-step optimization can be integrated with Bertrand's re-scheduling formulation to re-schedule a small group of activities that will execute in the near future, say around 50. This approach could optimally assign overtime to the activities that have the biggest impact on the baseline schedule and prevents change to the start time of all the activities in the schedule.

The priority-duration formulation may be tested for varying resource availability. This is an interesting topic under the RCPSP rescheduling policy. Not only does the varying resource availability bear the tendency to break the baseline schedule, but there can also be situations where it can help to improve the baseline schedule by planning the activities to execute early due to an increase in resource availability.

Bibliography

[1] E. L. J. Bertrand, Interviewee, *Interview with Royal Canadian Navy officer.* [Interview]. May 2019.

[2] J. MacLean, Interviewee, *Interviewee, Interview with Thales Head ERP Manager.* [Interview]. May 2019.

[3] L. Bianco, A. Mingozzi, V. Maniezzo and S. Ricciardelli, "An Exact Algorithm for the Resource Constrained Project Scheduling Problem Based on a New Mathematical Formulation," *Management Science,* vol. 44, no. 5, pp. 714-729, 1998.

[4] E. Croteau, "An On-Event Based Model for Resource Constrained Scheduling of Aircraft," Thesis, Dalhousie University, Halifax, Nova Scotia, 2015.

[5] J. Blazewicz, J. Lenstra and A. RinnooyKan, "Scheduling Subject to Resource Constraints: Classification and Complexity," *Discrete Applied Mathematics,* vol. 5, no. 1, pp. 11-24, 1983.

[6] C. Archetti and M. Speranza, "A survey on matheuristics for routing problems," *European Journal of Operational Research,* vol. 2, pp. 223-246, 2014.

[7] A. Augimeri, "Local Search with SubProblem Exact Resolution: Application to a real Resource-Constrained Project Scheduling Problem," Thesis, Polytechnic University of Turin, Turin, 2019.

[8] P. Baumann and N. Trautmann, "Resource-Constrained Project Scheduling with Project Management Information Systems," *Handbook on Project Management and Scheduling,* vol. 2, pp. 1385-1400, 2015.

[9] N. Rodríguez, A. Gupta, P. Zabala and G. Cabrera-Guerrero, "Optimization Algorithms Combining (Meta)heuristics and Mathematical Programming and Its Application in Engineering," *Mathematical Problems in Engineering,* pp. 1-3, 2018.

[10] "what is project management?," 2019. [Online]. Available: https://www.pmi.org/about/learn-about-pmi/what-is-project-management. [Accessed November 2019].

[11] Project Management Academy, "The Five Traditional Process Groups Explained," 2019. [Online]. Available: https://projectmanagementacademy.net/articles/five-traditional-process-groups/. [Accessed November 2019].

[12] Master of project academy, "24 Steps of Project Planning Process | What Are the Planning Process Group Activities," 2019. [Online]. Available: https://blog.masterofproject.com/project-planning-process-group-activities/. [Accessed November 2019].

[13] M. Vanhoucke, "Resource constrained project scheduling: What is my scheduling objective?," June 2016. [Online]. Available: http://www.pmknowledgecenter.com/baseline/dynamic-scheduling-introduction-to-baseline-scheduling. [Accessed September 2019].

[14] P. Brucker, A. Drexl, R. Mohring, K. Neumann and E. Pesch, "Resource-constrained project scheduling: Notation, classification, models, and methods," *European Journal of Operational Research,* vol. 112, no. 1, pp. 3-41, 1999.

[15] S. Hartmann and D. Briskorn, "A survey of variants and extensions of the resource-constrained project scheduling problem," *European Journal of Operational Research,* vol. 207, no. 1, pp. 1-14, 2010.

[16] F. Habibi, F. Barzinpour and S. Sadjadi, "Resource-constrained project scheduling problem: review of past and recent developments," *Journal of Project Management,* pp. 55-88, 2018.

[17] M. Vanhoucke, "Dynamic scheduling: An introduction to baseline scheduling," June 2012. [Online]. Available: http://www.pmknowledgecenter.com/baseline/dynamic-scheduling-introduction-to-baseline-scheduling. [Accessed May 2019].

[18] J. Couch, "Performance Robust Project Scheduling Polices for Naval Ship Maintenance," Thesis, Dalhousie University, Halifax, Nova Scotia, 2016.

[19] E. Bertrand, "Optimization of the Naval Surface Ship Resource-Constrained Project Scheduling Problem," Unpublished, Thesis, Dalhousie University, Halifax, Nova Scotia, 2020.

[20] P. Jeffery, Interviewee, *Interview with Thales Project Manager.* [Interview]. May 2019.

[21] United States Department of Labour, "Occupational Safety and Health Administration," 2019. [Online]. Available: https://www.osha.gov/SLTC/etools/shipyard/shiprepair/sr_index.html. [Accessed October 2019].

[22] S. Denpreuktham, "Would anything happen if you stand in front of a powerful military radar at the point when its strength is the greatest?," July 2017. [Online]. Available: https://www.quora.com/Would-anything-happen-if-you-stand-in-front-of-a-powerful-military-radar-at-the-point-when-its-strength-is-the-greatest. [Accessed October 2019].

[23] A. Kastor and K. Sirakoulis, "The effectiveness of resource levelling tools for resource constriant project scheduling problem," *International Journal of Project Management,* vol. 27, pp. 493-500, 2009.

[24] R. Kolisch, "Resource Allocation Capabilities of Commercial Project Management Software Package," *Interfaces,* vol. 29, no. 4, pp. 19-31, 1999.

[25] P. Weaver, "A Brief History of Scheduling," in *myPrimavera Conference*, Hyatt, Canberra, 2006.

[26] M. R. Walker and J. Kelley, "Critical-path planning and scheduling," in *Eastern Joint IRE-AIEE-ACM Computer Conference*, Boston, Massachusetts, 1959.

[27] D. Malcolm, J. H. Roseboom, C. E. Clark and W. Fazar, "Application of a Technique for Research and Development Program Evaluation," *Operations Research,* vol. 7, no. 5, pp. 646-669, 1959.

[28] F. Dilmaghani, "Critical Chain Project Management (CCPM) at Bosch Security Systems (CCTV) Eindhoven," Thesis, University of Twente, Enschede, The Netherlands, 2008.

[29] C. Carone, "The Differences Between CPM and Resource-Loaded Scheduling and How They Applied to The Martinez Tesoro Refinery Flare Header Replacement Project," California Polytechnic State University, San Luis Obispo, California, 2017.

[30] M. Matthews, "Resource scheduling: incorporating capacity into schedule construction," *Project Management Journal,* vol. 25, no. 2, pp. 44-54, 1994.

[31] C. Schwindt and J. Zimmermann, "Handbook on project management and sheduling vol.2," in *International Handbooks on Information Systems*, Springer International Publishing, 2015, pp. 1-1406.

[32] T. Guldemond, J. Hurink, J. Paulus and J. Schutten, "Time-constrained project scheduling," *Journal of Scheduling,* vol. 11, no. 2, pp. 137-148, 2008.

[33] J. Weglarz, "Project Scheduling with Continuously-Divisible, Doubly Constrained Resource," *Management Sciences,* vol. 27, no. 9, pp. 1040-1053, 1981.

[34] A. Moukrim and J. Carlier, "Storage Resources," in *Handbook on Project Management and Scheduling Vol. 1*, Springer International Publishing, 2015, pp. 177-189.

[35] E. Demeulemeester and W. Herroelen, "An efficient optimal solution procedure for the preemptive resource-constrained project scheduling problem," *European Journal of Operational Research,* vol. 90, no. 2, pp. 334-348, 1996.

[36] L. Bianco, M. Caramia and P. Dellolmo, "Solving a Preemptive Project Scheduling Problem with Coloring Techniques," in *In: Węglarz J. (eds) Project Scheduling. International Series in Operations Research & Management Science*, vol. 14, Boston, MA, Springer International Publishing, 1999, pp. 135-145.

[37] D. Debels and M. Vanhoucke, "The impact of various activity assumptions on the lead time and resource utilization of resource-constrained projects," *Computers & Industrial Engineering,* vol. 54, no. 1, pp. 140-154, 2008.

[38] S. E. Elmaghraby, Activity networks : project planning and control by network models, New york: Wiley, 1977.

[39] N. Nudtasomboon and S. Randhawa, "Resource-constrained project scheduling with renewable and non-renewable resources and time-resource tradeoffs," *Computers & Industrial Engineering,* vol. 32, no. 1, pp. 227-242, 1997.

[40] J. Grey, "Buffer Techniques For Stochastic Resource Constrained Project Scheduling With Stochastic Task Insertions Problems," Ph.D. dissertation, University of Central Florida, Orlando, Florida, 2007.

[41] M. Vanhoucke, "Dynamic Scheduling," 2012. [Online]. Available: http://www.pmknowledgecenter.com/content/dynamic-scheduling. [Accessed June 2019].

[42] F. Deblaere, E. Demeulemeester and W. Herroelen, "Reactive scheduling in multi-mode RCPSP," *Computers & Operations Research,* vol. 38, no. 1, pp. 63-74, 2011.

[43] "Integer programming," 2019. [Online]. Available: https://en.wikipedia.org/wiki/Integer_programming. [Accessed November 2019].

[44] O. Kone, C. Artigues, P. Lopez and M. Mongeau, "Comparison of mixed integer linear programming models for the resource-constrained project scheduling problem with consumption and production of resources," *Flexible Services and Manufacturing Journal,* vol. 25, no. 1, pp. 25-47, 2012.

[45] A. A. B. Pritsker, L. J. Waiters and P. Wolfe, "Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach," *Management Science,* vol. 16, no. 1, pp. 93-108, 1969.

[46] O. Kone, C. Artigues, P. Lopez and M. Mongeau, "Event-based MILP models for resource-constrained project scheduling problems," *Computers & Operations Research,* vol. 38, no. 1, pp. 3-13, 2011.

[47] "Solver," Wikipedia, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Solver. [Accessed Jan 2020].

[48] L. White, "what exactly is a "solver" in optimization?," StackExchange, 2016. [Online]. Available: https://stats.stackexchange.com/questions/241249/what-exactly-is-a-solver-in-optimization. [Accessed Jan 2020].

[49] G. Claassen, "What is the advantages of MIP model? Why the MIP model are proposed for the problems?," ReasearchGate, 2016. [Online]. Available: https://www.researchgate.net/post/what_is_the_advantages_of_MIP_model_Why_the_MIP_model_are_proposed_for_the_problems. [Accessed Feb 2020].

[50] R. Kolisch and S. Hartmann, "Heuristic Algorithms for Solving the ResourceConstrained Project Scheduling Problem: Classification and Computational Analysis," in *in: J. Weßglarz (Ed.), Handbook on Recent Advances in Project Scheduling*, vol. 1, Dordrecht, Kluwer Academic Publishers, 1999, pp. 147-178.

[51] V. Kenny, M. Nathal and S. Saldana, "Heuristic algorithms," May 2014. [Online]. Available: https://optimization.mccormick.northwestern.edu/index.php/Heuristic_algorithms. [Accessed November 2019].

[52] H. Lehtihet, "What are the differences between heuristics and metaheuristics?," July 2013. [Online]. Available: https://www.researchgate.net/post/What_are_the_differences_between_heuristics_and_metaheuristics. [Accessed November 2019].

[53] J. Patterson and E. Davis, "A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling," *Management Science,* vol. 21, no. 8, pp. 944-955, 1975.

[54] F. Boctor, "Some efficient multi-heuristic procedures for resource-constrained project scheduling," *European Journal of Operational Research,* vol. 49, no. 1, pp. 3-13, 1990.

[55] R. Kolisch, "Efficient priority rules for the resource-constrained project scheduling problem," *Journal of Operations Management,* vol. 14, no. 3, pp. 179-192, 1996.

[56] L. Ozdamar and G. Ulusoy, "A local constraint based analysis approach to project scheduling under general resource constraints," *European Journal of Operational Research,* vol. 79, no. 2, pp. 287-298, 1994.

[57] L. Ozdamar and G. Ulusoy, "An iterative local constraints based analysis for solving the resource constrained project scheduling problem," *Journal of Operations Management,* vol. 14, no. 3, pp. 193-208, 1996.

[58] R. Kolisch and A. Drexl, "Adaptive Search for Solving Hard Project Scheduling Problems," *Naval Research Logistics,* vol. 43, no. 1, pp. 23-40, 1996.

[59] A. Schirmer, "Case-Based Reasoning and Improved Adaptive Search for Project Scheduling," *Naval Research Logistics,* vol. 47, no. 3, pp. 201-222, 2000.

[60] R. Klein, "Bidirectional planning: Improving priority rule-based heuristics for scheduling resource-constrained projects," *European Journal of Operational Research,* vol. 127, no. 3, pp. 619-638, 2000.

[61] S. Diana, L. Ganapathy and A. Pundir, "Comparison of Different Priority Rules for the Resource Constrained Project Scheduling Problem," *NICMAR-Journal of Construction Management,* vol. 27, no. 2-3, pp. 27-37, 2012.

[62] "Metaheuristics," Wikipedia, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Metaheuristic. [Accessed December 2019].

[63] R. Pellerin, N. Perrier and F. Berthaut, "A survey of hybrid metaheuristics for the resource-constrained project schedulling problem," *European Journal of Operational Research,* vol. 280, pp. 395-416, 2020.

[64] J. Schulz, "Hybrid Solving Techniques for Project Scheduling Problems," Thesis, The Technical University of Berlin, Berlin, Germany, 2012.

[65] "Constraint programming," 2019. [Online]. Available: https://en.wikipedia.org/wiki/Constraint_programming#Constraint_solving. [Accessed November 2019].

[66] M. Palpant, C. Artigues and P. Michelon, "LSSPER: Solving the Resource-Constrained Project Scheduling Problem with Large Neighbourhood Search," *Annals of Operations Research,* vol. 131, pp. 237-257, 2004.

[67] T. A. M. Toffolo, H. G. Santos, M. A. M. Carvalho and J. A. Soares, "An integer programming approach to the multimode resource-constrained multiproject scheduling problem," *Journal of Scheduling,* vol. 19, no. 3, pp. 295-307, 2015.

[68] T. Wauters, J. Kinable, P. Smet, W. Vancroonenburg, G. V. Berghe and J. Verstichel, "The Multi-Mode Resource-Constrained Multi-Project Scheduling Problem," *Journal of Scheduling,* vol. 19, no. 3, pp. 271-283, 2014.

[69] R. K. Chakrabortty, A. Abbasi and M. J. Ryan, "Multi-mode resource-constrained project scheduling using modified variable neighborhood search heuristic," *International Transactions in Operational Research,* pp. 1-30, 2019.

[70] S. Fernandes and H. R. Lourenco, "Hybrids combining Local Search Heuristics with Exact Algorithm," in *Algoritmos Evolutivos y Bioinspirados*, Spain, 2007, pp. 269-274.

[71] M. A. Boschetti, V. Maniezzo, M. Roffilli and A. B. Rohler, "Matheuristics: Optimization, Simulation and Control," *Lecture Notes in Computer Science,* pp. 171-177, 2009.

[72] J. Puchingar and G. R. Raidl, "Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification," *Lecture Notes in Computer Science,* pp. 41-53, 2005.

[73] M. O. Ball, "Heuristics based on mathematical programming," *Surveys in Operations Research and Management Science,* vol. 16, pp. 21-38, 2011.

[74] L. Fortnow and S. Homer, "A Short History of Computational Complexity," *Bull. EATCS,* vol. 80, pp. 95-133, 2003.

[75] "NP-completeness," Wikipedia, 2020. [Online]. Available: https://en.wikipedia.org/wiki/NP-completeness.

[76] "Stephen Cook," wikipedia, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Stephen_Cook. [Accessed July 2020].

[77] J. Lenstra and A. H. G. Rinnooy Kan, "Complexity of Scheduling under Precedence Constraints," *INFORMS,* vol. 26, no. 1, pp. 22-35, 1978.

[78] R. Morrison, S. H. Jacobson, J. J. Sauppe and E. C. Sewell, "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning," *Discrete Optimization,* vol. 19, pp. 79-102, 2016.

[79] "Branch and bound," Wikipedia, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Branch_and_bound. [Accessed July 2020].

[80] J. Patterson, "Project Scheduling: The Effects of Problem Structure on Heuristic Performance," *Naval Research Logistics Quarterly,* vol. 23, no. 1, pp. 95-123, 1976.

[81] F. Nijs, "Project Scheduling: The Impact of Instance Structure on Heuristic Performance," Thesis, Delft University of Technology, Delft, The Netherlands, 2013.

[82] W. Herroelen and B. De Reyck, "Phase transitions in project scheduling," *Journal of the Operational Research Society,* vol. 50, no. 1, pp. 148-156, 1999.

[83] R. Kolisch, A. Sprecher and A. Drexl, "Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems," *Management Science,* vol. 41, no. 10, pp. 1693-1703, 1995.

[84] E. Demeulemeester, M. Vanhoucke and W. Herroelen, "RanGen: A Random Network Generator for Activity-on-the-Node Networks," *Journal of Scheduling,* vol. 6, no. 1, pp. 17-38, 2003.

[85] B. De Reyck and W. Herroelen, "On the use of the complexity index as a measure of complexity in activity networks," *European Journal of Operational Research,* vol. 91, no. 2, p. 347–366, 1996.

[86] S. Elmaghraby and W. Herroelen, "On the measurement of complexity in activity networks," *European Journal of Operational Research,* vol. 5, no. 4, pp. 223-234, 1980.

[87] "Gurobi Guidelines for Numerical Issues," February 2017. [Online]. Available: http://files.gurobi.com/Numerics.pdf. [Accessed July 2019].

[88] "IntFeasTol," 2019. [Online]. Available: https://www.gurobi.com/documentation/8.1/refman/intfeastol.html. [Accessed July 2019].

[89] S. Sahoo, "Univariate Analysis," July 2014. [Online]. Available: https://www.slideshare.net/drswaroopsoumya/univariate-analys. [Accessed August 2019].

[90] "90% confidence interval acceptable?," [Online]. Available: https://www.researchgate.net/post/90_Confidence_Interval_Acceptable. [Accessed November 2019].

[91] M. Vanhoucke, "Optimizing regular scheduling objectives: Schedule generation schemes," PM Knowledge Centre, 2012. [Online]. Available: http://www.pmknowledgecenter.com/dynamic_scheduling/baseline/optimizing-regular-scheduling-objectives-schedule-generation-schemes#:~:text=A%20schedule%20generation%20scheme%20determines,times%20to%20the%20project%20activities.&text=A%20serial%20schedule%2. [Accessed October 2019].

[92] R. Slowinski, "Two approaches to problems of resource allocation among project activities—a comparative study," *Journal of the Operational Research Society,* vol. 31, no. 8, pp. 711-723, 1980.

[93] D. Cooper, "Heuristics for Scheduling Resource-Constrained Projects: An Experimental Investigation," *Management Science,* vol. 22, no. 11, pp. 1186-1194, 1976.

[94] T. Pascoe, "Allocation of Resources - CPM," *Revue Francaise de Recherche Operationelle,* vol. 38, pp. 31-38, 1966.

[95] N. Christofides, R. Alvarez-Valdes and J. Tamarit, "Project scheduling with resource constraints: A branch and bound approach," *European Journal of Operations Research,* vol. 29, no. 3, pp. 262-273, 1987.

[96] B. De Reyck, W. Herroelen and E. Demeulemeester, "Algorithms for Scheduling Projects with Generalized Precedence Relations.," *International Series in Operations Research & Management Science,,* pp. 77-105, 1999.

Appendices

Appendix - 1

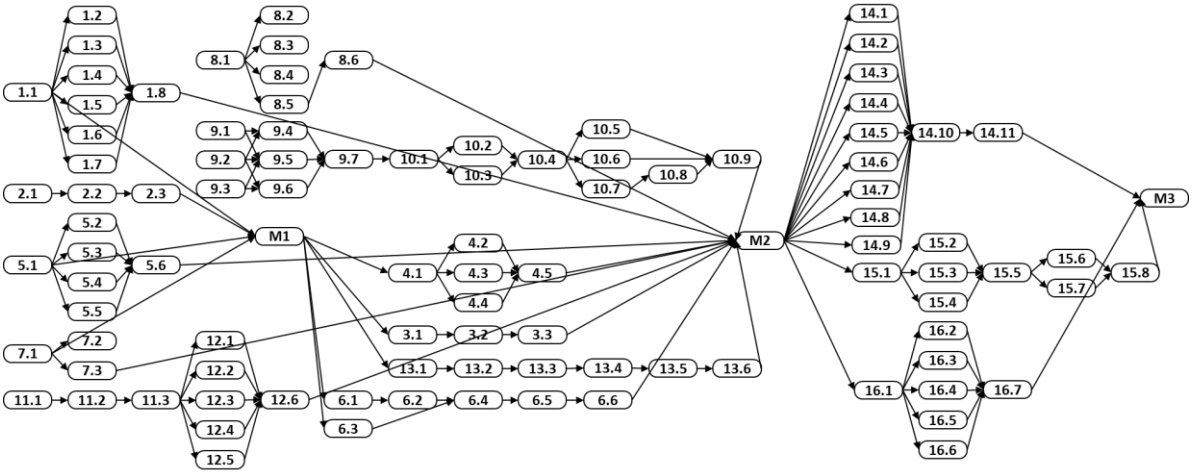


Figure 52. The network structure of Polytechnique University shared test instance

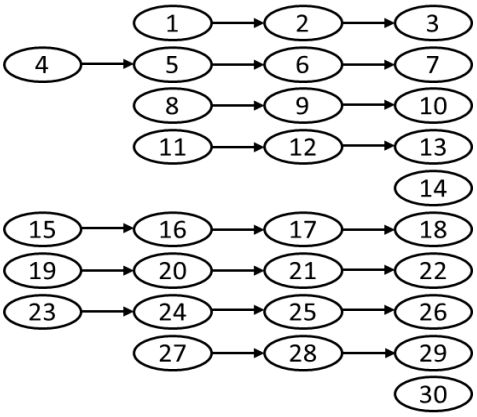


Figure 53. The network structure of Dal- Randomizer test instance

Appendix – 2

The priority- duration formulation and example data format

```
param H:= 40;  #Planning Horizon
param R:= 10;  #Resource Type
param n:= 10;  #No. of activities

set A, default{1..n}; # set of activities
set P within A cross A; #predecessor pairs
set RR, default {1..R}; #set of resources
param D{i in A}; #Duration of activity
param RD{i in A, k in RR}; #Resources demand for activity i
param AV{k in RR}; #Resource Availability
param ES{i in A} ; #Early Start time
param LS{i in A} ; #Late Start time
param PR{i in A}; #priority of the activity

var x{i in A, t in ES[i]..LS[i]}, binary;

minimize MS:  sum{i in A,t in ES[i]..LS[i]}  t * x[i,t] * ((1/PR[i])**5)* (0.01 + D[i])**1.1;

s.t. one{(i,j) in P} : sum{t in ES[j]..LS[j]} t*x[j,t] >= sum{t in ES[i]..LS[i]} t*x[i,t] + D[i];
s.t. two{t in 0..H,k in RR}: sum{i in 1..n, s in max(t-D[i]+1,ES[i])..min(LS[i],t)} RD[i,k]*x[i,s] <= AV[k];
s.t. three{i in A}:sum{t in ES[i]..LS[i]} x[i,t] = 1;

solve;

display MS,x;

end;
```

```
data;

set P :=
(2,3),
(6,7),
(8,9);

param:  D       ES      LS      PR:=
1       5       0       95      1
2       16      0       84      3
3       17      16      83      3
4       4       0       96      1
5       8       0       92      3
6       8       0       92      1
7       16      8       84      1
8       4       0       96      2
9       18      4       82      2
10      7       0       93      2   ;
```

```
param   AV   :=
1       35
2       35
3       35
4       35
5       35
6       35
7       35
8       35
9       35
10      35   ;

param   RD:    1       2       3       4       5       6       7       8       9       10  :=
1       0      0       0       0       7       0       0       0       0       0
2       0      0       0       26      0       0       0       0       0       0
3       0      0       0       0       0       10      32      0       0       0
4       0      0       0       0       0       0       0       0       0       0
5       1      0       0       0       18      0       0       0       0       0
6       0      0       0       0       16      0       0       0       0       0
7       0      0       0       0       16      0       35      0       0       0
8       0      0       0       0       0       0       0       0       0       35
9       0      1       0       0       0       0       0       0       0       0
10      1      5       0       0       0       0       0       0       0       0       ;

end;
```