Team 8

[Email address]

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

Innov8 data solutions

[Document subtitle]

Contents

[Executive Summary 2](#_Toc458762645)

[Introduction 3](#_Toc458762646)

[Method 4](#_Toc458762647)

[How we did it 4](#_Toc458762648)

[Options 5](#_Toc458762649)

[Algorithm 5](#_Toc458762650)

[Parallelisation 5](#_Toc458762651)

[Visualistation 5](#_Toc458762652)

[Results 6](#_Toc458762653)

[Discussion 7](#_Toc458762654)

[A\* Implementation 7](#_Toc458762655)

[Parallelisation 7](#_Toc458762656)

[Visualisation 7](#_Toc458762657)

[Conclusion 8](#_Toc458762658)

[References 9](#_Toc458762659)

# Executive Summary

# Introduction

## Problem Description

The scheduling problem is based on parallel computing where tasks are scheduled to run on multiple processors in an attempt to reduce total runtime. The process entails assigning each task to a processor in an order that preserves their dependencies and optimises performance. This is an NP-hard problem, meaning that an efficient algorithm has not yet been discovered to solve it optimally.

## Project Description

Our company was approached by a leading Parallel Computing Centre for a proposed solution to this problem. The client’s requirements are as follows:

* The program needs to produce a valid, optimal solution with minimal runtime.
* The program must contain a multithreaded implementation to improve the search performance.
* The search needs to be visualised.

The input to the program will contain all the tasks, their execution times, the dependencies between tasks and the communication costs. This data can be represented as a directed acyclic graph. In addition to the graph, the number of available processors will be passed into the program. Although the number of potential schedules is very large, it is still finite. This allows for the use of a combinatorial optimisation approach. The proposed algorithm, therefore, should take into consideration all possible valid schedules while avoiding schedules that are clearly non-optimal.

# Method

## How we did it

Planning, coding, tasks

# Options

## Algorithm

## Parallelisation

This we considered bt didn’t do

## Visualistation

# Results

# Discussion

## A\* Implementation

Stuff

## Parallelisation

ghf

## Visualisation

sss

# Conclusion

# References