Resource-Constrained Scheduling for Construction Projects

by

Margaretha Aletta Griebenow

Thesis presented in fulfilment of the requirements for the degree of Master of Engineering in the Faculty of Civil Engineering at Stellenbosch University

Supervisor: Dr. G.C. van Rooyen

March 2017

DECLARATION

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third-party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2017

SYNOPSIS

Project management and scheduling has been the topic of research for years, yet projects in industry are often completed after the deadline and significantly over budget. There is a large discrepancy between project scheduling in research and in practice. Even more worrying is a significant lack of research into construction project scheduling, an especially challenging field. The focus of this thesis will be project scheduling for construction projects. Advancements in project scheduling in research will make the required advancements in practice possible.

Two models will be investigated and compared, one utilising exact procedures and another utilising optimisation. Due to the limitations of exact procedures, this thesis will focus on developing a model capable of the optimisation of project schedules for the construction environment. The framework will be such that the entire solution procedure will utilise optimisation, and the format will allow more advanced objective functions to be utilised.

Construction projects are highly volatile. Two objective functions are presented which can lead to an improvement in the robustness of the schedule of the project, and therefore combat the effect of delays on the schedule of a construction project. Both functions are multi-objective and suited specifically to the construction environment. The functions minimise makespan while maximising the total slack, or distributed slack, in the schedule. The multi-objective format allows the relative importance of slack maximisation and makespan minimisation to be specified by the user, something which will be of great value to the construction industry where the robustness of a schedule is particularly important. The distributed slack and makespan function makes use of a risk model, where the importance of different activities having slack can be specified. This increases the likelihood that critical activities will have slack in the final schedule, and will be of great value to project managers.

SAMEVATTING

Projekbestuur en skedulering word reeds jare lank nagevors en in die praktyk gebruik. Desnieteenstaande word industrieprojekte dikwels laat voltooi en die begrotings word beduidend oorskry. 'n Groot gaping bestaan tussen projekskedulering in navorsing en in die praktyk. 'n Selfs groter probleem is dat daar te min navorsing oor konstruksie-projekskedulering gedoen word, 'n besonder uitdagende veld. navorsing oor konstruksie-projekskedulering gedoen word, 'n besonder uitdagende veld. Die fokus van hierdie tesis val op skedulering van konstruksieprojekte. Verbeteringe in projekskedulering in navorsing kan die benodigde vooruitgang in die praktyk moontlik maak.

Twee modelle sal ondersoek en vergelyk word. Die een maak gebruik van eksakte prosedures en die ander van optimering. As gevolg van die beperkings van eksakte prosedures sal die tesis fokus op die ontwikkeling van 'n model wat in staat is om projekskedules vir konstruksieprojekte te optimeer. Die raamwerk sal sodanig wees dat die oplossingsprosedure van optimering gebruik maak en die formaat sal dit moontlik maak om meer gevorderde doelwit-funksies te gebruik.

Konstruksieprojekte is onderhewig aan gereelde veranderinge en twee doelwit-funksies word voorgestel om die robuustheid van die skedule te verhoog. Daardeur word die effek van vertragings op die projekskedule teengewerk. Beide funksies kan multi-doelwitte hanteer en is spesifiek geskik vir die konstruksieomgewing. Die funksies minimeer die konstruksie tydspan, terwyl dit ook die totale tyd-speling, óf die verspreide tyd-speling van aktiwiteite in die projek sal maksimeer. Die multi-doelwit formaat maak dit moontlik dat die relatiewe belangrikheid van tyd-speling maksimering en tydspan minimering deur die gebruiker gespesifiseer word. Dit kan 'n groot bydrae lewer in die konstruksie industrie waar die robuustheid van 'n skedule besonder belangrik is. Die verspreide tyd-speling en tydspan funksie maak gebruik van 'n risiko model waarin die belangrikheid dat verskillende aktiwiteite tyd-speling moet hê gespesifiseer kan word. Dit verhoog die waarskynlikheid dat kritiese aktiwiteite wel tyd-speling sal hê in die finale skedule en dit sal van groot waarde wees vir projekbestuurders.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ACO	Ant Colony Optimisation
GUI	Graphical User Interface
RCPSP	Resource-Constrained Project Scheduling Problem
RCPSP-TT	Resource Constrained Project Scheduling Problem including Transfer Times

LIST OF MATHEMATICAL SYMBOLS

a	Number of resources available
b	Number of paths currently selected in flow
c_v	Coefficient of variation
$c(j)$	Risk value of activity j
d_j	Duration of V_j
$\bar{d}$	Average duration of all activities in project
e_{ij}	Precedence relationship between V_i and V_j
E	Dummy End Activity
f_{ij}	Resource flow from V_i to V_j
f_{max}	Maximum resource flow
$f(V)$	Resource Flow Path
$F\ (f(V))$	Flow
$F_F(f(V))$	Feasible Flow
g	Number of generations of ants in ACO
$\bar{g}$	Average number of generations to reach optimal solution
$G(V, X, R)$	Graph
m	Number of ants in each generation in ACO
r_j	Resource demand of V_j
$\bar{r}$	Average resource demand of all activities in project
R	Set of resources
R^2	Coefficient of determination
s_j	Start time of V_j
S	Dummy Start Activity = V_0
t_{ij}	Transfer time from V_i to V_j
X	Set of precedence relationships
V	Set of activities
V_j	Activity number j
$\in$	Is an element of
$\mathbb{N}$	Set of all natural numbers
α	Weighting factor used in the objective functions
β	Weighting factor used in the objective functions
γ	Weighting factor used in the objective functions
δ	Weighting factor used in the objective functions

θ	Weighting factor used in the objective functions
ρ	Evaporation rate
μ_m	Total resource flow along a resource flow path
μ_{max}	$a + \sum_{j \in V} r_j$
τ	Pheromone matrix

GLOSSARY

Ant Colony Optimisation	A meta-heuristic optimisation method that is capable of finding an optimal solution to a problem based on the technique that ants use to find the shortest path to a food source.
Backward Sweep	Moving backward in time from the project end and calculating the slack of each activity based on the latest time that an activity can end without delaying any successor.
Baseline Schedule.....	A schedule based on the expected evolution of a project, without resource allocations and without resource conflicts.
Distributed Slack	Total slack that is weighted according to the importance of each activity having slack.
Forward Sweep	Moving forward in time through the project activities and scheduling each activity to start once all predecessors have been completed, thus ensuring no resource conflicts exist.
Graph	A visual tool used to represent relationships and can be explained by its components, namely vertices and edges.
Late End	The latest point that an activity can end without delaying any other activity.
Late Start	The latest point that an activity can start without delaying any other activity.
Makespan	The total length of the project.
Meta-heuristic	A general algorithmic framework which can be applied to different optimisation problems with relatively few modifications.
Resource	An entity that contributes towards the completion of an activity, i.e. a person or crane.
Resource Allocation	Specifying which resource is assigned to a specific task
Robustness	The ability of a schedule to withstand unexpected disruptions.
Slack	The amount of time that an activity can be delayed without causing any successor activity to be delayed.
Start Time	The time that an activity is scheduled to start.
Total Slack	The sum of slack of all activities in a project.

CHAPTER 1

1 INTRODUCTION

The value of project management, and research into all aspects of good management, has received marked attention in recent years. Good project management can lead to consistently achieving project goals and deliverables. It can be the difference between a company achieving success or heading towards failure, both on specific projects and as a business.

One aspect of project management is the project schedule which is a crucial tool in ensuring the success of a project. A project schedule plays an important role in ensuring that a project is executed on time and within budget and can assist with better decision making. Schedules also act as a tool that benefits communication, collaboration, quality assurance, and safety and risk management.

The use of process models to assist in optimising complex processes has received marked attention in academia, and has proven beneficial regarding the analysis, simulation and optimisation of different processes. One field that has attracted much research is civil engineering projects, wherein the process being optimised is project scheduling. This focus on civil engineering projects is due to the unique challenges that one encounters when mapping the problem to a process model as well as the potential benefits of doing so successfully.

Civil engineering projects can be divided into two phases: the engineering phase and the construction phase. Some projects are fast tracked where the phases can be executed concurrently. The design and planning work is referred to as the engineering phase. The building of the project constitutes the construction phase. Civil engineering projects are unique, intricate and prone to changes throughout the project lifecycle. These factors present challenges during scheduling. Such projects have a large budget, scope, and size, as well as a variety of role players and resources – all factors which contribute to their complexity and unpredictability. The complex nature of the project directly affects the intricacy of the planning process and is the main contributing factor for the challenges faced while scheduling such projects.

The current industry standard is to schedule using human judgement and experience with the aid of rudimentary project management tools. These tools are not specialised for a specific industry, do not fully optimise the schedule produced and lack the amount of detail a project manager would require. The current method used in industry is cause for concern, especially when considering a large or relatively unknown project. The need for better tools to be used in industry is further emphasised when looking at the track record of certain high profile projects. The Scottish Parliament was 3 years late with a £364m, 730% cost overrun; the Boston Big Dig had a 6-year delay and a $19.2 billion, 685% cost overrun; and the Dubai Metro was 5 years late and approximately $3.6 billion and 85% over budget (Wilks 2015).

Much research has been done regarding project scheduling yet the industry has not caught up with the available research. In academia, functional models exist for the engineering phase which can now be implemented in industry, yet there is still a lack of research into the construction phase and how to account for the unique challenges that it presents. A basic model has been developed for construction phase scheduling that attempts to incorporate these unique requirements, but no accurate and usable model has been formulated. Significant advances need to be made in developing a functional model for the construction industry. This will be the focus of this research.

There are two significant differences when considering the construction phase of projects: construction sites can be immense and projects tend to be highly volatile. The size of a construction site is important due to the effect it has on the resources being used, namely that the time taken to transfer resources between activities becomes significant and needs to be accounted for. This is in contrast to the engineering phase of projects, in which personnel are the main resource and can switch between activities without incurring significant delays. Other resources used in the engineering phase such as software or stationery can be acquired with relative ease and will not drastically impact the schedule.

The volatile nature of these projects presents the second significant challenge. Whether the changes in the project are due to weather conditions, site conditions, delays, labour market volatility or material unavailability, they affect the project schedule. One can attempt to limit the risk of such changes, but the process model must still be able to easily deal with continuous modifications. Therefore, a scheduling model aimed at the construction industry should attempt

to make the schedule as robust as possible – i.e. able to handle disruptions without extending the completion date.

Addressing all of the concerns relating to project scheduling in industry and academia is beyond the scope of a single study. The focus of this thesis is the improvement of the scheduling of construction projects in the academic research environment with the aim of affecting the methods used in industry.

CHAPTER 2

2 STATE OF THE ART

The Civil Engineering Informatics Department of Stellenbosch University has been researching process modelling for several years, and this research in combination with other projects worldwide has led to considerable advancement in the field. A model has been developed that maps a schedule to a graph, where various graph theory methods are then available to assist in finding different solutions as well as an optimised solution for the schedule. A graph is able to represent relationships between objects, and can be referred to via its components: vertices and edges. In this specific process model, a vertex represents a project activity and an edge represents a technical dependency between two activities. An edge has direction, which indicates that one activity must be executed before another. This is referred to as a precedence relationship. An example of such a precedence relationship is that one must first dig a trench before laying a pipe. Although this concept might seem rudimentary, it is considered a major milestone in terms of project schedule optimisation.

The next important concept is that of a baseline schedule. Early research only accounted for technical precedence relationships and did not consider resource constraints in any way until the concept of a baseline schedule with no resource conflicts was developed (Eygelaar 2008). When resource demand is included and satisfied in a schedule this is referred to as the Resource-Constrained Project Scheduling Problem (RCPSP). The main objective of the RCPSP is the minimisation of the makespan of the schedule (Christodoulou 2016). In order to ensure a schedule has no resource conflicts, activities are shifted until the total resource demand in each time step is less than or equal to the total number of resources available. This concept is illustrated in Figure 1. It is important to note that a schedule without resource conflicts does not mean that resource allocations or the movement of resources are accounted for – but it is a crucial step towards developing such a model.

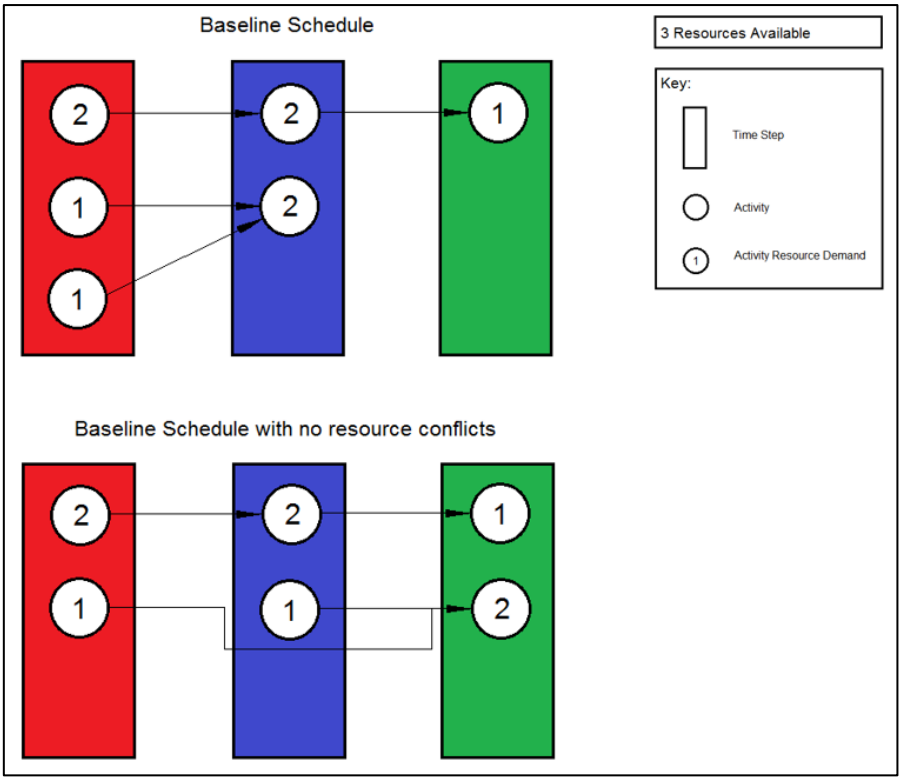


Figure 1: Resolving Resource Conflicts

In order to produce a realistic schedule resource allocations need to be accounted for explicitly. In addition to ensuring that no resource conflicts exist, the model needs to account for resource allocations and the movement of resources within the project. To address this need, the concept of resource precedence edges was introduced (Potgieter & Van Rooyen 2014). A resource precedence edge is added to the graph whenever a resource transfers from one task to another, and no technical precedence edge exists for the resource to move along. This concept is shown in Figure 2.

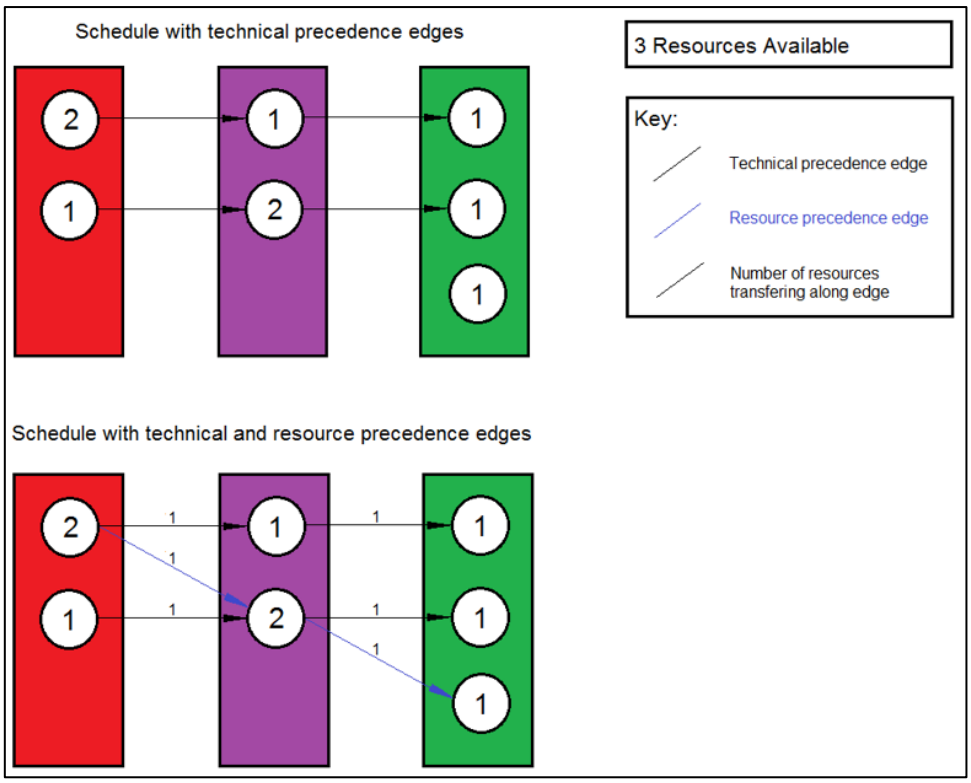


Figure 2: Resource Precedence Edges

The addition of resource precedence edges allows for resource allocations and their induced constraints to be incorporated into the model. This is a major development in project scheduling. Due to the movement of resources being accounted for by resource precedence edges, it is no longer necessary to reschedule the project when a change occurs. Activities can merely be shifted to get rid of scheduling conflicts. The addition of resource precedence edges therefore makes it possible for the model to absorb project disruptions without requiring re-optimisation of the schedule. Due to the stochastic nature of the optimisation techniques employed in schedule optimisation, re-optimisation could cause the schedule to change drastically which is of course extremely undesirable once a project is underway.

Interest in the RCPSP has grown over previous years. The RCPSP is an NP-Hard problem and is therefore extremely complex – especially with increased project size, number of resource types and constraints (Christodoulou 2016). The complexity of the problem leads to meta-heuristic optimisation algorithms being the preferred approach. Exact algorithms are either difficult to create or too computationally intensive, especially when considering projects of a realistic size. The current computational power of exact methods in research is around 60 activities whereas for larger projects the runtime becomes impractical (Potgieter 2014). This indicated that the

exact algorithms currently available are of no value as a solution for industry or when handling problems of a realistic size.

Various optimisation algorithms have been thoroughly researched and their benefits well documented. These algorithms are often capable of producing quality solutions for large projects. From numerous optimisation methods investigated, one approach produced excellent results when assessed against a library of benchmark problems - an Ant Colony Optimisation (ACO) algorithm that was adapted to suit the RCPSP by Merkle et al. (2002). This ACO algorithm proved to outperform several other heuristics tested (Merkle et al. 2002). Potgieter (2014) further developed the ACO and fine-tuned its parameters. This resulted in a framework able to produce high quality baseline schedules with minimised makespan for the engineering phase of civil engineering projects. The optimisation of the RCPSP is seen as an important development in project scheduling research. However, although many different techniques have been investigated and proven beneficial none have been adopted in practice (Potgieter 2014).

Often in research the project scheduling environment is considered static but in reality projects are plagued with uncertainty and disruptions (Herroelen & Leus 2002). What still needs to be accounted for in the model discussed thus far is resource allocations. The ability of the model to handle uncertainty and disruptions will be addressed during this process. The robustness of a schedule can be seen as its ability to handle disruptions. Improving the robustness of the model can be accomplished by selecting slack maximisation as the objective for the resource allocation process. The more slack that a schedule has the better it can recover from overruns and avoid exceeding the ultimate project deadline.

It has been shown that how resources are allocated can be used to optimise the total slack in a schedule (Potgieter & Van Rooyen 2014). Furthermore, the benefits of maximising total slack in a schedule have been highlighted yet it has received little attention in literature. As a first attempt, a heuristic algorithm capable of intelligent resource allocations to maximise the total slack of a project is proposed by Potgieter (2014). The benefits of making resource allocations based on a specific objective function should be clear, in that it limits the portion of the solution space that is to be explored. This is especially beneficial when dealing with large and/or complex projects (Potgieter & Van Rooyen 2014).

Distributing slack in an intelligent manner can further improve the robustness of a schedule. If slack is spread across the schedule it is less likely that a delay will disrupt other activities. This is especially true if one focuses on high-risk activities having more slack and low risk activities having less slack as they pose a lower risk of delay. It should be clear that slack distribution is as important, if not more so, than the total slack in a schedule.

The next important concept is that of resource transfer times. This is of particular importance in the construction environment. There are two different interpretations of resource transfer times namely physical and non-physical transfers. A physical transfer is when a resource must move location, for example a crane or digger on site. Non-physical transfers do not involve a change in location, for example the time required to set up equipment or for personnel to get familiar with a new activity. Non-physical transfers are considered insignificant in the model due to their duration being a negligible percentage of the total duration of the activities. It should be clear that physical transfer times will be significant in the construction phase of civil engineering projects, and due to the nature of the equipment being moved these times can be extensive.

The model developed by Potgieter (2014) is only applicable to the engineering phase of civil engineering projects. No attempt has been made to include transfer times as would be required for the model to be applicable to the construction phase. Due to the nature of construction projects and the potential for these transfer times to be extensive, this upgrade to the model is critically important. The difficulty of finding exact algorithms for the solution of the RCPSP has been well documented (Christodoulou 2016). However a first attempt to include transfer times in the current model was made by Griebenow (2014) utilising an exact approach. As stated the RCPSP is NP-Hard and the inclusion of resource transfer times in the model would further increase the complexity (Adhau et al. 2013). Therefore, the computation time requirements of exact procedures are further increased when utilised with a model including transfer times. As a result the algorithm presented by Griebenow (2014) is only applied to elementary examples but it shows that it is possible to consider transfer times in a project scheduling model. Therefore, a model that is applicable to the construction phase of civil engineering projects is finally possible – and within grasp.

When the RCPSP problem is extended to include transfer times, this is referred to as the Resource-Constrained Project Scheduling Problem including Transfer Times (RCPSP–TT). The exact algorithm developed for the RCPSP-TT utilises resource flows to find and represent a solution. In

contrast most approaches to the RCPSP represent solutions as activity lists or schedules which is a more suitable approach when utilising optimisation algorithms. The exact algorithm is therefore fundamentally different from most others, including the ACO algorithm developed by Potgieter (2014). A resource flow method was opted for due to the resource flow indicating the movement of a resource, thus resource transfers are immediately represented in the model. In an activity list approach resource transfers are not immediately represented and need to be added separately.

The influence of transfer times has been examined and its effect on a schedule and its makespan has been documented. Adhau et al. (2013) and Krüger & Scholl (2009) performed analyses in a multi-project environment where transfer times were only considered between projects. Griebenow (2014) examined small single-resource projects. Although the available research is limited, the results provide good insight. Transfer times have a significant influence on a variety of objective functions and their effect should not be ignored in the planning process. What is now lacking in research is a model capable of optimising a schedule where transfer times are included, where attaining these capabilities in industry makes it possible for industry to adopt these methods.

The next important factor to consider is the objective function to be used with the RCPSP-TT. For the RCPSP as described above the most common objective in literature is makespan minimisation. Often secondary objective functions are included such as the slack maximisation function which has been discussed. The slack maximisation objective presented by Potgieter (2014) is the first of its kind for the engineering planning environment. Functions such as these need to be further researched and developed. There is a distinct lack of research regarding a function that distributes slack instead of simply maximising it and this gap needs to be addressed. There is little research available on the optimisation of the RCPSP-TT and no literature on potential objective functions for such a model is available.

Due to the volatility and complexity of construction projects, it follows logically that slack maximisation and slack distribution functions should be the starting point when attempting to optimise the RCPSP-TT. The possibility of adapting the primary (makespan minimisation) and secondary (total slack maximisation) objective function structure as used by Potgieter (2014) into a multiple objective approach should also be investigated, where slack and makespan would be considered in one objective function. Such an objective function would make it possible to

allow for a slightly longer makespan in favour of more slack in a schedule – something of particular value in the construction environment.

CHAPTER 3

3 PROBLEM STATEMENT

3.1 THESIS OBJECTIVES

It is clear that project scheduling for the construction phase of civil engineering projects has numerous unique challenges which creates the need for a new project scheduling model specifically suited to the construction environment. The addition of transfer times is of specific importance due to the large impact such times can have on a schedule. The focus of this thesis will be the construction phase of civil engineering projects and the creation of a project scheduling model suited to this environment.

The exact algorithm that has been presented for the RCPSP-TT will be re-examined, and the model upgraded and tested for accuracy. Thereafter an attempt will be made to minimise the computational effort of the exact procedure and it will be evaluated whether an exact approach to the RCPSP-TT is too computationally intensive for use in industry.

As highlighted the Ant Colony Optimisation (ACO) algorithm has been shown to outperform many optimisation methods and has become a preferred algorithm in the project scheduling environment. The model created by Potgieter (2014), involving an ACO algorithm to optimise the baseline schedule and a heuristic approach to resource allocations, is therefore assessed regarding the possibility of upgrading the model for use with the RCPSP-TT. The quality of results provided by this model will then be evaluated through comparison with the exact approach. Furthermore, a significant upgrade will be investigated namely the optimisation of the resource allocation procedure. Through introducing an optimisation method for the resource allocation process, the model will finally be capable of incorporating sophisticated objective functions for both the RCPSP and RCPSP-TT.

The research presented in this thesis will serve as a first attempt to optimise project scheduling for the construction phase of civil engineering projects, and will introduce the possibility of incorporating more advanced objective functions due to the use of the optimisation method. This is considered a first step toward the ultimate goal of creating a model for use in industry that is capable of optimising the RCPSP-TT.

3.2 OUTLINE

A brief outline of the chapters in this thesis is presented below. This provides an overview of the document structure and assistance in understanding the flow of the document.

Chapter 4: Fundamental Mathematical Concepts

The mathematical concepts essential to understanding the remainder of this thesis will be presented. The topics covered include a graph, edge, resource flow path, feasible solution, schedule, slack and a framework.

Chapter 5: Framework One Overview: Exact Method

The model developed by Griebenow (2014) which uses an exact approach to the RCPSP-TT, hereafter referred to as Framework One, will be the focus of this chapter. The input, solution process and output of the framework will be described and the required improvements discussed.

Chapter 6: Framework Two Overview: Optimisation Method

The optimisation method developed by Potgieter (2014), hereafter referred to as Framework Two, will be presented and the input, solution process and output will be discussed. In order to make the framework suited to the RCPSP-TT certain essential upgrades are performed. These will be explained in detail.

Chapter 7: Framework One Method Improvement

Framework One will be examined in order to ensure that it is capable of exploring the entire solution space and discovering all possible schedules for a given project.

Chapter 8: Framework One Runtime Optimisation

The algorithms used in Framework One will be analysed and improved in an attempt to shorten the computational time for the framework to compute its output.

Chapter 9: Comparison of Framework One and Two

The exact procedure used in Framework One and the optimisation method in Framework Two will be compared, and the quality of the results produced by Framework Two assessed.

Chapter 10: Framework Two Method Improvement

The resource allocation procedure of Framework Two will be modified. The original heuristic method will be replaced with an optimisation algorithm and the results analysed.

Chapter 11: Objective Function Development

The development of an objective function for the construction environment is the focus of this chapter. A slack maximisation and slack distribution objective function will be developed. A multi-objective function is employed in both cases in order to minimise makespan while maximising the slack of the schedule.

Chapter 12: Conclusion and Recommendations

The thesis will be concluded by summarising the findings of the research. Aspects of the model requiring further development by future researchers as well as other potential topics for future research will be discussed.

CHAPTER 4

4 FUNDAMENTAL MATHEMATICAL CONCEPTS

In this chapter the formal definitions and notations of certain fundamental mathematical concepts will be introduced. These concepts, which are important to the remainder of the thesis, include the concepts of a graph, edge, resource flow path, feasible solution, schedule, slack and a framework.

4.1 A RESOURCE-CONSTRAINED PROJECT REPRESENTED AS A GRAPH

A project consists of a set of activities $V = \{V_0, V_1, \dots, V_{n+1}\}$, where each activity $V_j \in V$ has a duration $d_j \in \mathbb{N}$ and a resource demand $r_j \in \mathbb{N}$. Integer time units of duration are assumed while partial resource demand is considered to be not applicable. Activities are time-continuous which indicates that once an activity has commenced it cannot be interrupted. Note that the duration and resource demand of each activity are non-zero, except for the source and sink activities which will be explained in the paragraphs that follow.

A set of resources $R = \{i, ii, iii, ..\}$ is available, with the total number of available resources equal to a. Consequently, for every activity j the following rule holds: $r_j \leq a$. For the moment, all resources are considered to be of the same type.

A precedence relation X exists in the set of activities. Each ordered pair $e_{ij} \in X \subseteq \mathrm{V} \times \mathrm{V}$ indicates a finish-start precedence relationship from activity V_i to activity V_j. Additionally, each e_{ij} has a transfer time t_{ij} and a flow f_{ij}. The transfer time t_{ij} indicates the amount of time necessary for a resource to transfer from activity V_i to activity V_j, and f_{ij} indicates the number of resources transferring from V_i to V_j. The transfer time is defined during the project input and may be zero, but the flow is determined by the solution. It is important to note that a transfer time is only invoked when the flow along the edge is non-zero. The concept of a transfer time, as discussed, is a factor that is uniquely critical in the construction phase of civil engineering projects and can depend on a variety of factors. In this model the transfer times are specified directly, but these values can be based on the global location of activities, the project manager's knowledge, or

perhaps a more complex time function. The calculation method of the transfer times provides one possible opportunity for further research.

A project is modelled by a *Graph* $G(V, X, R)$ in which activities are represented by vertices and precedence relationships are represented by edges. A graph requires a source $V_0 = S$ which is a dummy start activity and a sink $V_{n+1} = E$ which is a dummy end activity, and all resources must flow through both of these activities. These two activities are the only activities with zero duration $d_0 = d_{n+1} = 0$ and zero resource demand $r_0 = r_{n+1} = 0$. When these two dummy activities are added, additional technical precedence edges are created between V_0 and all activities without predecessors, as well as from all activities without successors to V_{n+1}. It therefore follows that in the final graph $G(V, X, R)$ the only activity without predecessors is V_0 and the only activity without successors is V_{n+1}. Importantly, the project graph must be acyclic. Any cycle of tasks must be fused into a single, compound task.

4.2 Edge Types

The project graph can contain two different types of edges. The first are technical precedence edges, as discussed in Chapter 2.

The second edge type is a resource precedence edge $e_{ij} \in X$, and indicates the transfer of a resource from one activity to another. The resource precedence edges and the flow along each edge must be determined during the solution process. An example of this is when a work team, for example a shuttering team, must perform tasks in two locations. *Task A* is the placement of shuttering for the west wall of the structure and *Task B* the placement of shuttering for the east wall. The order in which the tasks are completed is irrelevant. The team then elects to complete *Task A* before commencing *Task B*, creating a resource precedence edge from *Task A* to *Task B*. This edge therefore indicates that one activity will be completed before the other as a result of both utilising the same resource, and is a significant constraint.

4.3 Resource Flow Path

A resource flow path $f(V)$ is defined as the route that a resource takes from the dummy start activity S to the dummy end activity E without any activity being visited twice. It is subsequently represented as an ordered set of activities $f(V) = \{k\}_{i=1}^{N}$ where k represents an activity V_k. In a

resource flow path, if one resource transfers from activity V_i to activity V_j, an edge $e_{ij} \in X$ exists with $f_{ij} = 1$.

Each resource flow path has a maximum flow f_{max}. This value represents the maximum number of resources that can flow along the path without any activity receiving more resources than required. f_{max} can therefore be easily determined as the minimum resource demand of all the activities that form part of the resource flow path, as shown in Equation 1.

$$f_{max} = min_i \, r_{k_i} \quad \textbf{1}$$

4.4 Feasible Solution

A solution for $G(V, X, R)$ is in the form of a number of resource flow paths, specifically one path per available resource. A set of resource flow paths are combined to form a flow F ($f(V)$). Any flow may be considered a solution to the graph, however not all flows are feasible in terms of the problem under consideration. When the following conditions are met, the flow F_F ($f(V)$) is called feasible and therefore a feasible solution to the RCPSP has been found:

- ✓ Resource demand of every activity is satisfied
- ✓ Every resource flows from source to sink

The criteria above mean that a feasible solution has been found when each activity has been allocated the correct number of resources from the pool of available resources. A small example indicating a feasible solution of a project with $a = 4$ can be seen in Table 4-1, with the technical precedence relationships of the project shown in Figure 3.

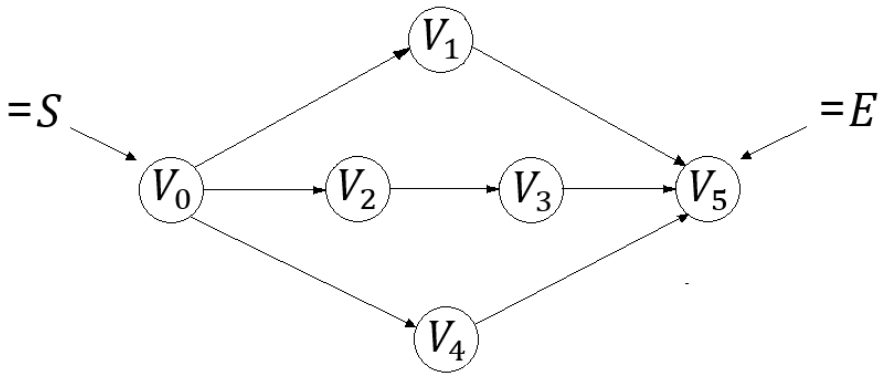


Figure 3: Example Project

Table 4-1: Example Project Feasible Solution Details

Activity	d_j	r_j	Allocated Resources
$V_0 = S$	0	0	
V_1	1	1	R_1 & R_2
V_2	2	1	R_3
V_3	1	1	R_3
V_4	1	1	R_4
$V_5 = E$	0	0	

In the above example, the movement of resources is as follows:

- R_1 and R_2: $S \rightarrow V_1 \rightarrow E$
- R_3: $S \rightarrow V_2 \rightarrow V_3 \rightarrow E$
- R_4: $S \rightarrow V_4 \rightarrow E$ =

4.5 SCHEDULE AND SLACK

From a feasible solution one can determine the associated feasible schedule of the project. Each activity j will be allocated a start time s_j. A schedule is created using a simple forward and backward sweep over the graph, whereby each activity is scheduled to start once all its predecessors have been completed. Note that sometimes more than one feasible schedule is possible for each feasible solution. For example, if the technical precedence edge e_{23} did not exist in the example above, two schedules would be possible. These two schedules would contain the edges $V_2 \rightarrow V_3$ and $V_3 \rightarrow V_2$ respectively.

When considering the predecessors of an activity, one can consider the resource-constrained or resource-unconstrained environment. The difference is that in the resource-unconstrained environment only technical precedence edges are considered, whereas for the resource-constrained environment resource precedence edges are also considered. The choice of environment will affect both the schedule and slack that is produced for a specific feasible solution.

Activity slack is defined as the amount of time that an activity can be delayed without causing any successor activity to be delayed. Two types of slack can be considered, namely resource-constrained slack and resource-unconstrained slack. The example project presented in Figure 3 is used to further explain the difference between the two types of schedules and slack. The project

has $a = 2$ and further details are indicated in Table 4-2. The associated resource-constrained and resource-unconstrained schedules are presented in Figure 4. For the constrained case the schedule is presented for the resource flow path R_1: $V_1 \rightarrow V_4$, as another schedule is also possible.

Table 4-2: Example Project Feasible Solution Details

Activity	d_j	r_j	Allocated Resources
$V_0 = S$	0	0	
V_1	1	1	R_1
V_2	2	1	R_2
V_3	1	1	R_2
V_4	1	1	R_1
$V_5 = E$	0	0	

Resource-Unconstrained Schedule and Slack

Activity	Time Unit 1	2	3	Slack
V_1	■			2
V_2	■	■		0
V_3			■	0
V_4	■			2
				4

Resource-Constrained Schedule and Slack

Activity	Time Unit 1	2	3	Slack
V_1	■			0
V_2	■	■		0
V_3			■	0
V_4		■		1
				1

Figure 4: Example Project Resource-Constrained and Resource-Unconstrained Schedules and Slack

As one can see from the example above, the type of environment considered significantly influences the schedule and slack results. In this thesis, the type of environment used is resource-constrained. Resource allocations are a crucial part of the project scheduling process - if they are not considered the results can be drastically skewed and unrealistic.

4.6 FRAMEWORK

A planning framework is a method of project planning in which each framework is often adapted to best suit a specific subset of planning problems. For example, certain frameworks cater to the engineering planning phase of projects whereas others are better suited to the construction phase of projects. A framework divides the project planning process into parts namely an input, planning and output phase - with specific methodology being used in each phase.

CHAPTER 5

5 FRAMEWORK ONE OVERVIEW: EXACT METHOD

The first software framework that is investigated in this thesis was developed by Griebenow (2014). A brief overview of the framework will be given by first referring to the required input, then the two phase solution process and lastly the output. Furthermore, the areas of the model that require improvements will be highlighted. Details of the software implementation of this framework are available in Appendix F in which the software implementation reflects the final version of the framework developed in this thesis.

5.1 INPUT

The required input for the model is a set of activities V, a set of technical precedence relationships X and a specified number of available resources a. The transfer time t_{ij} for every technical precedence relationship and every possible resource precedence relationship can be specified, otherwise it is assumed to be zero.

5.2 ADDITION OF RESOURCE PRECEDENCE EDGES

Before the two-part solution phase is commenced, resource precedence edges are added. For all pairs of activities V_i and V_j where a technical precedence relationship does not exist and $0 < r_k \leq a$ holds for both, a resource precedence edge e_{ij} is added. A set of rules by which the direction of this edge is selected is proposed in Griebenow (2014) and listed below. These edge direction selection criteria shorten the solution procedure but also vastly limit the solution space.

The algorithm cycles through all activity pairs where an edge needs to be added. The starting activity is found by moving along the list until a criterion is applicable to the activity pair.

1. Starting at the activity with an earlier starting time s_j
2. Starting at the activity with higher resource demand r_j
3. Starting at the activity with the earlier end time $s_j + d_j$
4. With random direction

5.3 PHASE 1: RESOURCE FLOW PATHS

The first phase of the solution process involves the calculation of all possible resource flow paths $f(V)$. The resource flow paths are calculated using a brute-force method of combinatorial nature which uses a recursive algorithm. The method begins at the dummy start activity and every time that the next activity needs to be selected, all eligible activities are viewed as branching decisions – in this way a complete solution tree can easily be formed for the problem. Every time that a branching decision needs to be made, the algorithm will explore every possible decision with the exception of those that cannot lead to new paths, involve visiting the same activity twice, or cannot lead to the dummy end activity. In this way, the solution tree is reduced and the algorithm becomes more efficient. Once all branches in the solution tree are explored, the entire available solution space has been explored and all possible resource flow paths have been found.

The algorithm commences by finding one path through the graph from the dummy start activity to the dummy end activity. A loop is then executed which uses a set of paths as input and produces a set of undiscovered paths as output. Initially the loop input is the first path that was discovered, and thereafter the input is all new paths discovered in the previous loop. Each path is processed individually by the loop. The path is shortened by one activity and a new route to the dummy end activity is attempted; this process is repeated until there are no activities left to remove from the given path. In this manner, the algorithm will continue running until no new paths are discovered in an execution of the loop – and all possible paths through the graph have been found.

Due to the combinatorial nature of the problem the solution tree can become enormous for problems of a practical size. This can cause the runtime of this framework to be too large to be practically useful, i.e. in industry. The topic of runtime improvement and whether the framework is practically implementable will be addressed in Chapter 8.

5.4 PHASE 2: FEASIBLE FLOWS

The second phase of the solution process consists of combining sets of resource flow paths to form a flow F ($f(V)$), where the input for this phase is the list of all resource flow paths found during Phase 1. The method used is a combinatorial brute force method that utilises recursion. When combining a set of resource flow paths, a rule is used to determine which combinations of paths have the potential to be feasible flows. Lemma 1 is a theorem presented by Herroelen & Leus (2002) which when true indicates that a flow is possibly feasible and when false a feasible

flow is impossible. Herroelen and Leus presented a branch-and-bound algorithm that is used for resource allocation, and this lemma assists in ensuring a feasible flow is possible during each step as the algorithm progresses. This lemma, as presented in Equation 2, is used in addition to the requirements of a feasible solution to determine if a flow is feasible.

$$\sum \mu_m = \mu_{max} \qquad 2$$

In the above equation, $\mu_{max} = a + \sum_{j \in V} r_j$ and $\sum \mu_m$ is the total source to sink flow of resources in the proposed flow F. This concept can be further explained as follows. $\mu_m = \sum f_{ij}$ represents the total resource flow along a resource flow path $f(V)$, and $\sum \mu_m$ is the total resource flows along all resource flow paths in the possible flow F. The value of μ_{max} can be predetermined for the graph and the value of μ_m can be calculated per resource flow path $f(V)$. By testing all combinations of paths, a possible flow is noted when $\sum \mu_m = \mu_{max}$. This possible flow is then tested against the feasible solution criteria to determine if a feasible flow and therefore feasible solution has been found. For a visual explanation of the concept, see Figure 5 which is based on the example project data given in Table 4-1. As can be seen, $\sum \mu_m = 9$, $a = 4$ and $\sum_{j \in V} r_j = 5$ and therefore $\mu_{max} = 9 = \sum \mu_m$.

$f(V)$	μ_m
0 - 1 - 5	2 + 2 = 4
0 - 2 - 3 - 5	1 + 1 + 1 = 3
0 - 4 - 5	1 + 1 = 2

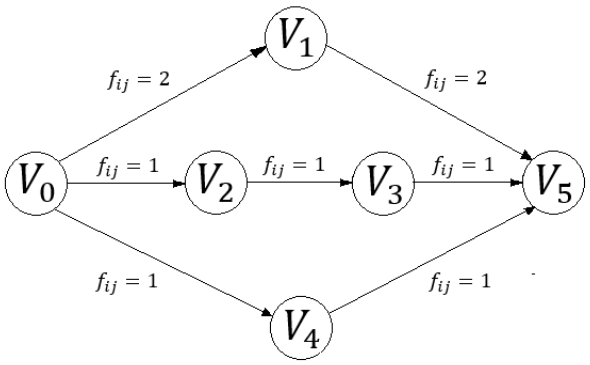


Figure 5: Source to Sink Resource Flow Example

The algorithm initialises by sorting all resource flow paths by order of descending μ_m value. A set of paths is then chosen, starting by selecting the first path in the set, and viewing all remaining possibilities as branching decisions. The entire solution tree is created in this way and then explored, meaning that all possible combinations of paths are analysed using a simple recursion method. Lemma 1 is used to determine which branches of a solution tree it is not necessary to

explore. Once all possible combinations are checked Phase 2 is complete, but all solutions found are not necessarily feasible solutions. Each solution is then checked in terms of the criteria for a feasible solution, see Chapter 4.4, and thus the final list of feasible flows is attained.

5.5 OUTPUT

This framework can produce all feasible solutions for a given input, with the solution space only restricted by the resource edge selection criteria used. For each solution the output includes the scheduled times and slack for each activity as well as resource flow paths utilised, hence producing a schedule with all the information required to execute the project.

5.6 POSSIBLE IMPROVEMENTS

The most significant flaw in Framework One is the limited solution space created by including only certain resource precedence edges instead of all possible edges. As a proof of concept this limitation was acceptable, but in order to determine all possible solutions and ensure the optimal solution is included one must include all edges. The method by which resource edges are added will be investigated in this thesis.

The exact procedures utilised in this framework indicate that if the entire solution space is explored all possible solutions will be found – ensuring that the optimal solution for the solution space is among them. It has been shown that the exact procedures researched thus far cannot handle problems of realistic size when considering RCPSP and RCMPSP projects (Krüger & Scholl 2009). If the framework being discussed cannot be vastly improved or adapted, it will indicate that the only realistic approach for RCPSP - TT is a heuristic or meta-heuristic.

By limiting the solution space through the resource edge direction selection criteria, the framework was simplified and therefore able to handle larger problems. In order to accurately compare the results of Framework One with other frameworks, as well as to ensure that all solutions are found, this simplification will be corrected. Chapter 7 will examine the edge selection method utilised by Framework One with the intention of including the entire solution space and Chapter 8 will investigate the enhancement of the runtime through various framework improvements.

CHAPTER 6

6 FRAMEWORK TWO OVERVIEW: OPTIMISATION METHOD

The second framework that will be investigated in this thesis is based on the one developed by Potgieter (2014). Potgieter's work focuses on the engineering planning phase, i.e. the engineering phase of civil engineering projects, and not the construction phase therefore certain shortcomings will need to be addressed. Specifically, transfer times do not play a role in the engineering-planning phase of projects since all resources are people that can switch from one activity to another without significant delay. An overview of the framework will be given by making reference to the required input, the two-part solution process, and lastly the available output. Furthermore, an overview of the initial upgrades to account for resource transfer times will be given. Details of the software implementation of this framework are available in Appendix G where the software implementation reflects the final version of the framework developed in this thesis.

6.1 INPUT

The input required by this framework is identical to that of Framework One, namely a set of activities V, a set of technical precedence relationships X and the number of available resources a. This framework is capable of handling multiple resource types but for the purposes of a comparison with Framework One only one resource type is used.

Note that after upgrading Framework Two to allow for transfer times, transfer time information will be required during input. The transfer time t_{ij} for any resource transfer can be specified, otherwise it is assumed to be zero.

6.2 PHASE 1: SCHEDULING

The first part of the solution process, named the Scheduling Phase, involves the generation of a baseline schedule without resource conflicts. Note that this schedule will be free of resource conflicts but will not yet have any resource allocations. The objective of the Scheduling Phase is

to generate a schedule with the shortest makespan. An Ant Colony Optimisation (ACO) algorithm is used to reach this objective. The ACO optimises the sequence in which activities are scheduled in order to achieve a schedule with the smallest makespan. Potgieter (2014) carefully fine-tunes and optimises the ACO parameter values and presents a sophisticated ACO, including the use of the heuristic developed by Merkle et al. (2002). For full details of the ACO algorithm used refer to Potgieter (2014).

Note that even though no resource allocations are made during this phase, the baseline schedule selected will have certain consequences in the remainder of the solution process. If one task is scheduled after another in the baseline schedule, i.e. $V_i \rightarrow V_j$, this order cannot change later in the solution process. Regardless of this limitation, Framework Two was shown to produce excellent results and this phase of Framework Two is used as is. This is convenient since a wide variety of sophisticated optimisation procedures have been developed over the years. It makes sense to reuse a tried and tested procedure.

6.3 Phase 2: Resource Allocation

The Resource Allocation Phase is the second phase of the solution process and the objective of this phase is to maximise the slack in the schedule. One of the only known slack based objective functions for the engineering planning environment is presented in Potgieter (2014). As a first attempt a heuristic approach to maximising the total slack is used. A greedy algorithm is used which makes one resource allocation at a time, and for each allocation the resource allocated is the one with the best objective function value. The objective function selected is to minimise the negative impact on the total slack of the graph. This method is founded on the assumption that a resource allocation with the smallest impact on the total slack will also be a good assignment in terms of the overall solution. The heuristic algorithm for the Resource Allocation Phase is as follows:

- For each resource assignment:
  - Calculate the *eligible resource set*
  - Make mock assignments of all eligible resources. Note that a mock assignment is a resource assignment that is not final, and instead it is an assignment made in order to evaluate the effects of making such an assignment.
  - Evaluate the objective function for each mock assignment
  - Select a resource to allocate based on the objective function
  - Make the final assignment of selected resource to the graph

Note that an *eligible resource set* comprises all available resources, in other words resources that have not already been assigned to an activity that overlaps with the current activity in the schedule. When a mock assignment is made a resource is assigned to the activity, the objective function is evaluated, and the mock assignment is then removed from the graph. The algorithm then selects the resource that performs best, i.e. with the smallest negative impact on total slack, as the final resource allocation.

Note '*for each resource assignment*' in the heuristic above - the order in which resource allocations are made is specified by the algorithm. Three factors were identified that affect the allocation order and therefore the outcome of the allocation process. These factors are the allocation schema, activity queue and resource queue. The allocation schema determines whether the allocation is done resource-by-resource or activity-by-activity. This is referred to as a resource-wise or activity-wise allocation schema. Note that the allocation schema is inconsequential for the purposes of this thesis, as one resource type is assumed. The following is resource-wise allocation while the opposite would be an activity-wise allocation:

- For each resource type
    - For each activity ($j \in V$)
        - Complete the heuristic above as defined for each resource assignment.

The second factor is the resource queue which represents the order in which resources are processed and could be by increasing or decreasing demand. The activity queue could either be ordered by increasing or decreasing start dates. The results were then analysed for each combination of the three elements above, resulting in eight different algorithms as well as a ninth random algorithm. The nine different heuristics were investigated and compared in Potgieter (2014) and the best performing algorithm was one with *increasing resource queue, increasing activity queue,* and a *resource-wise allocation schema*.

6.4 OUTPUT

Framework Two produces one optimised solution in the form of a schedule that includes resource allocations. For every activity one can view the start times, end times and resource allocations, as well as the resource flow paths for every resource. The Framework has a sophisticated output that is able to provide the user with resource-constrained and resource-unconstrained slack, the critical path and other useful information. Note that a critical path is the longest sequence of

activities in the schedule which must all be completed on time for the project to be finished by the scheduled completion date.

6.5 ESSENTIAL UPGRADES

As noted, Framework Two was designed for the engineering planning phase, and does not account for resource transfer times. The framework therefore requires certain fundamental upgrades before it is possible for it to be utilised for the construction phase of civil engineering projects, thus making these upgrades essential.

Firstly, it is important to note that two separate objective functions are used for the two phases of the solution process. Due to no resource transfer times being included in the current version of the framework, the makespan of the schedule will not change during the second phase when resource allocations are made. Therefore the separation of objective functions still produces a result in which both functions are optimised simultaneously. Furthermore, no resource transfers are taken into account when resource allocations are made. Both of the above-mentioned factors will differ when considering the construction phase of civil engineering projects.

In order to include resource transfer times in the current framework, dormant resource transfer activities are added to the graph. When a graph is created all resource transfer times will be created as activities with specified duration – but without any resource requirements, predecessors or successors. During the resource allocation process, when a resource assignment or mock assignment is made that would involve a resource transfer time this dormant activity becomes active, the activity gains a successor and predecessor, and the resource demand of the activity is increased by one. This means that resource transfer times will now be considered when resource allocations are made.

Due to resource transfer activities being added to the graph during Phase 2, the schedule will no longer be the same and activities will have to be shifted to account for the transfer times. This can be accomplished by a forward and backward sweep of the graph in order to recalculate the start and end times of all activities, as well as their slack values. The addition of resource precedence edges will ensure that the resulting schedule will still be free of resource conflicts (over-allocations). Consequently, it is not necessary to re-optimise the schedule at this point, i.e. to

execute Phase 1 again. If Phase 1 was executed again all previous decisions will be disregarded, i.e. the order of activities as produced by Phase 1, which is illogical.

Importantly, this modification causes the makespan of the schedule to change during the second phase, something which was not previously possible. The performance of the chosen resource allocation method and its effect on the makespan of the schedule now becomes uncertain and needs to be determined. An intelligent objective function will need to be incorporated into the Resource Allocation Phase in order to ensure that makespan minimisation is not overlooked during this phase, and to prevent unpredictable behaviour of the algorithm. The inclusion of an intelligent objective will direct the search for a solution and thus the result of the algorithm will be more predictable.

CHAPTER 7

7 FRAMEWORK ONE METHOD IMPROVEMENT

As pointed out in Chapter 5, the technique used in Framework One to select and add resource edges to the graph is flawed and limits the solution space. To accurately compare Framework One and Framework Two this discrepancy needs to be addressed.

7.1 ORIGINAL RESOURCE EDGE SELECTION TECHNIQUE

The edge selection criteria currently used to create resource precedence edges utilised by Framework One were presented in Chapter 5.2.

7.2 UPDATED RESOURCE EDGE SELECTION TECHNIQUE

The following updated list of edge selection criteria is proposed and will allow Framework One to find all possible solutions for a given project. All resource precedence edges will now be accounted for within the graph. Therefore any resource transfer becomes possible thus incorporating the entire solution space.

- Between any two activities with non-zero resource demand, edges in both directions
- An edge from every vertex with a non-zero resource demand to the dummy end vertex j_{n+1}
- An edge from the dummy start vertex j_0 to every vertex with non-zero resource demand

By comparing the original and updated edge selection techniques it will be clear whether the solution quality improves and/or the solution space increases. Note that with the updated edge selection technique the graph will no longer be acyclic during the solution process, and will only regain its acyclic characteristic once a specific solution is applied. This will affect which graph theory algorithms can be utilised as some algorithms are only applicable to acyclic graphs.

7.3 ESSENTIAL UPGRADES

The updated edge selection technique allows for edges in both directions between all activities with non-zero resource demand. This is a crucial modification in order to cover the entire solution

space. This modification however requires new feasibility checks to be added to the framework, to ensure that all technical precedence relationships are accounted for implicitly in both phases of the solution process and that no solution is illogical in nature.

Due to the updated resource edge selection technique, it now becomes possible for the framework to discover a path through the graph that does not adhere to the technical precedence requirements. To correct this, after Phase 1 of the framework is executed the resource flow paths discovered will be checked for any technical precedence violations. A basic algorithm is used that compares the technical predecessors of each activity in the graph to each resource flow path – and disregards any paths where conflicts are found.

Furthermore, it is important that no solution contains $V_i \rightarrow V_j$ and $V_j \rightarrow V_i$ simultaneously, as this would imply that activity V_i precedes V_j and vice versa which is not possible simultaneously. Consequently, an additional criterion for a feasible solution becomes necessary when utilising the updated edge selection technique. This additional criterion will be checked at the end of Phase 2 when all solutions are evaluated in terms of feasibility. The new feasible solution criterion stipulates that when a solution contains a path in which V_i precedes V_j, it cannot also contain a path in which V_j precedes V_i.

7.4 COMPARISON

Three simple example projects are used namely Project 1, Project 2 and Project 3. For the full detailed information on these projects refer to Appendix A, Appendix B and Appendix C. The information provided in the appendices includes all activities, their resource demand and duration, all technical precedence edges, and all resource precedence edges as created by Framework One for both the original and updated edge selection techniques. Furthermore, the distribution in terms of makespan and slack of all solutions is provided. Further details are then provided for solutions with the minimum makespan, namely the resource allocations, slack, and subsequent schedule of the solution.

Project 1, 2 and 3 will be simulated using Framework One for both the original and updated edge selection techniques. This simulation will indicate what percentage of the solution space was explored using the original technique, and whether the updated edge selection technique improves solution quality.

7.4.1 *RESULTS*

The number of solutions for Project 1 increased from 5 to 77, from 3 to 793 for Project 2, and from 2 to 6 for Project 3. Project 1 has four activities and Project 2 and 3 have 5 activities each but varying complexity, as can be seen from the number of feasible solutions available. Refer to Appendix A, Appendix B and Appendix C for further details.

The simulation results indicate that a large percentage of the solution space was unexplored with the original edge selection method whereas the updated edge selection technique will explore the entire solution space. Figure 6 illustrates what percentage of the solution space was discovered by the original edge selection technique.

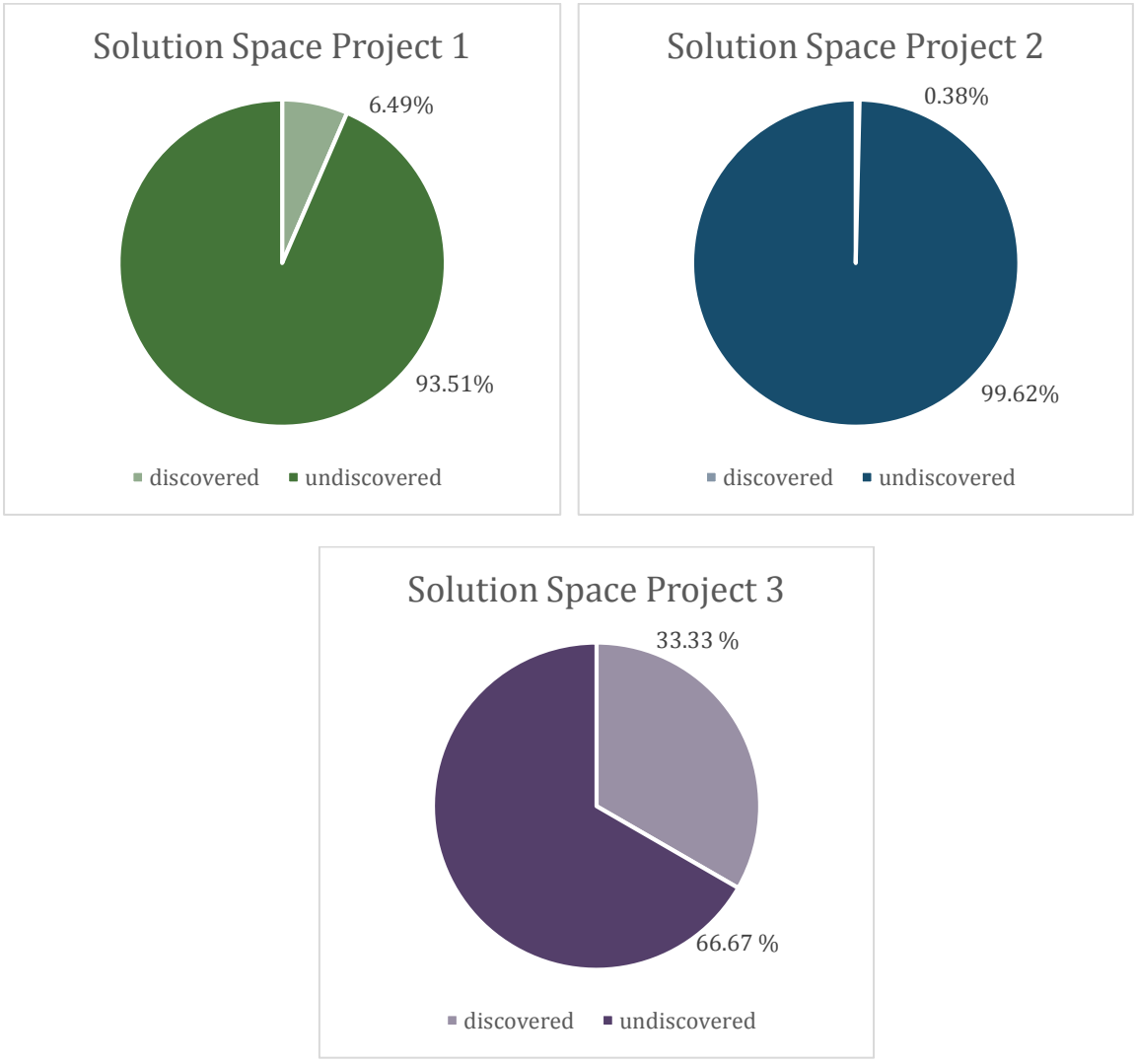
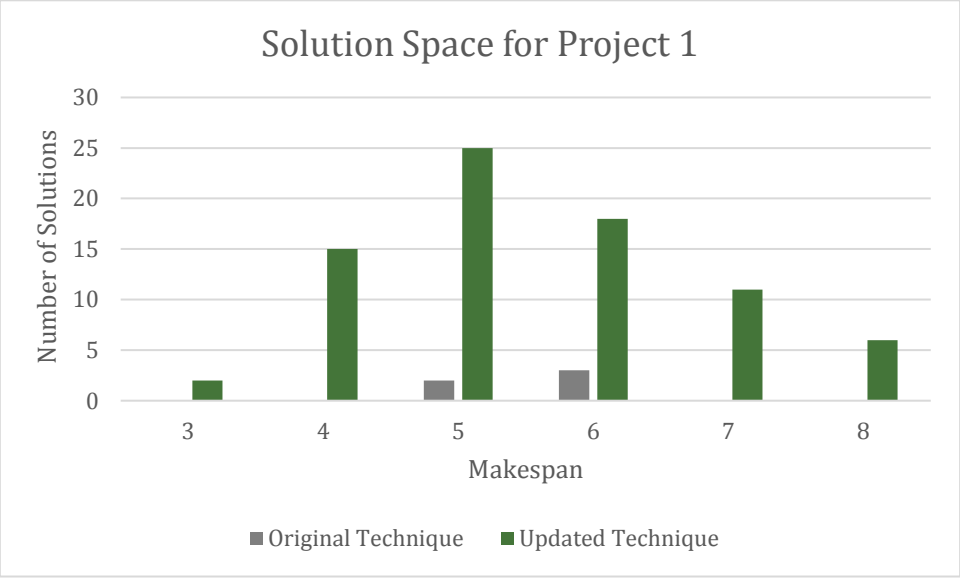
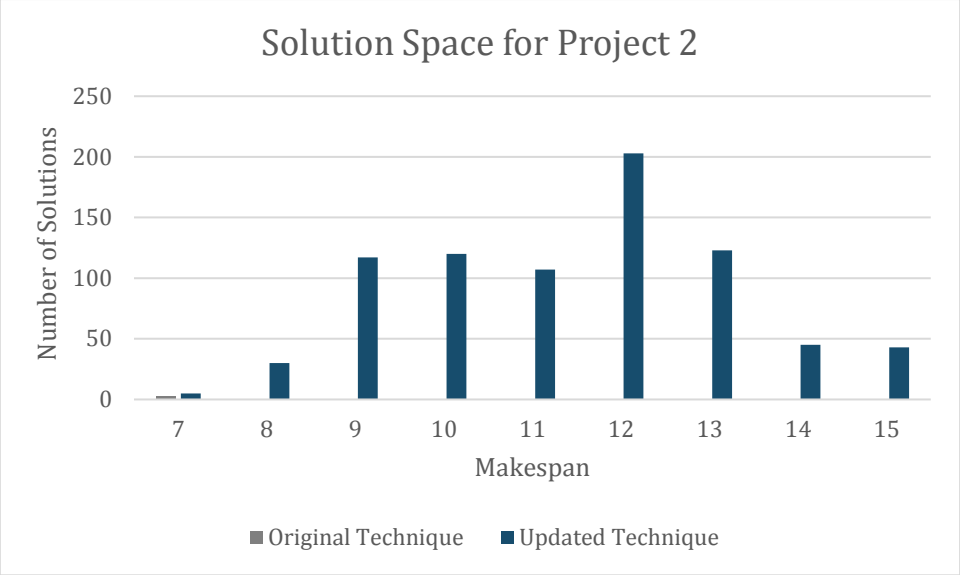


Figure 6: Percentage of Solution Space Explored Using Original Edge Selection Technique

In the three projects simulated, at least 67% of the solution space was undiscovered with the original edge selection technique. The results are considered substantial in terms of the increase in solution space, especially considering how small these projects are in relation to those of realistic size. From the three projects assessed, the percentage of the solution space that was undiscovered appears to increase with increased project size and complexity. It therefore follows that the undiscovered solution space could be significant for all projects especially when considering projects of practical size. Note that one potential area for further research would be to investigate whether a set of alternative resource precedence edge selection criteria exists. If an intelligent set of criteria exists which can limit the solution space without removing any optimal solutions, this would limit the computational effort of Framework One. This avenue is not pursued in this thesis as it falls outside the scope of the research.

What remains is to determine whether the newly available solution space improves the solution quality. In Figure 7 one can see the makespan of the different solutions found using the original edge selection technique, as well as the solutions discovered using the updated edge selection technique, and a comparison can be made. The number of solutions found for each makespan is shown.





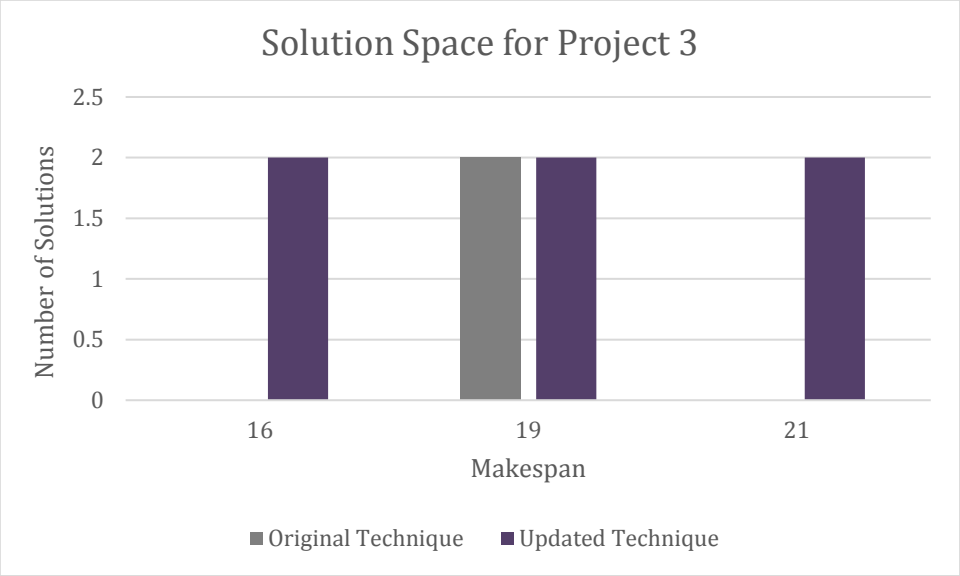


Figure 7: Detailed Solution Space per Edge Selection Technique

As can be seen in Figure 7 above, the new solution space includes solutions which are at least equal to or better than the original solutions in terms of makespan. It can therefore be concluded that the updated edge selection technique improves the number of solutions available, and is capable of providing solutions of improved quality. The updated edge selection technique is therefore permanently incorporated into Framework One. With this modification, it finally becomes possible to accurately compare the results of Framework One and Two.

CHAPTER 8

8 FRAMEWORK ONE RUNTIME OPTIMISATION

As outlined in Chapter 5, Framework One utilises exact procedures which cannot solve problems of realistic size and complexity in a reasonable amount of time. With the updated edge selection technique from Chapter 7, the complexity of the framework is further increased. In this chapter the two phase solution procedure will be revised to assess whether significant improvements to the runtime of the framework are possible. If no significant decrease in runtime can be brought about, a different, non-exact method will be required to find the optimal solution of the RCPSP-TT problem in practice.

8.1 PHASE 1: RESOURCE FLOW PATHS

The concept of a resource flow path has been defined in Chapter 4.3, and the current Original Method used to find resource flow paths in Phase 1 of Framework One has been described in Chapter 5.3. Three alternative methods to find resource flow paths were identified and are now presented. Their performance is also investigated.

8.1.1 *MATRIX METHOD*

The Matrix Method utilises Graph Theory and Path Algebra, and employs the transitive closure of the elementary path set matrix. The elementary path set matrix $\boldsymbol{A}$ can easily be established from a given graph, and the transitive closure of the matrix then calculated. The elementary path set matrix contains all paths of length one between activities, i.e. $\boldsymbol{A_{ij}}$ will contain the edge e_{ij} if it exists. With regards to the equation below the power $\boldsymbol{A}^0$ is equal to the identity matrix $\boldsymbol{I}_w$, and $\coprod$ refers to the union of two matrices. The transitive closure $\boldsymbol{A}^*$ corresponds to the complete path set matrix $\mathbf{W}$, and contains all paths from activity V_i to V_j in position $\boldsymbol{A}^*_{\boldsymbol{ij}}$. The formula used to calculate the closure from the elementary path set matrix is given in Equation 3.

$$A^* := I_w \coprod A \coprod A^2 \coprod A^3 \coprod \ldots = W \qquad \mathbf{3}$$

For further reading, the formula and method details are outlined on page 590 of Pahl & Damrath (2001). It is important to note that this method can only be used when the graph is acyclic else a

stable transitive closure cannot be found. All resource flow paths from the source to sink will be in the upper right position of the closure matrix ($\boldsymbol{A}^*_{\boldsymbol{1n}}$), which is the required output of Phase 1. This method can be further simplified by only calculating the top row of the matrix in each step of the formula, due to only the result in position (1, n) being needed.

8.1.2 *Column Matrix Method*

The Column Matrix Method is an algorithm by which any single column of the transitive closure matrix can be calculated, at the cost of having to execute a solution process. Again, it is only applicable when the graph is acyclic. The following set of equations, Equation 4-7, are utilised to determine the closure; for further reading see page 591 of Pahl & Damrath (2001). The **x** vector can be calculated for each k value, and as can be seen in the Equation 7 this corresponds to the k-th column of the closure of the elementary path set matrix. In this way a required column can be solved for, unlike the previous method where the full closure is calculated directly.

$$\boldsymbol{A}^* = \boldsymbol{I}_w \coprod \boldsymbol{A} \coprod \boldsymbol{A}^2 \coprod \ldots \rightarrow \boldsymbol{A}^* = \boldsymbol{A} \cdot \boldsymbol{A}^* \coprod \boldsymbol{I}_w \quad \textbf{4}$$

$$\boldsymbol{x} = \left(\boldsymbol{A} \cdot \boldsymbol{A}^* \coprod \boldsymbol{I}_w\right) \cdot \boldsymbol{e}_k = \boldsymbol{A} \cdot (\boldsymbol{A}^* \cdot \boldsymbol{e}_k) \coprod (\boldsymbol{I}_w \cdot \boldsymbol{e}_k) \quad \textbf{5}$$

$$\boldsymbol{x} = \boldsymbol{A} \cdot \boldsymbol{x} \coprod \boldsymbol{e}_k \quad \textbf{6}$$

$$\boldsymbol{x} = \boldsymbol{A}^* \cdot \boldsymbol{e}_k \quad \textbf{7}$$

8.1.3 *JGraphT Method*

JGraphT is a free graph library in Java that can support graph objects and provides a variety of algorithms (*http://jgrapht.org*). The algorithm of interest is called *KShortestPaths* and is capable of determining the *k* shortest paths through a graph, where *k* is defined by the user.

For the purposes of this chapter, the *k* value is set as the total number of paths found by the original Phase 1 method. The JGraphT Method is therefore only used to compare the run-time of the algorithm to the other methods, and if this method has the shortest runtime an accurate way to predict *k* will be investigated.

8.1.4 *COMPARISON*

As noted, when utilising the updated resource edge selection technique the graph is acyclic during the solution process. Due to this the graph theory methods that are eligible for use are restricted. Of the four methods being compared, only the Original and JGraphT Methods are compatible with the new edge selection technique. Analysis in this chapter will therefore initially be performed with the original edge selection techniques. If the most promising method is incompatible with the updated edge selection technique further investigation will be pursued.

Four methods to find the resource flow paths through a graph have been presented and are now compared. The average runtime of the method after five separate executions of a project is used as an indicator, where all testing is done with a personal computer in Windows Safe Mode. The use of Windows Safe Mode limits the use of the processor and makes the runtime of different tests comparable. All tests in this thesis were written in Java programming language and executed on a Gigabyte Q1742 notebook computer with an Intel i7-3610QM 2.3 GHz processor. The results produced by each method will be identical, due to each being capable of finding all resource flow paths through a given project. Three example projects are used for testing, Project 1, Project 2 and Project 3. For further details, refer to Appendix A, B and C. The results are available in Table 8-1.

Table 8-1: Phase 1 Method Runtime Comparison

	Project 1			Project 2			Project 3		
Method	**Average Runtime μ (nanoseconds)**	**Coefficient of Variation c_v**	**% Increase in runtime**	**Average Runtime μ (nanoseconds)**	**Coefficient of Variation c_v**	**% Increase in runtime**	**Average Runtime μ (nanoseconds)**	**Coefficient of Variation c_v**	**% Increase in runtime**
Original	2 704 147	1.21%		4 527 705	1.63%		2 716 998	1.02%	
Matrix	3 825 608	1.13%	41.47%	4 874 960	1.32%	79.80%	4 377 326	1.02%	61.40%
Column Matrix	4 858 806	0.76%	79.68%	6 102 178	2.30%	125.18%	5 956 083	0.54%	119.78%
JGraphT	28 386 218	4.70%	949.73%	30 920 708	1.48%	1042.98%	27 637 268	2.56%	921.56%

As one can see from the results the Original Method for finding resource flow paths, of brute force type, has the lowest average runtime. The trend seen in the increase in runtime between the different methods and the Original Method is similar for all three projects, which is a good indicator that the performance of the methods is consistent. The coefficient of variation during each simulation of five project executions can also be seen, and indicates that the results were grouped closely together. Both of the above-mentioned observations support that the data is consistent and the results reliable. The original method used for Phase 1 of Framework One is therefore the best performing method and it will be used from this point onwards.

8.2 PHASE 2: FEASIBLE FLOWS

In Chapter 4.4 the definition of a flow and feasible flow are given, and the criteria for each outlined. The current brute force method of Phase 2 has been outlined in Chapter 5.4 , and various upgrades to this method are investigated in this chapter.

8.2.1 *UNNECESSARY FLOWS ALGORITHM ADDITION*

An algorithm has been developed that will identify and remove unnecessary resource flow paths before Phase 2 commences. The algorithm is analysed in this chapter. A resource flow path is deemed unnecessary when it will never form part of a feasible solution, which would indicate the algorithm used in Phase 2 does not need to consider this resource flow path.

The Unnecessary Flows Algorithm uses a list of integer values, namely the μ_m values of all resource flow paths, ranked in descending order. Some values occur repeatedly. Since only a maximum of a numbers can be selected at one time, only a maximum of a repetitions of values need to be listed in the list of integers. Furthermore, if $\mu_m > \mu_{max}$ for any path – it does not have to be considered. After considering these two criteria, an updated list of μ_m values is created which the algorithm must consider. The algorithm utilises a simple recursion method to asses all possible combinations of a numbers from the set that will add up to μ_{max}. The μ_m values which never form part of such a combination are noted.

The Unnecessary Flows Algorithm is based on the principle used in Lemma 1, where a combination of $\mu_{\boldsymbol{m}}$ values represents a possible solution – as is explained in Chapter 5.4. The algorithm is therefore able to identify which resource flow paths are not able to form part of a feasible flow, and can therefore be disregarded. If a $\mu_{\boldsymbol{m}}$ value forms part of a solution, this indicates that any path with that $\mu_{\boldsymbol{m}}$ value must be considered by Phase 2. This proposed method can limit the number of paths that Phase 2 has to analyse and thus has the potential to significantly decrease the run-time of Phase 2. A visual explanation of the method is given in Figure 8.

Initial μ_m values:	*1 1 2 2 3 4 4 4 4 5 6 7 8*
a:	*3*
μ_{max}*:*	*7*
Updated μ_m values:	*1 1 2 2 3 4 4 4 5 6 7*
Possible solutions:	*1/2/4 and 1/1/5 and 2/2/3*
Final μ_m values:	*1 2 3 4 5*

Figure 8: Unnecessary Flows Algorithm

The effects of the Unnecessary Flows Algorithm on the runtime of Phase 2 of Framework One will now be analysed. The average runtime of Phase 2 is used as an indicator, where this is calculated for five separate executions of a project. Note that due to this testing using runtime as an indicator, it is performed in Windows Safe Mode. Framework One with and without the Unnecessary Flows Algorithm is capable of producing the same results, but the number of paths considered during the Phase 2 may differ and this difference is also recorded. Project 1, Project 2 and Project 3 are tested for both the original and updated edge selection methods in order to note any differences that may arise with increasing network complexity. The results are presented in Table 8-2 along with the coefficient of variation for the data sets in Table 8-3.

Table 8-2: Unnecessary Flows Algorithm Results

	Project 1		**Project 2**		**Project 3**	
	Without modification	With modification	Without modification	With modification	Without modification	With modification
	Average Runtime μ (nanoseconds)					
Original Edge Selection Technique						
Number Resource Flow Paths	*10*	*10*	*21*	*6*	4	4
Unnecessary Flows Algorithm	n/a	1 493 795	n/a	1 333 510	n/a	7 075 492
Phase 2	1 737 972	1 737 972	8 939 925	1 390 181	8 443 541	8 443 541
Total	**1 737 972**	**3 231 767**	**8 939 925**	**2 723 692**	**8 443 541**	**15 519 033**
Percentage Increase in Runtime		*86%*		*-70%*		*84%*
Updated Edge Selection Technique						
Number Resource Flow Paths	*45*	*45*	*99*	*99*	8	8
Unnecessary Flows Algorithm	n/a	2 295 490	n/a	11 255 051	n/a	11 497 977
Phase 2	119 684 444	119 684 444	18 463 846 364 397	18 463 846 364 397	763 644 314	763 644 314
Total	**119 684 444**	**121 979 934**	**18 463 846 364 397**	**18 463 857 619 448**	**763 644 314**	**775 142 291**
Percentage Increase in Runtime		*2%*		*0%*		*2%*

Table 8-3: Unnecessary Flows Algorithm Variation of Results

	Project 1		Project 2		Project 3	
	Without modification	With modification	Without modification	With modification	Without modification	With modification
	Coefficient of Variation c_v					
Original Edge Selection Technique						
Unnecessary Flows Algorithm	n/a	0.84%	n/a	1.13%	n/a	4.02%
Phase 2	0.97%	0.97%	4.58%	1.10%	1.72%	1.72%
Updated Edge Selection Technique						
Unnecessary Flows Algorithm	n/a	0.22%	n/a	0.56%	n/a	0.97%
Phase 2	1.85%	1.85%	0.65%	0.65%	1.52%	1.52%

The use of the Unnecessary Flows Algorithm has a positive effect on Project 2 for the original edge selection technique, yet for the other five simulations no positive effect is noted. For all three simulations utilising the updated edge selection technique, the increase in runtime is 2% or less. This is a very small change when considering that this algorithm has the potential to reduce the overall runtime. Where the improvement in runtime was noted, Project 2 saw a 71% reduction in the number of paths that Phase 2 needs to analyse, an 84% reduction in the runtime of Phase 2 as well as a 70% reduction in overall runtime. Furthermore, note that the coefficient of variation for all simulations is 4% or under, indicating that the variation of the data is minimal.

For the above data, there is an important trend with regards to the ratio of the runtime of the Unnecessary Flows Algorithm in comparison to that of Phase 2. With increasing complexity, represented here by the number of resource flow paths found during Phase 1, this ratio decreases - as can be seen in Figure 9.

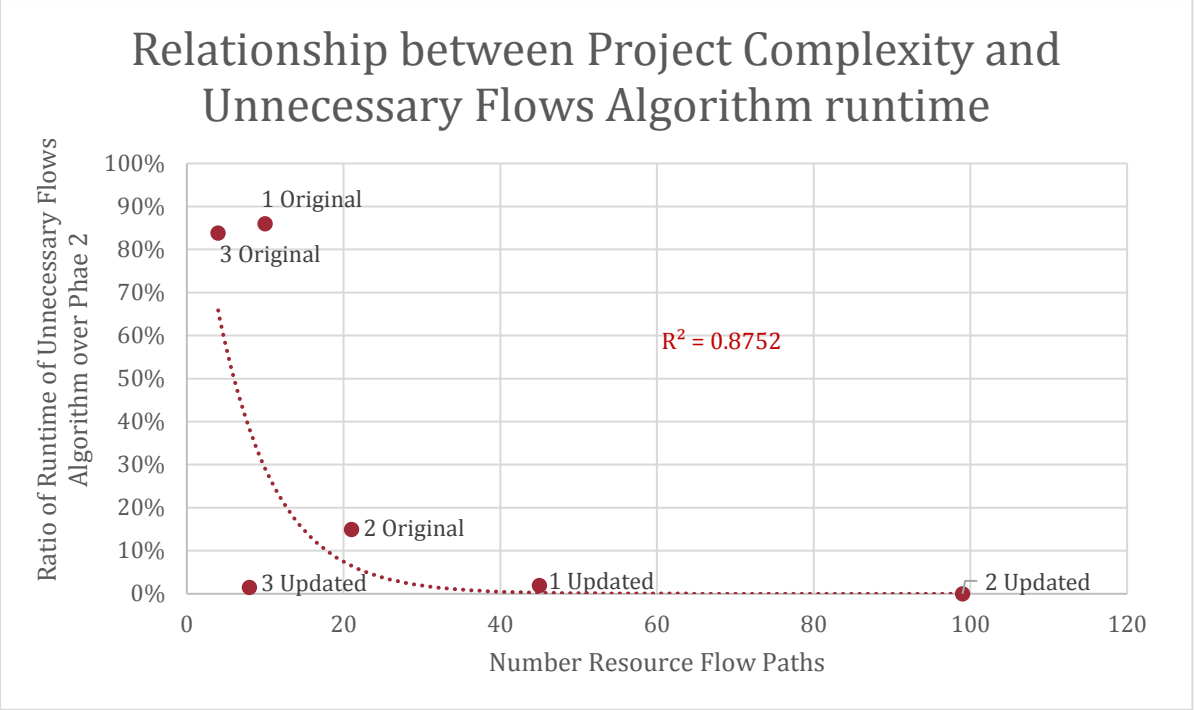


Figure 9: Relationship between Project Complexity and Unnecessary Flows Algorithm Runtime

There is a clear decreasing exponential trend that can be seen in Figure 9 with a R^2 of 0.8752 which indicates a strong correlation. This trend is significant as it indicates that the effect of the Unnecessary Flows Algorithm on total runtime could become insignificant for larger projects.

The data analysed in this section indicates that the Unnecessary Flows Algorithm will only reduce the number of paths in the minority of cases. However due to the very small increase in runtime when the algorithm is used with larger projects the potential benefit could outweigh the loss. It is therefore chosen to permanently incorporate the Unnecessary Flows Algorithm into Framework One.

8.2.2 *Phase 2 Programming Modifications*

Certain programming modifications were made to Phase 2 to improve runtime before any more advanced modifications are investigated. This includes carefully altering certain local variables that function more efficiently as global variables, as well as replacing all standard and enhanced for-loops involving objects with an iterator and while-loop. This proved to be the fastest method after performing a simple runtime comparison.

8.2.3 *PHASE 2 MODIFICATION ANALYSIS*

Modifications are possible that assist the recursion method used in Phase 2 in detecting that a branch of the solution tree will not result in any further feasible solutions. By introducing these modifications and assessing their effect on runtime, Phase 2 could be made more efficient. Two aspects that are potential sources of useful modifications are as follows:

- The number of resource paths selected
- Resource demand of each activity

The specific details of each modification will be presented in the remainder of this chapter. For analysis purposes each new criterion or modification will be added to the original Phase 2 algorithm one after another, the runtime compared to see if an improvement is noted - and a decision made to keep or discard the modification. If a modification is kept it is incorporated into the framework from that point onwards. Note that Phase 2 always produces the same outcome regardless of any modifications. Only the runtimes are affected.

The average runtime of five separate executions of a project is used as an indicator and testing is performed in Windows Safe Mode. Results are generated by simulating Project 1, Project 2 and Project 3 for both the original and updated edge selection techniques for each possible modification.

8.2.3.1 Number of Paths Selected

When considering the number of resource flow paths that have been selected as a factor, two modifications are possible. The first is simple - once a paths are selected no further feasible solutions are available for the remainder of that branch of the solution tree due to all resources already being utilised.

The second modification is more complex. It is based on the formula presented in Equation 8 which will be explained in detail. When this formula yields true, the branch of the solution tree is no longer capable of producing further feasible solutions.

$$\mu_{max} < minimum \sum \mu_m \quad \textbf{8}$$

$$\mu_{max} < \sum \mu_m + (a-b) \times smallest\ \mu_m$$

$$\mu_{max} - \sum \mu_m < (a-b) \times smallest\ \mu_m$$

The values of μ_{max} and a are available before Phase 2 commences and were discussed in Chapter 5.4. The $smallest\ \mu_m$ value is also available, and is the smallest μ_m value of all resource flow paths being considered. $\sum \mu_m$ and b are calculated based on the resource flow paths that are currently selected as part of the solution. $\sum \mu_m$ is that of all the paths at that point in the algorithm and b is the number of paths currently selected.

The logic behind the formula presented above is that when you have selected resource flow paths in number less than a, sometimes there are no paths available that will allow Lemma 1 to be satisfied for the current selection. This can be verified through calculating the minimum $\sum \mu_m$ one can reach by assuming that the remaining paths have $\mu_m = smallest\ \mu_m$. If the minimum $\sum \mu_m$ value that can be reached is greater than μ_{max} then this branch of the solution tree is no longer capable of producing further feasible solutions. An example showing the use of this modification is presented in Figure 10.

μ_m *values:*	*2 2 3 3 4*
a:	*3*
μ_{max}*:*	*8*
smallest μ_m*:*	*2*
Possible solutions:	*2/2/4 and 2/3/3*
Current paths selected:	*4/3*
b:	*2*
minimum $\sum \mu_m$*:*	*4+3+2 = 9*

Figure 10: Number of Paths Modification

The two modifications discussed above are now tested. The average runtime and coefficient of variation is noted for each of the two modifications and the % change in average runtime after each modification is also recorded.

Table 8-4: Phase 2 Number of Paths Selected Modification Results

	Project 1			Project 2			Project 3		
	Average Runtime μ (nanoseconds)	Coefficient of Variation c_v	% change in runtime	Average Runtime μ (nanoseconds)	Coefficient of Variation c_v	% change in runtime	Average Runtime μ (nanoseconds)	Coefficient of Variation c_v	% change in runtime
Original Edge Selection Technique									
Current Method	1 473 626	0.40%		1 433 733	1.17%		5 021 502	0.74%	
Modification 1	1 461 667	0.41%	-0.81%	1 459 793	0.83%	1.82%	5 095 843	0.46%	1.48%
Modification 2	1 483 800	0.41%	1.51%	1 491 297	0.71%	2.16%	5 174 826	0.46%	1.55%
Updated Edge Selection Technique									
Current Method	84 171 686	1.80%		11 609 243 717 427	2.30%		522 250 349	1.46%	
Modification 1	82 391 679	0.66%	-2.11%	11 711 133 866 875	0.85%	0.88%	514 275 170	1.51%	-1.53%
Modification 2	82 130 903	0.65%	-0.32%	11 785 735 462 233	0.99%	0.64%	519 587 523	1.06%	1.03%

From the above data one can see that after Modification 1 the % change in runtime varies from -2.11% to +1.82% for the four simulations, with an overall average decrease of -0.05%. This modification is kept for the following reasons: no significant negative effect on runtime was noted for any simulation; a positive effect on runtime occurred in half the simulations; and a slight decrease in runtime occurred on average.

The addition of Modification 2 causes a change in runtime varying from -0.32% to +2.16% from the original runtime, with an average change of +1.10%. It is again chosen to keep this modification due to the relatively small increase in runtime. With the use of these two modifications no significant increase in runtime is seen, but both modifications are capable of reducing the runtime when the layout of a graph is such that this modification becomes increasingly helpful.

8.2.3.2 Resource Demand Exceeded

The Resource Demand Exceeded modification tracks the number of resources still required by each activity throughout Phase 2, as the position of the algorithm in the solution tree changes. When an activity is assigned more resources than required, this indicates that an invalid solution will be produced by the remainder of that branch of the solution tree. Thus the branch does not need to be explored further.

This modification tracks how many resources are assigned to each activity. Every time a resource flow path is selected to form part of the current solution, the number of resources still required by each activity in the path is decreased by one. Once any activity has a resource requirement less than zero that branch of the solution tree no longer needs to be explored.

The effect of the modification on the runtime of Phase 2 is assessed and data recorded with and without the modification so that a comparison can be made. The average runtime, % change in runtime and coefficient of variation for five project executions is noted. The data is presented in Table 8-5.

Table 8-5: Phase 2 Resource Demand Exceeded Modification Results

	Project 1			Project 2			Project 3		
	Average Runtime μ (nanoseconds)	Coefficient of Variation c_v	% change in runtime	Average Runtime μ (nanoseconds)	Coefficient of Variation c_v	% change in runtime	Average Runtime μ (nanoseconds)	Coefficient of Variation c_v	% change in runtime
Original Edge Selection Technique									
Current Method	1 483 800	0.41%		1 491 297	0.71%		5 174 826	0.46%	
Modification	2 317 801	1.78%	56.21%	2 167 601	3.13%	45.35%	24 451 819	5.29%	372.51%
Updated Edge Selection Technique									
Current Method	82 130 903	0.65%		11 785 735 462 233	0.99%		519 587 523	1.06%	
Modification	11 843 625	7.08%	-85.58%	1 217 826 393	3.50%	-99.99%	282 447 458	6.57%	-45.64%

As can be seen from the data, when the original edge selection technique is utilised the modification increases the runtime of Phase 2. However for the updated edge selection technique the runtime decreases. Note that the updated edge selection technique is permanently incorporated into Framework One and thus a decrease in runtime should always be experienced with this algorithm. The trend can be attributed to the increased complexity of the graph when the updated edge selection technique is used. The increased complexity is due to the number of edges in the graph increasing and therefore the number of paths through the graph also increasing, which is the input of Phase 2. The trend between complexity, represented here by the number of resource flow paths considered by Phase 2, and the average number of times the Resource Demand Exceeded modification disregards a branch of the solution tree is plotted in Figure 11.

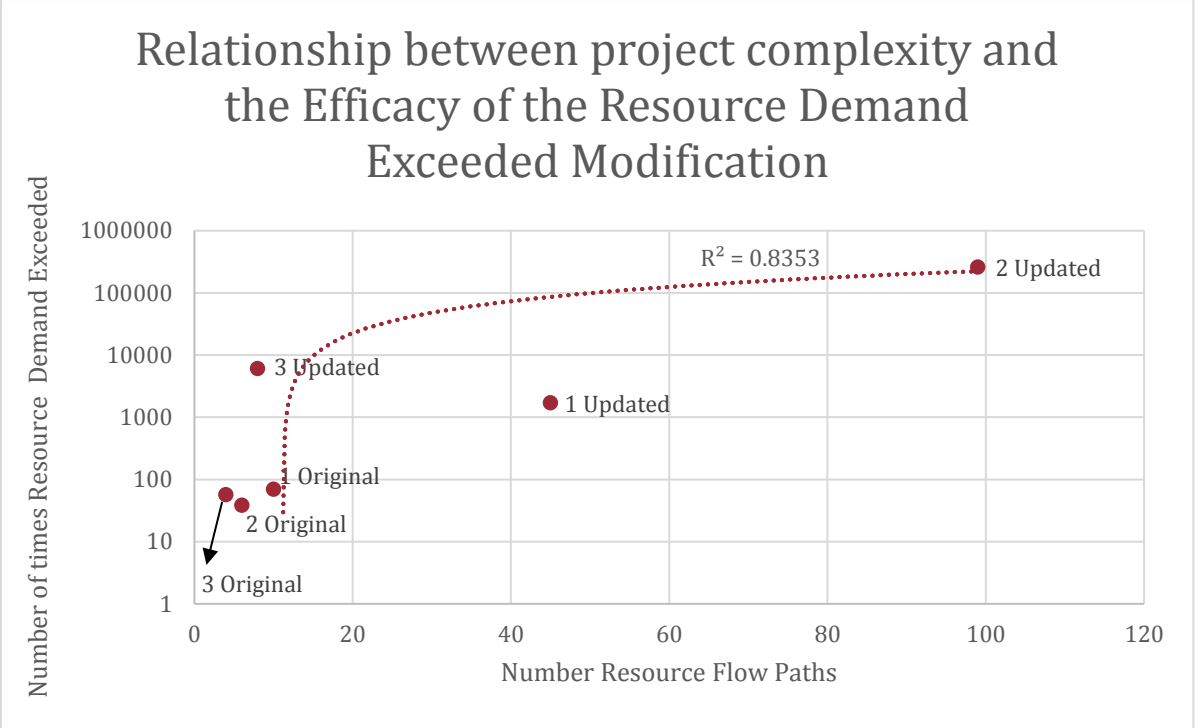


Figure 11: Relationship between Project Complexity and the Efficacy of the Resource Demand Exceeded Modification

In Figure 11 an increasing linear trend can be seen with a R^2 value of 0.8353 indicating a strong correlation. With an increased number of resource flow paths the Resource Demand Exceeded modification becomes more effective and it follows logically that the runtime will also decrease. The Resource Demand Exceeded modification is therefore kept.

8.3 Conclusion

In this chapter, the possibility of decreasing the runtime of Framework One was investigated. If the runtime of Framework One is reasonable for problems of realistic size, this framework will be usable in industry as a project management planning tool. If the framework remains too computationally heavy, another method will be required to find the optimal solution of the RCPSP-TT in practice.

For Phase 1 of the solution process the results indicated that the Original Method was the most efficient. All other methods investigated had a runtime of minimum 40% more during all simulations. This result is significant due to the Original Method being a brute force algorithm,

whereas the other algorithms employed graph theory principles which make the method more systematic yet not more efficient.

The Feasible Flows Phase, i.e. Phase 2, was adapted through the addition of the Unnecessary Flows Algorithm as a precursor as well as three separate modifications. The proposed modifications attempt to shortcut the algorithm through identifying branches of the solution tree that do not need to be explored and all modifications were permanently incorporated into the framework. The overall change in the runtime of Phase 2, including the time taken by the Unnecessary Flows Algorithm, is presented in Table 8-6.

Table 8-6: Overall Change in Runtime of Phase 2

	Project 1		Project 2		Project 3	
	Average Runtime μ (nanoseconds)	% change in runtime	Average Runtime μ (nanoseconds)	% change in runtime	Average Runtime μ (nanoseconds)	% change in runtime
Original Edge Selection Technique						
Initial	1 737 972		8 939 925		8 443 541	
Current	3 811 597	119.31%	3 501 111	-60.84%	24 451 819	189.59%
Updated Edge Selection Technique						
Initial	119 684 444		18 463 846 364 397		763 644 314	
Current	14 139 115	-88.19%	1 229 081 444	-99.99%	293 945 435	-61.51%

As can be seen from the data, four out of six data sets have a significant decrease in runtime of over 60%. To translate the decrease in runtime into more comprehensible terms, Project 2 with use of the updated edge selection technique goes from taking 5 hours 7 minutes 43.8 seconds to taking 1.2 seconds. This is a significant decrease. The reason for the decrease in runtime has been fully explained in this chapter and can be summarised as follows. The decrease in runtime for the data sets using the updated edge selection technique can be attributed to the Resource Demand Exceeded modification, and for Project 2 using the original edge selection technique the decrease in runtime can be attributed to the Unnecessary Flows Algorithm. Note that the original edge selection technique will not be officially used with Framework One and was only tested for data purposes. It is there that increased runtimes were registered. However for the updated edge selection technique, the modifications made in this chapter decrease the runtime in all cases.

As was the goal of this chapter, a decrease in the runtime of Framework One has been achieved. The question remains however whether the computation time of the framework is impractical for use in industry. Project 4 was executed with use of the updated edge selection technique and left to run for 72 hours, after which not even Phase 1 was complete which is the least computationally heavy of the two phases of Framework One. This is a very good indication that for larger projects the runtime of Framework One becomes unreasonable and the framework therefore becomes unusable. It is therefore clear that a meta-heuristic method is necessary to optimise the solution and that an exact method such as Framework One is not the correct approach to solving the RCPSP-TT. Framework One is recommended as a tool to assist in analysing the performance of other methods, and its use in this regard can be of value to future research through producing a full set of available solutions for a project which can be used for comparative purposes.

CHAPTER 9

9 COMPARISON OF FRAMEWORK ONE AND TWO

In the preceding chapters, two frameworks have been presented and modified so that a comparison between them can be made - namely Framework One, utilising exact procedures, and Framework Two, utilising optimisation. Framework One is capable of producing all solutions within a search space and is therefore computationally intensive, whereas Framework Two iterates towards an optimal result based on chosen objective functions.

Framework One and Framework Two will now be compared utilising data from Project 1, Project 2 and Project 3. In Chapter 8 it was shown that Framework One is not practical for problems of realistic size due to exact methods being too computationally intensive, thus this chapter will focus on comparing the quality and range of results. As noted in Chapter 6, Framework Two utilises a slack based objective function during Phase 2 which could influence the quality of the final result in terms of the makespan objective function. The quality of the results provided by Framework Two is therefore of particular interest and needs to be verified.

9.1 PROJECT 1

The full details of Project 1 are given in Appendix A.

9.1.1 *FRAMEWORK ONE RESULTS*

Framework One discovers a total of 77 solutions. The makespan distribution of these solutions was presented in Figure 7 and the distribution of total slack and makespan per solution is presented in Figure 12 which follows. In the figures that follow, the number of solutions discovered per makespan are shown, and colour coding is used to indicate the slack of the solutions.

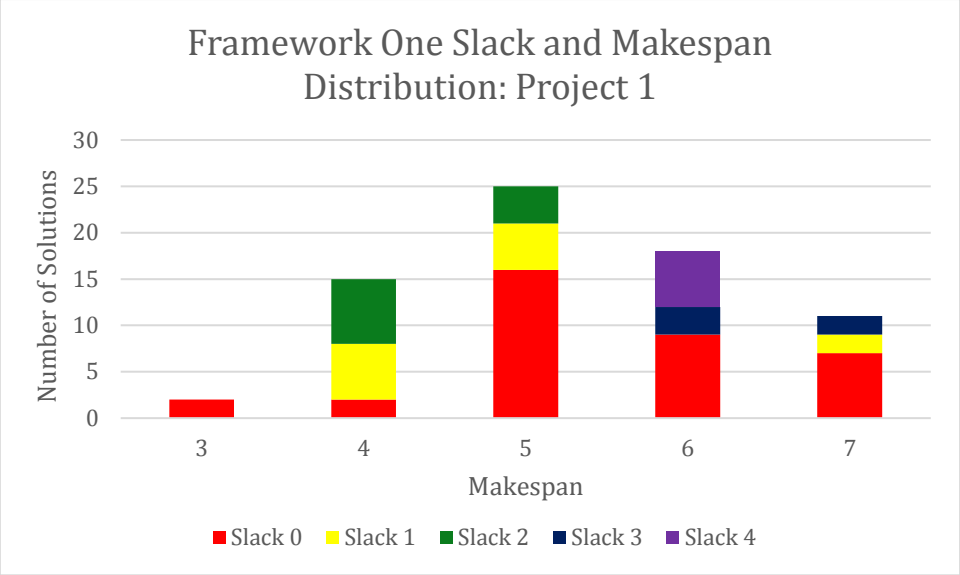


Figure 12: Framework One Project 1 Results

9.1.2 *Framework Two Results*

A data sample size of 10 is used for Framework Two. The results are presented in Figure 13.

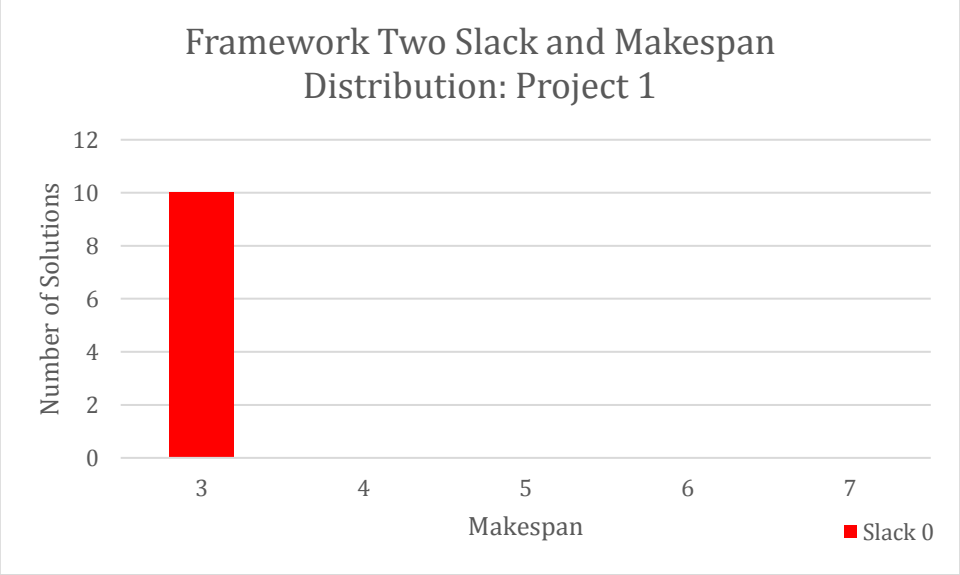


Figure 13: Framework Two Project 1 Results

9.1.3 *Comparison*

Framework Two consistently produces results with the shortest available makespan, but note that these solutions are not necessarily all identical. In Figure 12 two solutions with makespan 3

and slack 0 are seen, namely solution A.1 and A.2 from Appendix A. Out of the 10 data points of Framework Two, 50% was the first solution and 50% the second indicating that the two different solutions appear equally probable.

In the example above the probability of choosing either of the two solutions is approximately equal, but it is important to highlight that this is not always the case. Depending on the baseline schedule produced during Phase 1, the solution space available to Phase 2 is often restricted. For Project 1 two baseline schedules are possible, both with the minimum makespan of 3 units and these can be seen in Figure 14.

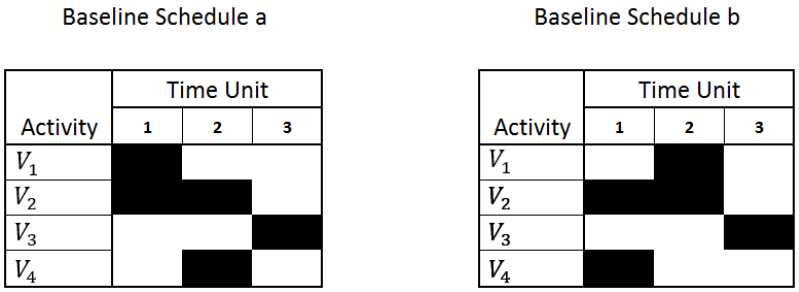


Figure 14: Baseline Schedules for Project 1

For Project 1, solution 1.1 can only be produced by Baseline Schedule a and solution 1.2 can only be produced by Baseline Schedule b. This is due to the ordering of activities, in this case $V_1 \rightarrow V_4$ or vice versa, not being able to change during Phase 2. For Project 1 the different baseline schedules produce identical results in terms of the makespan and slack objective functions, but this will not always be the case. The influence of the baseline schedule therefore needs to be noted, and in certain cases when comparisons are made the baseline schedule used will need to be specified.

9.2 PROJECT 2

The full details of Project 2 are given in Appendix B.

9.2.1 *Framework One Results*

Framework One identifies a total of 793 solutions for Project 2. The makespan distribution of these solutions can be seen in Figure 7 and the distribution of slack values per makespan can be seen in Figure 15.

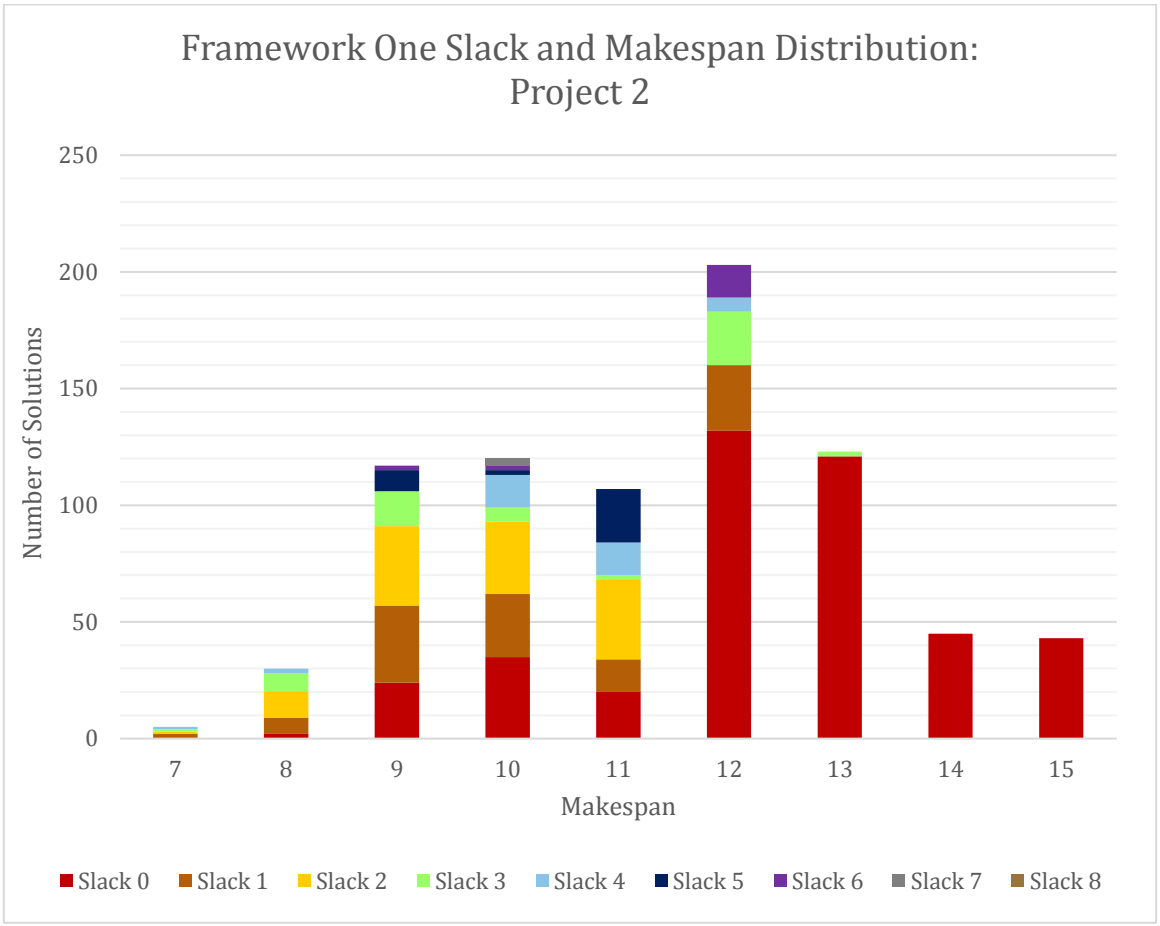


Figure 15: Framework One Project 2 Results

9.2.2 *Framework Two Results*

A data sample size of 10 is used for Framework Two. The results are presented in Figure 16.

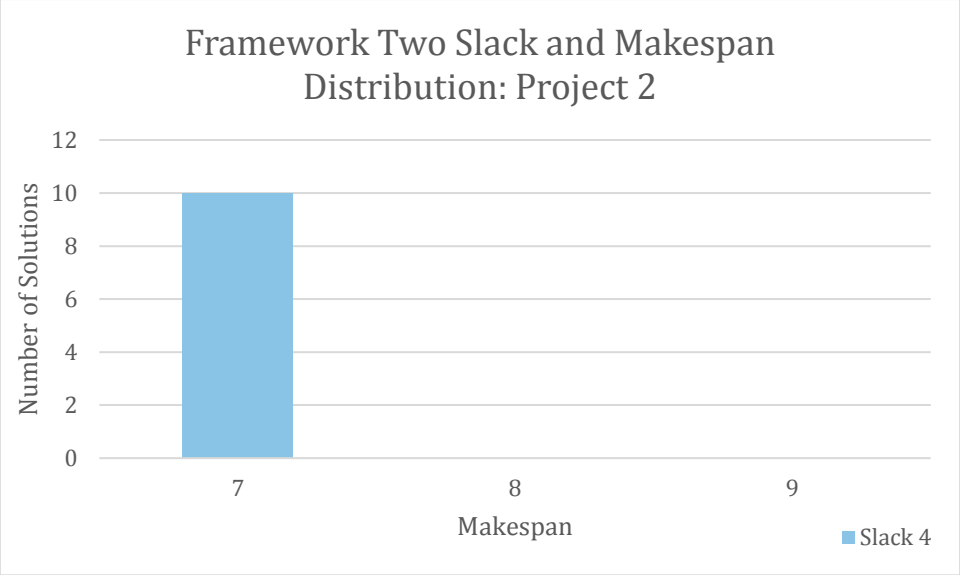


Figure 16: Framework Two Project 2 Results

9.2.3 *COMPARISON*

For Project 2 only one baseline schedule is possible and therefore the solution space is not limited by the baseline that is selected during Phase 1. The solution found by Framework Two for Project 2 is one with a makespan of 7 and slack 4. This is the optimal solution of which there is only one solution with those values. Framework Two is therefore shown to be capable of finding the optimal solution.

9.3 PROJECT 3

The full details of Project 3 are given in Appendix C.

9.3.1 *FRAMEWORK ONE RESULTS*

Framework One discovers a total of six solutions. The makespan distribution of these solutions was presented in Figure 7 and the distribution of total slack and makespan per solution is presented in Figure 17 which follows.

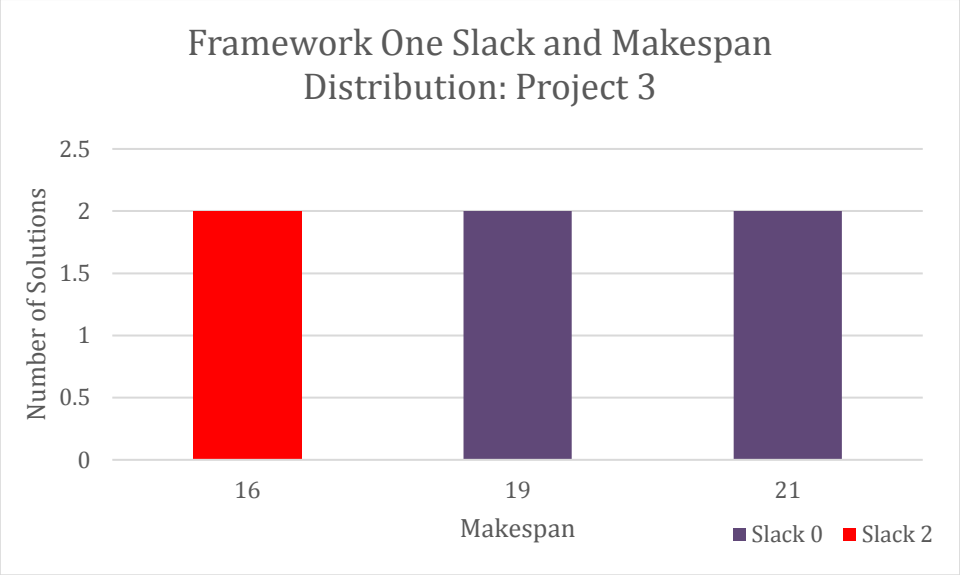


Figure 17: Framework One Project 3 Results

9.3.2 *Framework Two Results*

A data sample size of 10 is used for Framework Two. The results are presented in Figure 18.

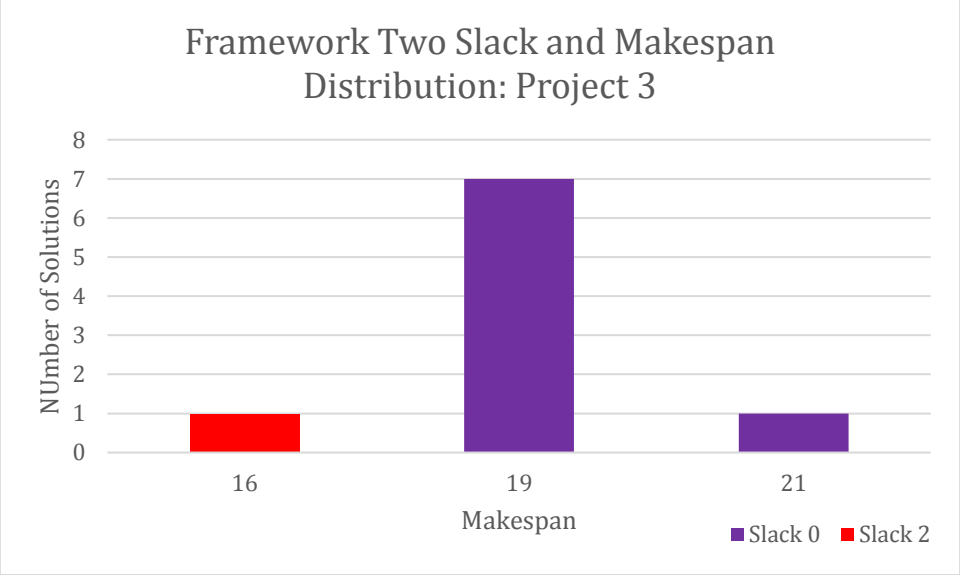


Figure 18: Framework Two Project 3 Results

9.3.3 *Comparison*

Three different baseline schedules are possible for Project 3, and in this case each baseline can only find solutions with a specific makespan. Baseline 1 can only produce two schedules with a

makespan of 16 units each, baseline 2 only two schedules with makespan 19 units and Baseline 3 can only produce two schedules with a makespan of 21. Therefore, the results of Framework Two can be attributed to the baseline schedule selected in Phase 1, not the algorithm in Phase 2.

In Chapter 9.1 the results alluded to each baseline schedule being equally probable for selection – but as one can see here, this is not always the case. The criterion used by Phase 1 of Framework Two to select a baseline schedule is makespan, and all three schedules have the same makespan. Therefore, the reason for the different schedules not being equally probable needs to be accounted for, and can be explained as follows. The specific code used in Framework Two is such that out of all baselines with minimum makespan discovered by Phase 1, the one found last will be selected. This reiterates the need to keep the baseline constant during certain testing in order to avoid the baseline selection affecting the results.

9.4 CONCLUSION

In this chapter it has become clear that Framework Two is capable of producing quality results, but one aspect can hinder the framework from finding the optimal result. The baseline schedule selected during the first stage can affect the solution space and thus the final solution selected by the framework. The baseline is selected in an optimised way, as was investigated in Potgieter (2014), and the framework was proven to produce good results. This factor does however become of particular concern during certain data comparisons, as will be the case in the chapters that follow where the baseline selected will need to be set in order to make accurate comparisons.

Further research into the selection of the baseline schedule could be useful – where a second criterion is introduced to select between baseline schedules with the same makespan. The upgrading of Phase 1 is not investigated due to time restrictions, as well as falling outside of the scope of this thesis. One criterion that should be investigated is the selection of a baseline based on the amount of transfer time the order of activities could induce.

Note that the greedy algorithm used in Phase 2 to make resource allocations is not an optimisation method but rather a heuristic - the algorithm never considers the overall effect of any single resource allocation but rather makes decisions step by step. Framework Two should be modified to use an optimisation algorithm in Phase 2 that will be capable of optimising the

complete set of resource allocations rather than one allocation at a time, as well as incorporating more sophisticated objective functions. This modification will be investigated in the Chapter 10.

CHAPTER 10

10 FRAMEWORK TWO METHOD IMPROVEMENT

Chapter 9 highlighted that Phase 2 of Framework Two can be improved by replacing the current heuristic approach to resource allocation with an optimisation meta-heuristic approach. An optimisation algorithm will now be introduced.

Currently the framework allocates resources sequentially. For each individual allocation, a resource is selected after comparing the effect of all eligible resource allocations. With the proposed modification, numerous resource allocation solutions will be evaluated. The quality of each solution will be evaluated by assessing the objective function. The solutions being considered will be compared and the best solution selected. This updated method allows for more sophisticated objective functions to be utilised with the model, since information from the entire set of resource allocations will be accessible when the objective function is evaluated.

Meta-heuristics have been thoroughly researched and their benefits proven, with the advantage of such algorithms being agreed upon. As previously highlighted, the Ant Colony Optimisation algorithm has been shown to outperform several other algorithms and has become a favoured algorithm in the project scheduling environment (Merkle et al. 2002). An ACO algorithm is therefore selected to optimise the resource allocation process in Phase 2 of Framework Two. A basic ACO algorithm will be implemented and tested in order to evaluate whether an optimisation method is advantageous and whether it does in fact allow for more advanced objective functions. If the outcome of this chapter indicates that the algorithm has potential, there will be an opportunity for future researchers to fine-tune the ACO parameter values and implement a more advanced version of the ACO algorithm.

In this chapter the ACO algorithm will be explained in detail and a comparison between the original and updated Phase 2 algorithm will be made to note any improvement or differences.

10.1 ANT COLONY OPTIMISATION ALGORITHM

The ACO meta-heuristic was developed with inspiration from the method that a colony of ants employs to search for food. Thus an overview of this strategy will be used as a starting point from which to explain ACO. The specific details of the proposed ACO algorithm will then be outlined.

10.1.1 *ANT COLONY BIOLOGICAL SYSTEM*

A colony of ants has the objective of locating the food source closest to the nest and does this using a simple yet efficient method. To start with all ants leave the nest and select a random direction in which to search. If an ant locates a food source, it will take some food and go directly back to the nest marking its path with a pheromone trail.

Pheromone has two properties that assist the colony to converge on the shortest path to a food source. Firstly, pheromone evaporates with time and therefore the further an ant walks the more the pheromone will evaporate. It thus follows logically that shorter paths will have a stronger pheromone scent. Secondly, ants are attracted to pheromone. When faced with choosing between different paths, paths with a stronger scent have a higher probability of being chosen by an ant. The pheromone on good paths is reinforced while the pheromone on longer paths evaporates away.

Given enough time, an ant colony will converge on the shortest path to a food source. For example, assume an experiment was set up with two paths to a food source - where one has a shorter length. Initially, the likelihood of an ant selecting either path is approximately equal. After a short amount of time the pheromone scent on the shorter path will be stronger due to the length of the path. This difference in scent strength will then continue to increase with time, due to more ants choosing the shorter path due to its stronger pheromone scent.

The above example can easily be translated into the problem of finding the shortest path through a graph which is one of the uses of the ACO algorithm. A path through a graph is directly related to a path to a food source and the ants will try to find the shortest path in both cases. In the optimisation environment this concept can also be applied to finding an optimal set of resource allocations for a project. A colony of artificial ants each selects a feasible set of resource allocations and pheromone is laid on the selection. For real ants a longer path would indicate that more pheromone evaporates and for the artificial ants more pheromone will evaporate from

solutions with worse quality. The manner in which the quality of a solution is measured can be specified by an objective function, for example minimising the makespan on the project. As is true of real ants, with time the artificial ants should converge on the solution with the strongest pheromone scent. This theoretical concept is adapted into a formal algorithm as will be explained in the following section.

10.1.2 *Basic ACO Algorithm*

An ACO algorithm consists of a colony of ants with g generations of m ants in each generation. Each ant in a generation will build a set of resource allocations for the project by selecting an eligible resource to allocate to each of the i resource allocations required. Note that each ant will find a feasible solution in the form of a set of resource allocations, such that all resource requirements of the project will have been satisfied. A solution is in the form of an integer vector of size i, where the value in each position is the number of the resource, $j \in R$, that is assigned to each of the i allocations.

The resource allocation decisions of each ant will be influenced by pheromone information, as generated using the best solutions found by the previous generations of ants. Pheromone information is stored in an $i \times j$ pheromone matrix τ, where the graph has i allocations and j resources. The i allocations represent the number of activities, with each activity listed once for each resource it requires. The pheromone matrix is initialised as per Equation 9 with $l = 1$ in accordance with the recommendation of Potgieter (2014).

$$\tau_{ij} = l \tag{9}$$

A high τ_{ij} value indicates that in previous generations when resource j was assigned to allocation i the ants found a good solution. The probability distribution used to select a specific resource allocation is calculated as shown in Equation 10. Only eligible resources λ are considered for allocation and a resource is considered eligible when it is not already allocated to the same activity.

$$p_{ij} = \frac{\tau_{ij}}{\sum_{h \in \lambda} \tau_{ih}} \tag{10}$$

After each generation a portion of the pheromone evaporates and additional pheromone is laid on the best solution trail of the current generation of ants. Pheromone will evaporate according to Equation 11, where ρ is the evaporation rate. This characteristic of the algorithm ensures that previous generations do not influence the behaviour of the algorithm for too long – also assisting the algorithm to converge. Thereafter, the pheromone values are updated according to Equation 12 for each resource allocation that was part of the best solution(s) of the current generation.

$$\tau_{ij} = \tau_{ij} \times (1 - \rho) \quad \textbf{11}$$

$$\tau_{ij} = \tau_{ij} + l \times \rho \quad \textbf{12}$$

The best solution in a generation is determined by comparing the quality of each solution, where the solution quality is represented by a fitness value yielded by the objective function. If more than one ant in a generation has the best fitness value, the pheromone matrix must be updated according to the solution of each of these ants. The pheromone matrix will then be updated according to Equation 12 for each of the best solutions – without any value $\boldsymbol{\tau_{ij}}$ being incremented twice. This concept is explained visually in Figure 19.

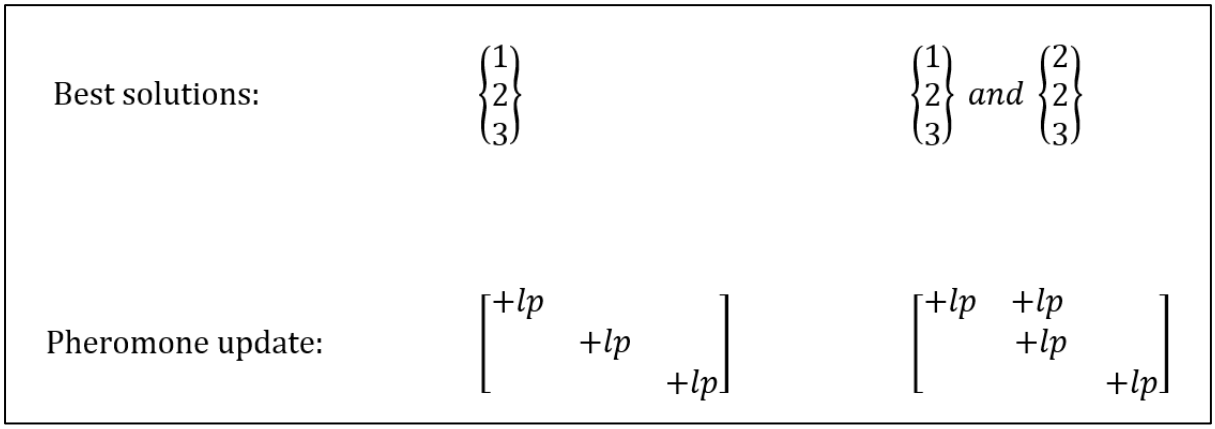


Figure 19: Pheromone Update Example

To determine the fitness value of a solution, an objective function is evaluated as will be discussed in Chapter 10.1.4. At the end of *g* generations the best solution found overall is then selected as the final resource allocation. If more than one solution exists with the best fitness value, one of the solutions is selected at random as the final solution. Numerous solutions with identical fitness are sometimes possible and this is more common when considering smaller projects or simpler objective functions.

10.1.3 *Algorithm Details*

The details of the ACO algorithm implemented will now be outlined as a step-by-step procedure:

1. Initialise pheromone matrix τ as per Equation 9
2. Generate an Ant Colony of m ants, each with a solution in the form of a set of resource allocations
3. Check the solution of each ant in terms of feasibility – if infeasible, let the ant form another solution.
4. Evaluate fitness of the solution of each ant by evaluating the objective function.
5. Evaporate pheromone according to Equation 11
6. Update pheromone matrix according to Equation 12 for best solution(s) in generation
7. Repeat Steps 2-6 for g generations
8. Select best solution
9. Make final resource allocations according to selected solution

10.1.4 *Objective Function*

Given a set of resource allocations as selected by an artificial ant, the solution of the ant needs to be assigned a fitness value. The fitness values are used to compare the solution quality of different ants. An objective function is selected and used to assign a fitness value - the higher the fitness value, the better the solution.

After an ant has formed a feasible solution of resource allocations, a wide variety of information is available that can be used by the objective function. This includes details of each resource allocation as well as all information regarding the schedule that the set of resource allocations will produce, such as the slack, start time and end time of each activity. Note that when evaluating the objective function, only activities with non-zero duration and non-zero resource demand will be considered.

10.2 Comparison

A comparison is now made between the original and updated method of Phase 2 of Framework Two. Phase 2 of Framework Two previously used a greedy algorithm in which each allocation made was the one with the least negative effect on slack, as was discussed in Chapter 6. The updated method is the ACO algorithm as discussed in this chapter. Project 1-5 are analysed, where

the baseline is kept constant for each project so as to make the available solution space identical during each test. For further details regarding Project 1-5 refer to Appendix A, B, C, D and E.

Four separate configurations will be compared in this chapter, where the first configuration is the original method. The updated ACO method will be tested with two elementary functions as well as a random function as objectives. The three objective functions are as follows:

1. Shortest makespan: Fitness = $\frac{1}{Makespan}$
2. Maximum total slack: Fitness = $\sum_{j \in V} slack_j$
3. Random allocation: Fitness = $random\ value$

For the ACO of the updated method the values of colony size m, number of generations g and evaporation rate ρ are set as 15, 100 and 0.15 respectively. Note that ACO parameter optimisation falls outside of the scope of this thesis and should not be of critical importance due to the relatively small size of the projects being tested. The one exception is Project 5 and thus special attention should be paid to these results. The selection of parameters is not an easy task and once good values are found they are not necessarily applicable to all problem instances (Potgieter 2014). Parameter selection can be optimised once the new methodology and optimisation functions are analysed and configured.

Objective function 3 will select a randomly generated fitness value between 0 - 10 to assign as the fitness value of the solution and consequently the final solution produced by this objective is also random. This objective is employed to simulate the current method used in industry to make resource allocations – one where no tools are used to assist project managers in making resource allocations. Resource allocations are not technically performed arbitrarily but project managers do not generally consider slack when making resource assignments – thus the results are somewhat random in this regard (Potgieter 2014).

The four configurations for comparison can therefore be summarised as follows:

- Configuration A: Original Method
- Configuration B: Updated Method Makespan Minimisation (Objective Function 1)
- Configuration C: Updated Method Slack Maximisation (Objective Function 2)
- Configuration D: Updated Method Random Objective (Objective Function 3)

10.2.1 *RESULTS*

The four configurations discussed in the previous section are now analysed and the results are presented in Table 10-1.

For configurations A through C, three project executions are completed and the average makespan and slack are displayed. Note that the average of multiple project executions is used in order to minimise the random component of an ACO. The column *Constant* indicates whether the three executions produced the same solution. For Configurations B and C, the average number of generations to reach the optimal solution $\bar{g}$ is also given.

10 project executions are completed for configuration D and the best solution is shown, where for the purposes of this comparison the best solution is the one with minimum makespan. If more than one of the 10 solutions has minimum makespan, the one with maximum slack is selected.

For comparison purposes the following three global bests are also presented, where all results for all configurations of a given project are considered:

- Best Makespan Solution: Smallest makespan and, if duplicate solutions exist, largest slack
- Next Best Makespan Solution: Second smallest makespan and, if duplicate solutions exist, largest slack
- Best Slack Solution: Largest slack and, if duplicate solutions exist, smallest makespan

Table 10-1: Framework Two Phase 2 Method Comparison

	Makespan	Slack	$\bar{g}$	Constant	Makespan	Slack	$\bar{g}$	Constant	Makespan	Slack	$\bar{g}$	Constant
	Project 1				Project 2				Project 3			
A	3	0		yes	7	4		yes	16	2		yes
B	3	0	2.33	yes	7	1.33	4.67	no	16	2	1	yes
C	6	4	1.67	yes	12	10	4	yes	16	2	1	yes
D	4	2			9	6			16	2		
Best Makespan Solution	3	0			7	4			16	2		
Next Best Makespan Solution	4	2			9	6			16	2		
Best Slack Solution	6	4			12	10			16	2		

	Project 4				Project 5			
A	65	7		yes	181	136.33		no
B	66.67	20.67	15.67	no	235.67	258	46	no
C	71	40	1.67	yes	292.33	569.67	48.33	no
D	70	8			261	230		
Best Makespan Solution	65	7			175	118		
Next Best Makespan Solution	66	25			178	137		
Best Slack Solution	71	40			285	699		

10.2.2 *Discussion and Conclusion*

A variety of conclusions can be drawn from the data presented in this chapter, and the relevant points will now be highlighted and discussed.

When comparing the original method to the updated method in terms of the makespan minimisation objective both perform similarly. However when the results of Project 5 is assessed, one sees that the original method is able to better minimise the makespan. The updated method, i.e. configuration B, has an average result that is worse than configuration A by 30.2%.

As noted, the size and complexity of Project 5 raised the question as to whether the basic parameter values used would allow the ACO to adequately search the solution space. The results are therefore suspected to be due to the ACO parameter values used, which could cause the search of the solution space to be insufficient. Two different factors support this hypothesis. The first factor is the $\bar{g}$ value of 46 for Project 5 configuration B where a maximum of 50 generations have been allowed. This high value indicates that with more generations of ants there is a chance that a better result could be found. The second factor is the amount of solution space configuration B explores, namely 50 generations of 15 ants each which is a maximum of 750 different solutions. The total solution space is approximately 6669, where each one of the 351 resource allocations could be allocated one of 19 resources, remembering that some solutions would be disregarded due to being unfeasible. Therefore, configuration B has only explored a small percentage of the solution space with the current parameter values. This accounts for suboptimum results. Parameter configuration is a complex task and falls outside of the scope of this thesis.

Note that both the original and updated methods perform well when compared to the random objective. – Therefore one can conclude that both methods would be an improvement from the current industry standard. Regardless of only basic parameter values used the ACO proved to find optimal results in most cases and near optimal results when a bigger project was tested.

The focus will now turn to comparing the slack in the given results. An objective of only slack maximisation is not realistic as this would mean that no regard is taken for the increase in makespan during the optimisation process. It is however of interest to note the performance of configuration C with regards to slack maximisation. For projects 1-4 configuration C consistently finds the solution with most slack during all three project executions but for Project 5 the results are not consistent. The average slack of the three executions is 18.5% lower than the optimum

which was found by configuration C, where the optimum was 699 units and the other two results 513 and 497 respectively. An optimisation method should reach near optimum and a trade-off is often made between computation time and the % deviation from the optimum. Considering how little computation time is taken when using this basic ACO, an 18.5% difference is an excellent result. With increased computation time due to a wider and better directed search a better result could be achieved.

One can now conclude that the different methods all achieve good results with the original method achieving particularly excellent results in terms of makespan minimisation. The ACO parameters of the updated method need to be further configured to allow the method to achieve better results.

Lastly, one interesting point needs to be highlighted. The difference between the best solution and next best solution is of particular interest and the amount of slack gained from the increase in makespan will now be noted for each project:

- Project 1: 1 unit increase in makespan = 2 unit increase in slack (200%)
- Project 2: 2 unit increase in makespan = 2 unit increase in slack (100%)
- Project 3: All solutions in available solution space are identical
- Project 4: 1 unit increase in makespan = 18 unit increase in slack (1800%)
- Project 5: 3 unit increase in makespan = 19 unit increase in slack (633%)

The results above are clear – not only should one consider slack as the second priority after makespan minimisation but as part of a dual optimisation. The construction industry is volatile and the importance of slack in such an environment has been discussed. Many project managers and businesses would consider trading a slightly longer schedule for an increase in slack to increase the robustness of a schedule. This is where the power of the updated method cannot be disputed. One is able to introduce a function to allow a dual optimisation of makespan and slack according to a ratio of the user's choice. A function of this nature will be investigated in Chapter 11.

CHAPTER 11

11 OBJECTIVE FUNCTION SELECTION

With the proposed upgrades to Framework Two, the first step has been made towards optimising the RCPSP-TT problem with the use of a sophisticated objective function. What remains is to demonstrate the types of objective functions that can be utilised and to propose an objective function that is a suited for use in the construction phase of civil engineering projects.

In this chapter an overview will be given of the different types of objective functions applicable to the RCPSP. Two objective functions will be presented which simultaneously minimise the makespan and maximise the slack in a schedule. The first will maximise the total slack and the second will maximise the distributed slack, where slack is weighted according to the importance of an activity having slack. This weight can be based on a combination of the risk of delay and effect of delay of the activity.

11.1 INTRODUCTION

When optimising the RCPSP, the selection of an objective function is crucial due to the magnitude of different feasible schedules that are possible. Even when selecting makespan minimisation as the main objective, there are still often many schedules with similar if not identical makespan. Numerous different objective functions have been formulated for the RCPSP. The eight categories of objectives will now be presented and the relevance of each noted; refer to Hartmann and Briskorn (2008) for an extensive overview or Potgieter (2014) for a summary.

- **Time-Based Objectives:** An object of this type aims to optimise a time-base aspect of the project. Examples of such functions include minimising the lateness or tardiness in a project with the most popular being makespan minimisation. In the engineering sector the final deadline of projects is generally the most important and thus makespan minimisation is the priority in most cases. In the construction phase of civil engineering projects the final deadline is also extremely important and can often lead to major penalties if not abided by.

- **Robustness-Based Objectives:** A robust schedule is one in which the effect of delays is limited. The more slack an activity has the less chance that a delay experienced by that activity will affect the remainder of the project. In the construction sector, projects are plagued with delays and time-overruns due to the complexity of projects and the large number of role players meaning robustness is of particular importance. Robustness is generally achieved through slack maximisation where various methods of maximising slack have been proposed.

- **Objectives for Rescheduling:** Rescheduling is typically done when delays or unexpected circumstances dictate that the schedule is no longer valid. When this occurs the schedule should be shifted such that the entire schedule does not change which will minimise any further delays. Due to the use of resource precedence edges in the current model, activities can simply be shifted and the project does not need to be rescheduled when a change occurs, thus the new schedule will be as similar to the original schedule as possible. Rescheduling objectives are therefore already dealt with in the structure of the current model.

- **Objectives Based on Renewable Resources:** An extension of the RCPSP is possible where the number of resources available in the project is not fixed but instead optimised based on an objective such as the total cost and makespan of the project. Often increasing the number of resources can lead to a decrease in makespan but a rise in total cost and thus requires careful selection. The model being dealt with in this thesis is not of this type and therefore such objective functions are not applicable.

- **Objectives Based on Non-Renewable Resources:** This group of objective functions are similar to those based on renewable resources but where the focus is on projects in which the resources are predominantly non-renewable. Again, these objective functions are not applicable to the current model.

- **Objectives Based on Costs:** When cost-based objectives are used, the model generally allows activity durations to be shortened for a certain cost which is also known as project crashing. These models then typically attempt to achieve a schedule with a specified deadline while minimising cost. This implies that resource requirements are not accounted for in the model but instead calculated after the final schedule is produced. These objectives are therefore not applicable to the model used in this thesis.

- **Net present value objectives:** Objectives of this type focus on the cash flow throughout a project and ensuring that there is always a positive cash balance. Such objectives fall outside of the scope of this thesis. Once a model is shown to be capable of optimising sophisticated objective functions for the construction environment, further work can be done to incorporate objectives involving the costs and other financial matters.

- **Multi-objectives:** All categories of objectives listed so far are focused on optimising a single objective function but it is of course possible to combine multiple objectives. A common strategy is to combine functions and weight each component based on the needs of the model and user. The model being discussed will employ this strategy. Where previously a multi-objective was utilised but split into one objective per phase of the solution process, a single multi-objective will now be used for Phase 2 of the solution process.

It should be clear that the main priority when scheduling in the construction environment is to minimise the makespan. Due to the complexity and volatility of such projects a robustness-based objective is an intelligent selection as a dual objective. A multi-objective approach will now be formulated and presented for the RCPSP-TT.

11.2 Total Slack And Makespan Function

A function will now be presented that combines the two objective functions discussed namely makespan minimisation and slack maximisation. Instead of utilising the functions as presented in Chapter 10.2 each function is normalised. The normalised slack maximisation function is presented in Equation 13 and the normalised makespan minimisation function in Equation 14. Note that the total slack in the schedule is calculated as $slack_T = \sum_{j \in V} slack_j$.

$$Fitness = \frac{slack_T - slack_{min}}{slack_{max} - slack_{min}} \quad \textbf{13}$$

$$Fitness = \frac{makespan_{max} - makespan}{makespan_{max} - makespan_{min}} \quad \textbf{14}$$

$slack_T$ – Total slack of the solution of the current ant

$Slack_{max}$ – Maximum slack of all ants

$Slack_{min}$ – Minimum slack of all ants

$Makespan$ – Total duration of the solution of the current ant

$Makespan_{max}$ – Maximum makespan of all ants

$Makespan_{min}$ – Minimum makespan of all ants

The maximum and minimum slack and duration are utilised in the equations above. These refer to the global values i.e. of all solutions evaluated up to the current point in time. After a generation of ants has completed their run, the maximum and minimum slack and duration values are updated. This update takes account of all previous generations of ants including the current generation. Thereafter the fitness values of the current generation of ants are calculated. The best solution discovered by the ACO thus far is always known at any point in the algorithm. Each time that the maximum and minimum values change the fitness of the best solution is updated. This allows for an accurate comparison between the best solution and all future ants.

The purpose of normalising the two objective functions is to ensure the range of values produced by each function is identical and thus assists when combining two different functions. Here, both functions will only produce fitness values which range between 0 and 1. The normalisation of these functions also simplifies the weighting of their importance - due to the range of values of each function being the same, the effect of a weighting factor is straightforward. The two functions are now combined with weighting factors to create a single objective function, as shown in Equation 15.

$$Fitness = \alpha \times \frac{slack_T - slack_{min}}{slack_{max} - slack_{min}} + \beta \times \frac{makespan_{max} - makespan}{makespan_{max} - makespan_{min}} \quad \textbf{15}$$

α, β - Weighting factors

where α + β = 1

α and β are the weighting factors that indicate the relevant importance of the two separate objectives and can be adjusted to suit the needs of the user. The performance of the above objective function will now be investigated as well as the effect of varying α and β**.** Note that for

any testing involving Project 1- 4, i.e. Experiment 1 and 2 described below, a duration of 1 unit as noted in the Appendices is used as 40 hours in the software implementation of Framework Two and in the data of this chapter. This is for ease of calculations and visualisation of the projects in the software implementation.

11.2.1 *EXPERIMENT 1*

For the first experiment, one generation of 200 ants is randomly generated for each project. For each of these 200 solutions, the objective function is evaluated for varying weighting parameters α and β and for each pair of the parameter values the best solution of the 200 ants is noted. Again, the baseline is kept constant to make the results comparable. Note that Project 3 is not tested, as only two solutions are possible when the baseline is kept constant. This experiment is conducted to determine the effect of varying the parameter values and whether a function of this type is valuable. The results are shown in Table 11-1, where colour is used to distinguish between different solutions for ease of view.

Table 11-1: Slack and Makespan Objective Parameter Comparison

		Project 1		Project 2		Project 4		Project 5	
α	β	Slack	Makespan	Slack	Makespan	Slack	Makespan	Slack	Makespan
1	0	240	160	400	480	1600	2840	505	301
0.9	0.1	240	160	400	480	1560	2800	503	278
0.8	0.2	240	160	400	480	1560	2800	503	278
0.7	0.3	240	160	320	360	1280	2720	503	278
0.6	0.4	240	160	320	360	1280	2720	503	278
0.5	0.5	80	160	320	360	1280	2720	418	258
0.4	0.6	0	120	120	280	520	2680	418	258
		80	160						
0.3	0.7	0	120	120	280	520	2680	418	258
0.2	0.8	0	120	120	280	520	2680	197	243
0.1	0.9	0	120	120	280	520	2680	197	243
0	1	0	120	120	280	520	2680	197	243
				80	280				

Key:
Solution with minimum makespan
Solution with maximum slack
Intermediate solution A
Intermediate solution B

Firstly, note that the above data does not indicate that these solutions are the optimum for the selected parameter values. Solutions were randomly generated in this experiment, with only a part of the solution space explored and without the use of an objective function in the ACO. The results above indicate that with varied weighting factors it is possible to obtain at least the following for each project: a solution with minimised makespan, one with maximised slack and at least one solution where a slightly longer makespan is selected to obtain more slack. For example, note the three solutions obtained for each Project with parameter values (0,1), (1,0) and (0.5,0.5) respectively.

The number of solutions available which are intermediaries between the solution with minimised makespan and the one with maximised slack is dependent on the available solution space. As can be seen in Projects 4 and 5, two different solutions can be found of this nature, see parameter values (0.5,0.5) as well as (0.8,0.2). This data indicates that varying parameter values makes dual optimisation possible: the minimisation of makespan and the maximisation of slack simultaneously.

11.2.2 *EXPERIMENT 2*

A second experiment is set up where the objective function is used with the ACO. The ACO is set up with 50 generations of 15 ants and an evaporation rate of 0.15. The parameter values are set as $\alpha = \beta = 0.5$. Three executions of Project 1, 2, 4 and 5 are simulated and the solutions recorded. The results, as shown in Table 11-2, are now evaluated and compared to the best solution selected from 200 random ants in Experiment 1.

Table 11-2: Slack and Makespan Objective Evaluation

Project 1		Project 2		Project 4		Project 5	
Slack	Makespan	Slack	Makespan	Slack	Makespan	Slack	Makespan
80	160	320	360	1440	2640	522	259
80	160	320	360	1280	2680	418	255
80	160	160	280	2840	2720	445	255

Key:
Same fitness as result in Experiment 1
Better fitness than result in Experiment 1

For Project 1 and 2 each of the three solutions obtained have identical fitness to the solution discovered in Experiment 1. Note that the third solution in Project 2 is new and was not one of the 200 solutions evaluated. For Project 4 and 5 the solutions discovered in this experiment are new and have a superior fitness to that of Experiment 1. The details of the fitness values are given in Table 11-3 where the maximum and minimum values are calculated considering only the four solutions listed. Calculating the fitness in this way will be referred to as relative fitness, as this makes the fitness values comparable. If the maximum and minimum values used when calculating the fitness of a solution are not equal, the fitness cannot be directly compared.

Table 11-3: Slack and Makespan Objective Fitness Comparison

Project 4			Project 5		
Slack	**Makespan**	**Fitness**	**Slack**	**Makespan**	**Fitness**
1440	2640	0.55	522	259	0.89
1280	2680	0.25	418	255	0.77
2840	2720	0.50	445	255	0.81
1280	2720	0.00	418	258	0.75

From the data in Table 11-3 it can be seen that the three solutions discovered for Project 4 and 5 during Experiment 2 are superior to that of Experiment 1. The data in Experiment 1 and 2 combined indicate that the objective function works as expected and is capable of optimising the solution of the ACO.

11.2.3 *EXPERIMENT 3*

More extensive testing of the presented Total Slack and Makespan objective function is now performed using Project 5. For each pair of parameter values α and β, three project executions are performed and the solution with the best relative fitness value recorded. These solutions can be seen in Table 11-4. These solutions are then plotted on a graph along with the 200 randomly generated solutions of Experiment 1 as can be seen in Figure 20.

Table 11-4: Slack and Makespan Objective Parameter Comparison

α	β	Slack	Makespan
1	**0**	587	283
0.9	**0.1**	562	289
0.8	**0.2**	538	258
0.7	**0.3**	653	298
0.6	**0.4**	625	284
0.5	**0.5**	445	255
0.4	**0.6**	266	225
0.3	**0.7**	250	240
0.2	**0.8**	322	235
0.1	**0.9**	276	239
0	**1**	271	237

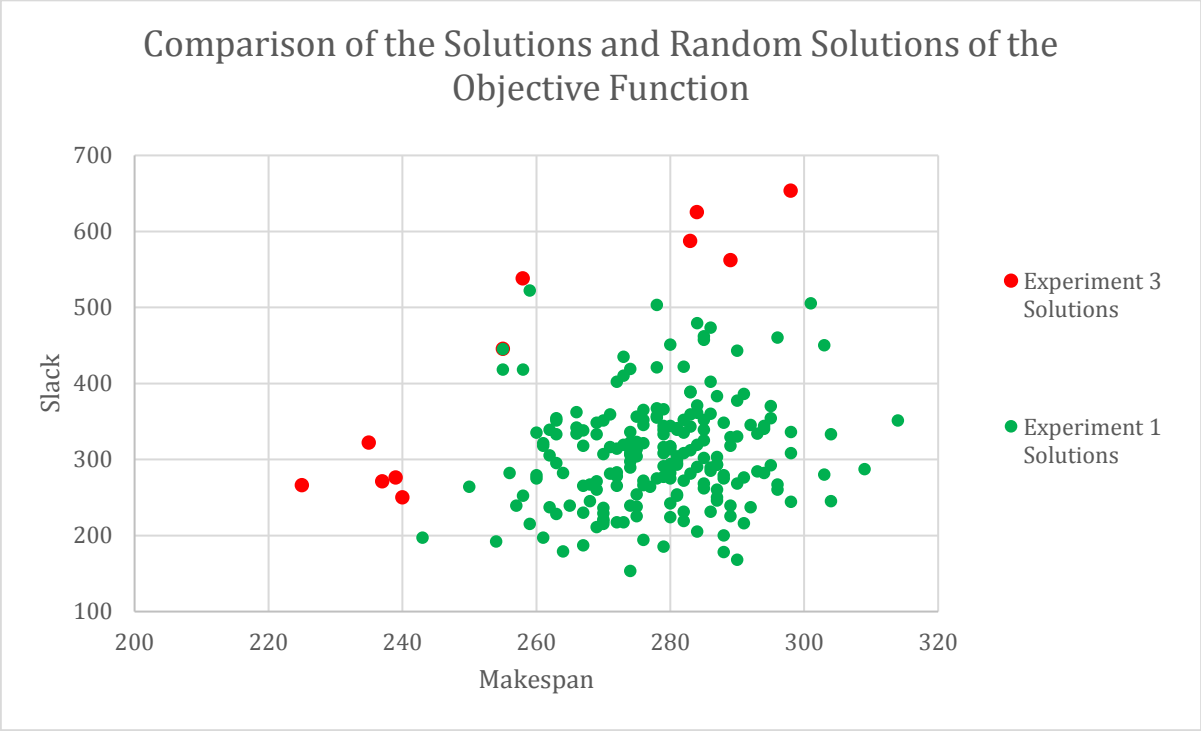


Figure 20: Total Slack and Makespan Objective Function

In Table 11-4 one can see that there is a general trend of decreasing slack and makespan with lowering α value. The trend appears to be very weak but there is a trend none-the-less. Further investigation will be required to determine the grouping of solutions due to changing parameter values. Figure 20 shows a clear difference between the solutions that were randomly generated and those obtained using the objective function. The solutions obtained with use of the proposed

Total Slack and Makespan objective are all grouped in the region with smaller makespan and/or larger slack proposed objective function works as desired.

11.2.4 *EXPERIMENT 4*

Experiment 4 was set up to determine the grouping of solutions with varying parameter values i.e. whether a change of parameters significantly affects the solutions obtained and whether the objective performs as expected in this regard. Three different pairs of parameter values were selected, namely (0.1,0.9), (0.5,0.5) and (0.9,0.1) and six data points obtained for each pair. These data points were then plotted along with the random solutions obtained in Experiment 1.

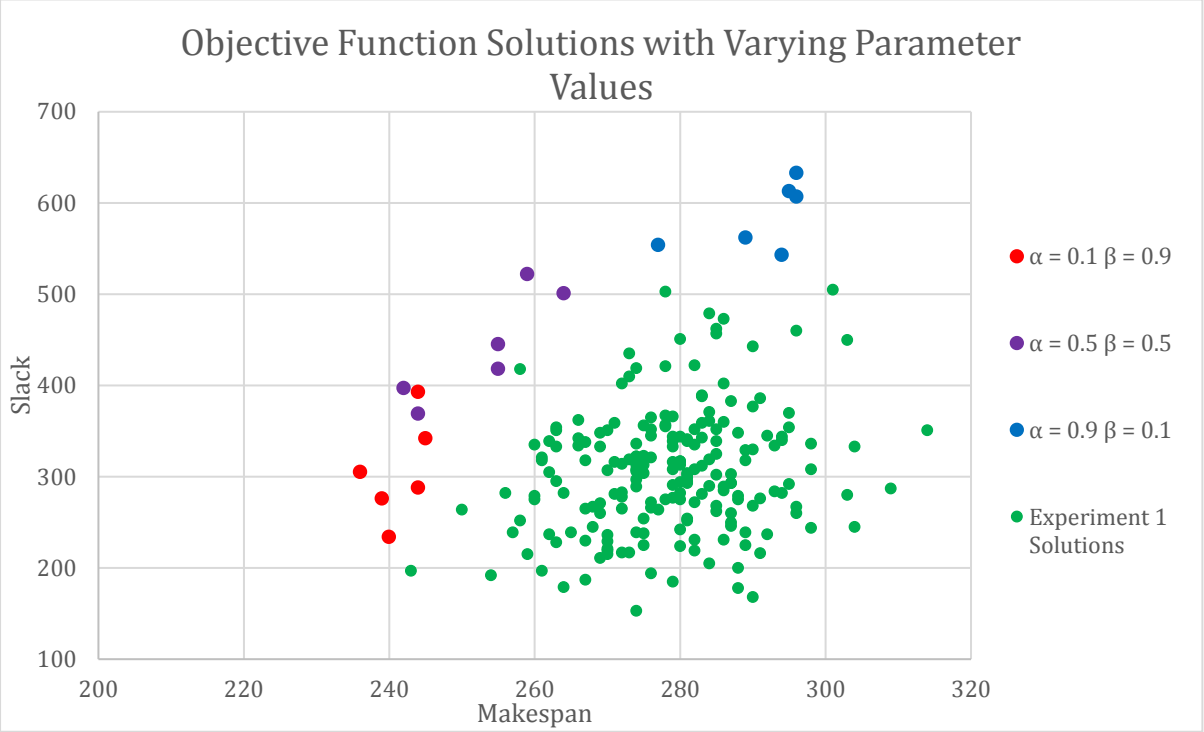


Figure 21: Total Slack and Makespan Objective Function with Varying Parameter Values

As per Experiment 3, it is clear from the above data that in terms of the selected criteria, the solutions obtained with the use of the Total Slack and Makespan objective function are superior to those obtained randomly. The data sets obtained with varying parameter values are also clearly grouped and the data points are shifted towards the criteria which carry more weight per the parameter values selected.

11.3 Distributed Slack And Makespan Function

Maximising total slack is not necessarily the most intelligent robustness-based objective in that other functions have the potential produce more robust schedules, and as has been discussed the distribution of slack is often more important. If the total slack of a schedule is large, this does not necessarily indicate that the slack is spread across the project as all the slack could be linked to a small number of activities.

For an example of the above concept, imagine two schedules where one has a total slack of 10 units and the other has 8 units. The first schedule has one low-importance activity with a slack of 10. The second schedule has four activities with a slack of 2 units each and one of these activities is considered to be of high-importance in terms of requiring slack. Based on the maximisation of total slack the first schedule would be selected but in reality the second schedule should be considered more valuable due to its improved distribution of slack. By distributing slack in an intelligent manner, a project will be more robust and able to better handle disruptions (Potgieter & Van Rooyen 2014). Equation 15 is easily adjusted to account for distributed slack by calculating the weighted slack using Equation 16, instead of total slack as before. The objective function will then guide solutions towards good slack distribution, instead of simply maximum total slack.

$$slack_T = \sum_{j \in V} c\,(j) \times slack_j \qquad \textbf{16}$$

$slack_T$ – Total slack in project

$slack_j$ – Slack of activity j

$c\,(j)$ – Risk value of activity j

It is important to note that with the updated method of calculating slack, the calculation of $slack_{min}$ and $slack_{max}$ as used in Equation 15 is also affected. These values must now be calculated based on the updated slack calculation, i.e. the weighted slack, and will therefore be the minimum and maximum adjusted slack of all solutions evaluated thus far, including the current generation.

What remains is to consider various factors that influence the optimal distribution of slack and formulate a risk model $c(V)$ that determines the risk associated with the delay of an activity. High-risk activities can then be assigned more weight in the updated slack calculation. The risk

associated with delay can be for a variety of reasons that indicate that an activity should have slack. For example: (i) the activity has a high risk of delay and therefore slack will prevent a delay from affecting the schedule, (ii) a delay of this activity will have a large impact on the schedule which can be prevented by slack. One suggestion is to consider slack more important for activities scheduled towards the end of the makespan or for activities with scarce resources (Potgieter 2014).

Another factor that should be considered is the duration of an activity. It is debatable whether a short activity requires more or less slack than one with a longer duration and this is dependent on the model in question and the type of project being considered. For the purposes of the current model, a longer activity is considered to require more slack and is thus weighted more heavily in the risk model. It is assumed that the longer the activity, the more time there is for delays to occur.

A risk model $c(V)$ will now be proposed for use with the Distributed Slack objective function. Based on the literature reviewed for this thesis, this is the first proposed function of its kind for the optimisation of the RCPSP-TT as well as the RCPSP. The format of the function to include a risk model allows future researchers to easily change the risk model in accordance with new research or to suit specific circumstances. To weigh the relative importance of the slack of an activity based on different attributes, a risk model is proposed as follows in Equation 17. The higher the $c(j)$ value of activity A_j is, the higher the risk associated with delay. Therefore more importance is placed on activity A_j having slack.

$$c(j) = \left(\frac{d_j + s_j}{0.5\ Makespan}\right)^{\gamma} \times \left(\frac{r_j}{\bar{r}}\right)^{\theta} \times \left(\frac{d_j}{\bar{d}}\right)^{\delta} \quad \textbf{17}$$

d_j – Duration of activity j

s_j – Starting time of activity j

r_j – Resource demand of activity j

$\bar{r}$ – Average resource demand of all activities

$\bar{d}$ – Average duration of all activities

γ, θ, δ – Weighting factors

This proposed risk model, although relatively simple, shows that a risk model can incorporate many factors with ease and weigh the factors by relative importance. The three factors included will increase the importance of activities that

- are scheduled for completion after the midpoint of the makespan;
- require more resources than the average;
- have a longer duration than the average.

The performance of the Distributed Slack and Makespan objective function will now be analysed and the effect of varying the weights γ, ρ and δ will be evaluated. Note that Experiment 5-7 involve Project 2, and a duration of 1 unit as noted in Appendix B will be used as 40 hours in the software implementation of Framework Two and in the data of this chapter. This is for ease of calculations and visualisation of the projects in the software implementation.

11.3.1 *EXPERIMENT 5*

For the first experiment regarding the Distributed Slack and Makespan objective, a random solution from Project 2 is selected, namely Solution 4 which can be viewed in Appendix B Section B.4. The risk values of each activity are assessed and the values of each component of the risk model displayed according to the following key, with parameter values γ, ρ and δ all equal to one.

- A = $\left(\frac{d_j + s_j}{0.5\ Makespan}\right)^{\gamma}$
- B = $\left(\frac{r_j}{\bar{r}}\right)^{\theta}$
- C = $\left(\frac{d_j}{\bar{d}}\right)^{\delta}$

The purpose of this experiment is to evaluate the way that activity attributes affect the different components of the risk model and therefore the overall risk value associated with an activity. The data is displayed in Table 11-5, where the makespan of the solution is 280 and "T1-2" indicates a transfer task between activity V_1 and V_2. Colour coding is used for ease of reading to indicate the size of the risk value component or risk value itself: [0,1], [1,2] and [2,∞].

Table 11-5: Risk Model for Project 2 Solution 4

Activity	Slack	Duration	Resource Demand	Start Time	A	B	C	$c(j)$	Adjusted Slack
V1	40	80	2	0	0.57	1.09	0.92	0.58	23.02
V2	0	120	2	0	0.86	1.09	1.38	1.29	0
V3	0	40	1	120	1.14	0.55	0.46	0.29	0
V4	0	120	1	0	0.86	0.55	1.38	0.65	0
V5	0	120	4	160	2.00	2.18	1.38	6.04	0
T2-5	0	40	1	120	1.14	0.55	0.46	0.29	0
Average:		86.67	1.83						

Key:
Value in range [0, 1]
Value in range [1, 2]
Value in range [2, ∞]

From the data in Table 11-5, the different factors of the risk model function appear to function as expected. Factor A is >1 for activities scheduled to end in the second half of the makespan i.e. V_3, V_5 and T2-5. Factor B considers activities V_1, V_2 and V_5 high risk due to their resource demand being higher than the average, and V_2, V_4 and V_5 are considered important by factor C due to their duration being above the average. Overall, V_5 is considered the highest risk activity which is logical as it has above average resource demand and duration and is scheduled to finish at the end of the makespan of the schedule.

The output of Experiment 5 indicates that the different factors of the risk model function as intended, and the effect of varying the attributes of activities will be further investigated to determine this with certainty. Note that the values $\gamma = \theta = \delta = 1$ strongly influence the final risk values associated with each activity. This can be seen by the $c(j)$ values which range from 0.29 to 6.04 in the above example. It is possible to lower the influence of the risk model by adjusting these parameters as will also be investigated in the following experiment.

11.3.2 *Experiment 6*

The effect of varying duration, resource demand and start time on the risk value of an activity will now be evaluated. Assuming the average duration and average resource demand as calculated in Experiment 5, example activities with varying attributes are created to determine the effect of

changing attributes on the risk value associated with an activity. The risk value $c(j)$ of each of the example activities is then evaluated for different parameter values γ, θ and δ, namely (1,1,1), (1,1,0), (1,0,1), (0,1,1) and (0.2,0.2,0.2). By making one of the parameters equal to zero, the influence of the factor on the risk model is removed, as anything to the power of zero equals one.

Table 11-6: Risk Model for Example Activities with Varying Attributes and Parameter Values

				γ=1 / θ=1 / δ=1				γ=1 / θ=1 / δ=0			γ=1 / θ=0 / δ=1			γ=0 / θ=1 / δ=1			γ=0.2 / θ=0.2 / δ=0.2			
Activity	**Duration**	**Resource Demand**	**Start Time**	**A**	**B**	**C**	$c(j)$	**A**	**B**	$c(j)$	**A**	**C**	$c(j)$	**B**	**C**	$c(j)$	**A**	**B**	**C**	$c(j)$
V1	80	4	0	0.57	2.18	0.92	**1.15**	0.57	2.18	**1.25**	0.57	0.92	**0.53**	2.18	0.92	**2.01**	0.89	1.17	0.98	**1.03**
V2	80	2	0	0.57	1.09	0.92	**0.58**	0.57	1.09	**0.62**	0.57	0.92	**0.53**	1.09	0.92	**1.01**	0.89	1.02	0.98	**0.90**
V3	120	2	0	0.86	1.09	1.38	**1.29**	0.86	1.09	**0.94**	0.86	1.38	**1.19**	1.09	1.38	**1.51**	0.97	1.02	1.07	**1.05**
V4	80	1	0	0.57	0.55	0.92	**0.29**	0.57	0.55	**0.31**	0.57	0.92	**0.53**	0.55	0.92	**0.50**	0.89	0.89	0.98	**0.78**
V5	120	1	0	0.86	0.55	1.38	**0.65**	0.86	0.55	**0.47**	0.86	1.38	**1.19**	0.55	1.38	**0.76**	0.97	0.89	1.07	**0.92**
V6	80	1	120	1.43	0.55	0.92	**0.72**	1.43	0.55	**0.78**	1.43	0.92	**1.32**	0.55	0.92	**0.50**	1.07	0.89	0.98	**0.94**
V7	120	1	120	1.71	0.55	1.38	**1.29**	1.71	0.55	**0.94**	1.71	1.38	**2.37**	0.55	1.38	**0.76**	1.11	0.89	1.07	**1.05**
V8	120	1	200	2.29	0.55	1.38	**1.73**	2.29	0.55	**1.25**	2.29	1.38	**3.16**	0.55	1.38	**0.76**	1.18	0.89	1.07	**1.12**
V9	40	1	200	1.71	0.55	0.46	**0.43**	1.71	0.55	**0.94**	1.71	0.46	**0.79**	0.55	0.46	**0.25**	1.11	0.89	0.86	**0.85**
Average:	86.67	1.83																		

Key:
Value in range [0, 1]
Value in range [1, 2]
Value in range [2, ∞]

The range of risk values $c(j)$ for each of the five sets of parameter values are as follows:

- γ = θ = δ = 1: $0.29 \leq c(j) \leq 1.73$
- γ = θ = 1 and δ = 0: $0.31 \leq c(j) \leq 1.25$
- γ = δ = 1 and θ = 0: $0.53 \leq c(j) \leq 3.16$
- θ = δ = 1 and γ = 0: $0.25 \leq c(j) \leq 2.01$
- γ = θ = δ = 0.2: $0.78 \leq c(j) \leq 1.12$

The key difference is between the first and last set of parameter values, where the effect of the risk model on the risk value of the activities is much lower when using γ = θ = δ = 0.2. By varying these parameter values the overall influence of the risk model on the distribution of slack in the schedule can be selected – whether the model favours slack on activities with a high risk value significantly, moderately or only slightly.

From the data in the table, as well as the summarised range of $c(j)$ values, the effect of removing the influence of a factor in the risk model can easily be seen. From the structure of the risk model

as well as the data above, it follows logically that one can easily adjust the relative influence of the different factors of the risk model to suit the needs of the user.

11.3.3 *EXPERIMENT 7*

An experiment is set up where four different solutions of Project 2 with minimum makespan are compared based on their adjusted slack values, as obtained with use of the risk model. We assume that the user wants the influence of the risk model to be significant and each factor to be equally important. Thus parameter values γ, θ and δ are set as 1. The four different solutions evaluated are Solution 1-4, as can be seen in Appendix B Section B.1 through B.4 for further details. For each of the four solutions the slack is adjusted according to the specified risk model and the adjusted slack is displayed only for activities which have slack. The results are displayed in Table 11-7.

Table 11-7: Comparison of Adjusted Slack of Various Project 2 Solutions

Activity	Slack	Duration	Resource Demand	Start Time	A	B	C	$c(j)$	Adjusted Slack	Total Slack
V3	80	40	1	160	1.43	0.57	0.50	0.41	32.65	
Average:		80.00	1.75		Makespan: 280				32.65	80
V1	80	80	2	0	0.57	1.23	1.00	0.70	56.26	
V3	40	40	1	200	1.71	0.62	0.50	0.53	21.10	
Average:		80.00	1.63		Makespan: 280				77.36	120
V1	40	80	2	0	0.57	1.00	0.92	0.53	21.10	
V3	120	40	1	120	1.14	0.50	0.46	0.26	31.65	
Average:		86.67	2.00		Makespan: 280				52.75	160
V1	40	80	2	0	0.57	1.00	0.92	0.53	21.10	
Average:		86.67	1.83		Makespan: 280				21.10	40

Based on the total slack of the solution, the solutions can be ranked in descending order as follows: 3, 2, 1, 4. When considering the adjusted slack however, the activities will be ranked 2, 3, 1, 4. The reason that Solution 2 will be favoured when utilising the Distributed Slack and Makespan function is due to activity V_1 and V_3 being considered higher risk than in Solution 3. This higher risk is due to a variety of reasons including: lower average duration and resource demand in Solution 2 as well as activity V_3 being scheduled later.

This experiment is a clear example showing the abilities of the Distributed Slack and Makespan objective function. The function can influence the solution that is selected and ensure that the selected schedule has slack on activities deemed to have a high risk associated with delay by the risk model.

11.4 Comparison of Functions: Experiment 8

For Experiment 8 the ACO algorithm is utilised with 50 generations of 15 ants, the evaporation rate $\rho = 0.15$ and the parameters of the risk model set as $\gamma = \theta = \delta = 1$. For each of the pairs of α and β values in Table 11-8, three data points are collected for both the Total Slack and Makespan objective function and the Distributed Slack and Makespan objective function. Therefore 27 solutions are available for each objective.

Table 11-8: Parameter Values used in Experiment 8

α	β
0.9	0.1
0.8	0.2
0.7	0.3
0.6	0.4
0.5	0.5
0.4	0.6
0.3	0.7
0.2	0.8
0.1	0.9

Three figures are now presented. The first two plot the solutions in terms of slack and adjusted slack against makespan for both the Total Slack and Distributed Slack objectives. This information is presented in Figure 22 and Figure 23. In Figure 24 the ratio of Adjusted Slack to Total Slack is plotted for each solution, which can be thought of as an indication of the amount of slack that is distributed to activities which are considered higher risk - the larger the ratio the better. Each figure is now presented then discussed.

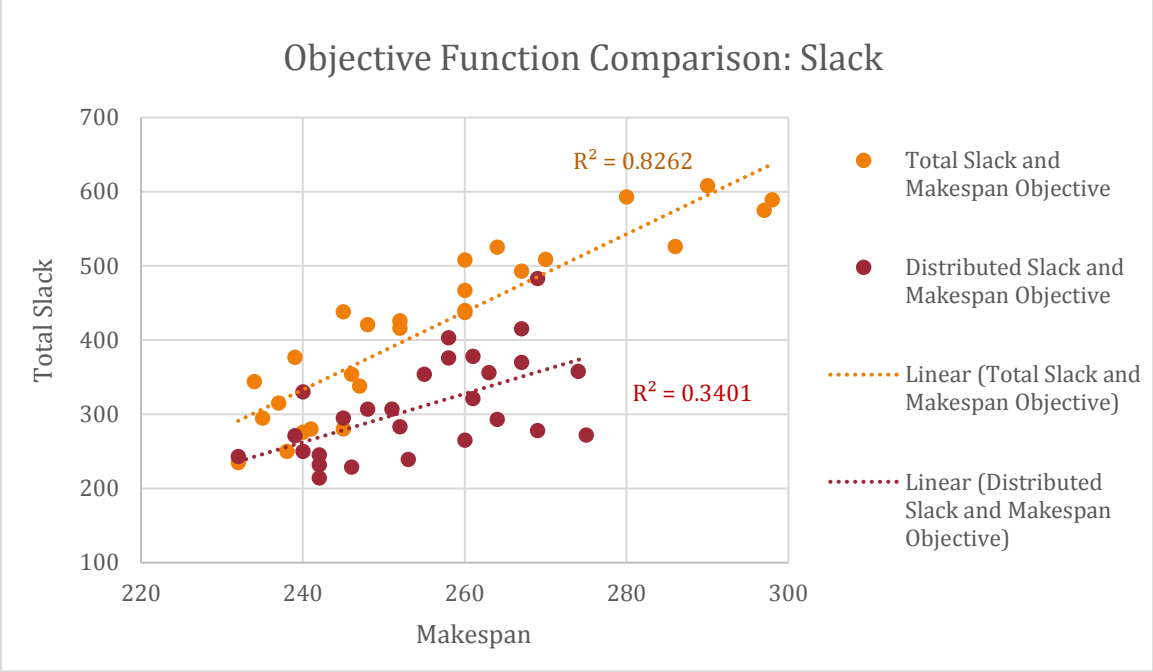


Figure 22: Objective Function Comparison: Slack

In Figure 22 the trend of total slack for both objective functions can be seen. The Total Slack and Makespan objective has superior slack results and the data set has a coefficient of determination R^2 = 0.8262 which indicates a strong correlation. The Distributed Slack and Makespan function has a weak correlation with R^2 = 0.3401, which follows logically due to this objective not optimising total slack but instead optimising the adjusted slack value. This data shows that the Total Slack and Makespan function optimises slack as expected and the Distributed Slack and Makespan function does not yield the same results.

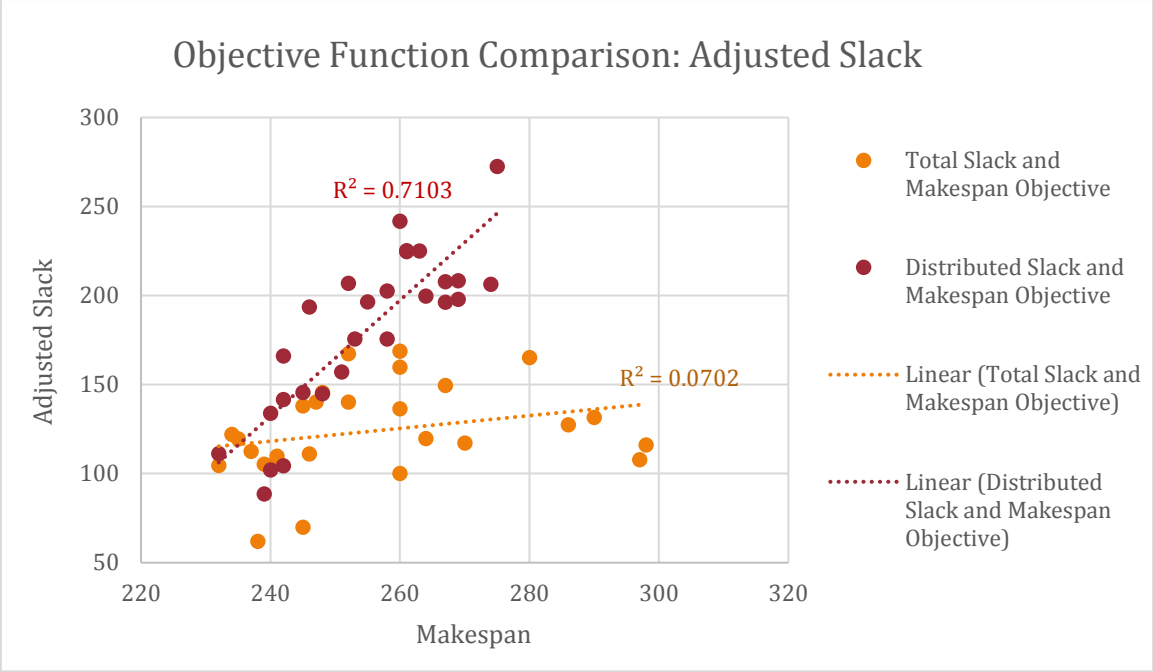


Figure 23: Objective Function Comparison: Adjusted Slack

In Figure 23 it can be seen that the adjusted slack results when utilising the Distributed Slack and Makespan function are superior to those of the Total Slack and Makespan function. The correlation of the data of the Distributed Slack objective is R^2 = 0.7103 and for the Total Slack objective R^2 = 0.0702, which is understandable due to which aspects of the solution each objective optimises. The two figures presented above indicate that with the use of the Distributed Slack and Makespan objective the solutions selected have slack distributed to high-risk activities and the distribution of slack is better than the result obtained by the Total Slack and Makespan objective.

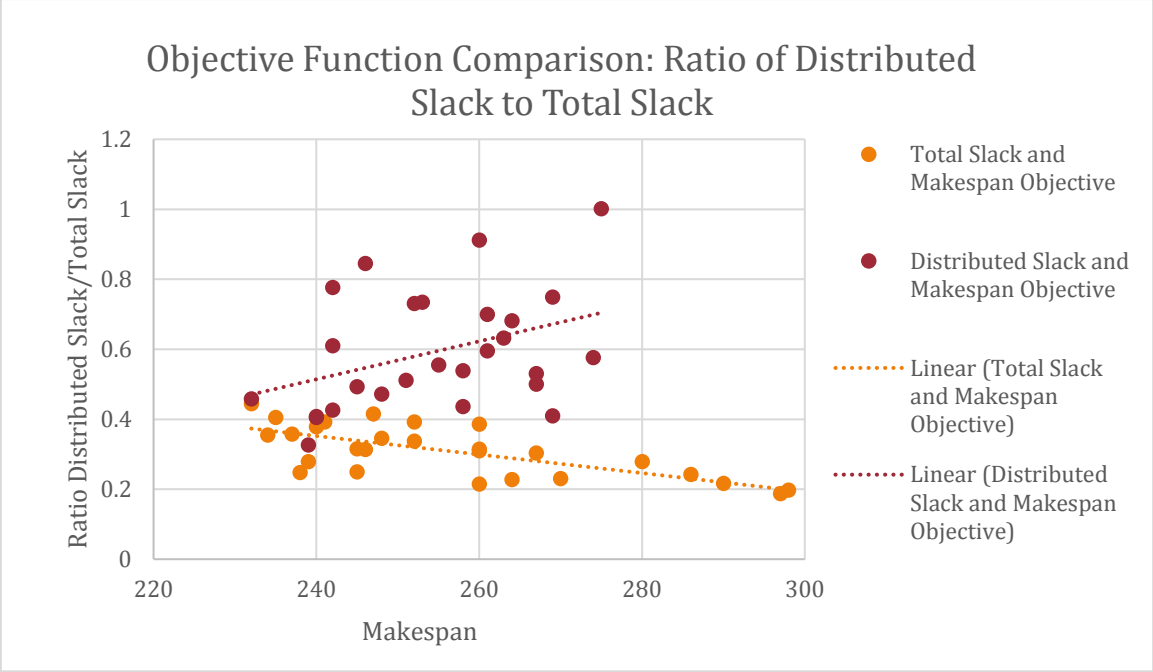


Figure 24: Objective Function Comparison: Ratio of Adjusted Slack versus Total Slack

Figure 24 further supports the conclusion that the use of the Distributed Slack objective yields results with improved adjusted slack values. The results of this section indicate that both functions produce results as intended, and that the Distributed Slack and Makespan function allows the user to distribute slack in an intelligent manner across the schedule.

11.5 Discussion and Conclusion

In this chapter two functions have been presented, namely the Total Slack and Makespan objective and the Distributed Slack and Makespan objective. Both functions are multi-objective functions which should maximise the slack in the schedule while minimising the makespan, with the importance of each objective controlled by a weighting factor. The functions will maximise the total slack and adjusted slack respectively, where adjusted slack is a measure of how well slack is distributed to activities considered to have a high risk associated with delay.

The results of the Total Slack and Makespan objective were compared to a set of random solutions, and the results were shown to be an improvement in terms of makespan minimisation as well as slack maximisation. Therefore the function has the desired influence. The parameter values α and β were also shown to allow the user to select to compromise an increased makespan for increased schedule robustness, a valuable tool for scheduling in a volatile environment.

The risk model utilised in the Distributed Slack and Makespan objective was carefully investigated, and the ability of the risk model to determine the risk associated with delay for a specific activity shown. Furthermore, the weighting factors of the risk model were shown to affect the results as expected. More importance can be placed on high-risk activities by increasing the overall influence of the risk model through higher risk values $c(j)$, achieved by increased weighting factors. The relative influence of the different factors of the risk model on the risk value associated with an activity can also be varied by adjusting the weighting factors.

The format of the proposed Distributed Slack and Makespan objective function allows for further development of the risk model to better suit the needs of the construction industry in general, or perhaps for specific types of projects or company objectives. The hope is that future researchers will continue to develop the risk model and adapt it as required. The risk model can be revised to include numerous other factors, for example the percentage of activities with slack. One could also consider activities with many successors to have a higher risk associated with delay due to the effect a delay of such an activity would have on the schedule. One could argue to include the total slack as a factor in the Distributed Slack and Makespan objective as well. Furthermore, when the model develops to include information regarding cost, objectives of this nature can also be incorporated into the objective function perhaps as a third objective in the multi-objective function.

The results of the experiments discussed in this chapter has shown the ability of both objective functions to influence the selected solution by making a compromise regarding slack and makespan. It has also demonstrated the ability of the Distributed Slack and Makespan objective to ensure that the selected schedule has slack on activities that are deemed to have a high risk associated with delay by the risk model. Both of the aforementioned developments can be of great value to the construction environment, as the project schedule can be selected ensuring that it is as robust as possible within the given restrictions.

CHAPTER 12

12 CONCLUSION AND RECOMMENDATIONS

12.1 CONCLUSION

Good project management can be crucial to a company's success, and project scheduling is an important tool in this regard. Schedules must not only be detailed and accurate but should be the optimal schedule for the projects and objectives in question. Optimised scheduling of civil engineering projects has been the focus of research for decades due to the numerous unique challenges presented by the field. However, there is a lack of research regarding scheduling for the construction phase of these projects.

Construction projects are specifically challenging to model due to their complex nature and unpredictability which increases the difficulty of the planning process. Research regarding the optimised scheduling of construction projects can lead to the better management of projects in industry and this could save vast amounts of time and money. The focus of this thesis was the optimised scheduling of construction projects with the hope that advancements in the academic research environment will affect the methods used in industry.

A basic model utilising exact procedures is available for the RCPSP-TT and this model was used as a starting point. The model was developed by Griebenow (2014) and is referred to as Framework One throughout the thesis. The manner in which resource precedence edges were added, i.e. the original resource edge selection technique, was flawed and required modification. An updated edge selection technique was incorporated. This finally allowed for all possible solutions to a project to be discovered using Framework One. The updated edge selection technique was utilised with three simple projects, which showed that at least 66.67% of the solution space was previously unexplored and new solutions of equal or improved quality were now available.

The runtime of the Framework One was then minimised through investigating various improvements, in order to determine if the model can analyse projects of realistic size with reasonable computation time. The modifications led to a runtime improvement of between 61.52% and 99.99% for the three projects tested. As was the aim, a considerable runtime

improvement was achieved. Even with this improvement, the model is too computationally intense. This was evident when Phase 1 of the solution procedure, the less computationally intense of the two phases, could not finish in 72 hours for a project with only 10 activities. A meta-heuristic method would therefore be required to solve the RCPSP-TT if realistically sized projects are to be analysed. Framework One can however be a valuable research tool in future due to the accuracy of results produced.

A model for the optimisation of the RCPSP was created by Potgieter (2014) utilising an Ant Colony Optimisation (ACO) algorithm to select a baseline schedule and thereafter utilising a heuristic method to allocate resources in the schedule. This model, i.e. Framework Two, was first upgraded to suit the RCPSP-TT. This was accomplished through the addition of transfer times to account for the movement of resources on a construction site. Framework One, although too computationally intense, is accurate. A comparison between the two models is therefore made in order to determine the accuracy of the solutions provided by Framework Two. Framework Two was shown to provide accurate results, limited only by the baseline selected during the first phase of the solution procedure. The computation time and abilities of Framework Two are immediately superior to Framework One and further modifications to make the model more suited to the RCPSP-TT were then undertaken.

The heuristic method of Phase 2, resource allocation, of Framework Two was replaced with an ACO algorithm. This modification means that rather than optimising one resource allocation at a time, the model will optimise the entire set of resource allocations. This finally makes it possible to utilise a sophisticated objective function with the model, due to the entire set of resource allocations being known when the objective function is evaluated.

The original heuristic method and updated ACO algorithm were compared. In most cases the results of both methods were of similar, good quality. The ACO algorithm was tested with very basic parameter values, which led to the solution produced being less optimal than the original method when a larger project was analysed, and ACO parameter configuration falls outside of the scope of this thesis. Parameter selection is not an easy task and parameters are dependent on the problem instance in question, which further complicates the configuration of these values.

Construction projects are highly volatile and prone to delays and disruptions. One way to limit the effects of such an event on the schedule is to ensure the schedule is as robust as possible. Slack

maximisation objectives have proven to be beneficial in this regard and could be used as a secondary objective, where the priority remains makespan minimisation. Research was conducted in a multi-objective format instead, which would be more suited to the construction environment as well as a more powerful objective. In this objective, the relative importance of makespan minimisation and slack maximisation can be specified. This allows a project manager or business to opt for a schedule with a slightly longer makespan in return for an increase in slack. The value of a function such as this cannot be disputed. This total slack and makespan objective function was tested and shown to direct the solution search towards an optimal result as desired. Varying the relative importance of the two objectives further directs the search accordingly.

A more sophisticated objective function was presented to conclude the research. Instead of maximising total slack the distributed slack is maximised. A risk model is used to assign a risk value to an activity, which determines the risk associated with the delay of that activity. Importance is placed on an activity having slack according to its risk value, and this is considered by the objective function. The proposed risk model includes factors for the duration, start time and resource demand of an activity, where the influence of each factor can be varied with weighting factors. Through varying the weighting factors in the distributed slack and makespan objective function, the user is able to vary the effect of the different factors on the risk value.

The distributed slack and total makespan function was tested and shown to direct the solution search as desired, towards functions with increase distributed slack. This development is significant. Not only is the model capable of optimising the makespan and slack in a project simultaneously, but the importance of slack on different activities can be specified.

The objectives of this thesis were met, and a model capable of optimising the solution of the RCPSP-TT has been presented. With this development, further research can now be conducted to make the model ready for industry, and a significant impact will be made when the model is eventually used in practice.

12.2 Recommendations for Further Work

Project scheduling is a broad topic and the optimisation of the RCPSP-TT a complex problem. Therefore many areas exist where further developments can be made. Potential topics for further research will now be presented.

12.2.1 *Multiple Resources*

The RCPSP-TT optimisation model presented in this thesis requires one major improvement to be practically usable: support for numerous resource types. Framework Two was originally capable of handling multiple resources but with the adaptions made in this thesis the functionality was removed. With the inclusion of transfer times, the optimisation of the resource allocation procedure and the inclusion of complex optimisation functions, the use of multiple resources becomes incredibly challenging. The fundamental software structure is capable of handling multiple resources but the entire solution procedure will need to be adapted, and research will be required regarding how best to manage multiple resources simultaneously.

12.2.2 *Testing Model Performance on Large Projects*

The largest problem instance used in this thesis contains 60 activities and 73 transfer times, which is a small project when compared to the magnitude of civil engineering projects in industry. Problem instances of varying size and complexity should be generated and tested using the proposed model. A problem benchmark library such as Progen is a potential database for such testing but transfer times must be included separately, perhaps through random generation.

12.2.3 *Accounting for Deadlines*

The model does not currently account for deadlines in any way but rather finds a schedule to meet the dual requirement of minimised makespan and maximised slack, with relative importance of each objective specified by the user. If the schedule generated does not meet deadline requirements of the project, the project manager would need to fast track activities through increasing the amount of resources allocated. Fast tracking activities does of course incur extra costs, increases the difficulty of managing the schedule and decreases the robustness of the schedule.

An alternative solution would be to extend the model to allow deadlines to be included. The model should be extended to allow multiple deadlines to be specified. Both the final project deadline and activity deadlines should be supported. This new functionality would ensure that the final schedule meets all deadline requirements. The model would need to be further upgraded to allow for activity fast tracking by allowing more resources to be incorporated into the project at a cost, given that the specified deadlines cannot be met in any other manner.

12.2.4 *Transfer Time Calculation*

The concept of transfer times has been introduced and in the presented model transfer times are specified by the user. Transfer times can however be calculated in a different manner. For example, a location can be linked to activities; a speed of movement linked to resources and transfer times then automatically calculated. This speed of movement can be based on actual data or determined by the project manager who can utilise his/her experience to determine the actual speed of transferring a specific resource. With the use of Building Information Modelling (BIM), spatial information of the activities will be automatically available and this would further simplify the project information input process due to the location of activities being predetermined.

The calculation of transfer times could be further developed. A factor indicating the unfavorability of a resource transfer can be included in the model and used to weigh transfer times during the scheduling process, thus making schedules which use unfavourable transfers less likely to be selected. If a resource transferring between two activities can do so easily it is given a factor value indicating it is favourable. A transfer can be deemed unfavourable if it would, for example, require specialised equipment to navigate the terrain or would interfere with other activities on the construction site.

Investigation into the calculation of resource transfer times has the potential to not only streamline the project information input process, but also to improve the accuracy of resource transfer times.

12.2.5 *Edge Selection Technique*

The resource edge selection technique of Framework One was revised in this thesis to allow the framework to discover all available solutions. Although Framework One has been deemed too computationally intensive, if one can update the current exact procedure to only search part of the solution space this could change. The existence of an alternative set of edge selection criteria which limits the solution space but does not disregard any optimal solutions should be investigated. An improved edge selection technique could be beneficial and increase the size of projects the framework can process, either making the framework usable in industry or improving its capabilities as a research tool, either of which would be beneficial.

To verify the existence of such an edge selection technique, one would need to analyse numerous projects. A comparison should then be made between all possible resource precedence edges in the graph as created by the updated edge selection technique, and all edges used by the optimal solutions. If a trend exists which can be used to select only edges needed by optimal solutions, a useful set of edge selection criteria has been discovered.

12.2.6 *Baseline Schedule Selection*

Phase 1 of the proposed framework currently selects the baseline schedule with the shortest makespan and when more than one baseline schedule exists with the shortest makespan, the last one to be discovered is selected. There is potential to introduce a secondary objective to assist with selecting between solutions with equal makespan or even to introduce a multi-objective function. One suggestion for a secondary objective is to select the baseline in which the order of activities can induce the least amount of transfer times. The topic requires research to determine how the selected baseline affects the available solution space of the RCPSP-TT and then select an objective accordingly.

12.2.7 *Ant Colony Optimisation Algorithm*

The resource allocation process of Phase 2 utilises a basic ACO algorithm. The parameters utilised by the resource allocation ACO need to be configured and a set of guidelines for parameter selection should be created, where good parameter values are dependent on the project characteristics. The parameters referred to here are the number of generations, number of ants per generation and the evaporation rate. A heuristic to guide the search of the ant colony should also be investigated as this could lead to earlier convergence and therefore a less computationally intensive algorithm.

Two areas could provide useful heuristics. The first is a heuristic that equalises resource usage. This would be achieved through favouring resources that have been utilised least thus far in the solution being generated. The second potential heuristic is to favour resource allocations that would not induce transfer times. Both heuristics should be investigated and their effects determined.

12.2.8 *COST OBJECTIVE*

The current model does not support any information regarding the cost of activities, cost of resource transfers, cash flow data or any other financial information. Once this information is represented in the model, objectives involving costs and similar financial indicators can be incorporated into the multi-objective used for scheduling. Even if the objective is not included in the optimisation process, the representation of this information is still a necessary development.

12.2.9 *RISK MODEL FOR SLACK DISTRIBUTION*

A risk model has been presented that assigns a risk value to an activity, and based on the literature reviewed for this thesis is the first of its kind. The risk value indicates the relative importance of an activity having slack, based either on the risk or the effect of delay of an activity. The factors included in the proposed risk model include start time, duration and resource demand of an activity. There are however many factors that could be included in such a risk model and further research should be undertaken to determine which factors should be included in the model. The factors which should be included will of course vary based on the intended use of the model and this should also be investigated.

Literature is available regarding the causes of delays on a construction site and this will be a good starting point from which to investigate alternative factors to include in the risk model. Any characteristic that increases the importance of an activity having slack should be considered, such as the number of successors an activity has. The delay of an activity with many successors will impact the schedule more heavily. Another factor that should be investigated is the cost associated with delay. These costs could be due to reasons such as penalties associated with missed deadlines, extending subcontractor contracts or extension of equipment leases.

12.2.10 *INDUSTRY APPLICATION*

The model presented in this thesis is the first model capable of finding an optimal solution to the RCPSP-TT. The model only needs to be extended to include multiple resource types before it can be applied to realistic projects. Further investigation into aspects highlighted in this chapter will improve the performance of the model. Once the required improvements are made, the capabilities of the model should be compared to the requirements of industry and the project managers that would use the software. Any necessary improvements should be made to create a

model that is industry-ready. The civil engineering industry needs to be introduced to the capabilities of the models in academia and an attempt needs to be made to bridge the gap. The feedback of industry is needed to make the model industry-ready and thereafter the benefits of such models need to become wide-spread knowledge to assist with bridging the gap between academia and industry.

12.2.11 *Project Calendar*

The software implementation of the current model allows for the project calendar to be configured and the work hours per day specified. The calendar configuration capabilities should be upgraded to allow the availability of specific resources to be restricted. This would be required when contractors have limited availability or when specialised equipment is only available for a certain period. The upgrade will of course complicate the solution process significantly and therefore require thorough research.

CHAPTER 13

13 REFERENCES

Adhau, S., Mittal, M. & Mittal, A. 2013. A multi-agent system for decentralized multi-project scheduling with resource transfers. *International Journal of Production Economics*, 146(2): 646-661.

Christodoulou, S. submitted. An Entropy-Based Heuristic for Resource-Constrained Scheduling. *Journal of Computing in Civil Engineering*.

Cronje, M. 2006. Engineering process model: Detection of cycles and determination of paths. Unpublished master's thesis. Stellenbosch: University of Stellenbosch.

Eygelaar, A.B. 2008. Resource constrained step scheduling of project tasks. Unpublished master's thesis. Stellenbosch: University of Stellenbosch.

Griebenow, M.A. 2014. Construction process modelling: Resource constrained scheduling with resource transfer times accounted for. Unpublished undergraduate's thesis. Stellenbosch: University of Stellenbosch.

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

Herroelen, W. & Leus, R. 2002. Stability and Resource Allocation in Project Planning. *Department Toegepast Economische Wetenschappen*, D/2002/2376/56.

Krüger, D. & Scholl, A. 2010. Managing and modelling general resource transfers in (multi-) project scheduling. *OR Spectrum*, 32(2): 369-394.

Krüger, D. & Scholl, A. 2009. A heuristic solution framework for the resource constrained (multi-)project scheduling problem with sequence-dependent transfer times. *European Journal of Operational Research*, 197(1): 492-508.

Merkle, D., Middendorf, M. & Schmeck, H. 2002. Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, 6: 333-346.

Pahl, P.J. & Damrath, R. 2001. *Mathematical Foundations of Computational Engineering.* Berlin, Germany: Springer.

Poppenborg, J. & Knust, S. 2015. A flow-based tabu search algorithm for the RCPSP with transfer times. *OR Spectrum,* 38:305-334.

Potgieter, I.J. 2014. Resource-Constrained Scheduling in the Engineering-Planning Phase of Civil Engineering Projects. Unpublished doctoral thesis. Stellenbosch: University of Stellenbosch.

Potgieter, I.J. & van Rooyen, G.C. Maximising slack in Resoure-Constrained Schedules: A Heuristic Approach. *Journal of Computing in Civil Engineering,* 10.1061/(ASCE)CP.1943-5487.000057, 04016018.

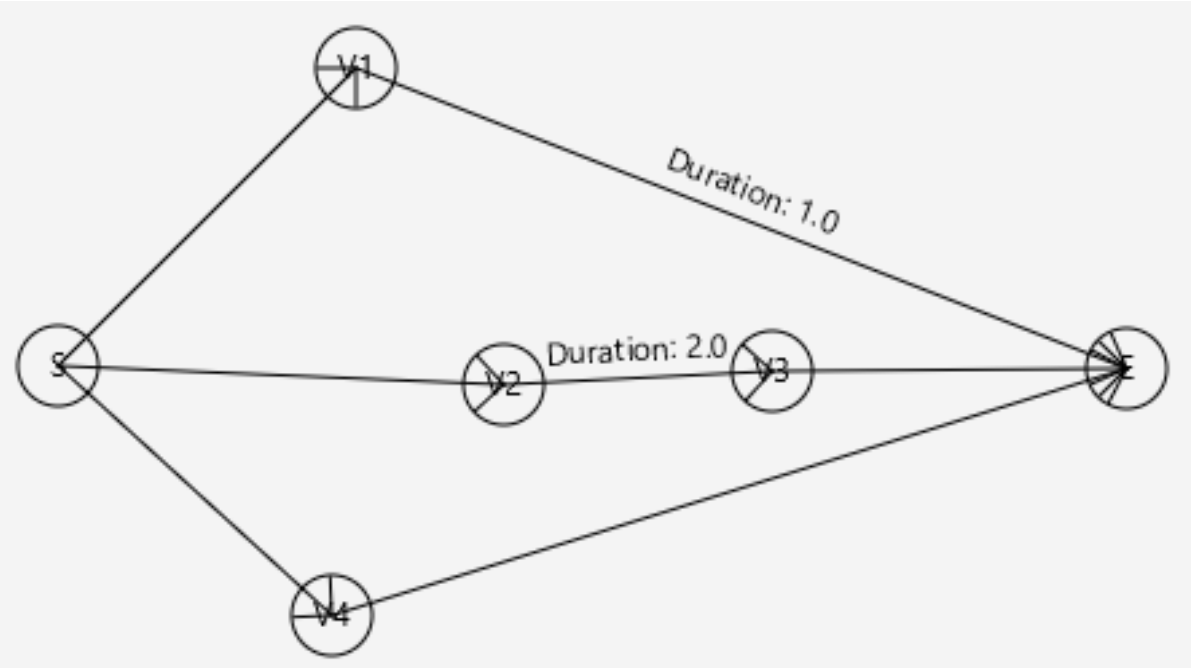
Potgieter, I.J. & van Rooyen, G.C. 2012. A multi-objective approach to project scheduling. Unpublished paper delivered at the Fourteenth International Conference on Computing in Civil and Building Engineering. 27 June, Moscow.

Wilks, S. 2015. *The century's most troublesome construction projects*. http://www.globalconstructionreview.com/perspectives/centurys-most-troublesome-construction-pr8oje8ct8s/.

APPENDICES

APPENDIX A: PROJECT 1

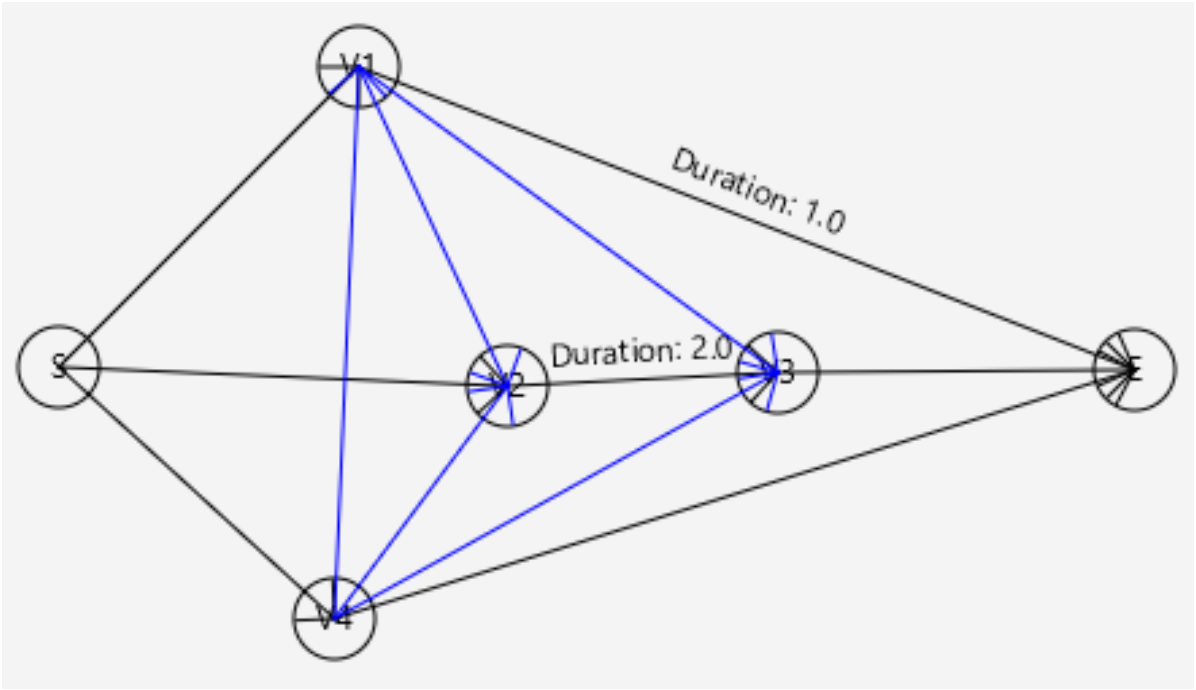
The data represented in this section is of all solutions of Project 1, as found utilising Framework One with use of the original and updated edge selection techniques. For both edge selection techniques, full details are then provided for all solutions with minimum makespan. To assist with interpretation, note that for the solutions one path is listed per resource, and the same path listed twice therefore indicates that two resources flow along the same path. Further, "Duration" values displayed above certain edges indicate the duration of the transfer, i.e. the transfer time. When no duration is displayed, it is zero.



Activity	d_i	r_i
S	0	0
V_1	1	1
V_2	2	1
V_3	1	2
V_4	1	2
E	0	0

3 resources available

Original Edge Selection Technique



Resource Edges:

V1V2

V1V3

V4V1

V4V2

V4V3

Resource Flow Paths (10):

S - V4 - V1 - V2 - V3 - E

S - V1 - V2 - V3 - E

S - V4 - V2 - V3 - E

S - V4 - V1 - V3 - E

S - V1 - V3 - E

S - V2 - V3 - E

S - V4 - V3 - E

S - V4 - V1 - E

S - V1 - E

S - V4 -

Original Edge Selection Technique Solution Details

5 Feasible Solutions

Summary of Solutions in terms of Makespan and Slack Distribution

Makespan	Slack 2	Slack 4
5	2	
6		3

The two solutions with minimum makespan are as follows.

Original Technique Feasible Solution #1

See Section A.1 for details

S - V4 - V1 - V3 - E

S - V2 - V3 - E

S - V4 - E

Slack: V4: 0.0 V1: 2.0 V3: 0.0 V2: 0.0

End time of selected flow: 5.0

Original Technique Feasible Solution #2

See Section A.2 for details

S - V4 - V1 - E

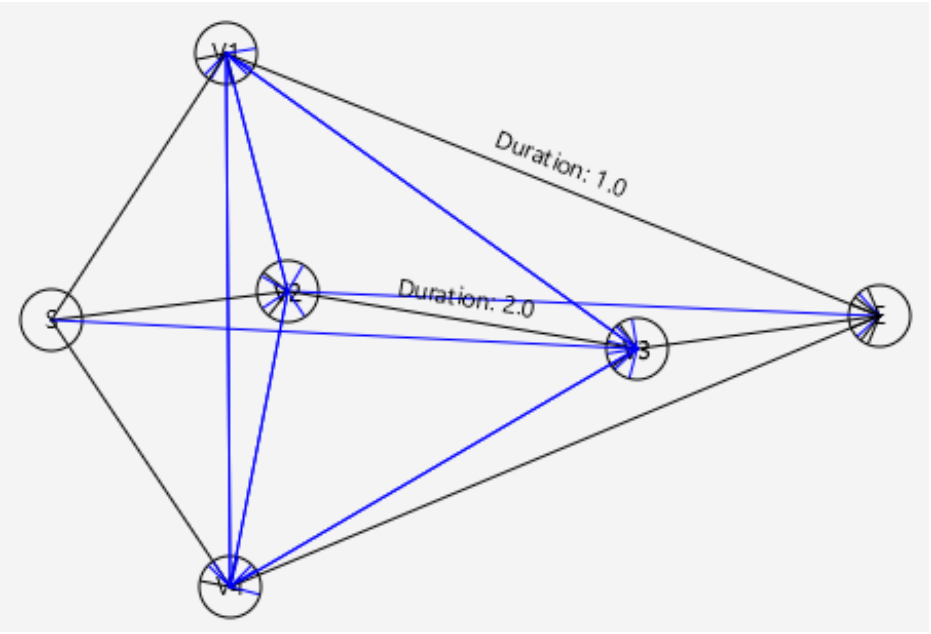
S - V2 - V3 - E

S - V4 - V3 - E

Slack: V4: 0.0 V1: 2.0 V3: 0.0 V2: 0.0

End time of selected flow: 5.0

Updated Edge Selection Technique



Resource Edges:

SV3
V2V4
V2V5
V4V2
V1V2/V2V1
V1V3/V3V1
V1V4/V4V1
V3V4/V4V3

Resource Flow Paths (45)

S - V1 - V2 - V3 - V4 - E
S - V1 - V2 - V4 - V3 - E
S - V1 - V4 - V2 - V3 - E
S - V2 - V3 - V1 - V4 - E
S - V2 - V1 - V4 - V3 - E
S - V2 - V1 - V3 - V4 - E
S - V2 - V4 - V1 - V3 - E
S - V2 - V3 - V4 - V1 - E
S - V2 - V4 - V3 - V1 - E
S - V4 - V2 - V3 - V1 - E
S - V4 - V1 - V2 - V3 - E
S - V4 - V2 - V1 - V3 - E
S - V1 - V4 - V2 - E
S - V1 - V4 - V3 - E
S - V1 - V2 - V3 - E
S - V1 - V2 - V4 - E
S - V1 - V3 - V4 - E
S - V2 - V3 - V1 - E
S - V2 - V1 - V4 - E
S - V2 - V3 - V4 - E
S - V2 - V1 - V3 - E
S - V2 - V4 - V1 - E
S - V2 - V4 - V3 - E
S - V3 - V1 - V4 - E
S - V3 - V4 - V1 - E
S - V4 - V2 - V3 - E
S - V4 - V3 - V1 - E
S - V4 - V1 - V2 - E
S - V4 - V1 - V3 - E
S - V4 - V2 - V1 - E
S - V1 - V3 - E
S - V1 - V4 - E
S - V1 - V2 - E
S - V2 - V3 - E
S - V2 - V1 - E
S - V3 - V4 - E
S - V3 - V1 - E
S - V2 - V4 - E
S - V4 - V2 - E
S - V4 - V3 - E
S - V4 - V1 - E
S - V1 - E
S - V2 - E
S - V3 - E
S - V4 - E

Updated Edge Selection Technique Solution Details

77 Feasible Solutions

Summary of Solutions in terms of Makespan and Slack Distribution:

Makespan	Slack 0	Slack 1	Slack 2	Slack 3	Slack 4
3	2				
4	2	6	7		
5	16	5	4		
6	9			3	6
7	7	2		2	
8	6				

The two solutions with minimum makespan are as follows.

Updated Technique Feasible Solution #1 = Original Technique Feasible Solution #1

See Section A.1 for details

S - V1 - V4 - V3 - E

S - V4 - V3 - E

S - V2 - E

Slack: V4: 0.0 V2: 0.0 V1: 0.0 V3: 0.0

End time of selected flow: 3.0

Updated Technique Feasible Solution #1 = Original Technique Feasible Solution #2

See Section A.2 for details

S - V4 - V1 - V3 - E

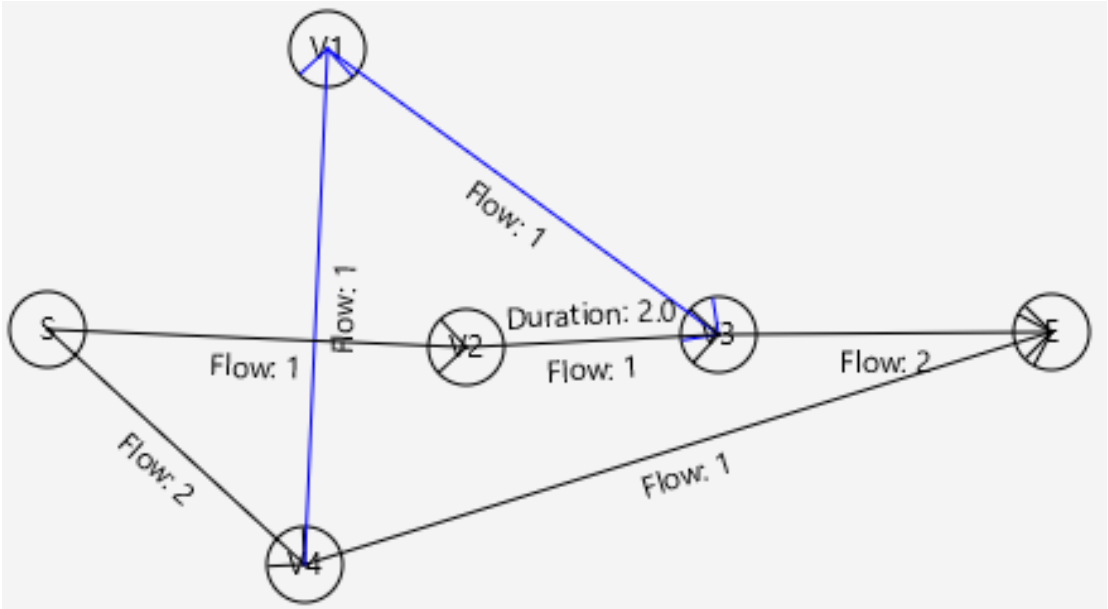
S - V4 - V3 - E

S - V2 - E

Slack: V4: 0.0 V2: 0.0 V1: 0.0 V3: 0.0

End time of selected flow: 3.0

A.1 SOLUTION



Resource Flow Paths:

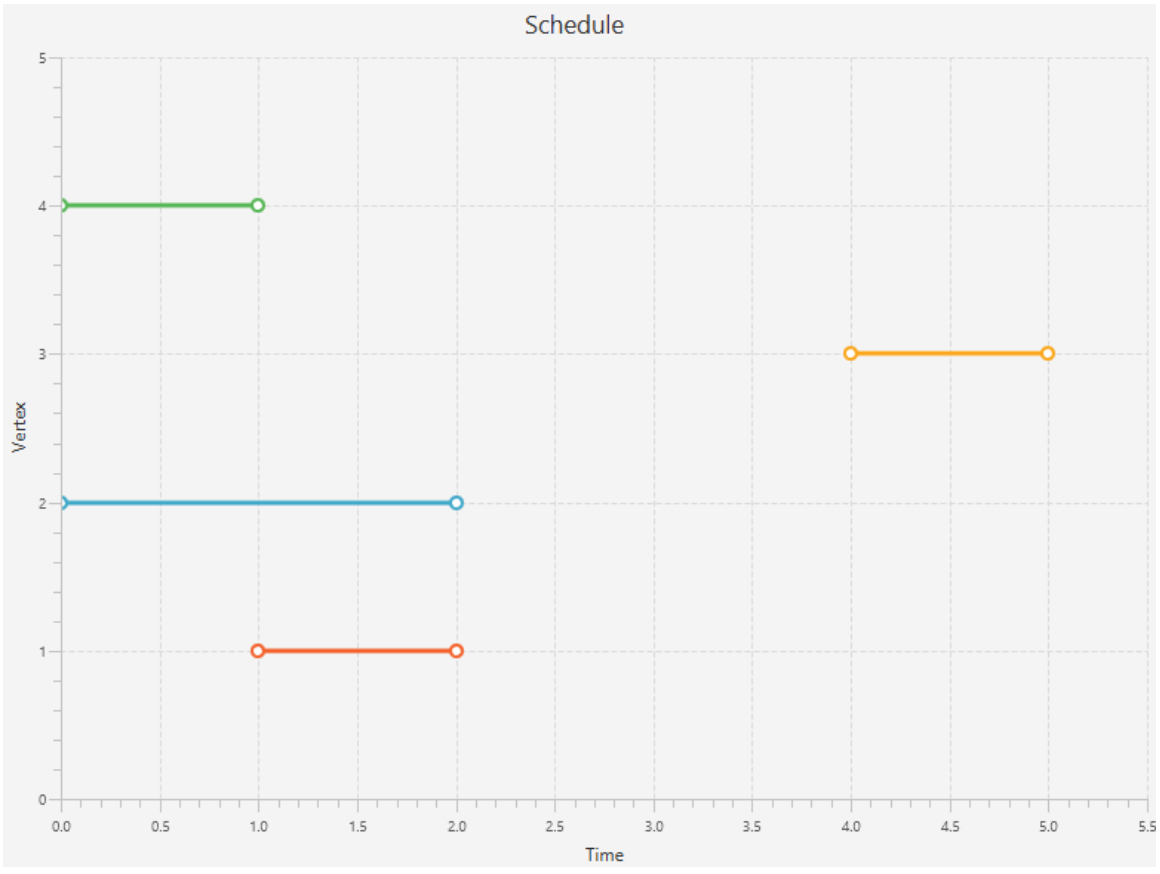
S - V4 - V1 - V3 - E

S - V2 - V3 - E

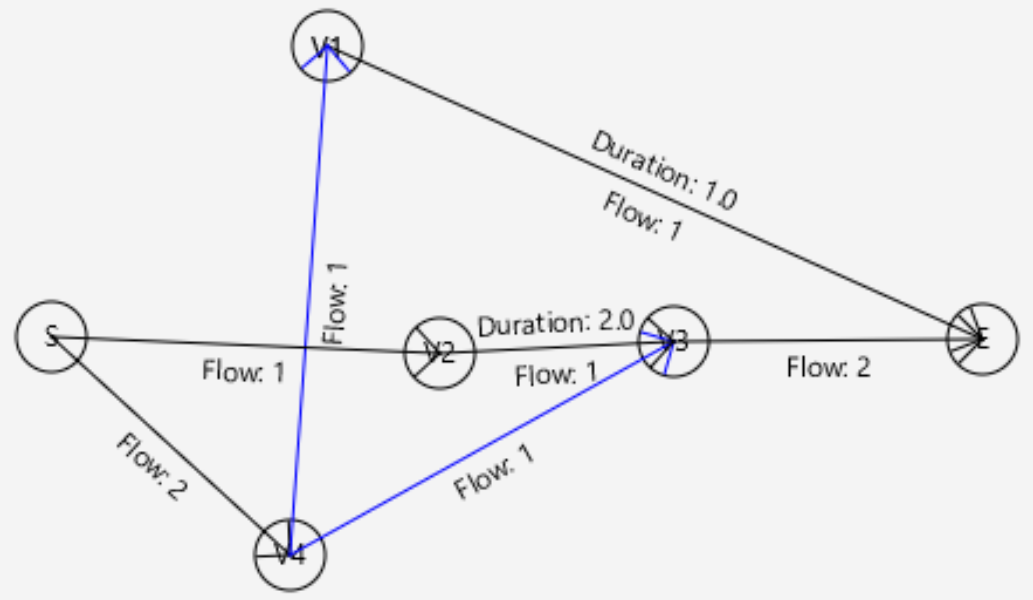
S - V4 - E

Slack: V4: 0.0 V1: 2.0 V3: 0.0 V2: 0.0

End time of selected flow: 5.0



A.2 SOLUTION



Resource Flow Paths:

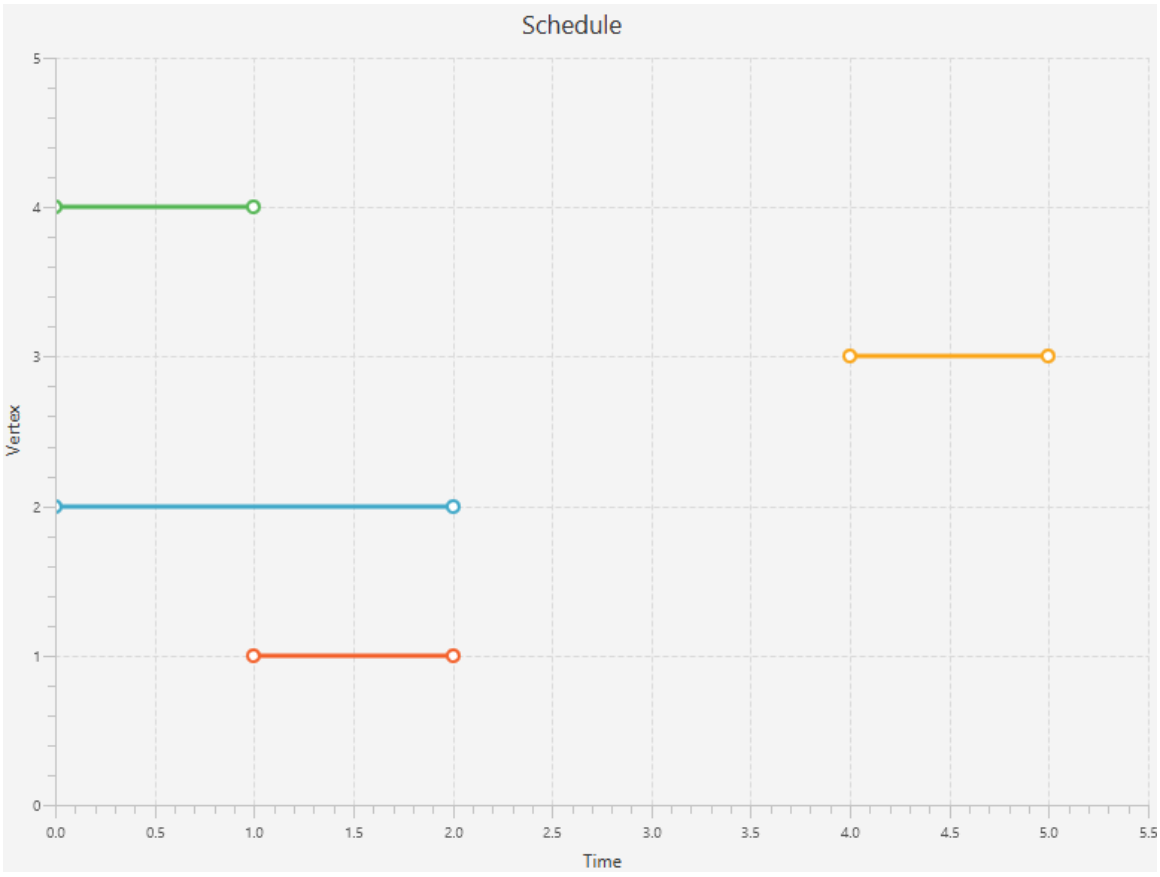
S - V4 - V1 - E

S - V2 - V3 - E

S - V4 - V3 - E

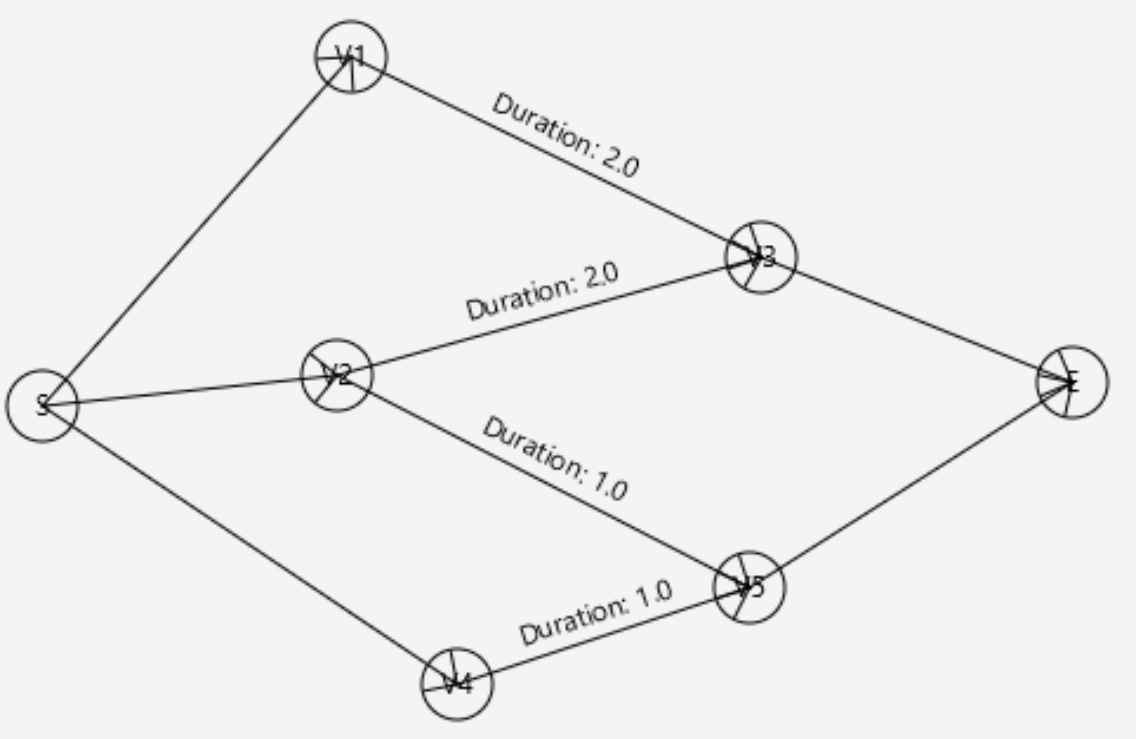
Slack: V4: 0.0 V1: 2.0 V3: 0.0 V2: 0.0

End time of selected flow: 5.0



APPENDIX B: PROJECT 2

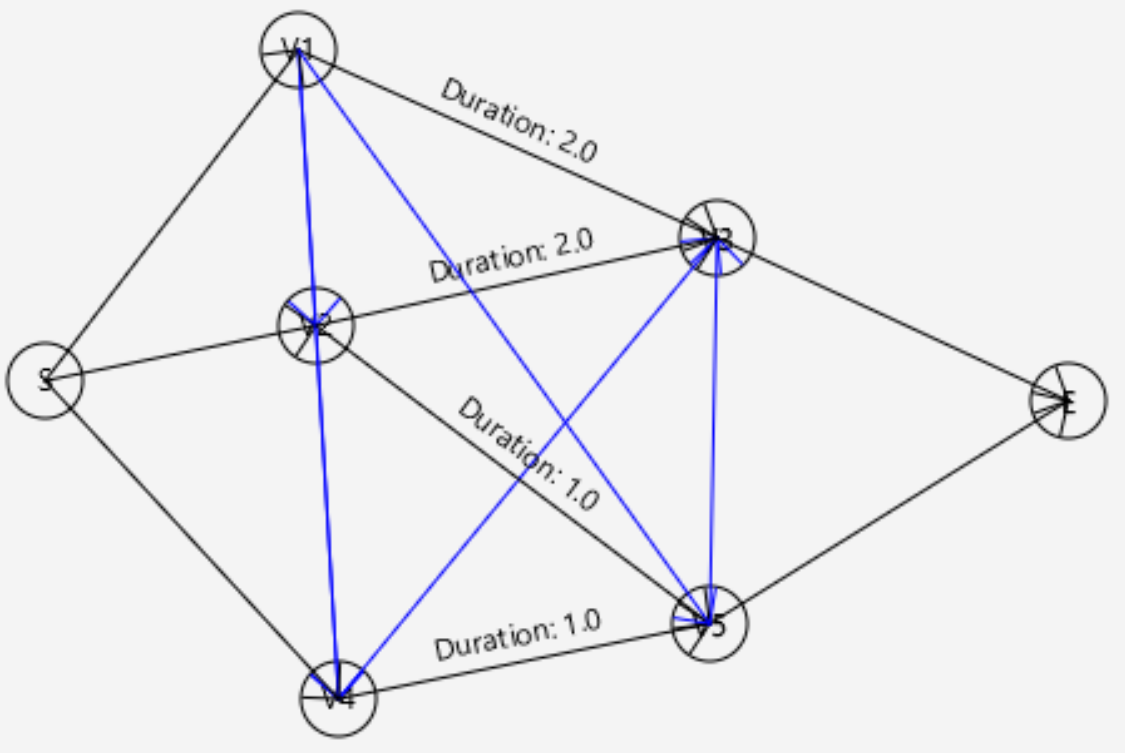
The data represented in this section is of all solutions of Project 2, as found utilising Framework One with use of the original and updated edge selection techniques. For both edge selection techniques, full details are then provided for all solutions with minimum makespan. To assist with interpretation, note that for the solutions one path is listed per resource, and the same path listed twice therefore indicates that two resources flow along the same path. Further, "Duration" values displayed above certain edges indicate the duration of the transfer, i.e. the transfer time. When no duration is displayed, it is zero.



Activity	d_i	r_i
S	0	0
V_1	2	2
V_2	3	2
V_3	1	1
V_4	3	1
V_5	3	4
E	0	0

5 resources available

Original Edge Selection Technique



Resource Edges:

V1V2

V1V4

V1V5

V2V4

V4V3

V5V3

Resource Flow Paths (21 → 6):

Paths removed by the Unnecessary Flows Algorithm of Chapter 8.2.1 shown ~~like this~~

~~S - V1 - V2 - V4 - V5 - V3 - E~~

~~S - V1 - V4 - V5 - V3 - E~~

~~S - V1 - V2 - V4 - V5 - E~~

~~S - V1 - V2 - V5 - V3 - E~~

~~S - V1 - V2 - V4 - V3 - E~~

~~S - V2 - V4 - V5 - V3 - E~~

~~S - V1 - V2 - V3 - E~~

~~S - V1 - V5 - V3 - E~~

~~S - V1 - V2 - V5 - E~~

~~S - V1 - V4 - V5 - E~~

~~S - V1 - V4 - V3 - E~~

~~S - V2 - V5 - V3 - E~~

~~S - V2 - V4 - V5 - E~~

~~S - V2 - V4 - V3 - E~~

~~S - V4 - V5 - V3 - E~~

S - V1 - V3 - E

S - V1 - V5 - E

S - V2 - V5 - E

S - V2 - V3 - E

S - V4 - V5 - E

S - V4 - V3 -

Original Edge Selection Technique Solution Details

3 Feasible Solutions

Summary of Solutions in terms of Makespan and Slack Distribution:

Makespan	Slack 2	Slack 3	Slack 4
7	1	1	1

The three solutions with minimum makespan are as follows.

Original Technique Feasible Solution #1

See Section B.1 for details

S - V1 - V5 - E

S - V1 - V3 - E

S - V2 - V5 - E

S - V2 - V5 - E

S - V4 - V5 - E

Slack: V4: 0.0 V3: 2.0 V2: 0.0 V5: 0.0 V1: 0.0

End time of selected flow: 7.0

Original Technique Feasible Solution #2

See Section B.2 for details

S - V2 - V3 - E

S - V1 - V5 - E

S - V1 - V5 - E

S - V2 - V5 - E

S - V4 - V5 - E

Slack: V4: 0.0 V3: 1.0 V2: 0.0 V5: 0.0 V1: 2.0

End time of selected flow: 7.0

Original Technique Feasible Solution #3

See Section B.3 for details

S - V4 - V3 - E

S - V1 - V5 - E

S - V1 - V5 - E

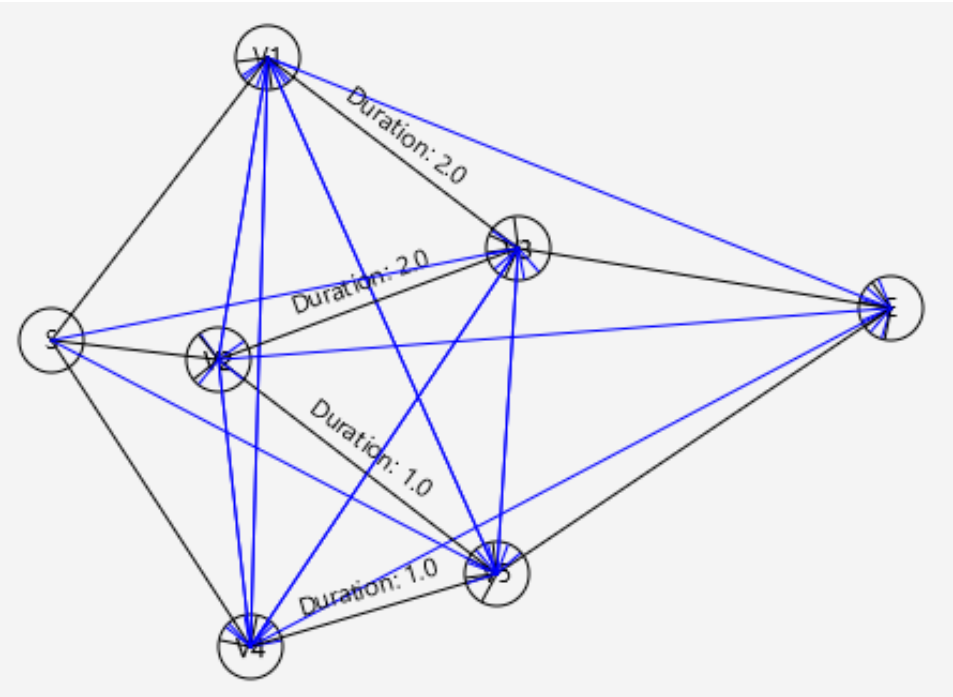
S - V2 - V5 - E

S - V2 - V5 - E

Slack: V4: 0.0 V3: 3.0 V2: 0.0 V5: 0.0 V1: 1.0

End time of selected flow: 7.0

Updated Edge Selection Technique



Resource Edges:

SV3
SV5
V1V6
V2V6
V3V4/V4V3
V4V6
V1V2 / V2V1
V1V4 / V4V1
V1V5 / V5V1
V2V4 / V4V2
V3V5 / V5V3

Resource Flow Paths (99):

S - V1 - V2 - V4 - V5 - V3 - E
S - V1 - V4 - V2 - V5 - V3 - E
S - V1 - V2 - V4 - V3 - V5 - E
S - V1 - V2 - V3 - V4 - V5 - E
S - V1 - V4 - V2 - V3 - V5 - E
S - V2 - V4 - V1 - V5 - V3 - E
S - V2 - V4 - V5 - V1 - V3 - E
S - V2 - V1 - V4 - V3 - V5 - E
S - V2 - V1 - V4 - V5 - V3 - E
S - V2 - V1 - V3 - V4 - V5 - E
S - V2 - V4 - V1 - V3 - V5 - E
S - V4 - V2 - V1 - V3 - V5 - E
S - V4 - V2 - V5 - V1 - V3 - E
S - V4 - V1 - V2 - V3 - V5 - E
S - V4 - V2 - V1 - V5 - V3 - E
S - V4 - V1 - V2 - V5 - V3 - E
S - V1 - V4 - V3 - V5 - E
S - V1 - V2 - V3 - V5 - E
S - V1 - V4 - V2 - V5 - E
S - V1 - V3 - V4 - V5 - E
S - V1 - V2 - V5 - V3 - E
S - V1 - V2 - V3 - V4 - E
S - V1 - V4 - V5 - V3 - E
S - V1 - V2 - V4 - V5 - E
S - V1 - V4 - V2 - V3 - E
S - V1 - V2 - V4 - V3 - E
S - V2 - V5 - V1 - V3 - E
S - V2 - V1 - V4 - V3 - E
S - V2 - V4 - V5 - V1 - E
S - V2 - V3 - V4 - V5 - E
S - V2 - V1 - V3 - V4 - E
S - V2 - V4 - V3 - V5 - E
S - V2 - V4 - V1 - V3 - E
S - V2 - V4 - V1 - V5 - E
S - V2 - V1 - V5 - V3 - E
S - V2 - V1 - V4 - V5 - E

S - V2 - V1 - V3 - V5 - E
S - V2 - V4 - V5 - V3 - E
S - V4 - V1 - V3 - V5 - E
S - V4 - V2 - V5 - V3 - E
S - V4 - V5 - V1 - V3 - E
S - V4 - V1 - V2 - V3 - E
S - V4 - V1 - V2 - V5 - E
S - V4 - V2 - V1 - V3 - E
S - V4 - V1 - V5 - V3 - E
S - V4 - V2 - V3 - V5 - E
S - V4 - V2 - V1 - V5 - E
S - V4 - V2 - V5 - V1 - E
S - V1 - V2 - V3 - E
S - V1 - V4 - V2 - E
S - V1 - V2 - V5 - E
S - V1 - V3 - V5 - E
S - V1 - V4 - V3 - E
S - V1 - V5 - V3 - E
S - V1 - V4 - V5 - E
S - V1 - V2 - V4 - E
S - V1 - V3 - V4 - E
S - V2 - V1 - V4 - E
S - V2 - V4 - V1 - E
S - V2 - V3 - V5 - E
S - V2 - V3 - V4 - E
S - V2 - V4 - V5 - E
S - V2 - V5 - V3 - E
S - V2 - V4 - V3 - E
S - V2 - V1 - V5 - E
S - V2 - V5 - V1 - E
S - V2 - V1 - V3 - E
S - V3 - V4 - V5 - E
S - V4 - V2 - V5 - E
S - V4 - V1 - V3 - E
S - V4 - V1 - V2 - E
S - V4 - V2 - V3 - E
S - V4 - V5 - V3 - E
S - V4 - V1 - V5 - E
S - V4 - V3 - V5 - E
S - V4 - V2 - V1 - E
S - V4 - V5 - V1 - E
S - V5 - V1 - V3 - E
S - V1 - V2 - E
S - V1 - V3 - E
S - V1 - V5 - E
S - V1 - V4 - E
S - V2 - V4 - E
S - V2 - V5 - E
S - V2 - V3 - E
S - V2 - V1 - E
S - V3 - V5 - E -
S - V3 - V4 - E -
S - V4 - V1 - E
S - V4 - V3 - E
S - V4 - V2 - E
S - V4 - V5 - E
S - V5 - V1 - E -
S - V5 - V3 - E
S - V1 - E
S - V2 - E
S - V3 - E
S - V4 - E
S - V5 - E

Updated Edge Selection Technique Solution Details

793 Feasible Solutions

Summary of Solutions in terms of Makespan and Slack Distribution:

Makespan	Slack 0	Slack 1	Slack 2	Slack 3	Slack 4	Slack 5	Slack 6	Slack 7
7		2	1	1	1			
8	2	7	11	8	2			
9	24	33	34	15		9	2	
10	35	27	31	6	14	2	2	3
11	20	14	34	2	14	23		
12	132	28		23	6		14	
13	121			2				
14	45							
15	43							

The five solutions with minimum makespan are as follows.

Updated Technique Feasible Solution #1 = Original Technique Feasible Solution #1

See Section B.1 for details

S - V1 - V5 - E

S - V4 - V5 - E

S - V2 - V5 - E

S - V2 - V5 - E

S - V1 - V3 - E

Slack: V3: 2.0 V5: 0.0 V1: 0.0 V4: 0.0 V2: 0.0

End time of selected flow: 7.0

Updated Technique Feasible Solution #2 = Original Technique Feasible Solution #2

See Section B.2 for details

S - V1 - V5 - E

S - V1 - V5 - E

S - V4 - V5 - E

S - V2 - V3 - E

S - V2 - V5 - E

Slack: V3: 1.0 V5: 0.0 V1: 2.0 V4: 0.0 V2: 0.0

End time of selected flow: 7.0

Updated Technique Feasible Solution #3 = Original Technique Feasible Solution #3

See Section B.3 for details

S - V1 - V5 - E

S - V1 - V5 - E

S - V4 - V3 - E

S - V2 - V5 - E

S - V2 - V5 - E

Slack: V3: 3.0 V5: 0.0 V1: 1.0 V4: 0.0 V2: 0.0

End time of selected flow: 7.0

Updated Technique Feasible Solution #4

See Section B.4 for details

S - V4 - V3 - V5 - E

S - V1 - V5 - E

S - V2 - V5 - E

S - V2 - V5 - E

S - V1 - E

Slack: V3: 0.0 V5: 0.0 V1: 1.0 V4: 0.0 V2: 0.0

End time of selected flow: 7.0

Updated Technique Feasible Solution #5

See Section B.5 for details

S - V4 - V3 - V5 - E

S - V1 - V5 - E

S - V1 - V5 - E

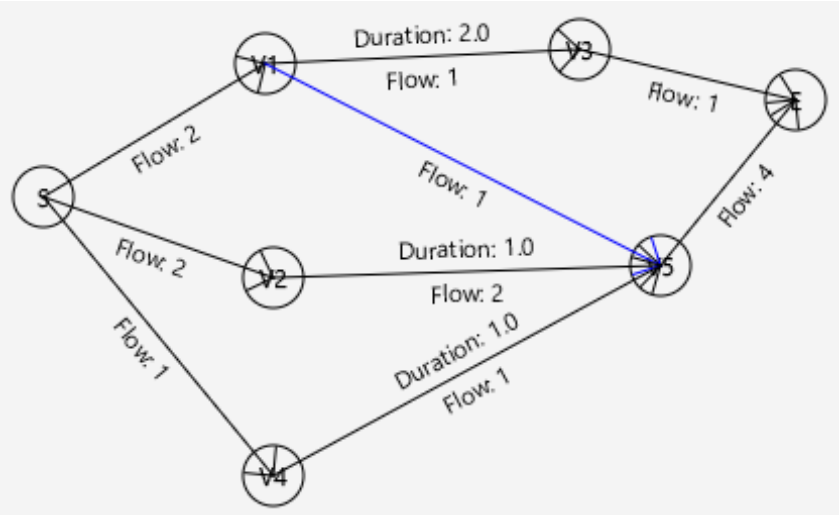
S - V2 - V5 - E

S - V2 - E

Slack: V2: 0.0 V5: 0.0 V3: 0.0 V4: 0.0 V1: 1.0

End time of selected flow: 7.0

B.1 SOLUTION 1



Resource Flow Paths:

S - V1 - V5 - E

S - V4 - V5 - E

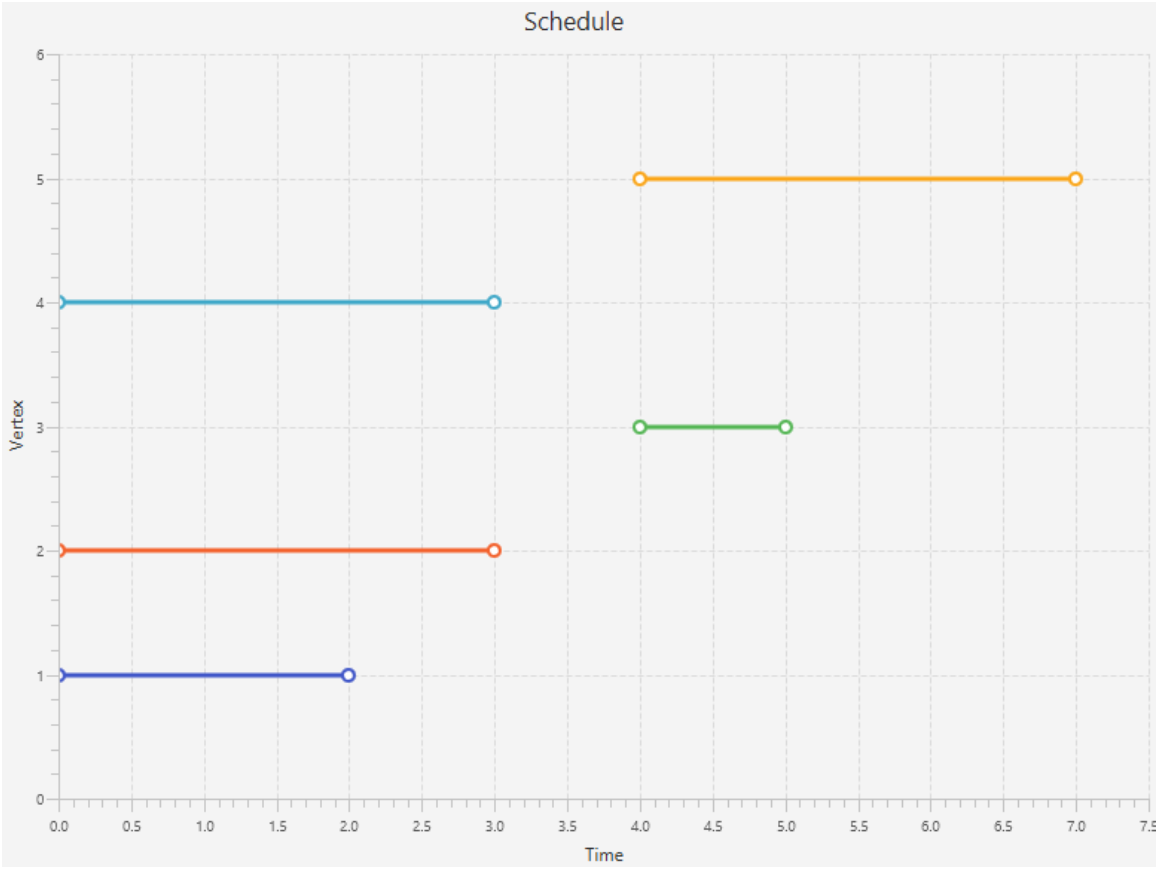
S - V2 - V5 - E

S - V2 - V5 - E

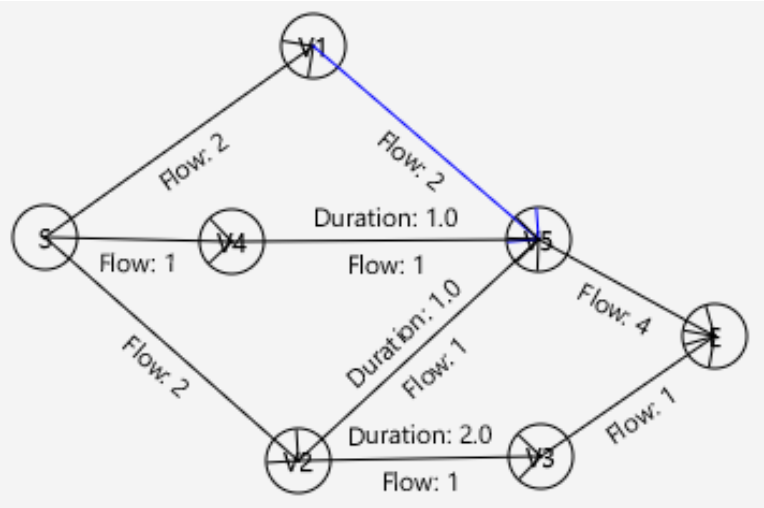
S - V1 - V3 - E

Slack: V3: 2.0 V5: 0.0 V1: 0.0 V4: 0.0 V2: 0.0

End time of selected flow: 7.0



B.2 SOLUTION 2



Resource Flow Paths:

S - V1 - V5 - E

S - V1 - V5 - E

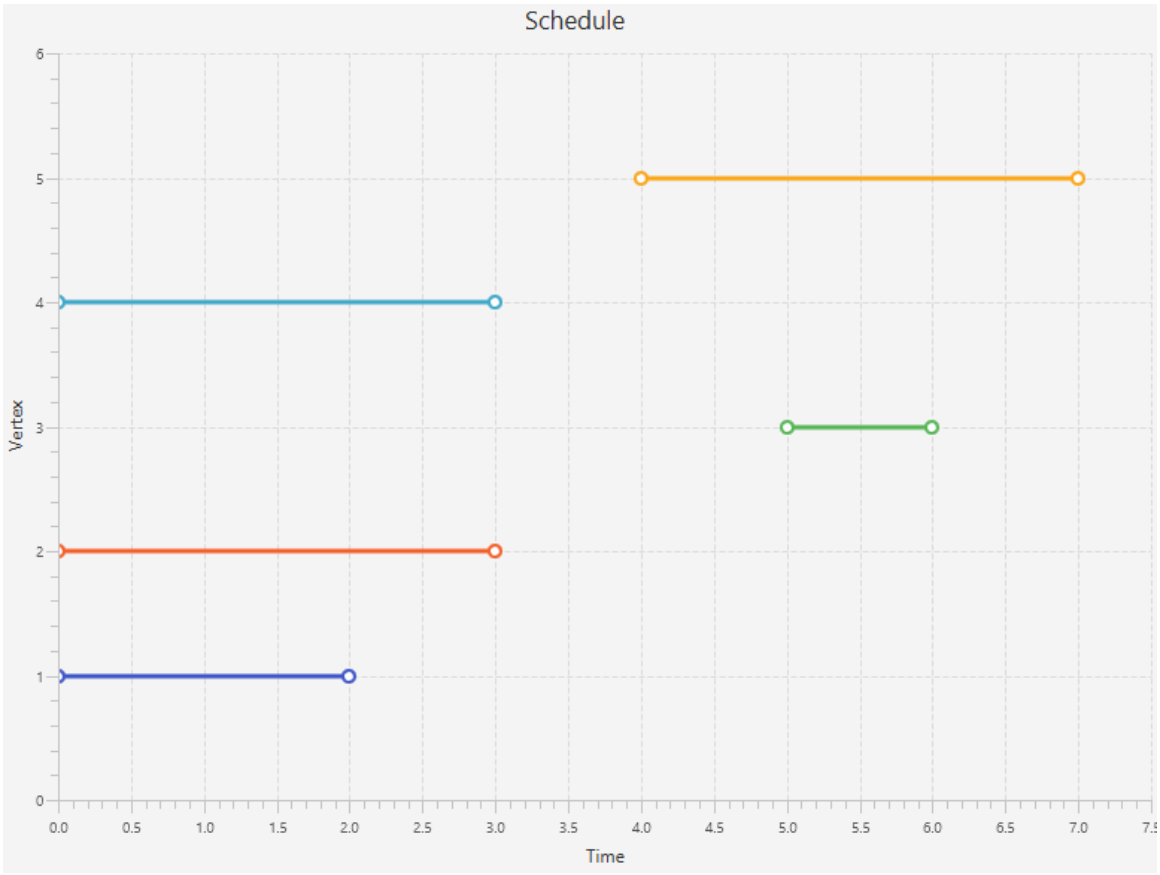
S - V4 - V5 - E

S - V2 - V3 - E

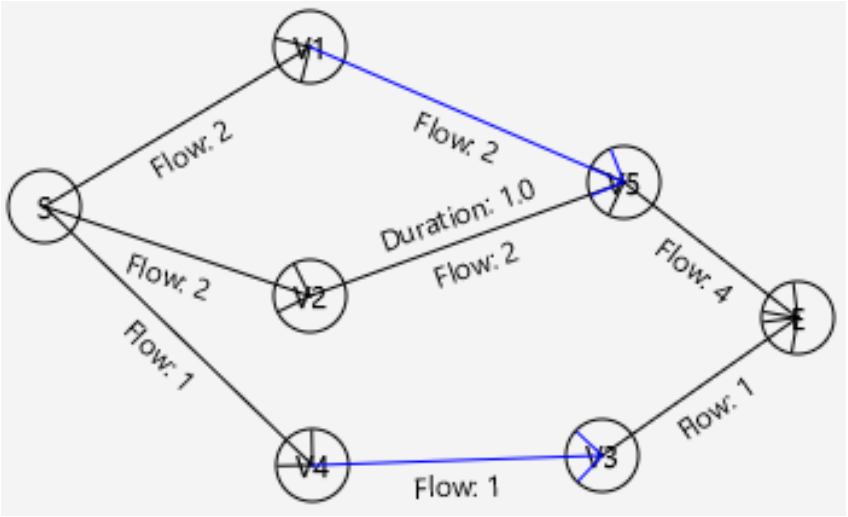
S - V2 - V5 - E

Slack: V3: 1.0 V5: 0.0 V1: 2.0 V4: 0.0 V2: 0.0

End time of selected flow: 7.0



B.3 SOLUTION 3



Resource Flow Paths:

S - V1 - V5 - E

S - V1 - V5 - E

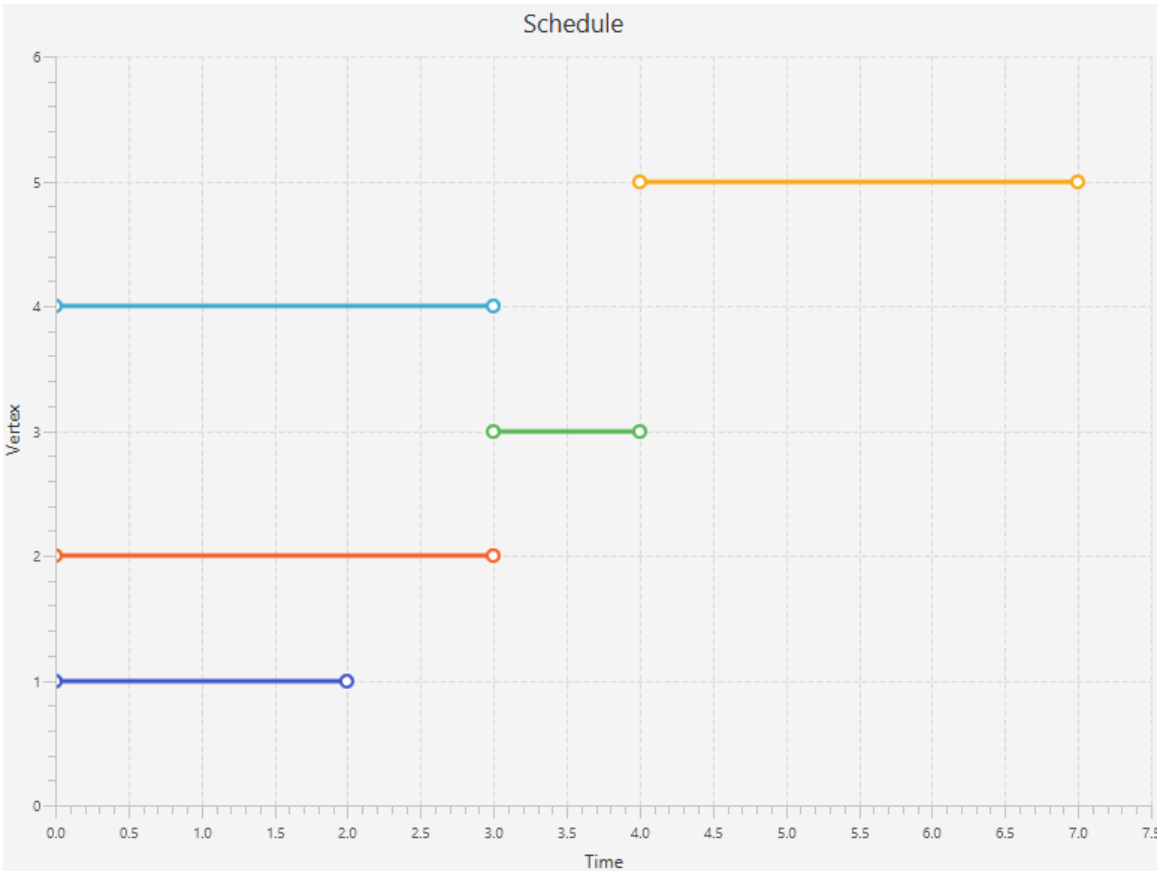
S - V4 - V3 - E

S - V2 - V5 - E

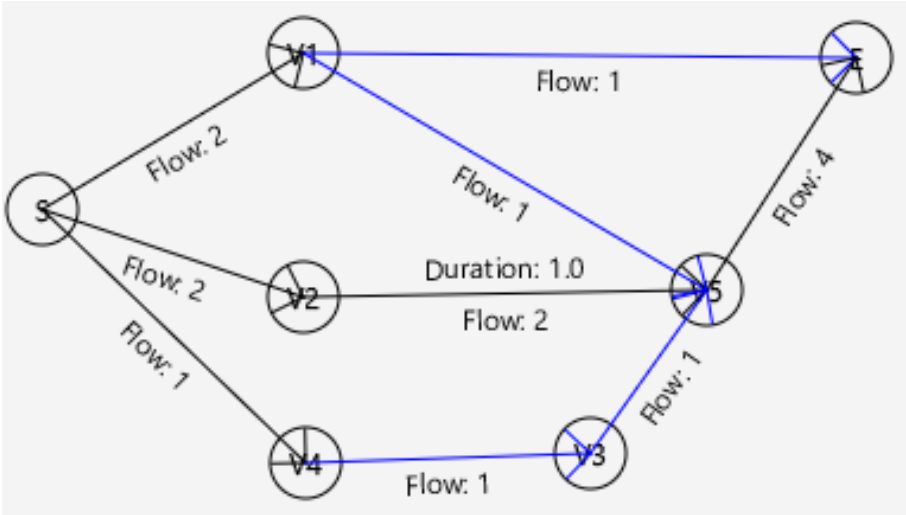
S - V2 - V5 - E

Slack: V3: 3.0 V5: 0.0 V1: 1.0 V4: 0.0 V2: 0.0

End time of selected flow: 7.0



B.4 SOLUTION 4



Resource Flow Paths:

S - V4 - V3 - V5 - E

S - V1 - V5 - E

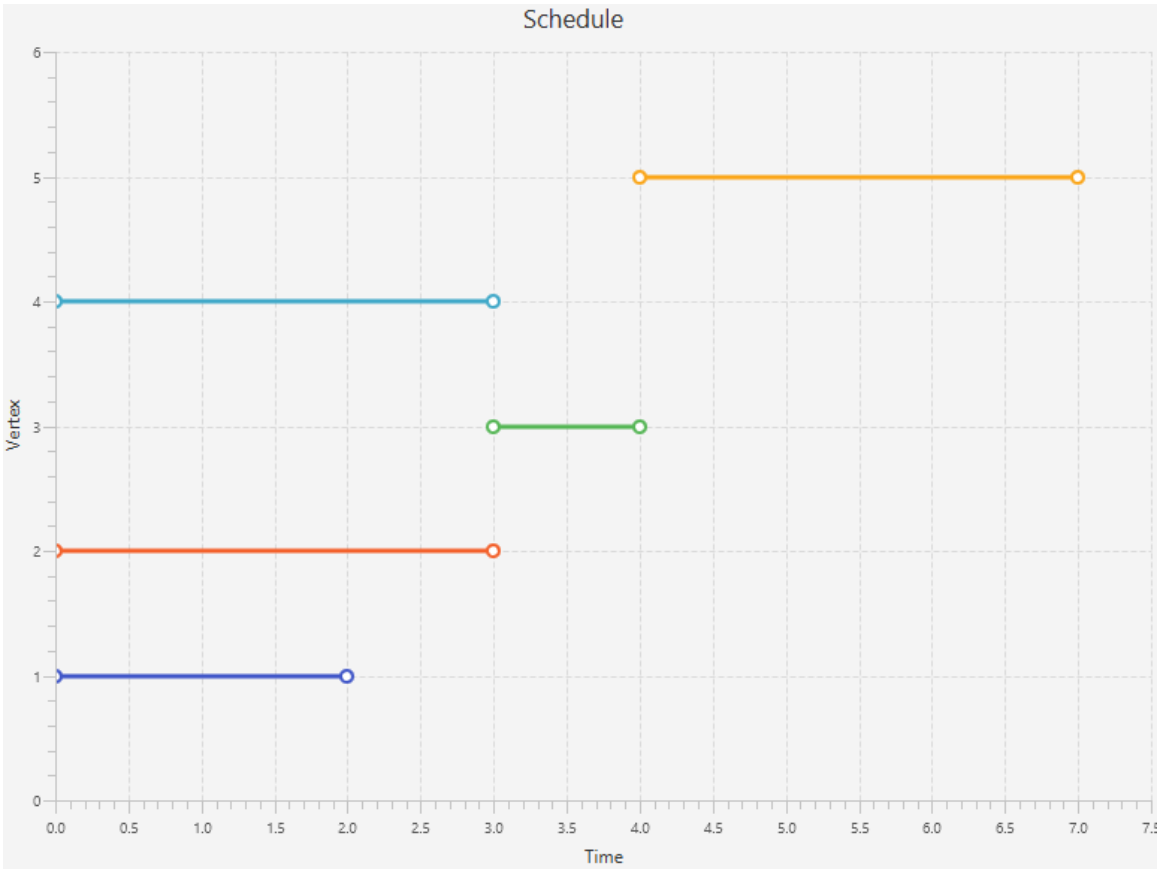
S - V2 - V5 - E

S - V2 - V5 - E

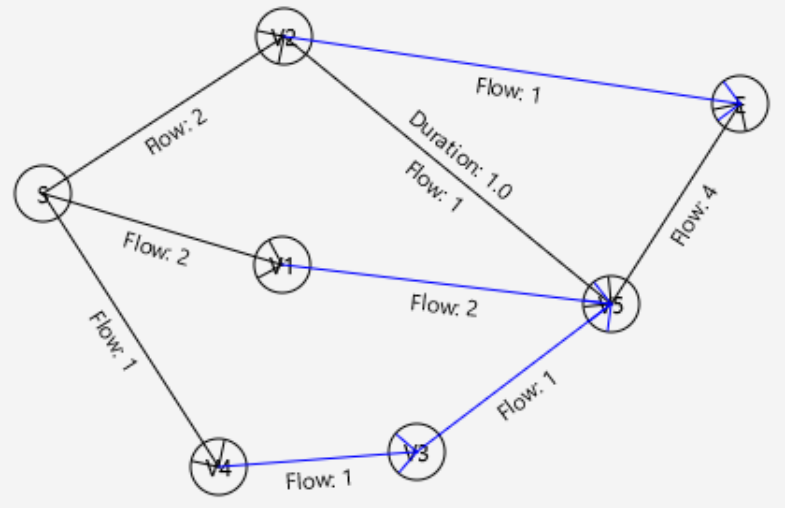
S - V1 - E

Slack: V3: 0.0 V5: 0.0 V1: 1.0 V4: 0.0 V2: 0.0

End time of selected flow: 7.0



B.5 SOLUTION 5



Resource Flow Paths:

S - V4 - V3 - V5 - E

S - V1 - V5 - E

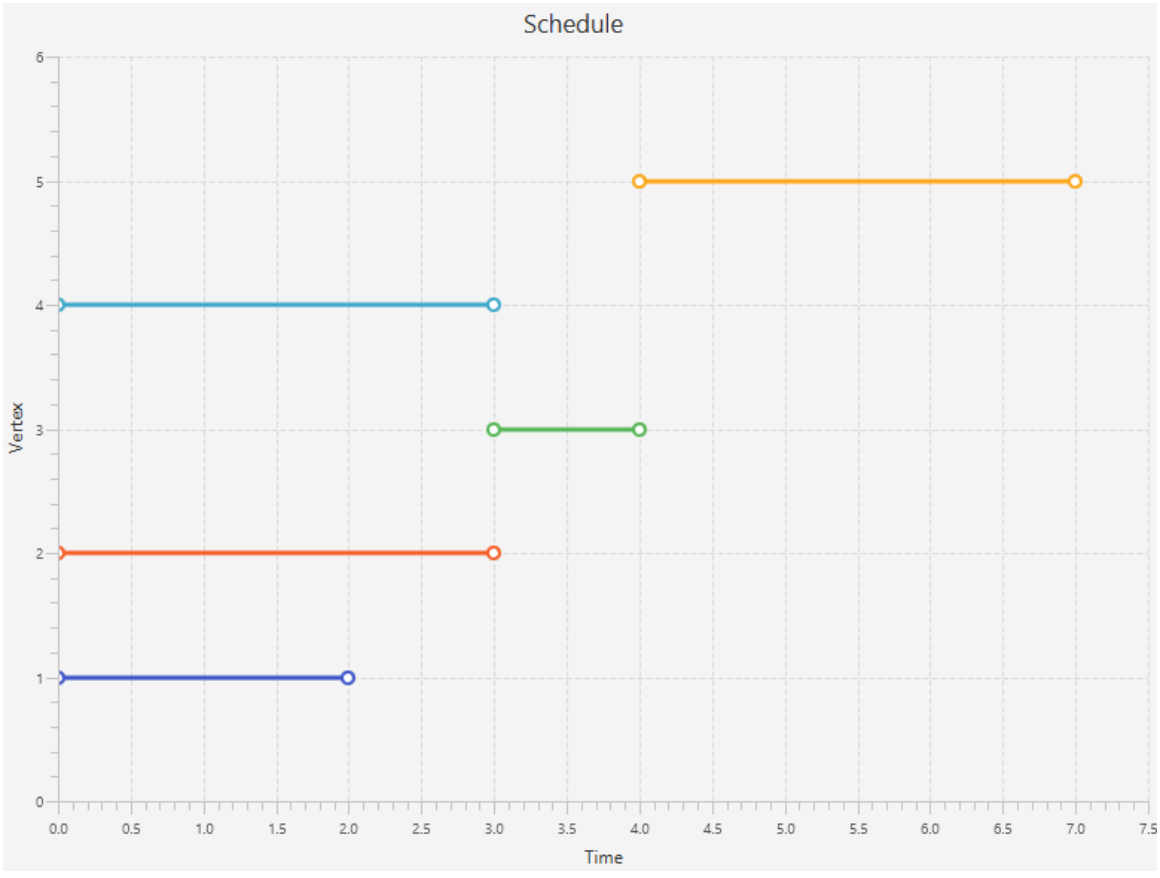
S - V1 - V5 - E

S - V2 - V5 - E

S - V2 - E

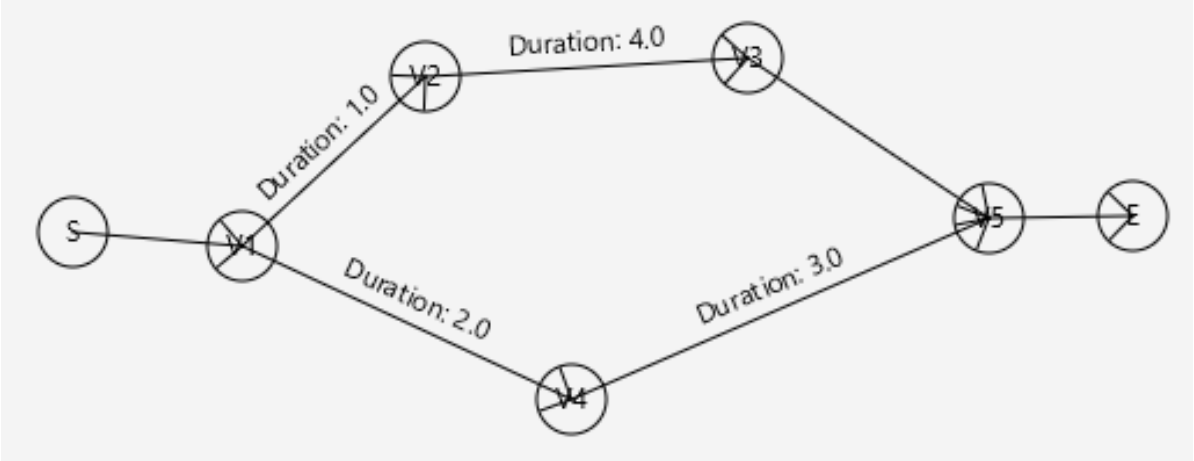
Slack: V2: 0.0 V5: 0.0 V3: 0.0 V4: 0.0 V1: 1.0

End time of selected flow: 7.0



APPENDIX C: PROJECT 3

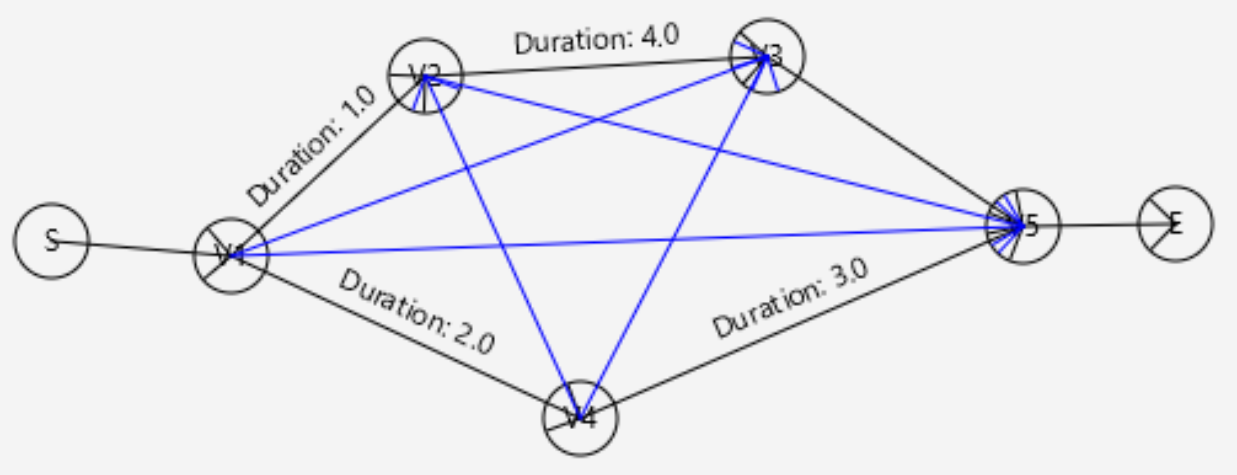
The data represented in this section is of all solutions of Project 3, as found utilising Framework One with use of the original and updated edge selection techniques. For both edge selection techniques, full details are then provided for all solutions with minimum makespan. For each solution the resource flow path used by each individual resource is listed separately, thus resulting in repetition when two resources travel along the same path. To assist with interpretation, note that for the solutions one path is listed per resource, and the same path listed twice therefore indicates that two resources flow along the same path. Further, "Duration" values displayed above certain edges indicate the duration of the transfer, i.e. the transfer time. When no duration is displayed, it is zero.



Activity	d_i	r_i
S	0	0
V_1	5	6
V_2	1	5
V_3	1	2
V_4	4	6
V_5	2	6
E	0	0

6 resources available

Original Edge Selection Technique



Resource Edges:

V1V3

V1V5

V2V5

V4V2

V4V3

Resource Flow Paths (4):

S - V1 - V4 - V2 - V3 - V5 - E -

S - V1 - V4 - V3 - V5 - E -

S - V1 - V4 - V2 - V5 - E -

S - V1 - V4 - V5 - E -

Original Edge Selection Technique Solution Details

2 Feasible Solutions

Summary of Solutions in terms of Makespan and Slack Distribution:

Makespan	Slack 0
19	2

The two solutions with minimum makespan are as follows.

Original Technique Feasible Solution #1

See Section C.1 for details

S - V1 - V4 - V2 - V3 - V5 - E

S - V1 - V4 - V2 - V5 - E

S - V1 - V4 - V2 - V5 - E

S - V1 - V4 - V2 - V5 - E

S - V1 - V4 - V2 - V5 - E

S - V1 - V4 - V3 - V5 - E

Slack: V1: 0.0 V4: 0.0 V5: 0.0 V3: 0.0 V2: 0.0

End time of selected flow: 19.0

Original Technique Feasible Solution #2

See Section C.2 for details

S - V1 - V4 - V2 - V3 - V5 - E

S - V1 - V4 - V2 - V3 - V5 - E

S - V1 - V4 - V2 - V5 - E

S - V1 - V4 - V2 - V5 - E

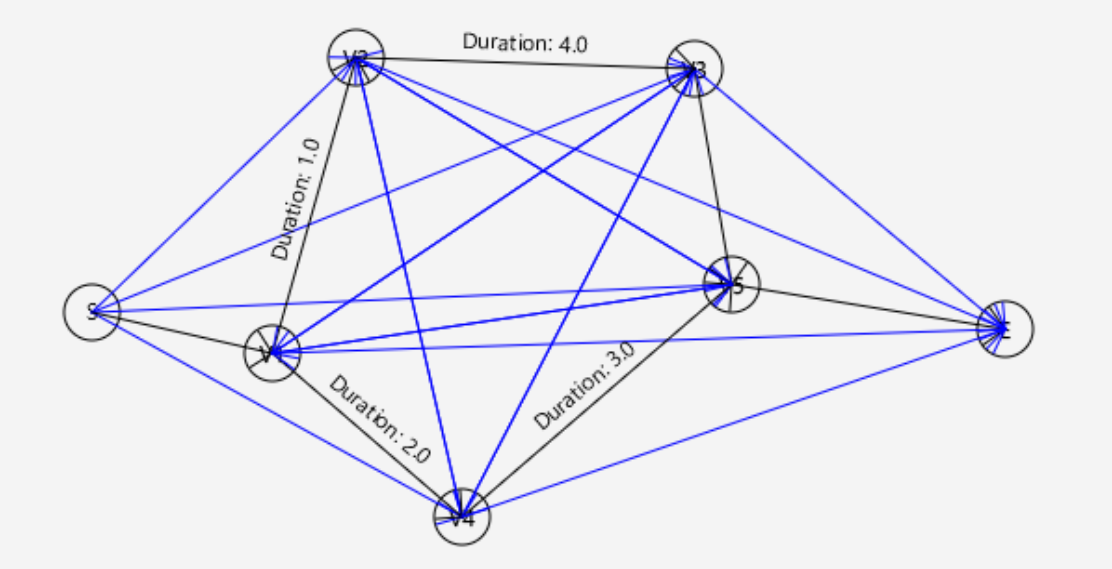
S - V1 - V4 - V2 - V5 - E

S - V1 - V4 - V5 - E

Slack: V1: 0.0 V4: 0.0 V5: 0.0 V3: 0.0 V2: 0.0

End time of selected flow: 19.0

Updated Edge Selection Techniqu



Resource Edges:

SV2
SV3
SV4
SV5
V1V3 / V3V1
V1V5 / V5V1
V1V6
V2V4 / V4V2
V2V5 / V5V2
V2V6
V3V4 / V4V3
V3V6
V4V6

Resource Flow Paths (8):

S - V1 - V2 - V3 - V4 - V5 - E
S - V1 - V4 - V2 - V3 - V5 - E
S - V1 - V2 - V4 - V3 - V5 - E
S - V1 - V4 - V3 - V5 - E
S - V1 - V2 - V4 - V5 - E
S - V1 - V3 - V4 - V5 - E
S - V1 - V4 - V2 - V5 - E
S - V1 - V4 - V5 - E

Updated Edge Selection Technique Solution Details

6 Feasible Solutions

Summary of Solutions in terms of Makespan and Slack Distribution:

Makespan	Slack 0	Slack 2
16		2
19	2	
21	2	

The two solutions with minimum makespan are as follows.

Updated Technique Feasible Solution #1 = Original Technique Feasible Solution #1

See Section C.1 for details

S - V1 - V4 - V2 - V3 - V5 - E

S - V1 - V4 - V2 - V5 - E

S - V1 - V4 - V2 - V5 - E

S - V1 - V4 - V2 - V5 - E

S - V1 - V4 - V2 - V5 - E

S - V1 - V4 - V3 - V5 - E

Slack: V1: 0.0 V4: 0.0 V5: 0.0 V3: 0.0 V2: 0.0

End time of selected flow: 19.0

Updated Technique Feasible Solution #2 = Original Technique Feasible Solution #2

See Section C.2 for details

S - V1 - V4 - V2 - V3 - V5 - E

S - V1 - V4 - V2 - V3 - V5 - E

S - V1 - V4 - V2 - V5 - E

S - V1 - V4 - V2 - V5 - E

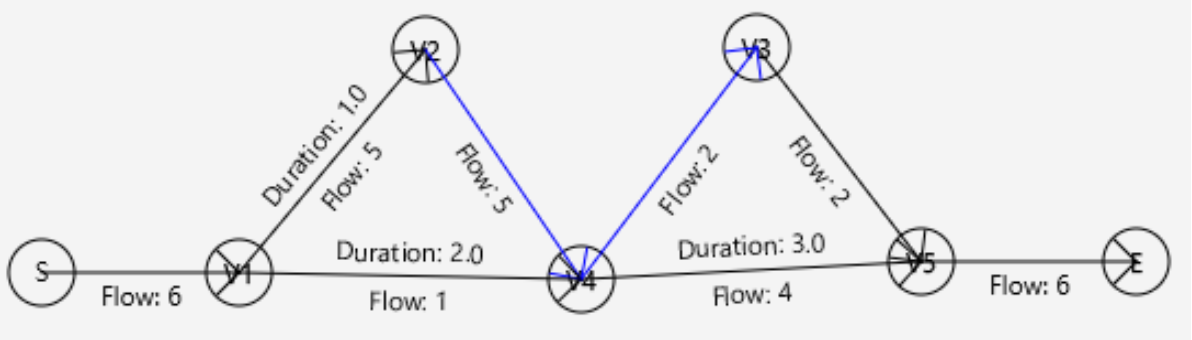
S - V1 - V4 - V2 - V5 - E

S - V1 - V4 - V5 - E

Slack: V1: 0.0 V4: 0.0 V5: 0.0 V3: 0.0 V2: 0.0

End time of selected flow: 19.0

C.1 SOLUTION 1



Resource Flow Paths:

S - V1 - V4 - V2 - V3 - V5 - E

S - V1 - V4 - V2 - V5 - E

S - V1 - V4 - V2 - V5 - E

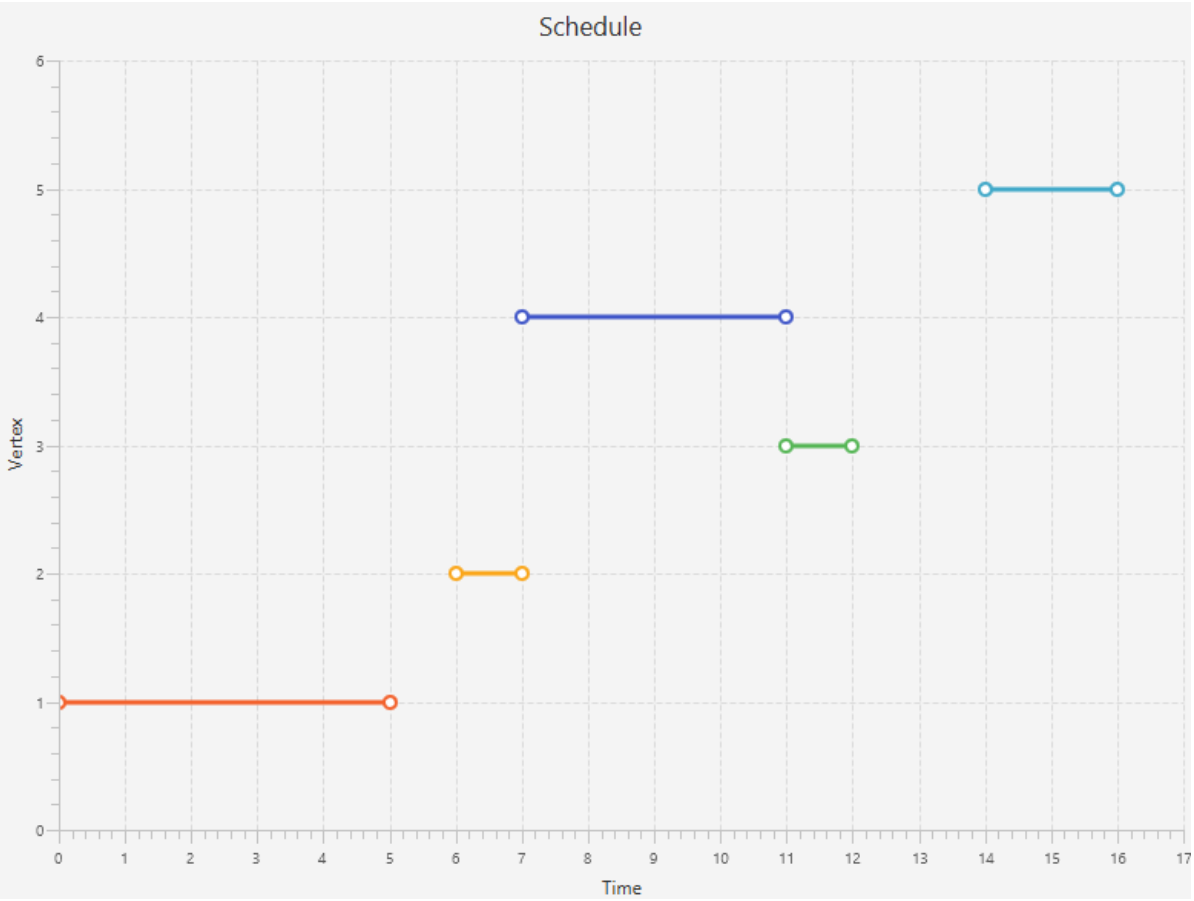
S - V1 - V4 - V2 - V5 - E

S - V1 - V4 - V2 - V5 - E

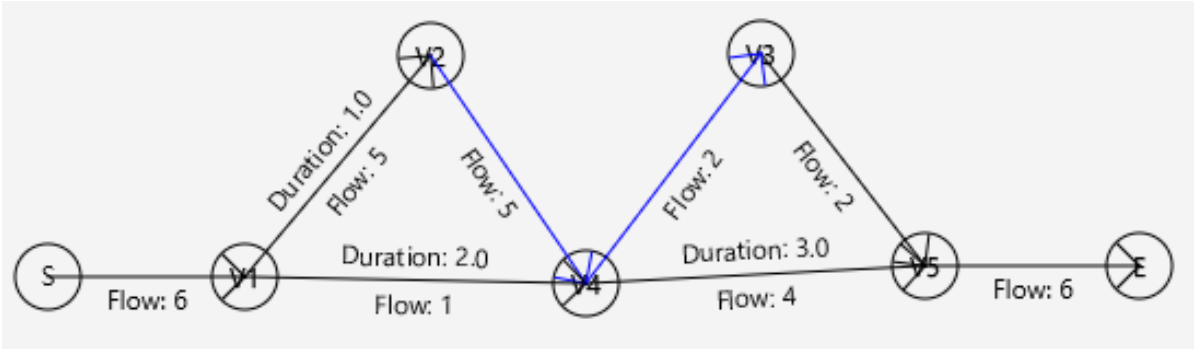
S - V1 - V4 - V3 - V5 - E

Slack: V1: 0.0 V4: 0.0 V5: 0.0 V3: 0.0 V2: 0.0

End time of selected flow: 16.0



C.2 SOLUTION



Resource Flow Paths:

S - V1 - V4 - V2 - V3 - V5 - E

S - V1 - V4 - V2 - V3 - V5 - E

S - V1 - V4 - V2 - V5 - E

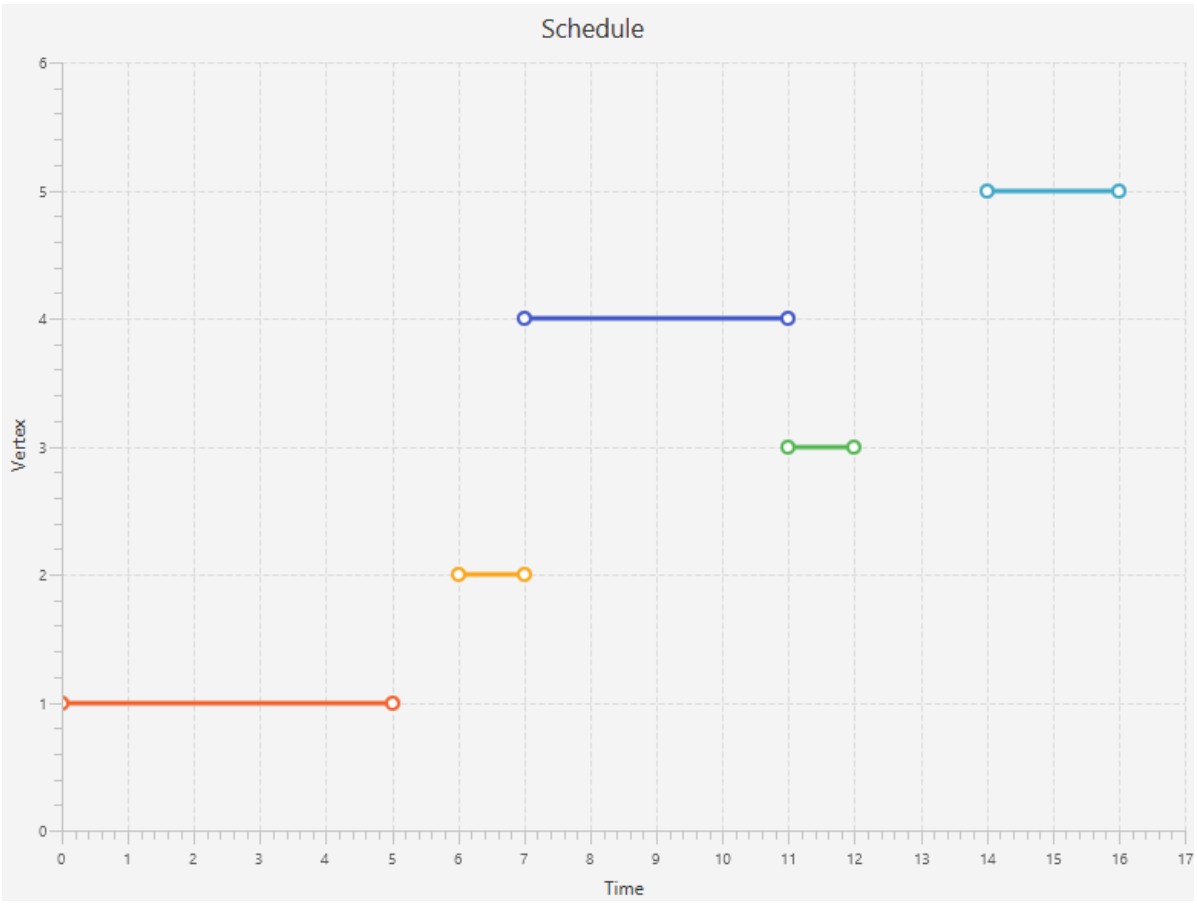
S - V1 - V4 - V2 - V5 - E

S - V1 - V4 - V2 - V5 - E

S - V1 - V4 - V5 - E

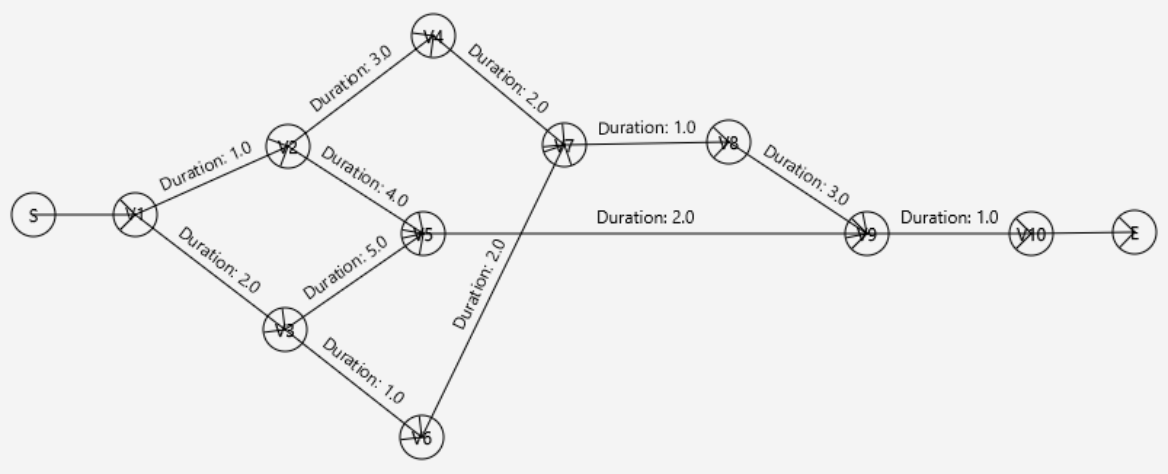
Slack: V1: 0.0 V4: 0.0 V5: 0.0 V3: 0.0 V2: 0.0

End time of selected flow: 16.0



APPENDIX D: PROJECT 4

The data represented in this section is the project input information of Project 4, including technical precedence relationships, transfer times, activity duration and resource requirements as well as the resource availability for the project. Due to the size and complexity of the project Framework One is unable to find all solutions in a reasonable amount of time, and therefore this information is unavailable. To assist with interpretation, "Duration" values displayed above certain edges indicate the duration of the transfer, i.e. the transfer time. When no duration is displayed, it is zero.

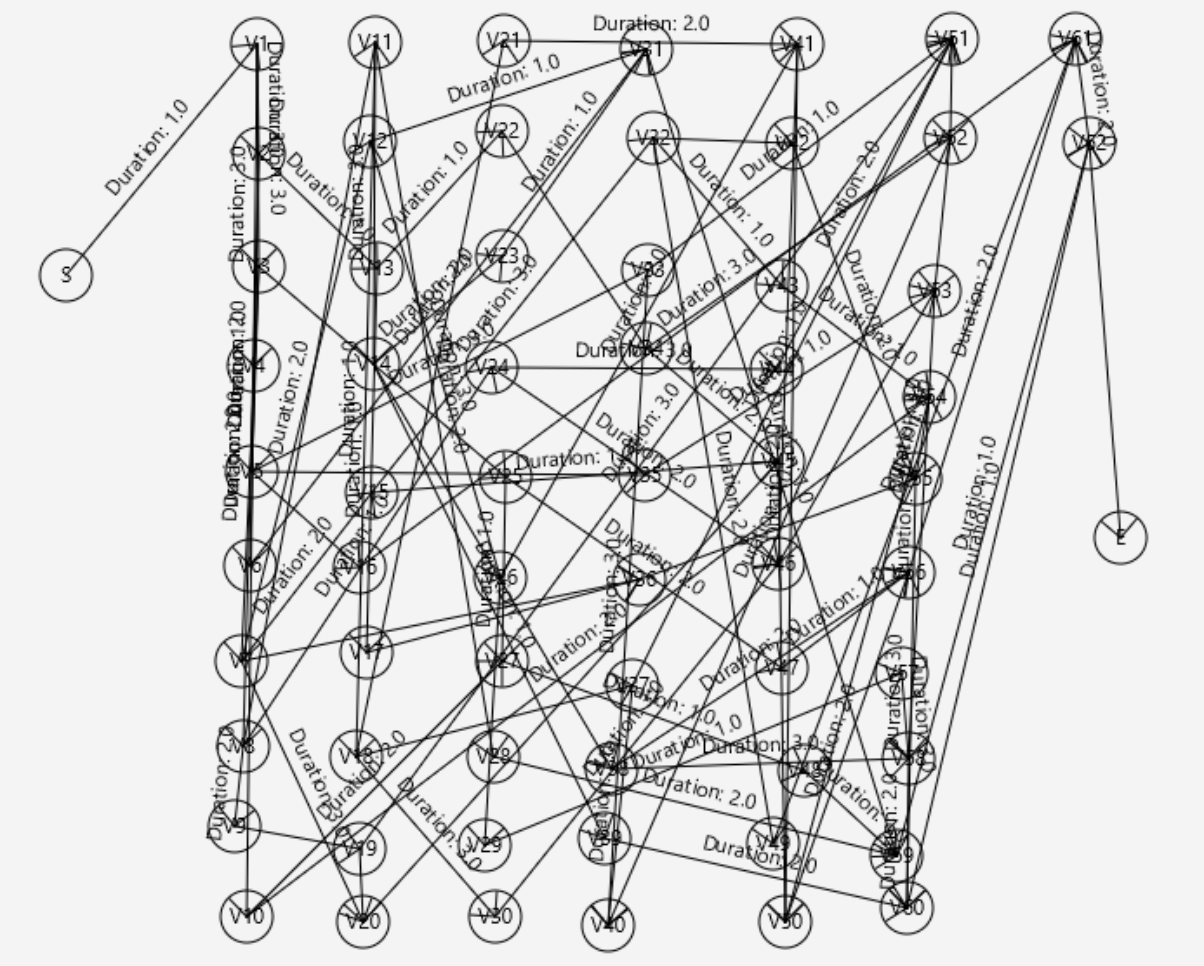


Activity	d_i	r_i
S	0	0
V_1	9	3
V_2	9	6
V_3	8	8
V_4	6	9
V_5	9	8
V_6	3	6
V_7	7	4
V_8	1	4
V_9	9	3
V_{10}	2	5
E	0	0

9 resources available

APPENDIX E: PROJECT 5

The data represented in this section is the project input information of Project 5, including technical precedence relationships, transfer times, activity duration and resource requirements as well as the resource availability for the project. A table is provided with transfer times, as the project complexity means these cannot be read off the provided figure. Due to the size and complexity of the project Framework One is unable to find all solutions in a reasonable amount of time, and therefore this information is unavailable. The edges added utilising the original and updated resource edge selection techniques are available, and are also provide



19 resources available

Activity Details:

Activity	d_i	r_i	Activity	d_i	r_i	Activity	d_i	r_i	Activity	d_i	r_i
S	0	0	V_{16}	7	5	V_{32}	8	9	V_{48}	5	5
V_1	0	19	V_{17}	7	6	V_{33}	1	2	V_{49}	10	10
V_2	4	9	V_{18}	6	3	V_{34}	5	4	V_{50}	1	4
V_3	9	5	V_{19}	3	8	V_{35}	1	1	V_{51}	7	6
V_4	7	3	V_{20}	8	3	V_{36}	10	4	V_{52}	9	5
V_5	2	10	V_{21}	6	1	V_{37}	5	9	V_{53}	7	9
V_6	1	9	V_{22}	4	6	V_{38}	9	3	V_{54}	1	5
V_7	6	1	V_{23}	10	10	V_{39}	1	5	V_{55}	7	5
V_8	8	2	V_{24}	4	8	V_{40}	5	10	V_{56}	10	5
V_9	10	7	V_{25}	7	1	V_{41}	7	5	V_{57}	3	6
V_{10}	8	10	V_{26}	10	1	V_{42}	6	3	V_{58}	8	9
V_{11}	7	8	V_{27}	2	8	V_{43}	9	2	V_{59}	5	7
V_{12}	1	3	V_{28}	7	5	V_{44}	8	5	V_{60}	3	1
V_{13}	2	10	V_{29}	3	1	V_{45}	2	8	V_{61}	7	10
V_{14}	4	1	V_{30}	1	10	V_{46}	1	10	V_{62}	0	19
V_{15}	9	1	V_{31}	8	5	V_{47}	8	5	E	0	0

Transfer Time Details:

Start	End	t_{ij}	Start	End	t_{ij}	Start	End	t_{ij}	Start	End	t_{ij}
S	V_1	1	V_{11}	V_{18}	1	V_{26}	V_{41}	2	V_{42}	V_{55}	2
V_1	V_2	3	V_{11}	V_{28}	2	V_{27}	V_{48}	1	V_{43}	V_{54}	1
V_1	V_3	3	V_{12}	V_{26}	3	V_{28}	V_{59}	2	V_{44}	V_{51}	2
V_1	V_4	3	V_{12}	V_{31}	1	V_{29}	V_{57}	1	V_{47}	V_{56}	1
V_2	V_6	2	V_{13}	V_{17}	1	V_{30}	V_{46}	3	V_{48}	V_{59}	1
V_2	V_8	2	V_{13}	V_{22}	1	V_{31}	V_{59}	1	V_{49}	V_{61}	2
V_2	V_{13}	1	V_{14}	V_{23}	2	V_{32}	V_{43}	1	V_{50}	V_{56}	2
V_3	V_5	1	V_{14}	V_{39}	1	V_{32}	V_{49}	2	V_{53}	V_{58}	1
V_3	V_7	2	V_{15}	V_{32}	3	V_{33}	V_{40}	3	V_{54}	V_{55}	3
V_3	33	3	V_{15}	V_{45}	1	V_{33}	V_{51}	1	V_{55}	V_{60}	3
V_6	31	2	V_{16}	V_{61}	3	V_{34}	V_{45}	2	V_{56}	V_{61}	2
V_7	12	2	V_{18}	V_{30}	3	V_{35}	V_{53}	1	V_{57}	V_{58}	3
V_7	15	2	V_{19}	V_{51}	3	V_{37}	V_{40}	1	V_{58}	V_{60}	2
V_7	20	3	V_{21}	V_{41}	2	V_{37}	V_{51}	1	V_{59}	V_{62}	1
V_8	V_9	2	V_{23}	V_{31}	1	V_{38}	V_{56}	2	V_{60}	V_{62}	1
V_8	24	1	V_{24}	V_{44}	3	V_{38}	V_{58}	3	V_{61}	V_{62}	2
V_{10}	27	2	V_{24}	V_{46}	2	V_{39}	V_{60}	2			
V_{10}	54	2	V_{25}	V_{27}	1	V_{40}	V_{52}	1			
V_{11}	V_{14}	3	V_{25}	V_{47}	2	V_{41}	V_{50}	2			

APPENDIX F: FRAMEWORK ONE GRAPHICAL USER INTERFACE

A java software implementation was created of Framework One. This software allows the user to input data about the project to be analysed, execute the two-part solution process, and access the available solutions. The different aspects of the Graphical User Interface (GUI) will now be highlighted and explained, where the data being used for explanation in this Chapter is Project 1. The steps to be followed to set up and analyse a project will also be summarised in Chapter 0 for convenience. The software discussed in this chapter will be provided on a CD accompanying the thesis.

F.1 MAIN VIEW OF APPLICATION

The main view of the GUI can be seen in Figure 25 where four main components of the GUI are visible. The Menu Bar can be seen at the top of the screen, the Side Panel on the left, the Text Output at the bottom and a Display Panel on the top right. Figure 25 indicates the view once project information has been loaded, and this process will be highlighted in Chapter 0.

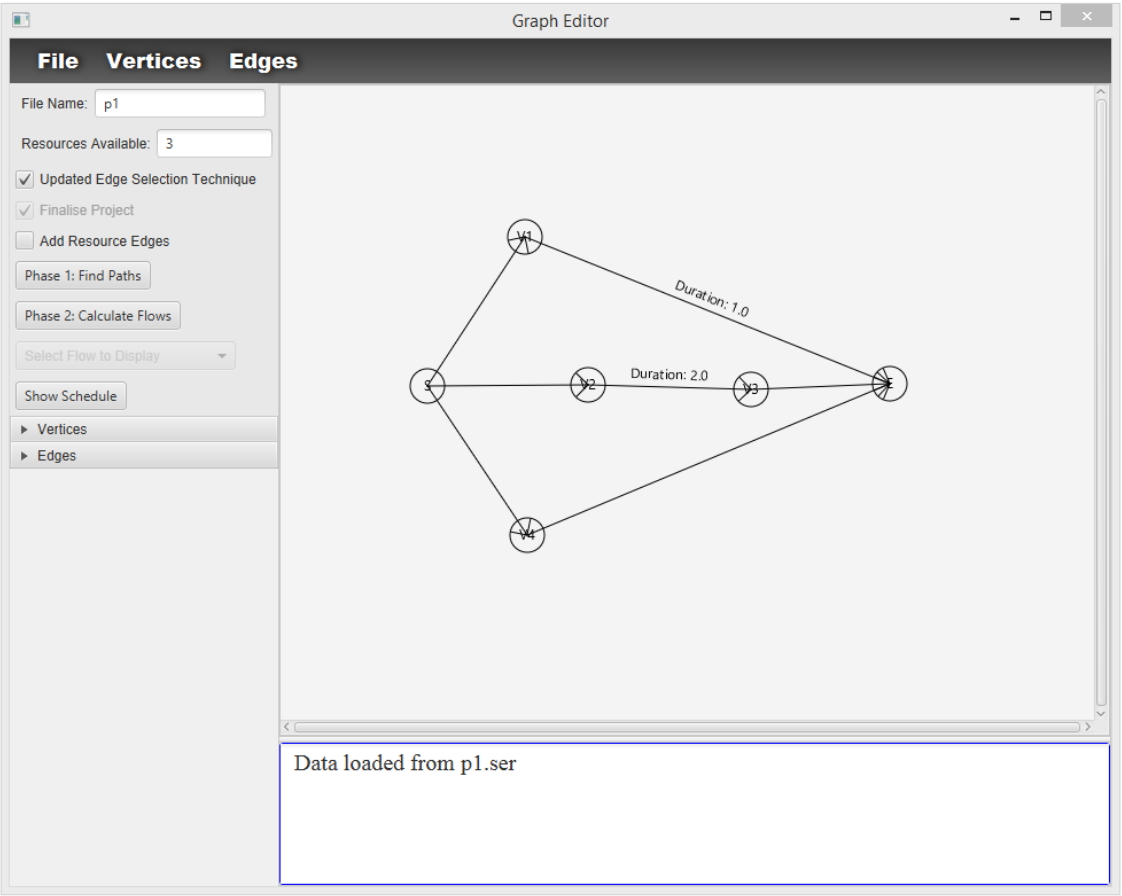


Figure 25: Framework One GUI Main View

On the Display Panel, the layout of Project 1 can be seen. The activities $V_1 - V_4$, dummy start and end activities S and E as well as lines representing precedence relationships, i.e. edges, are visible. Note that any transfer times t_{ij} will be displayed along the applicable edge, as is displayed along edge X_{23} in the figure.

F.2 MENU BAR

The Menu Bar is used to access some components of the functionality of the GUI. As seen in Figure 25, three menus are available, where each has a drop-down list as per Figure 26.

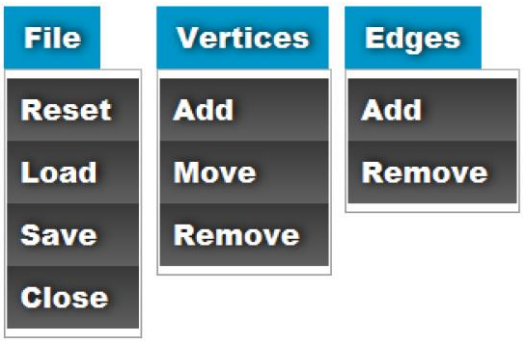


Figure 26: Framework One GUI Menus

In the first menu, *Reset* is used to refresh the GUI and remove any project information that has been inputted, and *Close* is used to exit the GUI. *Load* and *Save* can access and store project information, where the project name should first be entered in the *File Name* field in the Side Panel. Note that a project may only be saved if it has been finalised, in other words the dummy start and end activities must have been added, and this procedure will be further discussed in Chapter 0.

The *Vertices* menu is used to *Add, Move* and *Remove* activities from the project. Once selecting the desired menu option, one must click on the Display Panel to place the activity in the selected location or delete the activity in the selected location, or click and drag to move an activity. The *Edges* menu operates in a similar manner. To *Add* an edge the user must click on the start activity and then the end activity, and to remove an edge click on the edge itself.

F.3 Side Panel

The Side Panel is used to access the majority of the functionality of the GUI. In Figure 27 two separate images are shown. The first is the Side Panel when the *Vertices* drop-down table is shown and the second when the *Edges* drop-down table is shown. When neither of the tables has been selected, the GUI will appear as in Figure 25.

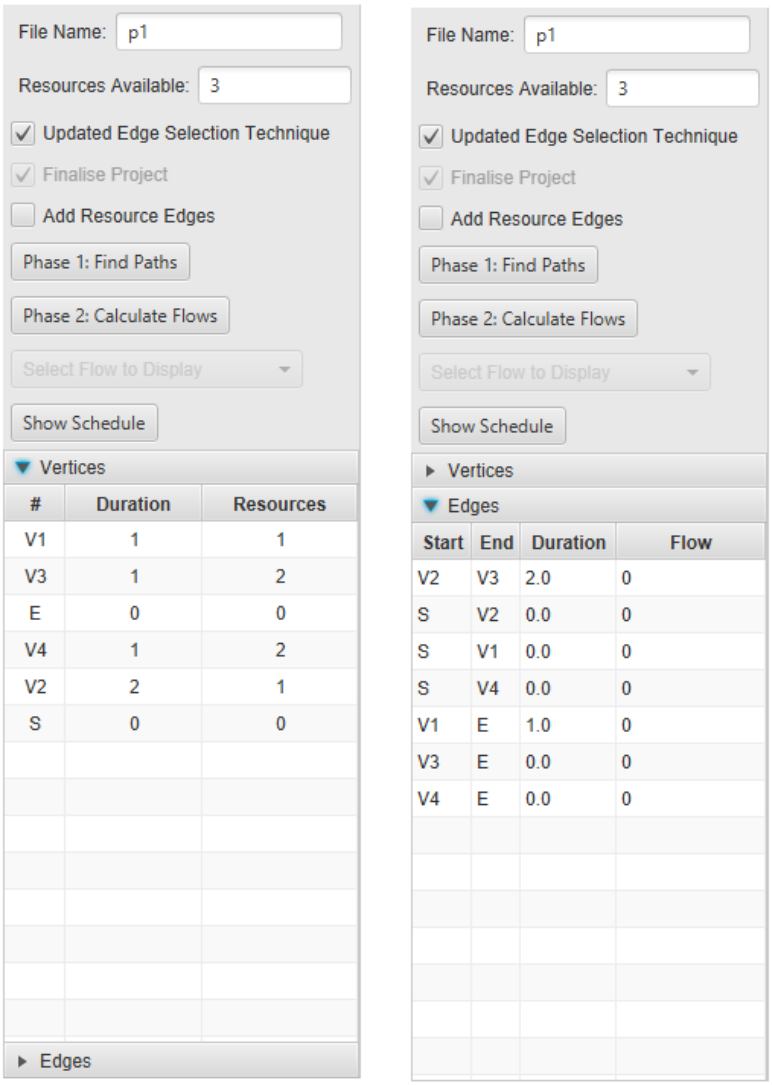


Figure 27: Framework One GUI Side Panel

By selecting *Updated Edge Selection Technique*, the framework will be executed with the updated edge selection Technique as presented in Chapter 7, else the original edge selection technique will be used. Selecting *Add Resource Edges* then adds the resource precedence edges to the graph in accordance with the specified technique. Resource Edges will be dispayed on the Display Panel in blue.

Phase 1: Find Paths and *Phase 2: Calculate Flows* perform the two phase solution process as highlighted in Chapter 5. After Phase 1 all the resource flow paths discovered will be displayed in the Text Output, and after Phase 2 all the feasible flows will be displayed. *Select Flow to Display* will now become available, and one will be able to select any of the available solutions, i.e. Feasible Flows. The Display Panel will then show the selected solution, and when selecting *Show Schedule*

the schedule of the selected solution will be displayed as in Figure 28. The different colours shown in the schedule diagram indicates different activities.

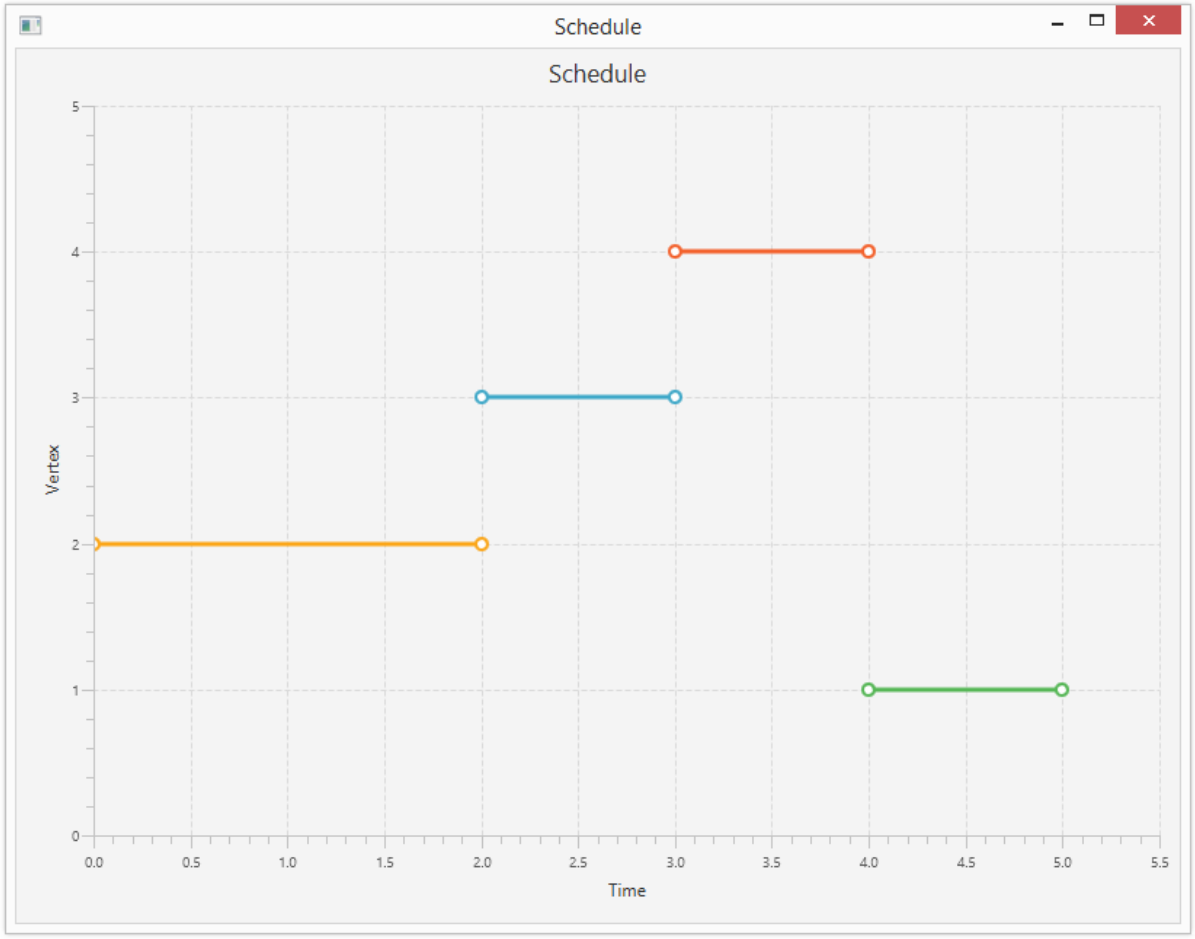


Figure 28: Framework One GUI Schedule

The *Verticies* and *Edges* tables are used to specify information regarding the activities and precedence relationships. For each activity V_j, the *Duration* d_j and number of *Resources* r_j must be specified. For each precedence relationship X_{ij}, the *Duration*, i.e. Transfer Time t_{ij}, can be specified. Double click on the number to be changed, specify a new value, and press Enter. This process is completed during project information input, as will be discussed in Chapter 0. The *Edges* table also has a *Flow* column which will display the flow f_{ij} along an edge when a specific solution is selected.

The use of *Resources Available* and *Finalise Project* will be noted in the following section when the project information input procedure is discussed.

F.4 INPUTTING PROJECT DATA

The user can either load a pre-saved project, or create a new project.

F.4.1 LOADING DATA

To load a pre-saved project:

1. Input project name at *File Name* in the Side Panel
2. *File – Load*

The project information will now be loaded. The project will be viewable on the Display Panel, the *Vertices* and *Edges* tables in the Side Panel will be populated, and in the Text Output "*Data loaded from…"* will be displayed. The available projects to load are: *P1, P2, P3, P4, P5,* representing projects 1 through 5 as utilised in this thesis.

F.4.2 ENTERING DATA

To create a new project:

1. *Vertices – Add*
   Repeat for each vertex, i.e. activity, to be added.
2. *Edges – Add*
   Repeat for each edge, i.e. precedence relationship, to be added.
3. Enter the number of *Resources Available* in the Side Panel
4. Complete *Vertices* table in Side Panel
   For each vertex the *Duration* and number of *Resources* must be specified.
5. Complete *Edges* table in Side Panel
   For each edge the *Duration*, i.e. Transfer Time, can be specified, else it is zero.
6. Select *Finalise Project* in Side Panel
   Select the location of the start vertex and then the end vertex on the Display Panel.

Once project information has been entered, the information can be saved by filling in a name at *File Name* and then selecting *File – Save.*

F.5 DISPLAY PANEL

The display panel has two modes, namely the normal project information display or the display when a solution has been selected. The manner of selecting a solution has been discussed in Chapter F.3. The two different modes are shown in Figure 29 and Figure 30.

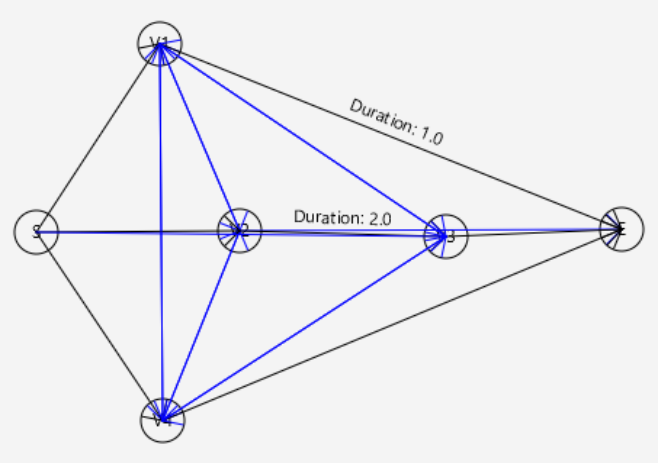


Figure 29: Framework One GUI Normal Project Information Display

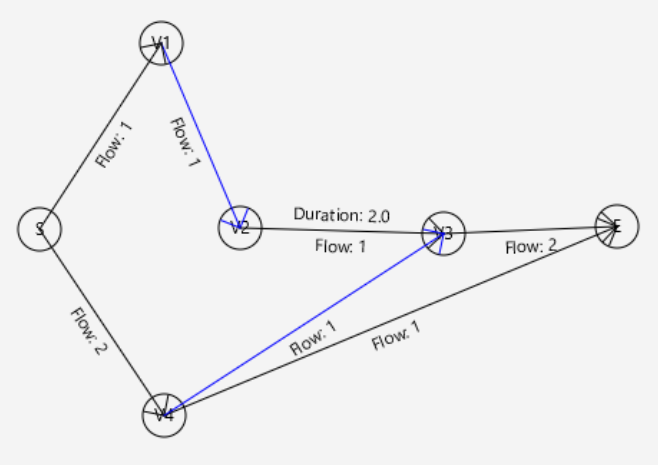


Figure 30: Framework One GUI Solution Display

During normal project display, all project information is shown as per Figure 29, except for the blue edges. These edges are only displayed once the user elects to add resource precedence edges to the graph. If resource precedence edges are not yet added, project information will be displayed as per Figure 25. When the user selects to display a specific solution, the display mode will change: only edges used to transfer resources will be visible and the flow of resources on each edge will be displayed, as per Figure 30.

Note the “Duration” values displayed above certain edges. This refers to the duration of an edge, namely the transfer time associated with the transfer. When no duration is displayed, it is zero.

F.6 TEXT OUTPUT AREA

The text output area at the bottom of the GUI allows the user to see information regarding the project and solution process. Some examples of the information displayed after certain operations are as follows:

- **Add Resource Edges**

  Displays which edges have been added.

  - *Resource Edges Added:*
  - *V1V3*

- **Phase 1: Find Paths**

  The paths discovered by Phase 1 are displayed, as well as the number of paths remaining after the Unnecessary Flows Algorithm is executed.

  - *ORIGINAL METHOD PHASE 1:*
  - *Number of Paths: 45*
  - *Path: S - V1 - V4 - V2 - V3 - E –*
  - *….*
  - *Number of Paths after Unnecessary Flows Algorithm: 45*

- **Phase 2: Calculate Flows**

  All different solutions, i.e. feasible flows, are listed including the paths, slack and duration of each solution. One path is listed per resource, and the same path listed twice therefore indicates that two resources flow along the same path.

  - *UPDATED METHOD PHASE 2:*
  - *Feasible flows #1*
  - *Path: S - V2 - V4 - V1 - E -*
  - *Path: S - V3 - V4 - E -*
  - *Path: S - V3 - E -*
  - *Slack: V4: 0.0 V2: 0.0 V1: 0.0 V3: 0.0*
  - *End time of selected flow: 6.0*

- **Show Schedule**

  Displays the scheduled time of all activities in the selected solution, as well as the late start (LS), late end (LE) and slack of each activity

  - *Start time, end time, LS, LE, slack per activity:*
  - *S : 0.0 , 0.0 , 0.0 , 0.0 , 0.0 START*
  - *V4: 3.0 , 4.0 , 0.0 , 0.0 , 0.0*
  - *V2: 0.0 , 2.0 , 0.0 , 0.0 , 0.0*
  - *V1: 4.0 , 5.0 , 0.0 , 0.0 , 0.0*
  - *V3: 2.0 , 3.0 , 0.0 , 0.0 , 0.0*
  - *E : 6.0 , 6.0 , 0.0 , 0.0 , 0.0 END*

F.7 GUI USER MANUAL

The procedure below is a guideline for a user to input project data, analyse the project and view all available solutions and the related schedules.

1. Input graph data as per 0
2. Select or unselect '*Updated Edge Selection Technique*'
3. *Add Resource Edges*
4. *Phase 1: Find Paths*
5. *Phase 2 Calculate Flows*
6. *Select Flow to Display* and *Show Schedule* to visualise a solution and its schedule

F.8 CONCLUSION

This Appendix outlines the functionality of the GUI for Framework One, and for convenience a step by step procedure to utilise the GUI is also provided. The information regarding all activities, precedence relationships and resource flow paths is available, as well as each solution and the related schedule.

APPENDIX G: FRAMEWORK TWO GRAPHICAL USER INTERFACE

A java software implementation of Framework Two, named ProBaSE, was available as per Potgieter (2014). The software was updated throughout the thesis as changes to the model were made, and functionality was added as required. The software allows the user to import data, perform the two-phase solution process, and view the selected solution. Any changes to the Graphical User Interface (GUI) will be specifically noted, and the functionality of the GUI thoroughly explained. The steps required to utilise the software will be summarised in section G.8 for convenience, and the software will be provided on a CD accompanying this thesis. Note that the project data used during this chapter is that of Project 5. For further reading, an overview of the GUI was provided by Potgieter (2014) in Chapter 5.6 and 6.5.

G.1 MAIN VIEW OF APPLICATION

The main view of the application can be seen in Figure 31. On the right one can see a Side Panel with four different tabs, namely *General*, *Dates*, *Resource Usage* and *Paths*. In the figure the *General* tab is displayed from which the solution process is initiated, and the *Dates, Resource Usage* and *Paths* tabs are related to the output information. The remainder of the main application view is a where project information will be displayed in the form of a Gantt chart.

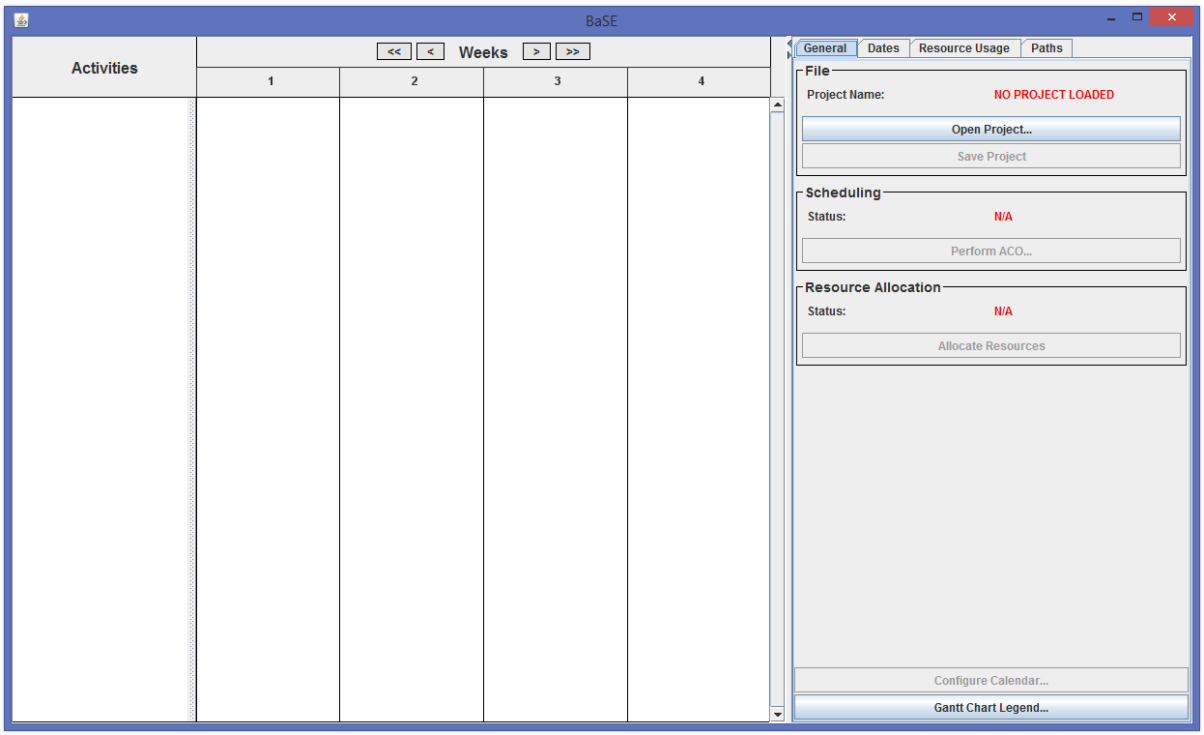


Figure 31: Framework Two GUI Main View

G.2 General Tab

The *Open Project* and *Save Project* buttons allow the user to import and export a project. The creation of the required input file will be discussed in section 0. The *Perform ACO* and *Allocate Resources* buttons initiate the two part solution process, as introduced in Chapter 6.

When selecting *Perform ACO* a window appears, as shown in Figure 32, where the user can adjust the ACO input parameters for the baseline selection of Phase 1.

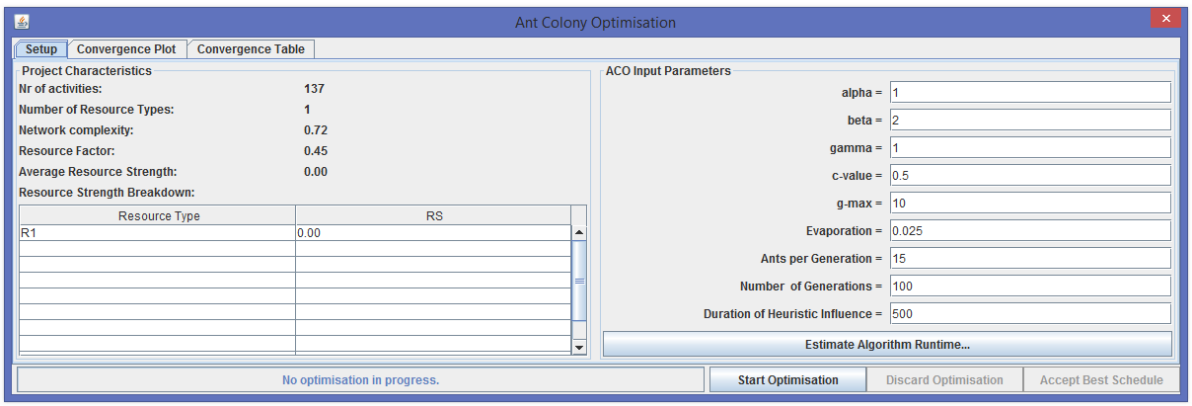


Figure 32: Framework Two Phase 1 ACO Window

As can be seen in the figure, default ACO input parameter values are provided. These values are considered a good starting point, and guidelines for the further adjustment of these parameters are presented by Potgieter (See section 5.5 of Potgieter 2014). On the left of the window, important project characteristics are automatically calculated to assist with this process. These properties include the number of activities, number of resource types, network complexity, resource scarcity and the resource strength of the project. Note that for the model presented in the thesis there will only be one resource type, and the number of activities noted will include transfer tasks. The *Estimate Algorithm Runtime* button will allow the user to estimate the runtime based on the specified input parameters, and can further assist with parameter selection when only a certain amount of time is available for the scheduling process.

After the optimisation has been complete, the user can analyse the behaviour of the ant colony in the *Convergence Plot* and *Convergence Table* tabs of the window. Once the user selects to *Accept Best Schedule,* the Gantt chart of the selected baseline schedule will be displayed by the GUI. For further details about the analysis functionality, refer to Potgieter (2014).

By selecting *Allocate Resources*, Phase 2 of the solution process is commenced. A window appears to let the user select the ACO parameters as well as the objective function parameters and the window is shown in Figure 33. The ACO used in Phase 2, the objective functions utilised by Phase 2, and therefore the window are all new features added to the model and GUI.

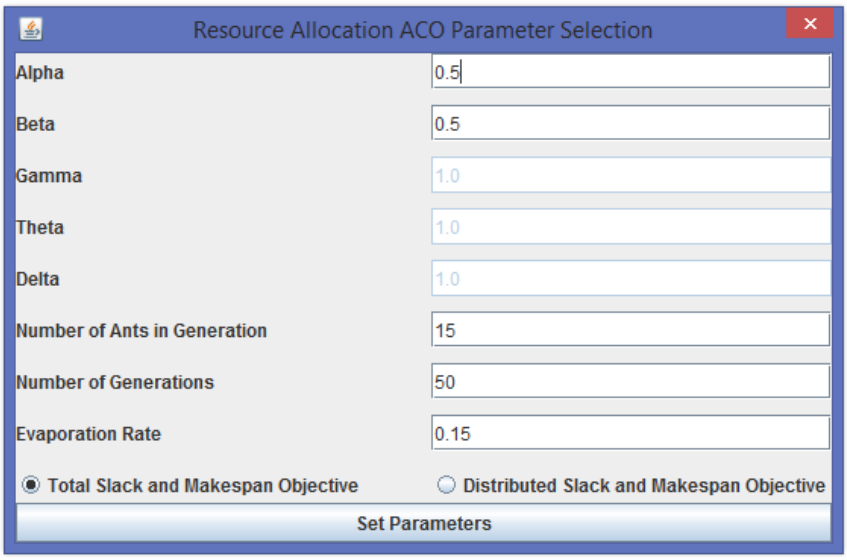


Figure 33: Framework Two Phase 2 ACO Window

The meaning of each parameter was discussed in Chapter 11. When *Total Slack and Makespan Objective* is selected, the objective function presented in section 11.2 will be utilised, else the objective function in section 11.3 will be used. By selecting *Set Parameters*, the window will close, Phase 2 will commence, and when completed the final schedule will be displayed by the GUI.

Configure Calendar allows the user to specify the hours of the standard work week, the start date and time of the project, as well as specify days that do not conform to the standard work week. This means that schedule information can be displayed by time and date. The window used to configure the calendar is shown in Figure 34.

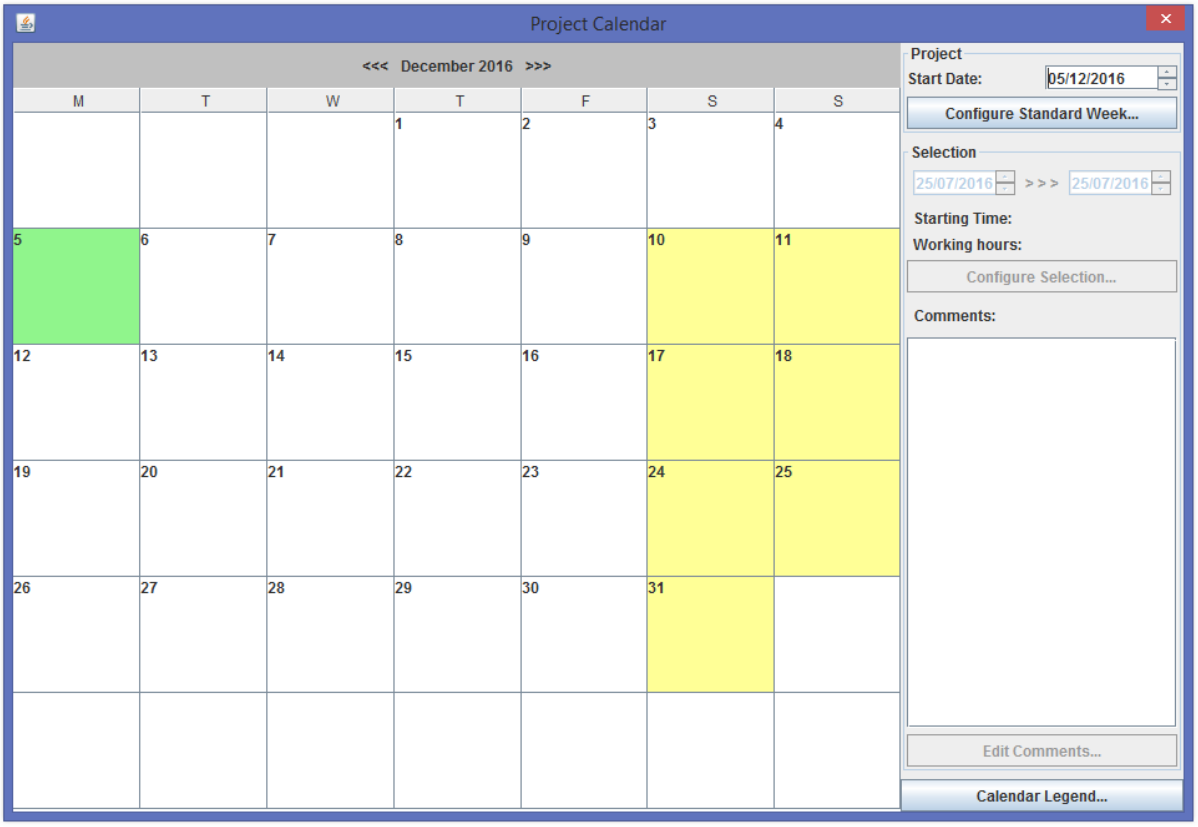


Figure 34: Framework Two Project Calendar Configuration Window

G.3 GANTT CHART DISPLAY

The baseline schedule produced after Phase 1 as well as the final schedule produced after Phase 2 will be displayed in the Gantt Chart Display area, shown in Figure 35. Through use of the different tabs in side panel, different attributes of the schedule can then be viewed, as will be

explained in the sections that follow. As can be viewed in the figure, transfer tasks are listed separately. 'T11-14' indicates a resource transfer from V11 to V14.

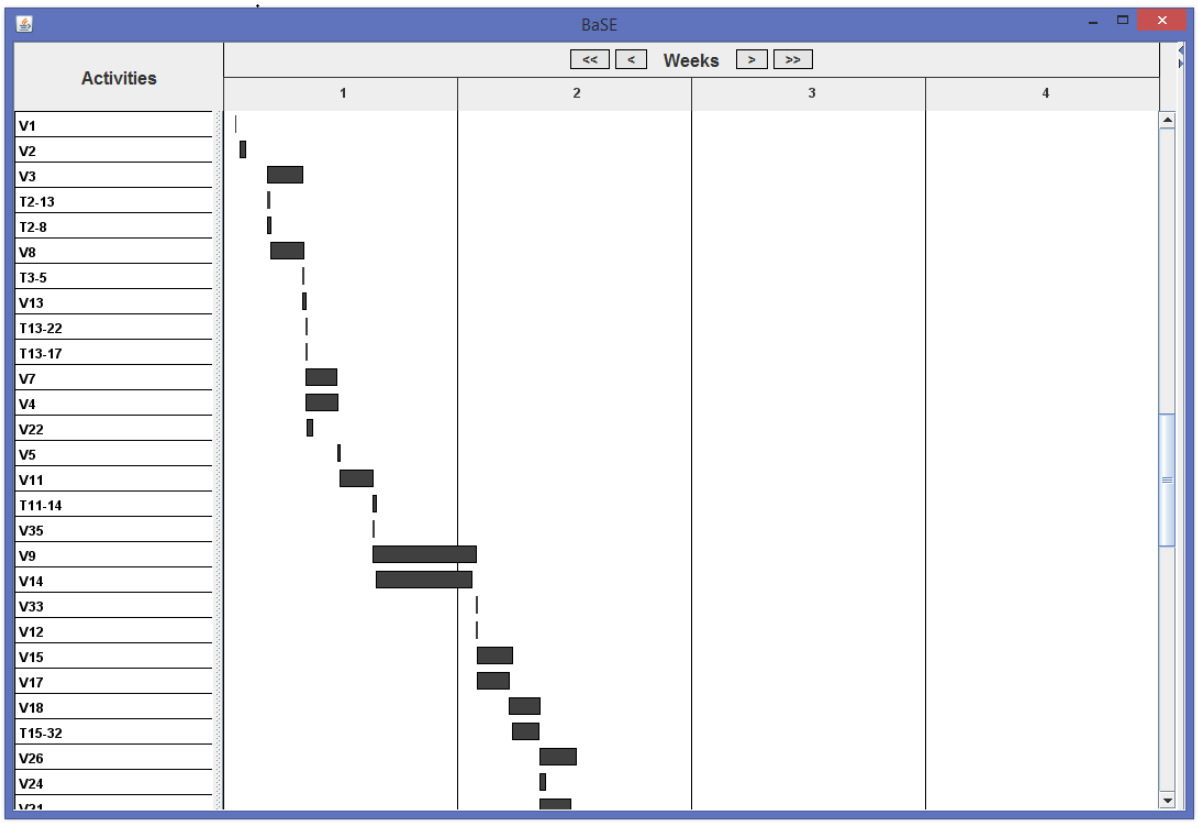


Figure 35: Framework Two Gantt Chart Display

G.4 DATES TAB

The dates tab shows a variety of information regarding the scheduled times of an activity, for the resource-constrained and resource-unconstrained environment. When an activity is selected in the Gantt chart, information is displayed as per Figure 36. The information displayed includes the start and end times, late start and finish as well as slack for both the constrained and unconstrained case.

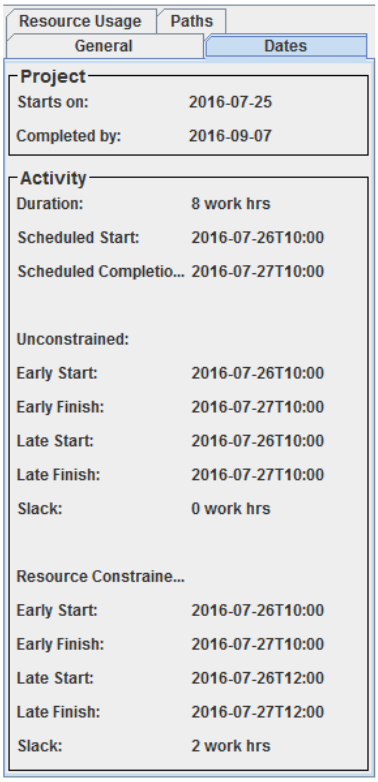


Figure 36: Framework Two Dates Tab

G.5 Resource Usage Tab

When an activity is selected in the Gantt chart, the resource usage tab shows the resource requirements of the activity as per Figure 37. When resource allocations have been completed, the resource allocations will also be displayed.

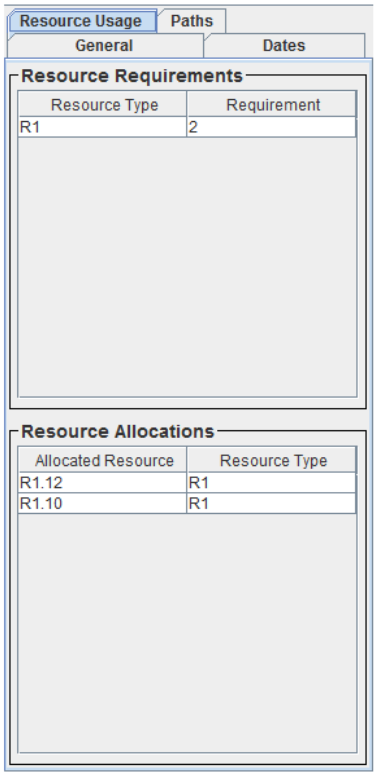


Figure 37: Framework Two Resource Usage Tab

G.6 PATHS TAB

The paths tab allows the user to view activity dependency paths, critical paths and resource paths, as shown in Figure 38.

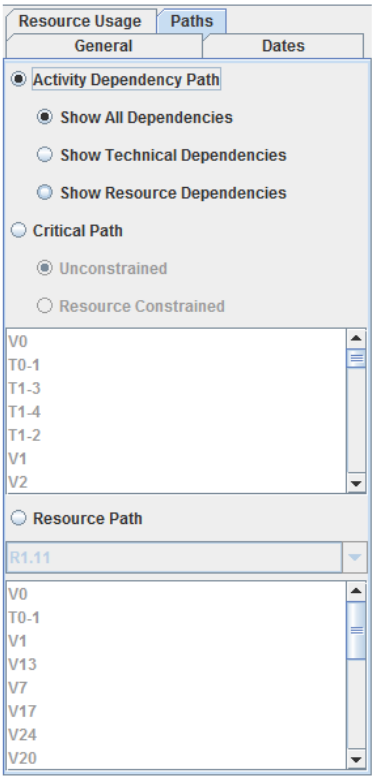


Figure 38: Framework Two Paths Tab

Activity dependency paths show the predecessors and successors of an activity. The predecessors and successors can be calculated considering technical dependencies, resource dependencies, or both. An example of visualising activity dependency paths can be seen in Figure 39, where predecessors are shown in green and successors in orange.

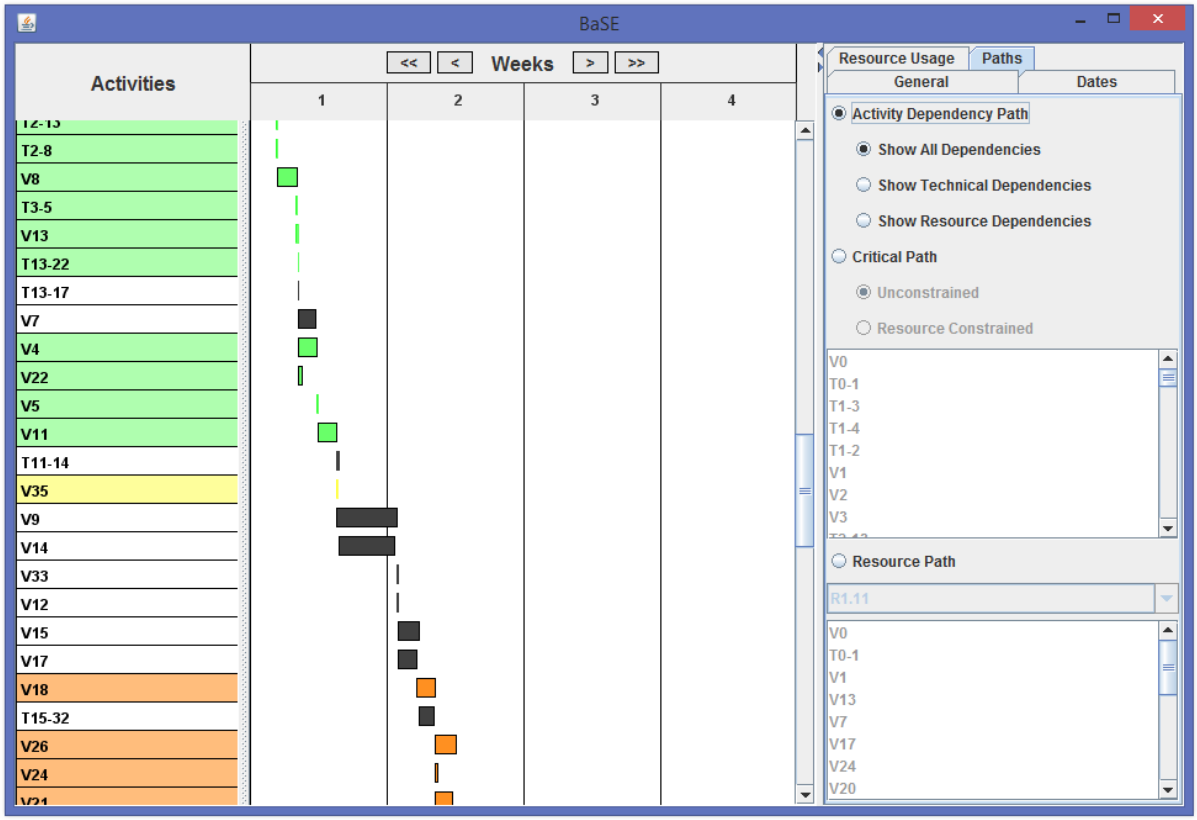


Figure 39: Framework Two Activity Dependency View

Critical paths can be viewed for either the resource-constrained or resource-unconstrained case. The critical activities will be shown in red, as per Figure 40.

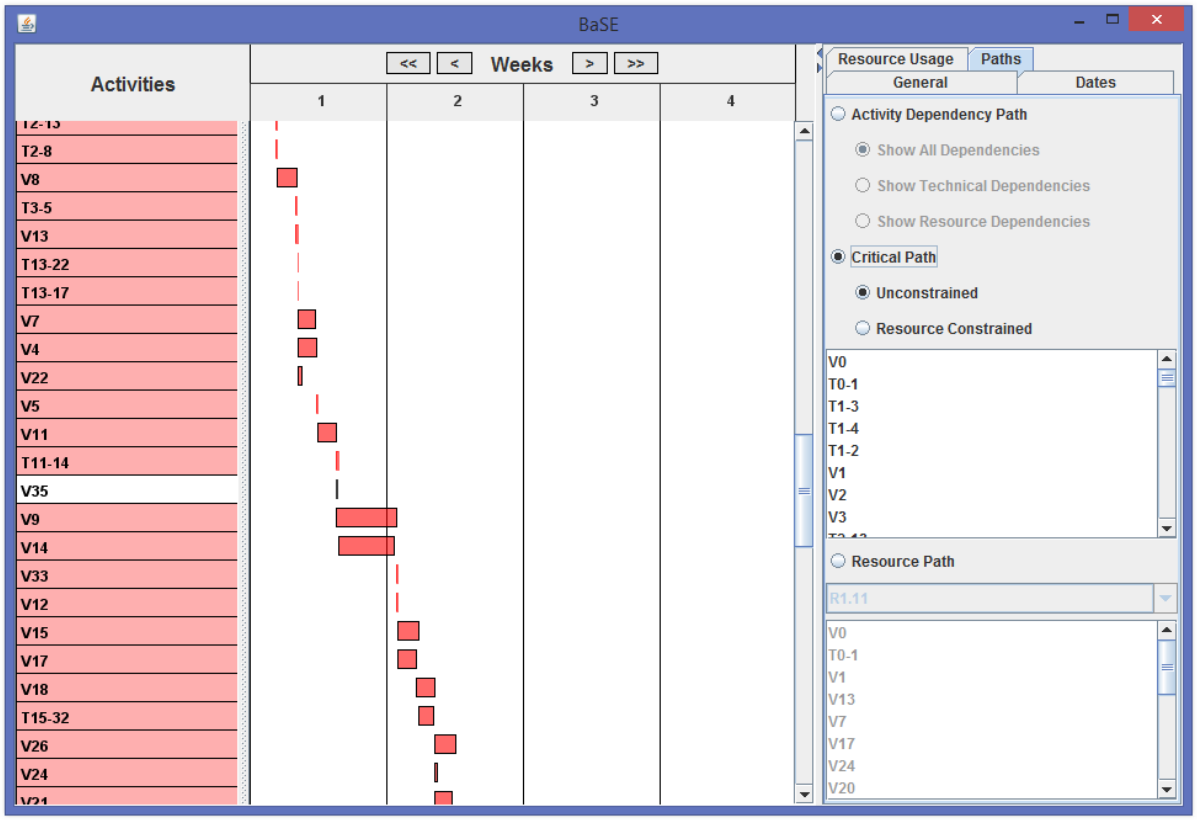


Figure 40: Framework Two Critical Path View

A resource flow path, i.e. resource path, has been defined. When selecting a specific resource, the movement of the resource through the schedule will be shown in blue, as per Figure 41.

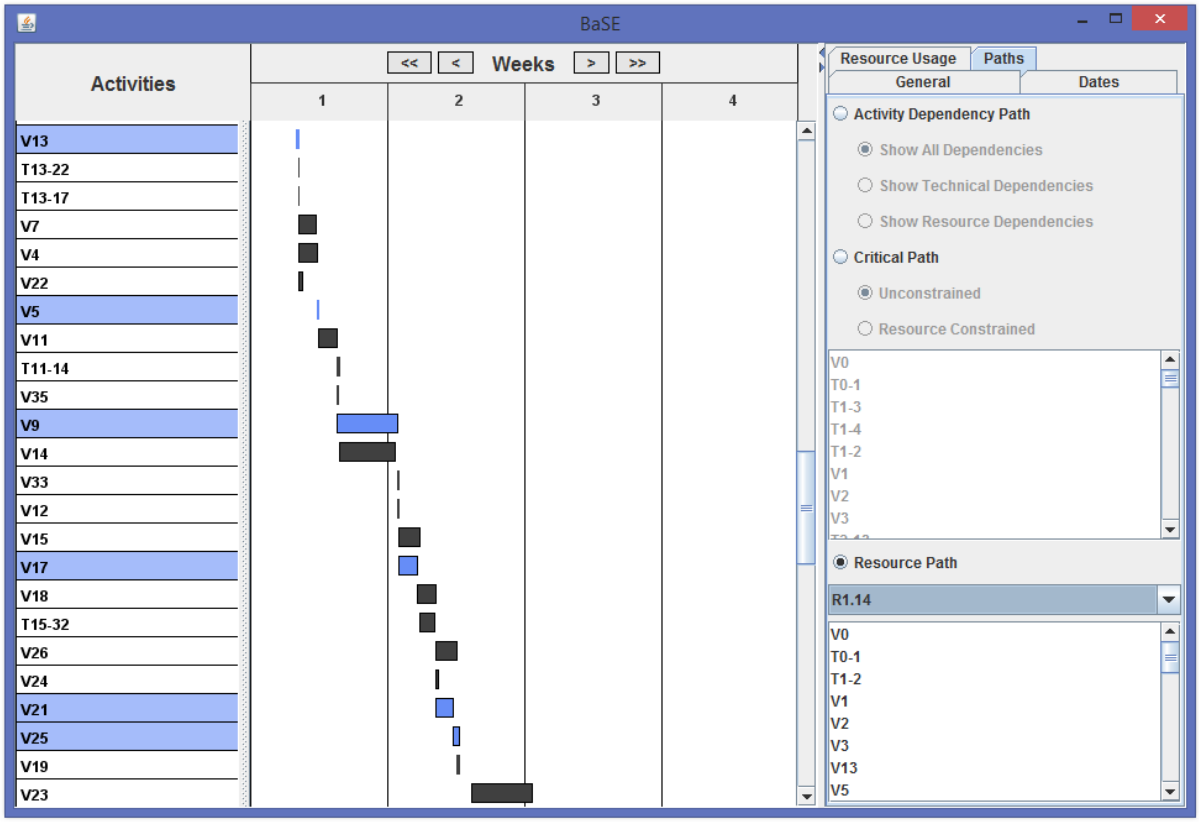


Figure 41: Framework Two Resource Path View

G.7 INPUTTING PROJECT INFORMATION

Project information files are stored as XML files. Examples of input files are given along with the source code of Framework Two, and due to XML files being readable should be easily understandable. Files of this nature can then be created by the user. For the purposes of this thesis and to simplify the project information input process, a converter named Probase_filecreator.java was created in order to convert files saved by Framework One to the correct format required by Framework Two. This file is provided along with the source code of Framework One on a CD attached to the thesis.

Numerous project files are available to load, namely Project 1 through 5, with and without the baseline set. Note that Projects 1 through 4 are represented in the software with 1 time unit equal to 40 hours to visualise these projects more easily. In the appendix for each of these projects, as well as in the thesis, the time units are used unless specifically stated that the unit in converted as stated here.

G.8 GUI USER MANUAL

The following is a guideline for the user to input project data, analyse the project data and view the details of the selected solution.

1. Import project information
2. *Perform ACO*
3. Select Phase 1 ACO parameters
4. *Allocate Resources*
5. Select Phase 2 ACO parameters and objective function parameters
6. View solution details by using the *Dates, Resource Usage* and *Paths* tabs

G.9 CONCLUSION

Chapter 0 has given an overview of the functionality of the software implementation of Framework Two, and outlined the procedure to be followed to use the application. The information of all activities, resources and precedence relationships will be available, as well as the final schedule including all resource allocations, resource flow paths and other required scheduling information.