Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Explainable Optimal Scheduling for Workforce Management using Argumentation

Author:
Jennifer Leigh

Supervisors:
Prof. Francesca Toni
Dr. Dimitrios Letsios
Mr. Lucio Machetti

Submitted in partial fulfilment of the requirements for the MSc Degree in Computing of Imperial College London

September 2023

Abstract

This project explores the application of argumentation theory to scheduling optimisation problems relating to workforce management. Motivated by the need to enhance user workflow at *Terranova*, we extend existing frameworks to address complex scheduling challenges, providing natural language reasoning for scheduling decisions to improve transparency and trust in AI systems.

We implement this extended theory in a proof-of-concept tool to analyse and enhance scheduling processes. The tool significantly improves user accuracy and efficiency, offering a valuable resource for users seeking to enhance their schedules. This contributes to more efficient and user-friendly scheduling solutions, with implications across various industries.

Acknowledgements

I would like to thank Prof. Francesca Toni and Dr. Dimitrios Letsis for their support, guidance and constructive feedback throughout my project. I'd also like to thank Mr. Lucio Machetti and Dr. Alessandro Mella from *Terranova* for their support during the development and deployment of the user study, as well as for providing valuable suggestions and feedback during the building of my scheduling tool.

Lastly, I'd like to thank my friends and family who have aided me during my project, particularly by allowing me to test my tool on their laptops (and occasionally reconfigure their Windows environment to make my code run).

Contents

Chapter 1

Introduction

In the field of operational logistics and resource management, a central question arises: when faced with a set of tasks and a group of machines, how can we optimise task assignments to minimise the overall completion time? This inquiry has practical applications in various domains, from creating school timetables to organising nurse schedules[1]. These scheduling challenges can be formulated as optimisation problems, where the objective is to minimise a cost function while adhering to constraints related to task-machine assignments.

Argumentation theory, initially introduced to formalise common-sense reasoning and clarify the underlying logic in human conversations[2], has found relevance in addressing some aspects of such optimisation problems. It aids in understanding the relationship between abduction (deducing the most likely conclusion from known facts) and logical programming[3]. In the era of growing artificial intelligence and its opaque decision-making processes, argumentation offers a means to provide natural language explanations for the decisions that lead to specific outcomes.

1.1 Motivation

This project delves into the domain of scheduling optimisation, with a specific focus on applying argumentation theory using data from *Terranova*[4]. Frameworks and mapping previously defined by Čyras et al.[5] involve the allocation of jobs to machines while optimising the makespan, considering fixed job-machine requirements. However, scheduling challenges often extend beyond these boundaries, encompassing complexities such as varying task durations, multiple optimisation criteria, spatial considerations, nuanced decision distinctions, and temporal constraints.

In particular, consider the following motivation questions:

1. What if machines take different lengths of time to complete the same job?
2. What if there are multiple optimisation criteria that should be considered, rather than just processing time/makespan?
3. Can we consider distance as an optimisation criterion?
4. What are these “fixed decisions”? What if we want to differentiate between them?
5. What if there are timing limitations on when jobs can be completed or during which machines can work?

These are all questions that arise in the case of *Terranova*. *Terranova* develop software providing "end-to-end support for Smart Metering, Smart Grid and Smart Workforce management processes and user workflow"[4].

In particular, there exists a pronounced demand for enhancing the transparency of solutions in scheduling problems. The opaqueness inherent in optimality tools often leaves users in the dark about the rationale behind the generated schedules, causing a lack of confidence and understanding between users and their automated scheduling systems. Our objective is to formulate comprehensive frameworks that offer users straightforward, easily comprehensible explanations and practical suggestions for enhancing scheduling outcomes. Importantly, this should be accomplished without necessitating a deep understanding of intricate algorithms or unfamiliar notations.

Another compelling motivation for this project stems from the evolving needs of organisations to adapt and make real-time adjustments to their schedules. Frequently, operational conditions within work environments undergo modifications during scheduled activities, necessitating immediate alterations in job allocation and workforce management.

Our motivation is rooted in the practical needs of companies like *Terranova*, which grapple with the computational complexity of finding optimal solutions to scheduling problems. To enhance operational efficiency, such companies often resort to suboptimal algorithms. While these algorithms yield reasonably efficient results, there is often room for improvement, leaving users uncertain about the efficiency of potential enhancements.

1.2 Objectives

This project has two objectives: firstly, we aim to harness argumentation theory as a valuable tool to address a specific scheduling optimisation challenge, leveraging data from *Terranova*. This optimisation problem involves a range of constraints and variables, extending the frameworks and mappings presented by Čyras et al.[5]. Our aim is to provide comprehensive answers to the questions raised in Section 1.1 while extending the framework to deliver explanations in accessible, natural language (Chapters 3 and 6).

Secondly, we intend to enhance the original web application developed by Karamlou[6], which served as a proof-of-concept for Čyras et al.'s work. Our goal is to adapt this tool to be compatible with *Terranova*'s data, enabling it to provide insightful assessments of the company's schedules (Chapter 4). We aim to demonstrate that our extended theory and proof-of-concept tool can genuinely improve the user experience, offering clear explanations for suboptimal schedules and practical suggestions for enhancements (Chapter 5). Our objective is to bridge the gap between computational efficiency and user-friendly decision support.

By pursuing these objectives, we aim to contribute to the evolving field of scheduling optimisation. Our work seeks to empower companies like *Terranova* with tools to improve operational efficiency and provide users with transparent insights and sensible suggestions for their scheduling challenges.

Chapter 2

Background

Previous endeavors have explored the provision of natural language explanations for planning problems. For instance, Vasileiou et al.[7] presented a logic-based framework constructed on a series of axioms and algorithms that assess the validity of a given query. Sukkerd, Simmons, and Garlan[8] proposed an alternative framework based on a Markov decision process, extending it to account for cost functions and identify Pareto-optimal planning solutions. Notably, their user studies demonstrated increased trust in the system due to the explanations provided regarding trade-off rationale.

This project builds upon the work of previous research that explored explainable scheduling using argumentation theory[5]. We commence by providing an overview of the scheduling problem and its real-world applications. Subsequently, we delve into the background of argumentation theory and the mapping from problem to framework.

2.1 The Makespan Scheduling Problem

The makespan scheduling problem is an optimisation problem for n independent jobs $\mathcal{J} := \{1, 2, \ldots, n\}$ to be executed by m machines $\mathcal{M} := \{1, 2, \ldots, m\}$. It aims to assign jobs to machines in such a way that the total length of time to complete the jobs across all the machines is minimised, as described by Graham[9] and originally implemented using the example of nurse rostering by Warner and Prawda[1]. Each job may only be executed by one machine, and any machine is capable of completing any job.

Definition 2.1. *Consider a set of n independent jobs $\mathcal{J} := \{1, 2, \ldots, n\}$ with associated processing times $\boldsymbol{p} = \{p_1, p_2, ..., p_n\}$, to be completed by m machines $\mathcal{M} := \{1, 2, \ldots, m\}$.*

*The **optimisation problem for makespan scheduling** is as follows:*

$$\min_{C_{\max}, C_i, x_{i,j}} \quad C_{\max} \tag{2.1}$$

$$C_{\max} \geq C_i \qquad i \in \mathcal{M} \tag{2.2}$$

$$C_i = \sum_{j=1}^{n} x_{i,j} p_j \qquad i \in \mathcal{M} \tag{2.3}$$

$$\sum_{i=1}^{m} x_{i,j} = 1 \qquad j \in \mathcal{J} \tag{2.4}$$

$$x_{i,j} \in \{0, 1\} \qquad i \in \mathcal{M}, j \in \mathcal{J} \tag{2.5}$$

*where $C_{\max} = \max_{i=1}^{m}\{C_i\}$ is the **makespan** (the length of time needed to complete all jobs).*

*A **schedule** S of $(\mathcal{M}, \mathcal{J})$ defines each $x_{i,j}$ where $i \in \mathcal{M}, j \in \mathcal{J}$, satisfying equation (2.5). In this way, it can be seen as the $m \times n$ matrix $S \in \{0,1\}^{m \times n}$*

We take a schedule S in this case to be a potential solution (feasible or not, optimal or not).

Definition 2.2. *A schedule S is **feasible** iff it satisfies equations (2.2)-(2.4) in Definition 2.1.*

Example 2.1. *Consider 2 machines and 3 jobs, where $\boldsymbol{p} = \{2, 3, 2\}$ are the associated processing times. Consider the following schedule S_1:*

$$x_{1,1} = x_{1,3} = x_{2,2} = 1$$
$$x_{1,2} = x_{2,1} = x_{2,3} = 0$$

This schedule is valid since it satisfies (2.5) of Definition 2.1. An alternate schedule S_2 may define $x_{i,j}$ as:

$$x_{1,1} = x_{1,3} = x_{2,3} = 1$$
$$x_{1,2} = x_{1,2} = x_{2,2} = 0$$

with the matrix forms of both schedules being

$$S_1 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, S_2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

Note S_1 describes an optimal solution, where $C_1 = C_{\max} = 4$, while S_2 describes an infeasible solution where both machines complete job 3 and no machine completes job 2.

2.2 Abstract Argumentation (AA) Framework

The origin of the abstract argumentation framework (AA) begins with Dung's logical implementation of "attacking" arguments[10]. The argument for this framework comes from the idea that the winner of a debate comes from the player who "has the last word". In [10], Dung introduces some key definitions used in this section. We use a graph notation to illustrate our argumentation framework, as in previous related work e.g. García et al.[11], Fan and Toni[12].

Definition 2.3. *An **argumentation framework** $AF = (Args, \rightsquigarrow)$ is a directed graph where:*

- *Args is a set of arguments (vertices), and*
- *$\rightsquigarrow$ is a binary "attack" relation on Args where $a \rightsquigarrow b$ means that a attacks b. This is represented on a graph as an edge from a to b. The negation of this, e.g. $a \not\rightsquigarrow b$, indicates that a does not attack b.*

We can extend the definition of the attack relation. For a sets of arguments A:

- $A \rightsquigarrow b \iff \exists a \in A : a \rightsquigarrow b$.
- $b \rightsquigarrow A \iff \exists a \in A : b \rightsquigarrow a$.

It follows that for sets A, B, we have $A \rightsquigarrow B \iff \exists a \in A, b \in B : a \rightsquigarrow b$.

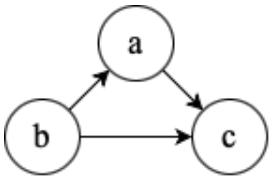


Figure 2.1: A basic argumentation framework example, presented as a directed graph.

Figure 2.1 represents an argumentation framework where *Args* $= \{a, b, c\}$ with attacks $a \rightsquigarrow c, b \rightsquigarrow a, b \rightsquigarrow c$.

Definition 2.4. *An **extension** E of Args, $E \subseteq Args$, is*

1. ***conflict-free** iff $E \not\rightsquigarrow E$,*
2. *a **stable** extension iff E conflict-free, $E \rightsquigarrow a, \forall a \in Args \backslash E$.*

Example 2.2. *Returning to our AF in Figure 2.1, suppose we have extension $E_1 = \{b\}$. Clearly, it is conflict-free since b does not attack itself. Since b attacks both a and c, E_1 is also a stable extension. Alternately, consider $E_2 = \{a, c\}$. Since $a \rightsquigarrow c$, it is not conflict-free. Also, since $E_2 \not\rightsquigarrow b$, E_2 is not stable.*

In this paper, Dung[10] goes on to introduce the terms *acceptable* ($A \in$ *Args*, for any $B \in$ *Args* : $B \rightsquigarrow A \implies S \rightsquigarrow B$, with respect to set of arguments S), *admissible* (conflict-free S, all arguments in S acceptable with respect to S), *preferred* (extension which is the maximal admissible set of AF) and *grounded*. Recent research has utilised and extended such terms: Ulbricht and Wallner[13] extend the concept of extensions by introducing σ-extensions, with their own set of properties and abilities to offer natural language descriptions beyond extensions. In particular, there has been extensive development in argumentation and its links to artificial intelligence in recent decades[14].

2.3 Mapping A Schedule Using Argumentation

Here we summarise the work of Čyras et al.[5] to create a mapping between a scheduling problem as defined in Section 2.1 and the AA framework defined in Section 2.2. The paper focuses on 3 dimensions as routes of explanation: feasibility, efficiency and the ability to satisfy predetermined fixed decisions within schedules. These 3 dimensions are all represented by different AFs.

The definitions in this section are taken directly from Čyras et al.[5].

2.3.1 Feasibility

Feasibility enforces that each job must be assigned to exactly one machine. No machines can be left unassigned or assigned multiple times.

Definition 2.5. *The **feasibility AF** $(Args_F, \rightsquigarrow_F)$ is given by:*

- $Args_F = \{a_{i,j} : i \in \mathcal{M}, j \in \mathcal{J}\}$,
- $a_{i,j} \rightsquigarrow_F a_{k,l} \iff i \neq k, j = l$.

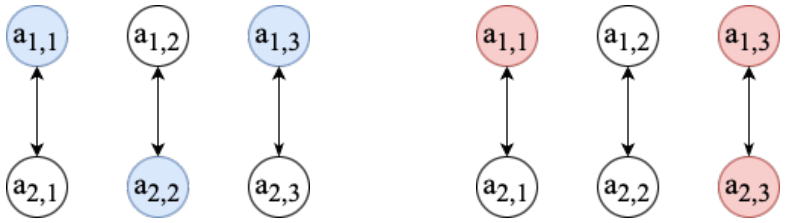


Figure 2.2: Corresponding extensions for Example 2.1. Left: S_1 (extension in blue); Right: S_2 (extension in red).

In summary, all possible combinations of machine and job are represented by an argument, and any combination of distinct arguments (and such, machines) with the same job must attack each other. The following mapping can now be defined:

Definition 2.6. *Let $(Args_F, \rightsquigarrow_F)$ be the feasibility AF. A schedule S and extension $E \subseteq Args_F$ are **corresponding**, $S \approx E$ iff*

$$x_{i,j} = 1 \iff a_{i,j} \in E.$$

Example 2.3. *Consider Example 2.1. We can derive corresponding extensions for S_1 and S_2, by the definition above. Indeed, we have $S_1 \approx E_1 := \{a_{1,1}, a_{1,3}, a_{2,2}\}$ and $S_2 \approx E_2 := \{a_{1,1}, a_{1,3}, a_{2,3}\}$. The resulting extensions are shown in Figure 2.2.*

The following important property of the above mapping comes from Čyras et al.[5].

Theorem 2.7 (Čyras et al.[5])**.** *Let $(Args_F, \rightsquigarrow_F)$ be the feasibility AF. For any $S \approx E$, S is feasible $\iff$ E is stable.*

We generalise the case of Example 2.3 to consider the construction and validation algorithms for all feasibility AFs.

Lemma 2.8. *The feasibility AF can be constructed in $O(nm^2)$ time. Verification of whether an extension $E \subseteq Args_F$ is stable requires up to $O(n^2m^2)$ time.*

Proof. This proof is taken from Lemma 4.2, Čyras et al.[5].

A feasibility AF consists of $O(mn)$ arguments. For each job n we have $m(m-1)$ attacks by Definition 2.5, so construction takes $O(nm^2)$ time. For an extension $E \subseteq Args_F$, we must check for all i, i', j, j' where $a_{i,j}, a_{i',j'} \in E$, if $a_{i,j} \rightsquigarrow_F a_{i',j'}$. To iterate through all $(i,j) \in \mathcal{O} \times \mathcal{A}$ takes $O(mn)$, so verification takes $O(m^2n^2)$ time. □

2.3.2 Efficiency

In the makespan problem, we want to minimise the maximum total processing time any one machine takes to complete all their assigned paths. For example, if machine 1 takes 60 minutes and machine 2 takes 80 minutes, than our makespan is 80 minutes.

To decide whether one schedule is optimal compared to another, we must define a finite set of properties by which we can improve a non-optimal schedule.

Definition 2.9. *A **critical** job $j \in \mathcal{J}$ satisfies $C_i = C_{\max}$ for $i \in \mathcal{M} : x_{i,j} = 1$.*

We define the following properties for $i \neq i'$:

1. ***Single Exchange Property** (SEP): $C_i - C_{i'} \leq p_j$;*
2. ***Pairwise Exchange Property** (PEP): $\forall j' \neq j, x_{i',j'} = 1 : p_j > p_{j'} \implies C_i + p_{j'} \leq C_{i'} + p_j$.*

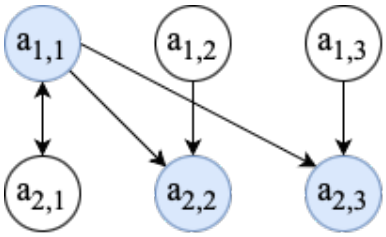


Figure 2.3: Example of optimality AF for schedule $S : x_{1,1} = x_{2,2} = x_{2,3} = 1$.

*We say S is **efficient** iff S is feasible and satisfies both SEP and PEP.*

Example 2.4. *Consider 2 machines and 3 jobs, where $\boldsymbol{p} = [3, 5, 6]$. Consider the optimal schedule S with $x_{1,1} = x_{1,2} = x_{2,3} = 1$ and all other values 0. Since $C_{max} = 11 = C_1$, the critical jobs in our problem are jobs 1 and 2. We have for SEP:*

$$C_1 - C_2 = 2 \leq 3 < 5$$

and for PEP:

$$6 > 5, 6 > 3$$

so both SEP and PEP are satisfied.

It can be shown that every efficient schedule satisfies SEP and PEP[5]. Note that efficient does not imply optimal. For example, consider schedules which might be improved by a three-way exchange, as opposed to a single or pairwise exchange.

Definition 2.10. *For feasibility AF $(Args_F, \rightsquigarrow_F)$ and schedule S, the **optimality AF** $(Args_S, \rightsquigarrow_S)$ is defined as:*

- $Args_S = Args_F$,
- $\rightsquigarrow_S = (\rightsquigarrow_F \setminus \{(a_{i,j}, a_{i',j'}) : C_i = C_{\max}, x_{i,j} = 1, C_i > C_{i'} + p_j\}) \cup$
  $\{(a_{i',j'}, a_{i,j}) : C_i = C_{\max}, x_{i,j} = 1, x_{i',j'} = 1, i \neq i', j \neq j', p_j > p_{j'},$
  $C_i + p_{j'} > C_{i'} + p_j\}$

Example 2.5. *Consider the problem as in Example 2.4, with schedule S such that $x_{1,1} = x_{2,2} = x_{2,3} = 1$. Then $C_{max} = C_2$. The optimality AF for this schedule is show in Figure 2.3.*

2.3.3 Fixed Decisions

Up until now, all assignments of jobs to any machine have been "legal", even if they might not be efficient schedules, so long as they fix the feasibility criteria (every job allocated to exactly one machine). Consider the example of 2 machines and 3 jobs. Suppose job 1 cannot be allocated to machine 1 (maybe it doesn't have the right parts to complete the job). Alternatively, perhaps job 3 must be assigned to machine 2. These fixed decisions can also be modelled as argumentation frameworks, allowing us to add additional constraints into our problem.

Definition 2.11. *Let the set of **negative fixed decisions** $D^- = \mathcal{M}^- \times \mathcal{J} \subseteq \mathcal{M} \times \mathcal{J}$ satisfy*

$$(i, j) \in D^- \implies x_{i,j} = 0,$$

*and similarly, the set of **positive fixed decisions** $D^+ = \mathcal{M}^+ \times \mathcal{J} \subseteq \mathcal{M} \times \mathcal{J}$ satisfy*

$$(i, j) \in D^+ \implies x_{i,j} = 1.$$

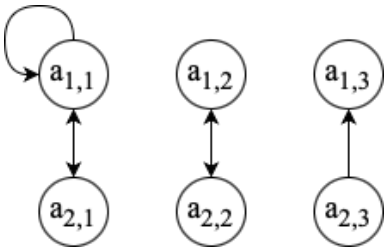


Figure 2.4: Example of fixed decision AF, with $(1,1) \in D^-, (2,3) \in D^+$.

Example 2.6. *Consider a problem with 2 machines and 3 jobs. For our examples above (job 1 cannot be allocated to machine 1; job 3 must be allocated to machine 2), we have* $(1,1) \in D^-, (2,3) \in D^+$.

With these sets formally defined, we now introduce the fixed decision AF:

Definition 2.12. *For feasibility AF* $(Args_F, \rightsquigarrow_F)$ *and fixed decisions* $D = (D^-, D^+)$*, the* ***fixed decision AF*** $(Args_D, \rightsquigarrow_D)$ *is defined as:*

- $Args_D = Args_F$,
- $\rightsquigarrow_D = (\rightsquigarrow_F \cup \{(a_{i,j}, a_{i,j}) : (i,j) \in D^-\}) \setminus \{(a_{k,l}, a_{i,j}) : (i,j) \in D^+, (k,l) \in \mathcal{M} \times \mathcal{J}\}$.

Example 2.7. *Figure 2.4 shows a fixed decision AF corresponding to the problem in Example 2.6. Note that* $a_{1,1}$ *attacks itself, while attacks to* $a_{2,3}$ *have been removed from its initial feasibility AF.*

Lemma 2.13 (Čyras et al.[5])**.** *Given a schedule* S*, the fixed decision AF* $(Args_D, \rightsquigarrow_D)$ *can be constructed in* $O(n^2m^2)$ *time. Verification of whether an extension* $E \subseteq Args_D$ *is stable requires up to* $O(n^2m^2)$ *time.*

2.3.4 Explanations

Using the 3 AFs defined so far in Section 2.3, Čyras et al.[5] go on to define a set of explanations as to why a given schedule might be infeasible, inefficient or violate fixed decisions, giving the specific edge responsible for the violation.

Definition 2.14. *For a given schedule* S *and set of fixed decisions* D*,* $E \approx S$*,* $(Args, \rightsquigarrow) \in \{(Args_F, \rightsquigarrow_F), (Args_S, \rightsquigarrow_S), (Args_D, \rightsquigarrow_D)\}$*, for an attack* $a \rightsquigarrow b, a, b \in E$*:*

- $(a,b) \in \rightsquigarrow_F \implies S$ *not feasible,*
- $(a,b) \in \rightsquigarrow_S \setminus \rightsquigarrow_F \implies S$ *not efficient,*
- $(a,b) \in \rightsquigarrow_D \setminus \rightsquigarrow_F \implies S$ *violates fixed decisions.*

For a non-attack $E \not\rightsquigarrow b, b \notin E$*:*

- $\rightsquigarrow = \rightsquigarrow_F \implies S$ *not feasible,*
- $\rightsquigarrow = \rightsquigarrow_S$ *and* $b \rightsquigarrow_S E \implies S$ *not efficient,*
- $\rightsquigarrow = \rightsquigarrow_D, b$ *unattacked* $\implies S$ *violates fixed decisions.*

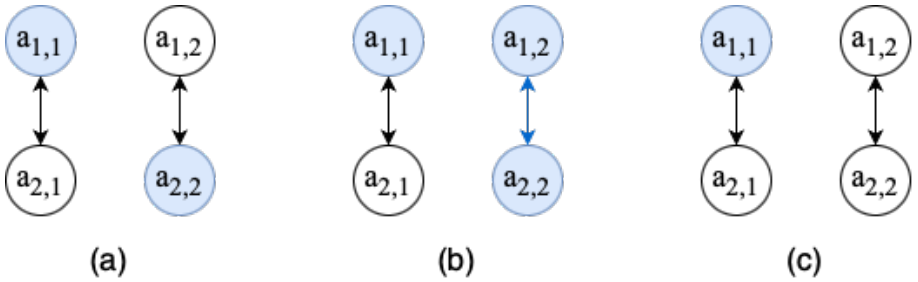


Figure 2.5: Feasibility AF examples

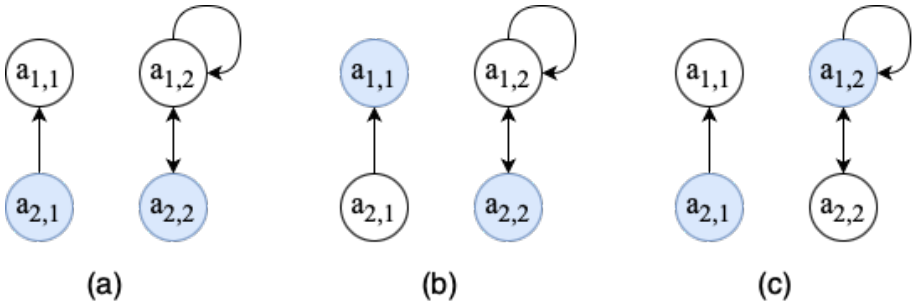


Figure 2.6: Fixed decision AF examples

Example 2.8. *Figures 2.5, 2.6, 2.7 gives the following explanations (the arguments corresponding to the extension $E \approx S$ are highlighted in blue):*

1. *Feasibility AF (Figure 2.5):*
   (a) *A feasible schedule.*
   (b) *Not feasible since $a_{1,2} \rightsquigarrow a_{2,2}$ where $a_{1,2}, a_{2,2} \in E$.*
   (c) *Not feasible since $E \not\rightsquigarrow a_{1,2}, E \not\rightsquigarrow a_{2,2}$.*
2. *Fixed Decision AF (Figure 2.6):*
   (a) *Schedule satisfies fixed decisions.*
   (b) *Positive fixed decision violated since $E \not\rightsquigarrow a_{2,1}$.*
   (c) *Negative fixed decision violated since $a_{1,2} \rightsquigarrow a_{1,2}$.*
3. *Optimality AF (Figure 2.7):*
   (a) *Schedule is efficient.*
   (b) *Inefficient schedule since $a_{1,2} \rightsquigarrow a_{2,1}$ (schedule can be improved by swapping jobs 1 and 2).*
   (c) *Inefficient schedule since $E \not\rightsquigarrow a_{2,1}$ (schedule can be improved by moving job 1 to machine 2).*

Note that while the feasibility and fixed decision AFs can be constructed independent of a schedule, the optimality AF requires a given schedule S so it can test the relevant inequalities for i such that $C_i = C_{max}$.

In recent years, there have been various attempts to design user interfaces that are able to offer natural language explanations for the scheduling decisions taken, but these generally build on frameworks separate to that given above[15][16].

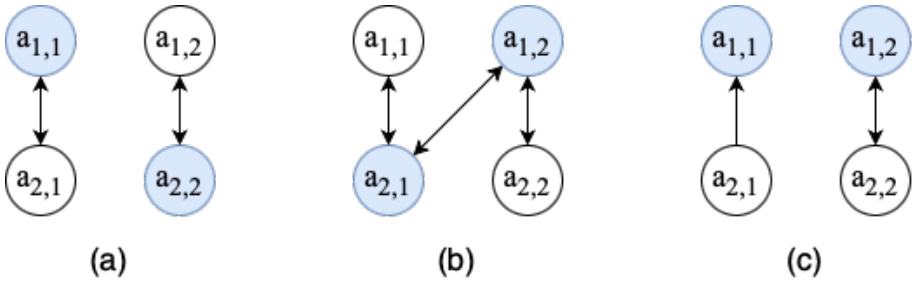


Figure 2.7: Optimality AF examples

2.4 Priority Scheduling Algorithms

Fixed-priority preemptive scheduling in real time-systems ensures that a processor executes the highest priority task of those that are ready to be completed. As stated by Coffman and Denning[17]: "The basic objective of [makespan scheduling] algorithms .... is to schedule task systems so that they execute in minimum time". Importantly, the job-shop scheduling algorithms do not accommodate the timing requirement of these more complex problems[18][19]. For example, Liu and Layland[20] provide a scheduling algorithm, dependent on 4 assumptions; namely, it is assumed that all processes have some common "deadline" such that, after this, they are released.

Most priority-oriented scheduling attempts focus on the scheduling of jobs in an operating system, considering priority in terms of a time-sensitive deadline rather than a numerical value. In the context of management systems, Jin and Yu[21] provide a scheduling algorithm where each task has an associated priority value. Their algorithm can be seen in Figure 2.8. At the time of this project, this appears to be the extent of research into numerical priority values as a factor in optimal scheduling.

2.5 Travelling Salesman Problem

We include here some preliminary background on the travelling salesman problem (TSP), which relates to our theory in later chapters regarding shortest routes between jobs. The aim of the TSP is to find the shortest route for a travelling salesman who starts at a home city, must visit a given set of other cities, then return home to the starting city.[22]

Definition 2.15. *Consider a set of n distinct cities $\mathcal{C} := \{C_1, C_2, \ldots, C_n\}$, each characterised by its geographical coordinates in a Euclidean space, denoted as $\boldsymbol{p}_i = (x_i, y_i)$, where $i \in \{1, 2, \ldots, n\}$. Then the **Traveling Salesman Problem (TSP)** can be formulated as the optimisation problem:*

$$\min_{\pi} \sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(1)} \tag{2.6}$$

$$\text{subject to } \pi \text{ is a permutation of } \{1, 2, \ldots, n\}, \tag{2.7}$$

where π represents a permutation of the set $\{1, 2, \ldots, n\}$, defining the order in which the cities are visited in the tour.

It can be shown that the TSP is NP-hard. There are various ways the TSP can be written in the form of an integer programming problem, as shown by Finke, Funn and Claus[23]

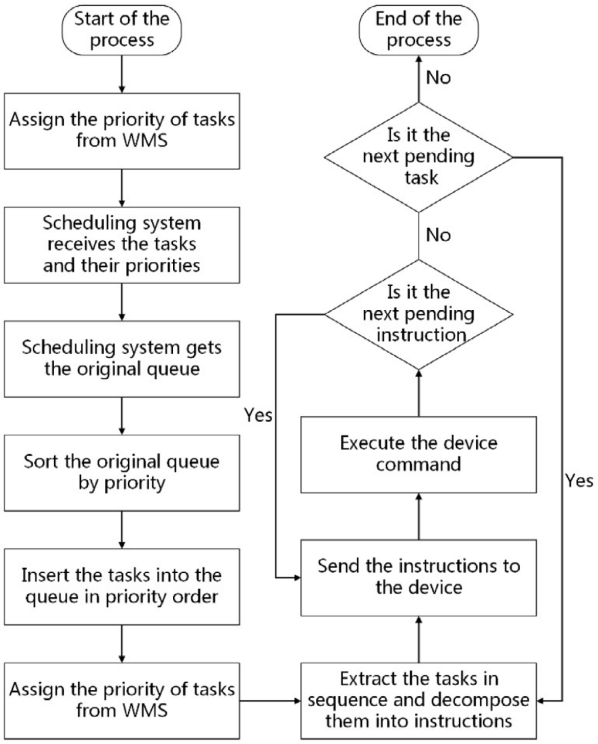


Figure 2.8: High priority scheduling algorithm proposed by Jin and Yu[21]

and Padberg and Sung[24]. Lenstra and Rinnooy Kan[25] illustrate a number of scenarios, including scheduling, which can be formatted as a travelling salesman problem. Some algorithms, such at the branch-and-bound method[26], can be shown to give optimal solutions, but may require infeasible running times. Alternatively, there are various computationally-feasible, sub-optimal algorithmic solutions for the TSP. Lin[27] utilises the concept of λ-optimality:

Definition 2.16. *Consider a graph where every city is represented by a vertex. We define a* ***link*** *as the edge connecting two cities.*

A route is λ***-optimal*** *(or* λ*-opt) if, for any set of* λ *links, it is impossible to find a route with smaller cost by replacing any links in the set with any other set of* λ *links.*

Clearly all routes are 1-optimal, and it can be shown a route of n cities is optimal iff it is n-optimal. Croes[28] initially proposed the 2-opt method as a sub-optimal solution to the TSP, which takes a route that crosses over itself and reorder so it does not.

Theorem 2.17. *The following are equivalent:*

1. *A route is 2-optimal.*
2. *A route is optimal relative to inversion (i.e. reversing the order of a set of neighbouring cities in the route).*
3. *The route does not intersect itself.*

Clearly, to check all pairs of links takes time $O(n^2)$. Lin[27] goes one step further in suggesting 3-opt routes are the optimal value of λ to use for this sub-optimal approach, with a significantly higher probability of finding an optimal solution than with 2-opt routes. The time complexity of this is $O(n^3)$.

Definition 2.18. *A route is* ***optimal relative to inversion and insertion*** *if no k consecutive cities on the original route can be removed and reinserted (as is or inverted) between any two consecutive remaining cities to produce a tour of lesser cost, for any value of k.*

Theorem 2.19 (Taken from Lin[27])**.** *A route is optimal relative to inversion and insertion iff it is 3-optimal.*

2.6 Dataset

The dataset provided by *Terranova* forms the basis of our problem. The full description of dataset properties can be seen in Appendix A. The route planning system takes 3 lists as inputs: activities, operators and instruments, and outputs a list of activities with associated operators, required instruments and a list of routes connecting the activities.

Activities for workforce management might involve maintenance at a specific practice or an emergency callout. These activities might require specific skills or instruments to complete the task; for example, a van might be needed (the instrument), along with the corresponding skill of a driver's license. These activities must be completed by operators who are human workers completing a shift (usually 8:00-13:00 then 14:00-18:00).

Compare the *Terranova* case with the makespan scheduling problem as outlined in Section 2.1: we can consider activities here in a similar way to jobs in the makespan problem and operators in a similar way to machines – this provides us with a mapping from the *Terranova* dataset to a subset problem, explainable with argumentation. We can go one step further to consider the similarities between fixed decisions and the constraints (skills, instruments) outlined in the dataset; notably, these constraints act as a subset of negative fixed decisions and can be mapped as such in our problem. This is discussed further and more extensively in the next chapter.

Activities

Each activity has the following properties:

1. `activity_id` is the unique identifier for a given activity.
2. `Y, X, Z` are the map co-ordinates where the activity takes place.
3. `duration` contains
   (a) `default`, the default time in seconds needed for an operator to complete the activity.
   (b) `specific` contains a list of corresponding `operator_ids` and their `value` i.e. the time for this specific operator to complete the activity (in seconds).
4. `start_timestamp` and `end_timestamp` specific the date-times between which the activity must be completed.
5. `priority` from 1-5, with 1 assigned to tasks of the highest priority.
6. `must_start_after` gives the `activity_id` of the activity that must be completed before this can start (if any).
7. `needed_capacity` of vehicle (in litres), if required. Vehicles are listed as instruments with `instrument_type = "VE*"`.

8. `needed_skills` of operator completing activity (if any).
9. `needed_instruments` by `instrument_type` to complete activity (if any).

Operators

Each operator has the following properties:

1. `operator_id` is the unique identifier for a given operator.
2. `start_position` contains the initial `Y, X, Z` co-ordinates where the operator begins its shift.
3. `returning_position` contains the `Y, X, Z` co-ordinates where the operator must end its shift.
4. List of `skills` the operator has (if any).
5. `working_shift` gives a list of `start_timestamps` and `end_timestamps` (date-times) where the operator is available to work.

Instruments

This additional set of inputs for this model contains the following properties:

1. `instrument_id` is the unique identifier for a given instrument.
2. `instrument_type` is a 3-character string specifying if any additional properties apply to this instrument. Most importantly, `"VE*" = Vehicle`.
3. `capacity` in litres if `instrument_type = "VEH"`, and `null` otherwise.
4. List of `needed_skills` operator must have to use the given instrument.

Chapter 3

Theory

The following chapter outlines the key theory which extends the framework from Section 2.3, in line with the additional information provided in Section 2.6.

We begin to adapt the model by finding a mapping which relates a simplified version of our problem as closely as possible to the theory outlined in Section 2.3. To do this, we must reformat our problem as a mixed integer linear programming formulation: let $\mathcal{A}$ be the set of activities (identified by their `activity_id`) and $\mathcal{O}$ the set of operators (identified by their `operator_id`), with $x_{i,j} = \{0, 1\}$ the binary decision variable denoting if operator $i \in \mathcal{O}$ completes activity $j \in \mathcal{A}$.

Notably, in line with the terms used in the dataset, we refer to all "machines" as "operators", and use the terms "job" and "activity" interchangeably.

Definition 3.1. *Let $\mathcal{A}$ be the set of all activities/jobs and $\mathcal{O}$ the set of all operators in our problem. As such, we define the feasibility AF (in the context of our dataset) as:*

- $Args_F = \{a_{i,j} : i \in \mathcal{O}, j \in \mathcal{A}\}$,
- $a_{i,j} \rightsquigarrow_F a_{k,l} \iff i \neq k, j = l$.

Besides the renaming of sets, the mappings are identical from problem to feasibility AF as in Definition 2.5 and, as such, Lemma 2.8 holds for this new definition.

3.1 Efficiency with Extended Cost

Recall the makespan scheduling problem as in Definition 2.1:

$$
\begin{aligned}
\min_{C_{\max}, C_i, x_{i,j}} \quad & C_{\max} & \\
& C_{\max} \geq C_i & i \in \mathcal{M} \\
& C_i = \sum_{j=1}^{n} x_{i,j} p_j & i \in \mathcal{M} \\
& \sum_{i=1}^{m} x_{i,j} = 1 & j \in \mathcal{J} \\
& x_{i,j} \in \{0, 1\} & i \in \mathcal{M}, j \in \mathcal{J}
\end{aligned}
$$

where $C_{\max} = \max_{i=1}^{m} \{C_i\}$ is the makespan. Čyras et al.[5] only considered time within the "cost" of the schedule for any one operator. However, the dataset in Section 2.6 lists many

other variables that may contribute to determining optimality, other than a simple processing time. In particular, we look at extending the "cost" calculation to include more complex processing times (variable between operators) and explore multiple variables contributing to this cost (priorities).

3.1.1 Variable Processing Times

Not all operators may take the same time to complete a job. For example, suppose operator 2 is more skilled than operator 1, so can complete a given job in 40 minutes, compared to operator 1 taking 60 minutes.

Definition 3.2. *Let $m := |\mathcal{O}|, n := |\mathcal{A}|$. Then we define $P \in M_{m,n}(\mathbb{R})$ to be the $m \times n$ matrix representing the **processing times** for a set of jobs $j \in \mathcal{A}$ assigned to a set of operators $i \in \mathcal{O}$. p_{ij} is the time taken for operator i to complete a job j.*

Example 3.1. *For the example given above, $P = (60, 40)^T$ where the units are in minutes.*

3.1.2 Priority

Not all jobs are of equal importance when it comes to scheduling. For the example of nurse rostering, suppose you have 2 jobs: a time-critical operation and a routine check up on another patient. It is more important to get the operation completed (and at that, as soon as possible) than the check up.

Definition 3.3. *Let $n := |\mathcal{A}|$. Then we define $\boldsymbol{q}, |\boldsymbol{q}| = n$, the **priority vector** for a set of jobs, such that q_j denotes the priority of job $j \in \mathcal{A}$.*

Definition 3.4. *The **individual cost** $c_{i,j}^{P,q}$ (or shorthand $c_{i,j}$) for a job j and operator i, with processing times P and priority $\boldsymbol{q}$,is defined as*

$$c_{i,j}^{P,q} = \alpha q_j + \beta p_{i,j}$$

where $\alpha + \beta = 1$.

Here, α and β represent the weighting towards priority and processing times respectively. Adjusting the ratio of α to β then allows us to vary the influence certain variables carry when calculating the final cost. Note that setting either $\alpha = 0$ or $\beta = 0$ gives a problem similar to that illustrated in Chapter 2.

Definition 3.5. *Consider a set of n independent jobs $\mathcal{A} := \{1, 2, \ldots, n\}$ with associated processing time matrix $P \in M_{m,n}(\mathbb{R})$ (where $p_{i,j}$ denotes the element in the ith row and jth column) and priority vector $\boldsymbol{q} = \{q_1, q_2, ..., q_n\}$, to be completed by m operators $\mathcal{O} := \{1, 2, \ldots, m\}$.*

*The **optimisation problem for individual cost scheduling** is as follows:*

$$\min_{C_{\max}, C_i, x_{i,j}} C_{\max} \tag{3.1}$$

$$C_{\max} \geq C_i \qquad i \in \mathcal{O} \tag{3.2}$$

$$C_i = \sum_{j=1}^{n} x_{i,j} c_{i,j}^{P,q} \qquad i \in \mathcal{O} \tag{3.3}$$

$$\sum_{i=1}^{m} x_{i,j} = 1 \qquad j \in \mathcal{A} \tag{3.4}$$

$$x_{i,j} \in \{0, 1\} \qquad i \in \mathcal{O}, j \in \mathcal{A} \tag{3.5}$$

where $C_{\max} = \max_{i=1}^{m} \{C_i\}$.

For clarity, we simply use $c_{i,j}$ to denote individual cost in the remainder of this report.

3.1.3 Optimising Individual Cost Scheduling

Now we have redefined our problem, we revise our definitions of SEP and PEP for the updated "individual" cost variable.

Definition 3.6. *For a critical job $j \in \mathcal{A}$ satisfying $C_i = C_{\max}$ for $i \in \mathcal{O} : x_{i,j} = 1$, we define the following properties for $i \neq i'$, processing times P and priority $\boldsymbol{q}$:*

1. ***Single Exchange Property for Variable Processing Time and Priority** (SEPVPTQ): $C_i - C_{i'} \leq c_{i',j}$;*
2. ***Pairwise Exchange Property for Variable Processing Time and Priority** (PEPVPTQ): $\forall j' \neq j, x_{i',j'} = 1 : c_{i,j} > c_{i,j'} \implies C_i + c_{i',j'} \leq C_{i'} + c_{i',j}$.*

*We say S is **efficient in individual cost** iff S is feasible and satisfies both SEPVPTQ and PEPVPTQ.*

It can be shown that every optimal schedule with respect to individual cost satisfies SEPVPTQ and PEPVPTQ. To avoid repetition, we omit the proof here, but the method follows that in Lemma 3.15, from which this result follows directly.

Example 3.2. *Consider the following problem with 2 operators with*

$$P = \begin{pmatrix} 5 & 3 \\ 2 & 6 \\ 3 & 3 \end{pmatrix}^T, \boldsymbol{q} = (2, 4, 1)$$

and schedule

$$S = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Let $\alpha = \beta = 0.5$. Calculating the individual costs:

$$c_{i,j} = \frac{1}{2} \times \begin{pmatrix} 7 & 6 & 4 \\ 5 & 10 & 4 \end{pmatrix}_{i,j} = \begin{pmatrix} 3.5 & 3 & 2 \\ 2.5 & 5 & 2 \end{pmatrix}_{i,j}.$$

So $C_1 = 5.5$ and $C_2 = 5$.

Checking SEPVPTQ:

$$5.5 - 5 \leq 2, 2.5$$

so SEPVPTQ is satisfied for S. For PEPVPTQ:

$$3.5 > 3, 5.5 + 5 > 5 + 2.5$$

so PEPVPTQ is violated for S.

3.2 Schedule Order and Distance

One notable inclusion in the dataset in Section 2.6 is the addition of distance parameters. Indeed, some jobs are based at different locations, with operators both having a fixed starting and ending position for their shifts.

Definition 3.7. *We define the **schedule order for an operator** i **and schedule** S as*

$$\sigma_i = \{j_1, j_2, ..., j_{N_i}\} \subseteq \mathcal{J},$$

where N_i is the number of jobs allocated to operator i in schedule S and $j_k = j_{k'} \iff k = k'$.

The schedule order allows us to track the order jobs are completed in S, since S is essentially a boolean array that can only track assignment rather than order.

Definition 3.8. *Consider a schedule S and operator i with schedule order σ_i, and $|\sigma_i| = N_i$. Then we define the notation $\sigma_i[k] = j_k$ for $1 \leq k \leq N_i$ where j_k is the kth entry in σ_i.*

Example 3.3. *Consider a schedule S where jobs $3, 1, 2$ are allocated to operator 1 in this order. So we have $x_{1,3} = x_{1,1} = x_{1,2} = 1$ and $\sigma_1 = \{3, 1, 2\}$. Furthermore, we have $\sigma_1[1] = 3, \sigma_1[2] = 1$ and $\sigma_i[3] = 2$.*

3.2.1 Distance

We define a metric to allow us to calculate distances between jobs, given the coordinates information in Section 2.6.

Definition 3.9. *For a job $j \in \mathcal{A}$, we define its associated 3D coordinates as $(j_x, j_y, j_z) \in \mathbb{R}^3$. We define the **distance metric** δ:*

$$\delta : \mathcal{A} \times \mathcal{A} \to \mathbb{R} \tag{3.6}$$

$$j_1, j_2 \mapsto \sqrt{(j_{1_x} - j_{2_x})^2 + (j_{1_y} - j_{2_y})^2 + (j_{1_z} - j_{2_z})^2} \tag{3.7}$$

as the distance between 2 jobs, $j_1, j_2 \in \mathcal{A}$.

We must also bear in mind that all operators have a specified starting and ending position, at which they must begin/end their shifts.

Definition 3.10. *Let $i \in \mathcal{O}$ be an operator with associated schedule order σ_i for a schedule S. We define:*

1. ***Start position*** $s_i = (s_{i_x}, s_{i_y}, s_{i_z}) \in \mathbb{R}^3$ *where the operator must begin their shift.*
2. ***Returning position*** $f_i = (f_{i_x}, f_{i_y}, f_{i_z}) \in \mathbb{R}^3$ *where the operator must end their shift.*

We also define two special cases of the distance metric $\delta : \mathbb{R}^3 \times \mathcal{A} \to \mathbb{R}$ for an operator $i \in \mathcal{O}$:

1. $\delta(s_i, j) := \sqrt{(s_{i_x} - j_x)^2 + (s_{i_y} - j_y)^2 + (s_{i_z} - j_z)^2}$;
2. $\delta(f_i, j) := \sqrt{(f_{i_x} - j_x)^2 + (f_{i_y} - j_y)^2 + (f_{i_z} - j_z)^2}$.

The special cases of δ allow us to calculate the distance between the starting and ending positions of a given operator and the first/last job on their individual schedule σ_i.

Example 3.4. *Consider an operator i with $s_i = f_i = (0, 0, 0)$. Let the location of j be $(3, 4, 0)$. Suppose we have a schedule S such that $x_{i,j} = 1$ and $x_{i,j'} = 0 \forall j' \neq j$. Then $\delta(s_i, j) = \delta(j, f_i) = \sqrt{3^2 + 4^2 + 0} = 5$.*

Definition 3.11. *The **extended cost** C_i **with schedule ordering** σ_i for an operator i with starting position $s_i = (s_{i_x}, s_{i_y}, s_{i_z})$ and returning position $f_i = (f_{i_x}, f_{i_y}, f_{i_z})$ is defined as*

$$C_i = \sum_{j=1}^{N} x_{i,j} c_{i,j} + \gamma \left(\delta(s_i, \sigma_i[1]) + [\sum_{k=1}^{k_i - 1} \delta(\sigma_i[k], \sigma_i[k+1])] + \delta(\sigma_i[k_i], f_i) \right)$$

where $c_{i,j}$ is the individual cost for a job j and δ is the distance metric.

Example 3.5. *Continuing from Example 3.4, let $p_{i,j} = 5$ and $q_j = 2$. Let $\alpha = \beta = 0.5$ and $\gamma = 1$. Then*

$$C_i = 0.5(5 + 2) + 1(5 + 5) = 13.5.$$

Note that, unlike previous definitions of cost, extended cost relies on a given schedule ordering to calculate the distance component for each operator.

Definition 3.12. *Consider a set of n independent jobs $\mathcal{A} := \{1, 2, \ldots, n\}$ with associated processing time matrix $P \in M_{m,n}(\mathbb{R})$ (where $p_{i,j}$ denotes the element in the ith row and jth column) and priority vector $\boldsymbol{q} = \{q_1, q_2, ..., q_n\}$, to be completed by m operators $\mathcal{O} := \{1, 2, \ldots, m\}$.*

*The **optimisation problem for extended cost scheduling** is as follows:*

$$\min_{C_{\max}, C_i, x_{i,j}} \quad C_{\max} \tag{3.8}$$

$$C_{\max} \geq C_i \tag{3.9}$$

$$C_i = \sum_{j=1}^{N} x_{i,j} c_{i,j} + \gamma \left(\delta(s_i, \sigma_i[1]) + [\sum_{k=1}^{k_i - 1} \delta(\sigma_i[k], \sigma_i[k+1])] + \delta(\sigma_i[k_i], f_i) \right) \tag{3.10}$$

$$\sum_{i=1}^{m} x_{i,j} = 1 \tag{3.11}$$

$$c_{i,j} = \alpha q_j + \beta p_{i,j} \tag{3.12}$$

$$x_{i,j} \in \{0, 1\} \tag{3.13}$$

$$i \in \mathcal{O}, j \in \mathcal{A} \tag{3.14}$$

where $C_{\max} = \max_{i=1}^{m} \{C_i\}$ and $\delta, s_i, f_i, \sigma_i$ as defined in 3.11 and α, β, γ arbitrary values such that $\alpha + \beta = 1$.

3.2.2 Optimising Extended Cost Scheduling

We define another set of notation related to σ_i, allowing us to easily access previous jobs and next jobs in the schedule. This will aid us in upcoming proofs.

Definition 3.13. *Let σ_a be a schedule order for an operator a. Suppose we have $\sigma_a[b] = c$ for some job c and some index b. Then we define $c^- := \sigma_a[b-1], c^+ := \sigma_a[b+1]$.*

In other words j^- and j^+ are the jobs before and after j in an individual schedule σ_i respectively. As in Section 3.1.3, we extend SEP to accommodate these new variables as defined in the problem.

Definition 3.14. *Let δ be the distance metric and σ_i the schedule order for a given schedule S and operator i. For a critical job $j \in \mathcal{A}$ satisfying $C_i = C_{\max}$ for $i \in \mathcal{O} : x_{i,j} = 1$, we define the following properties for $i \neq i'$ for all $\sigma_{i'}[z] = w$:*

1. ***Extended Single Exchange Property*** *(SEP+):* $\forall w \neq j, x_{i',w} = 1 : C_i - C_{i'} \leq c_{i',j} + \gamma[\delta(w,j) + \delta(j,w^+)] - \gamma\delta(w,w^+)$
2. ***Extended Pairwise Exchange Property*** *(PEP+):* $\forall j' \neq j, x_{i',j'} = 1 :$

$$\gamma([\delta(j^-,j) + \delta(j,j^+)] - [\delta(j^-,j') + \delta(j',j^+)]) > [c_{i,j'} - c_{i,j}]$$
$$\implies C_i - C_{i'} \leq [c_{i',j} - c_{i',j'}] + \gamma([\delta(j'^-,j) + \delta(j,j'^+)] - [\delta(j'^-,j') + \delta(j',j'^+)]).$$

We say S is ***extended cost efficient*** *iff S is feasible and satisfies both SEP+ and PEP+.*

Lemma 3.15. *Every optimal schedule with respect to extended cost satisfies SEP+ and PEP+.*

Proof. SEP+: Consider the schedule S obtained by moving job j from machine i to i', placed directly after a given job $w \in \sigma_{i'}$, keeping the remaining job assignments and orders as in $S*$.

Then $C_i(S) = C_i(S^*) - c_{i,j} - \gamma[\delta(j^-,j) + \delta(j,j^+)] + \gamma\delta(j^-,j^+)$ and $C_{i'}(S) = C_{i'}(S^*) + c_{i',j} + \gamma[\delta(w,j) + \delta(j,w^+)] - \gamma\delta(w,w^+)$. Note $C_{i''}(S) = C_{i''}(S^*)$ for $i'' \neq i, i'$.

Since job j is critical, $C_i(S^*) \geq C_{i'}(S^*)$, and since S^* is optimal, $C_{i'}(S) \geq C_i(S^*)$ since S cannot obtain a lower extended cost. Putting this together:

$$\begin{aligned} & C_{i'}(S) \geq C_i(S^*) \\ \implies & C_{i'}(S^*) + c_{i',j} + \gamma[\delta(w,j) + \delta(j,w^+)] - \gamma\delta(w,w^+) \geq C_i(S^*) \\ \implies & C_i(S^*) - C_{i'}(S^*) \leq c_{i',j} + \gamma[\delta(w,j) + \delta(j,w^+)] - \gamma\delta(w,w^+) \end{aligned}$$

PEP+: Consider the schedule S obtained by moving job j from machine i to i', and job j' from i' to i, maintaining the same positions in schedule orders σ_i and $\sigma_{i'}$, with all order job assignments and orders remaining the same. We have

$$C_i(S) = C_i(S^*) + [c_{i,j'} - c_{i,j}] + \gamma([\delta(j^-,j') + \delta(j',j^+)] - [\delta(j^-,j) + \delta(j,j^+)])$$

and

$$C_{i'}(S) = C_{i'}(S^*) + [c_{i',j} - c_{i',j'}] + \gamma([\delta(j'^-,j) + \delta(j,j'^+)] - [\delta(j'^-,j') + \delta(j',j'^+)]).$$

As before, since job j is critical, $C_i(S^*) \geq C_{i'}(S^*)$, and since S^* is optimal,

$$\max\{C_i(S), C_{i'}(S)\} \geq C_i(S^*)$$

since S cannot obtain a lower extended cost.

If $C_i(S) \geq C_i(S^*)$, then

$$[c_{i,j'} - c_{i,j}] \geq \gamma([\delta(j^-,j) + \delta(j,j^+)] - [\delta(j^-,j') + \delta(j',j^+)]).$$

Alternately, if $C_{i'}(S) \geq C_i(S^*)$,

$$\begin{aligned} & C_{i'}(S) \geq C_i(S^*) \\ \implies & C_{i'}(S^*) + [c_{i',j} - c_{i',j'}] + \gamma([\delta(j'^-,j) + \delta(j,j'^+)] - \\ & [\delta(j'^-,j') + \delta(j',j'^+)]) \geq C_i(S^*) \\ \implies & C_i(S^*) - C_{i'}(S^*) \leq [c_{i',j} - c_{i',j'}] + \gamma([\delta(j'^-,j) + \delta(j,j'^+)] - \\ & [\delta(j'^-,j') + \delta(j',j'^+)]). \end{aligned}$$

□

Corollary 3.16. *Every optimal schedule with respect to individual cost satisfies SEPVPTQ and PEPVPTQ.*

Proof. As above, setting $\gamma = 0$. □

We can now utilise the definitions in this section to define an updated "optimality" AF. For the purposes of this report, we will refer to this as the efficiency AF, since the AF doesn't necessarily imply a fully optimal schedule.

Definition 3.17. *For feasibility AF $(Args_F, \rightsquigarrow_F)$ and schedule S, the **extended cost efficiency AF** $(Args_{S+}, \rightsquigarrow_{S+})$ is defined as:*

- $Args_{S+} = Args_F$,
- $$\rightsquigarrow_{S+} = \Big(\rightsquigarrow_F \setminus \big\{ (a_{i,j}, a_{i',j}) : C_i = C_{\max}, x_{i,j} = 1 : \exists w \neq j, x_{i',w} = 1 : C_i - C_{i'} > c_{i',j} + \gamma[\delta(w,j) + \delta(j,w^+)] - \gamma\delta(w,w^+) \big\} \Big) \bigcup \Big\{ (a_{i',j'}, a_{i,j}) : C_i = C_{\max}, x_{i,j} = 1, x_{i',j'} = 1, i \neq i', j \neq j', \gamma\big([\delta(j^-,j) + \delta(j,j^+)] - [\delta(j^-,j') + \delta(j',j^+)]\big) > c_{i,j'} - c_{i,j}, C_i - C_{i'} > c_{i',j} - c_{i',j'} + \gamma\big([\delta(j'^-,j) + \delta(j,j'^+)] - [\delta(j'^-,j') + \delta(j',j'^+)]\big) \Big\}$$

In other words:

1. *If an argument $a_{i,j} \in E$ violates SEP+ due to $a_{i',j}$, remove any attacks from $a_{i,j}$ to $a_{i',j}$;*
2. *If arguments $a_{i,j}, a_{i',j'} \in E$ cause a violation of PEP+, add an attack from $a_{i',j'}$ to $a_{i,j}$.*

As before, we refer to the extended cost efficiency AF as $(Args_S, \rightsquigarrow_S)$ for the remainder of this section for ease.

Theorem 3.18. *For feasibility AF $(Args_F, \rightsquigarrow_F)$, schedule S and $S \approx E$, let $(Args_S, \rightsquigarrow_S)$ be the efficiency AF for extended cost. Then E is stable in $(Args_S, \rightsquigarrow_S)$ iff S is feasible and satisfies SEP+ and PEP+.*

Proof. This proof is derived from Theorem 4.3 from Čyras et al.[5].

Let E be stable in $(Args_{S+}, \rightsquigarrow_{S+})$. We know E is conflict-free in $(Args_F, \rightsquigarrow_F)$ since attacks removed for SEP+ only make asymmetric attacks symmetric and attacks added for PEP+ are between arguments that do not already attack each other in $(Args_F, \rightsquigarrow_F)$ (since $i \neq i'$). So S is feasible by Theorem 2.7 (taken directly from[5]).

As E is stable for all $j \in \mathcal{A}, a_{i,j} \in E$, it follows that $a_{i,j} \rightsquigarrow_{S+} a_{i',j}$ where $i' \neq i$. So we do not need to remove attacks from $\rightsquigarrow_F$ to get to $\rightsquigarrow_{S+}$, so

$$w \neq j, x_{i,j} = x_{i',w} = 1, C_i - C_{i'} > c_{i',j} + \gamma[\delta(w,j) + \delta(j,w^+)] - \gamma\delta(w,w^+)$$

cannot hold for any (i, i', j), leading to SEP+ being satisfied.

Similarly, since E is conflict-free we have $j' \neq j, x_{i,j} = x_{i',j'} = 1$,

$$\gamma([\delta(j^-,j) + \delta(j,j^+)] - [\delta(j^-,j') + \delta(j',j^+)]) > [c_{i,j'} - c_{i,j}],$$
$$C_i - C_{i'} > [c_{i',j} - c_{i',j'}] + \gamma([\delta(j'^-,j) + \delta(j,j'^+)] - [\delta(j'^-,j') + \delta(j',j'^+)])$$

cannot hold for any (i, i', j, j') so S satisfies PEP+.

As in Theorem 4.3 from Čyras et al.[5], since S is feasible, E is stable in $(Args_F, \rightsquigarrow_F)$. S also satisfies SEP+ and PEP+ so $\rightsquigarrow_{S+}=\rightsquigarrow_F$. So S feasible, satisfies SEP+, PEP+ $\implies$ E stable in $(Args_{S+}, \rightsquigarrow_{S+})$. □

Corollary 3.19. *Every makespan scheduling problem can be represented by an extended cost scheduling problem.*

Proof. Simply set $\alpha = \gamma = 0, \beta = 1$. □

Example 3.6. *Consider the following problem, consisting of 3 jobs and 2 operators. For jobs 1-3, we have locations* $(3, 4, 0), (5, 12, 0), (5, 12, 0)$ *respectively. Both operators begin and end their shifts at the origin* $(0, 0, 0)$.

$$P = \begin{pmatrix} 120 & 120 \\ 60 & 60 \\ 30 & 60 \end{pmatrix}^T, \boldsymbol{q} = (1, 4, 3)$$

$\alpha = 0.9, \beta = 0.1, \gamma = 1$. *Consider the schedule*

$$S = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Calculating our costs, we have

$$c_{i,j} = \begin{pmatrix} 12.9 & 9.6 & 5.7 \\ 12.9 & 9.6 & 8.7 \end{pmatrix}_{i,j},$$

$$C_1 = 12.9 + 5.7 + (5 + \sqrt{68} + 13) \approx 44.85, C_2 = 9.6 + (13 + 13) = 35.6.$$

So $C_{max} = C_1 \approx 44.85$. *We can now do the following checks to build our efficiency AF:*

- *Move job 1 from operator 1 to 2 at start:* $C_1 - C_2 = 44.85 - 35.6 = 9.25$*,*
  $c_{i',j} + [\delta(w, j) + \delta(j, w^+)] - \gamma\delta(w, w^+) = 12.9 + [5 + \sqrt{68}] - 13 \approx 13.15.$
  $9.25 < 13.15 \implies$ *satisfies SEP+.*
- *Move job 1 from operator 1 to 2 after job 2: same as above.*
- *Move job 3 from operator 1 to 2 at start:* $c_{i',j} + [\delta(w, j) + \delta(j, w^+)] - \gamma\delta(w, w^+) = 8.7 + [13 + 0] - 13 = 8.7.$
  $C_1 - C_2 = 9.25 > 8.7 \implies$ *violates SEP+.*
- *Swap jobs 1 and 2:* $c_{i,j'} - c_{i,j} = 0$*,*
  $\gamma([\delta(j^-, j) + \delta(j, j^+)] - [\delta(j^-, j') + \delta(j', j^+)]) = [5 + \sqrt{68}] - [13 + 0] \approx 0.246 > 0$*,*
  $C_i - C_{i'} = 9.25$*,*
  $c_{i',j} - c_{i',j'} + \gamma\big([\delta(j'^-, j) + \delta(j, j'^+)] - [\delta(j'^-, j') + \delta(j', j'^+)] = 12.9 - 9.6 + ([5 + 5] - [13 + 13] = -12.7 < 9.25 \implies$ *violates PEP+.*
- *Swap jobs 3 and 2:* $c_{i,j'} - c_{i,j} = 9.6 - 5.7 = 3.9$*,*
  $\gamma([\delta(j^-, j) + \delta(j, j^+)] - [\delta(j^-, j') + \delta(j', j^+)]) = [\sqrt{68} + 13] - [\sqrt{68} + 13] = 0 < 3.9 \implies$ *satisfies PEP+.*

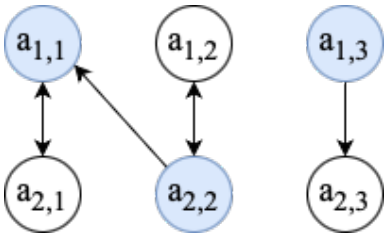


Figure 3.1: Example 3.6 efficiency AF

The corresponding efficiency AF is shown in Figure 3.6.

Lemma 3.20. *Given a schedule S and schedule order σ_i for all operators $i \in \mathcal{O}$, the extended cost efficiency AF $(Args_{S+}, \rightsquigarrow_{S+})$ can be constructed in $O(m^2n^2)$ time.*

Verifying whether an extension $E \subseteq Args_{S+}$ such that $E \approx S$, is stable can be done in $O(m^2n^2)$ time.

Proof. Recall that there may be multiple $i \in \mathcal{O}$ that satisfy $C_{max} = C_i$. Recall from Lemma 2.8, the construction of the feasibility AF takes $O(nm^2)$. To remove attacks, we must first identify the critical operators, taking $O(m)$ time. Then, for each critical operator (of which there are a maximum of m), to iterate over $(j, i', w) \in \mathcal{A} \times \mathcal{O} \times \mathcal{A}$ and check $x_{i,j} = 1, w \neq j, x_{i',w} = 1$ takes $O(mn^2)$ times. For each of these, calculating the distance metrics and the inequality comparisons take $O(1)$. So, in total, removing attacks takes $O(m^2n^2)$.

To add attacks, we iterate over $(j, i', j') \in \mathcal{A} \times \mathcal{O} \times \mathcal{A}$ for each critical operator and check $x_{i,j} = 1, x_{i',j'} = 1, i \neq i', j \neq j'$ which takes $O(m^2n^2)$ time. Calculating distances ($O(1)$) and checking inequalities ($O(1)$) means the addition of attacks also takes $O(m^2n^2)$ time.

As with Lemma 2.8, we determine if $E \subseteq Args_{S+}$ is stable by checking if E is conflict-free, taking $O(m^2n^2)$ time, and if E attacks every argument in $Args_{S+} \backslash E$, also taking $O(m^2n^2)$ time. □

3.3 Mapping Skill Constraints to Fixed Decisions

Recall fixed decisions as in Subsection 2.3.3, allowing us to specify if jobs cannot/must be completed by specific operators. We want to extend this to a specific case, where D^- is determined by further input information from the dataset (Section 2.6), specifically skill constraints that operators must satisfy to carry out the corresponding job.

Definition 3.21. *Let $\mathcal{K}$ be a set of **skills**, $\mathcal{K} = \{k_1, k_2, ..., k_\tau\}$ where $|\mathcal{K}| = K$. Then we define:*

- *The set of **skill prerequisites** $\mathcal{K}_j \subseteq \mathcal{K}$ for a job $j \in \mathcal{A}$,*
- *The set of **operator skills** $\mathcal{K}_i \subseteq \mathcal{K}$ for an operator $i \in \mathcal{O}$.*

By definition, any operator i assigned a job j must fulfill all skill criteria. In other words, a feasible (with relation to fixed decisions) schedule S must satisfy $\mathcal{K}_j \subseteq \mathcal{K}_i \forall i \in \mathcal{O}, j \in \mathcal{A}$.

Definition 3.22. *The **skill constraints AF** $(Args_{D_S}, \rightsquigarrow_{D_S})$ is defined as:*

- $Args_{D_S} = Args_F$,
- $\rightsquigarrow_{D_S} = (\rightsquigarrow_F \cup \{(a_{i,j}, a_{i,j}) : \exists k \in \mathcal{K}_j, k \notin \mathcal{K}_i\})$.

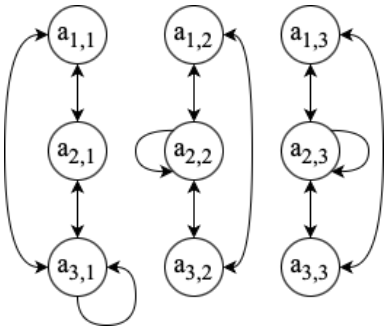


Figure 3.2: Example 3.7 skill requirement AF

Lemma 3.23. *Violating skill requirements are negative fixed decisions. In other words:*

$$\exists k \in \mathcal{K}_j : k \notin \mathcal{K}_i \implies (i,j) \in D^-.$$

Proof. We will use the contrapositive. By definition, if (i,j) is not a negative fixed decision, then it is a valid assignment within the schedule. By definition, we then must have $\mathcal{K}_j \subseteq \mathcal{K}_i$ if the skillsets are valid. The contrapositive of $\exists k \in \mathcal{K}_j : k \notin \mathcal{K}_i$ is $\forall k \in \mathcal{K}_j : k \in \mathcal{K}_i \iff \mathcal{K}_j \subseteq \mathcal{K}_i$ so the contrapositive holds. □

So the skill requirements can directly determine D^-, allowing us to use the previous theory as in Subsection 2.3.3 to explain any schedule S.

Example 3.7. *Consider a problem consisting of 3 operators and 3 jobs.*

- *Operator skills:*
  1. *001, 002, 003*
  2. *001, 003*
  3. *002, 003*
- *Job skill requirements:*
  1. *001*
  2. *002*
  3. *002, 003*

So $D^- = \{(2,2),(2,3),(3,1)\}$. *Figure 3.2 shows the corresponding skill requirements (fixed decision) AF.*

Lemma 3.24. *Given a schedule S, a set of job skill prerequisites and a set of operator skills, the skill constraints AF can be constructed in $O(mnK + nm^2)$ time.*

Verifying whether an extension $E \subseteq Args_S$ such that $E \approx S$, is stable can be done in $O(n^2m^2)$ time.

Proof. To get from skills to negative fixed decisions, we must iterate over each skill, job and operator $(k,j,i) \in \mathcal{K} \times \mathcal{A} \times \mathcal{O}$ and check $k \in \mathcal{K}_j, k \notin \mathcal{K}_i$, taking $O(mnK)$ for K total skills. There are no positive fixed decisions so we can ignore this array completely, or for every $(i,j) \in \mathcal{A} \times \mathcal{O}$ set the fixed decision as false (taking $O(mn)$ time).

By Lemma 2.13, we have that from these fixed decisions, the fixed decision AF can be constructed in $O(nm^2)$ time. As such, we have the overall construction time is $O(mnK + nm^2)$.

Verification follows trivially from Lemma 2.13. □

3.4 Individual Efficiency

Unlike the previous framework from Čyras et al.[5], which did not differentiate between the order of jobs, the component of distance and travel means that each operator's individual schedule should also be optimised, beyond the assignment of jobs. Recall the travelling salesman problem discussed in Section 2.5. Our individual schedule problem can be seen as a direct application of the TSP.

In this section, we prove our hypothesis that there exists an extension of SEP/PEP that allows us to utilise argumentation to solve this problem. Similarly to the TSP, we take inspiration from λ-optimality, and apply the ideas of insertion and inversion from Definition 2.18 to adapt our previous optimality principles to construct an argumentation framework in $O(n^2)$ (for fixed m).

Definition 3.25. *Let δ be the distance metric and σ_i the schedule order for a given schedule S and operator i. For $i \in \mathcal{O}, j \in \mathcal{A} : x_{i,j} = 1$, we define the following properties for all $\sigma_i[z] = w$:*

1. ***Individual Single Exchange Property** (ISEP): $\forall w \neq j, x_{i,w} = 1 : \delta(w,j) + \delta(j,w^+) - \delta(w,w^+) \geq \delta(j^-,j) + \delta(j,j^+) - \delta(j^-,j^+)$*
2. ***Individual Pairwise Exchange Property** (IPEP): $\forall j' \neq j, x_{i,j'} = 1 :$*

$$\begin{aligned}&\delta(j'^-,j) + \delta(j,j'^+) + \delta(j^-,j') + \delta(j',j^+)\\ \geq\ &\delta(j'^-,j') + \delta(j',j'^+) + \delta(j^-,j) + \delta(j,j^+).\end{aligned}$$

*We say S is **individual cost efficient** iff S is feasible and satisfies both ISEP and IPEP.*

Lemma 3.26. *Every optimal schedule with respect to individual cost satisfies ISEP and IPEP.*

Proof. ISEP: Let $D_i(S)$ be the objective distance function for a schedule S, where

$$D_i = \delta(s,\sigma_i[1]) + [\sum_{k=1}^{k_i-1} \delta(\sigma_i[k],\sigma_i[k+1])] + \delta(\sigma_i[k_i],f).$$

Let S* be an optimal schedule. If we move job j from its position between j^- and j^+ to a new position between w and w^+, we have

$$\begin{aligned}D_i(S) = D_i(S*) &- [\delta(j^-,j) + \delta(j,j^+)] + \delta(j^-,j^+)\\ &+[\delta(w,j) + \delta(j,w^+)] - \delta(w,w^+).\end{aligned}$$

Since $S*$ is optimal, any other schedule route will be longer and such $D_i(S) \geq D_i(S*)$. From this we get ISEP.

IPEP: consider two jobs j, j', both in optimal schedule $S*$ where $x_{i,j} = x_{i,j'} = 1$. Let job j be between j^- and j^+, and similarly j' between j'^- and j'^+. If we then have a schedule S such that j and j' swap positions in the order of jobs, we have:

$$\begin{aligned}D_i(S) = D_i(S*) &- [\delta(j'^-,j') + \delta(j',j'^+) + \delta(j^-,j) + \delta(j,j^+)]\\ &+[\delta(j'^-,j) + \delta(j,j'^+) + \delta(j^-,j') + \delta(j',j^+)].\end{aligned}$$

As with ISEP, we have by assumption $D_i(S) \geq D_i(S*)$ from which we deduce IPEP. □

Example 3.8. *Consider an operator i with 3 jobs at the following locations:*

1. $(3, 4, 0)$
2. $(5, 12, 0)$
3. $(5, 12, 0)$

and the operator beginning and ending its shift at $(0, 0, 0)$*. Suppose the individual schedule is* $\sigma_i = \{2, 1, 3\}$*. We can then check the following (not a conclusive list):*

- *Move job 1 to start:*
  $\delta(w, j) + \delta(j, w^+) - \delta(w, w^+) = 5 + \sqrt{68} - 13 \approx 0.246$
  $\delta(j^-, j) + \delta(j, j^+) - \delta(j^-, j^+) = 13 + 13 - 0 = 26 > 0.246 \implies$ *violates ISEP.*
- *Swap jobs 2 and 3 (*$j' = 2, j = 3$*):* $\delta(j'^-, j) + \delta(j, j'^+) + \delta(j^-, j') + \delta(j', j^+) = 13 + \sqrt{68} + \sqrt{68} + 13 \approx 42.49$
  $\delta(j'^-, j') + \delta(j', j'^+) + \delta(j^-, j) + \delta(j, j^+) = 13 + \sqrt{68} + \sqrt{68} + 13 \approx 42.49 \implies$ *satisfied IPEP.*
- *Swap jobs 1 and 3 (*$j' = 1, j = 3$*):* $\delta(j'^-, j) + \delta(j, j'^+) + \delta(j^-, j') + \delta(j', j^+) = 0 + 0 + 0 + 5 = 5$
  $\delta(j'^-, j') + \delta(j', j'^+) + \delta(j^-, j) + \delta(j, j^+) = \sqrt{68} + \sqrt{68} + \sqrt{68} + 13 \approx 37.74 > 5 \implies$ *violates IPEP.*

Recall our 2 options resulting in a violation of an argumentation framework:

1. An attack $a \rightsquigarrow b, a, b \in E$.
2. A non-attack $E \not\rightsquigarrow b, b \notin E$.

If we are considering arguments within the same extension, the latter option is not possible. Hence, we must find another way to map the SEP condition.

Definition 3.27. *For feasibility AF* $(Args_F, \rightsquigarrow_F)$ *and schedule* S*, the* ***extended cost efficiency AF*** $(Args_{IS}, \rightsquigarrow_{IS})$ *is defined as:*

- $Args_{IS} = Args_F$*,*
- $\rightsquigarrow_{IS} = \rightsquigarrow_F \bigcup \Big\{ (a_{i,j}, a_{i,j}) : \exists w \neq j, x_{i,w} = 1 :$
  $\delta(w, j) + \delta(j, w^+) - \delta(w, w^+) < \delta(j^-, j) + \delta(j, j^+) - \delta(j^-, j^+) \Big\}$
  $\bigcup \Big\{ (a_{i,j'}, a_{i,j}) : x_{i,j} = x_{i,j'} = 1, j \neq j',$
  $\delta(j'^-, j) + \delta(j, j'^+) + \delta(j^-, j') + \delta(j', j^+) < \delta(j'^-, j') + \delta(j', j'^+) + \delta(j^-, j) + \delta(j, j^+) \Big\}.$

Example 3.9. *Figure 3.3 gives the final AF for the problem described in Example 3.8.*

In the individual efficiency AF, any violation of ISEP/IPEP will always be from an attack within the extension. As such, despite both conditions now being reflected in the framework, we should now find extra steps must be taken to discern which type of violation we have (i.e. check whether $a \rightsquigarrow a$ attacks itself or not).

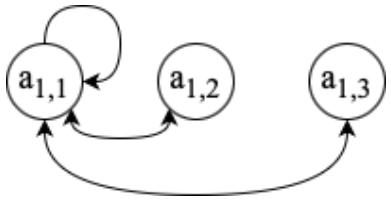


Figure 3.3: Example 3.9 individual cost efficiency AF

Lemma 3.28. *Given a schedule S and schedule order σ_i for all operators $i \in \mathcal{O}$, the individual cost efficiency AF $(Args_{IS}, \rightsquigarrow_{IS})$ can be constructed in $O(nm(n+m))$ time.*

Verifying whether an extension $E \subseteq Args_S$ such that $E \approx S$, is stable can be done in $O(m^2n^2)$ time.

Proof. From Lemma 2.8, the construction of the feasibility AF takes $O(nm^2)$ for the first set of attacks.

For the second set of attacks, every $i \in \mathcal{O}$, we iterate over $(j, w) \in \mathcal{A} \times \mathcal{A}$ and check $w \neq j, x_{i,w} = 1$ taking $O(mn^2)$ time each. The last inequality and distance metrics can be calculated in $O(1)$ so in total the addition of self-attacks takes $O(mn^2)$.

For the final set of attacks, we iterate over $(j, j') \in \mathcal{A} \times \mathcal{A}$ for each critical operator and check $x_{i,j} = 1, x_{i,j'} = 1, j \neq j'$ which takes $O(mn^2)$ time. Calculating distances ($O(1)$) and checking inequalities ($O(1)$) means the addition of these attacks also takes $O(mn^2)$ time. In total, we have $O(nm^2) + O(mn^2) + O(mn^2) = O(nm^2 + mn^2)$.

As with Lemma 2.8, we determine if $E \subseteq Args_S$ is stable by checking if E is conflict-free (first checking for self-attacks and then regular conflicting attacks), taking $O(m^2n^2)$ time. □

Definition 3.29. *For a given schedule S and set of skills $\mathcal{K}$, $E \approx S$, $(Args, \rightsquigarrow) \in \{(Args_F, \rightsquigarrow_F), (Args_{S+}, \rightsquigarrow_{S+}), (Args_{D_S}, \rightsquigarrow_{D_S}), (Args_{IS}, \rightsquigarrow_{IS})\}$, for an attack $a \rightsquigarrow b, a, b \in E$:*

- $(a, b) \in \rightsquigarrow_F \implies$ *S not feasible,*
- $(a, b) \in \rightsquigarrow_{S+} \setminus \rightsquigarrow_F \implies$ *S not efficient,*
- $(a, b) \in \rightsquigarrow_{D_S} \setminus \rightsquigarrow_F \implies$ *S violates skill requirements.*
- $(a, b) \in \rightsquigarrow_{IS} \setminus \rightsquigarrow_F \implies$ *S not individual schedule efficient,*

For a non-attack $E \not\rightsquigarrow b, b \notin E$:

- $\rightsquigarrow = \rightsquigarrow_F \implies$ *S not feasible,*
- $\rightsquigarrow = \rightsquigarrow_{S+}$ *and* $b \rightsquigarrow_{S+} E \implies$ *S not efficient,*

Note that non-attacks are not possible for $(Args_{D_S}, \rightsquigarrow_{D_S}), (Args_{IS}, \rightsquigarrow_{IS})$ since neither AF have attacks removed to get from $(Args_F, \rightsquigarrow_F)$ to their respective AFs.

3.5 Instruments

Instruments can be allocated to operators in a similar fashion to jobs. Unlike job allocation, there is no efficiency component involved (instruments don't carry some associated value) and there is no ordering between instruments assigned to a specific operator.

Definition 3.30. *$\mathcal{I}$ is defined as the set of instruments for a given problem. We define $\mathcal{K}_\tau \subseteq \mathcal{K}$ as the set of skill requirements for instrument $\tau \in \mathcal{I}$.*

Instrument allocation obeys the following rules:

1. *All instruments must be allocated to exactly one operator.*
2. *Any skill constraints must be satisfied by the assigned operator for a given instrument.*

Lemma 3.31. *Instrument allocation can be modelled by argumentation for explainable scheduling[5], without optimality/efficiency AF.*

Proof. From Definition 3.30, rule 1 can be modelled by the feasibility AF by its definition. Rule 2 can be modelled by mapping skill requirements to negative fixed decisions by Lemma 3.23. □

Example 3.10. *Consider 2 operators with the following skills:*

1. *A0D 4D7 E60*
2. *E60*

and instruments $I0-3$. Let $I1$ have skill requirements $\{A0D, E60\}$, and all other instruments be requirement-free.

Then

$$SI = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

is a feasible and valid instrument assignment, but

$$SI' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

is not since it violates the skill constraints for $I1$.

3.5.1 Job Instrument Constraints

As well as skill requirements, some jobs may also require specific instruments that the assigned operator must carry on their shift.

Definition 3.32. *$\mathcal{I}_j \subseteq \mathcal{I}$ is defined as the set of instrument requirements for job $j \in \mathcal{A}$.*

To define a framework to assess the relations between jobs and instruments, we should first consider how this differs from job-operator and instrument-operator relations. Note that both of these operate on a many-to-one basis, since multiple jobs/instruments can be allocated to one operator, but only one operator is associated with each job/instrument. For job-instrument relations, many jobs may require many instruments.

We will attempt to extend some aspects of the argumentation framework here to accommodate the many-to-many nature of the job-instrument relation. We do this by reversing the roles that the assignments and fixed decisions play, constructing an AF from our job/instrument allocations and comparing against the constraints.

Definition 3.33. *Let n be the number of jobs and t the number of instruments in a problem. We define $\zeta \in \{0,1\}^{n \times t}$ as the job-instrument constraint matrix, such that*

$$\zeta_{j,\tau} = 1$$

if instrument τ is a requirement for job j, and 0 otherwise.

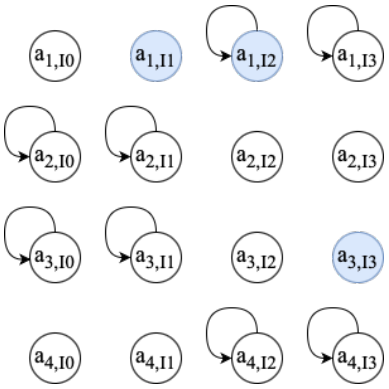


Figure 3.4: Example 3.11 job-instrument assignment AF

Definition 3.34. *For n jobs and t instruments, consider job allocation schedule S and instrument allocation schedule SI. We define S_i and SI_i as the set of jobs and instruments assigned to operator $i \in \mathcal{O}$ respectively.*

*The **job-instrument assignment AF** $(Args_{JI}, \rightsquigarrow_{JI})$ is defined as:*

- $Args_{JI} = \{a_{j,\tau} : j \in \mathcal{O}, \tau \in \mathcal{I}\}$,
- $\rightsquigarrow_{JI} = \{(a_{j,\tau}, a_{j,\tau}) : \tau \in S_i, j \notin SI_i, i \in \mathcal{O}\}$.

We have now constructed an AF from our two schedules for job/instrument assignment, which will allow us to check if a given set of instrument requirements would be violated by these independent assignments. Our new "schedule" here will be our instrument requirements for each job.

Definition 3.35. *For a given job-instrument constraint matrix ζ, $E \approx \zeta$, $(Args, \rightsquigarrow) \in (Args_{JI}, \rightsquigarrow_{JI})$, for an attack $a \rightsquigarrow a, a \in E$:*

- $(a, a) \in \rightsquigarrow_{JI} \implies$ *job-instrument requirements are violated.*

Example 3.11. *Continuing from Example 3.10, suppose we also have jobs 1-4 with the following schedule assignment:*

$$S = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

and job-instrument constraint matrix:

$$\zeta = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Using SI from Example 3.10, Figure 3.4 shows the job-instrument assignment AF with ζ highlighted in blue where $\zeta_{j,\tau} = 1$. Here we can see there is a violation since $a_{1,I2} \rightsquigarrow a_{1,I2}$; in other words, $I2$ is an instrument requirement for job 1, however they are allocated to different operators.

Lemma 3.36. *Given job and instrument assignment schedules S, SI and job-instrument constraint ζ, the job-instrument constraint AF can be constructed in $O(mnt)$ time, where t is the number of instruments in the problem.*

Verifying explanations can be done in $O(nt)$ time (note $E \approx \zeta$ is not stable since E does not defend any argument of $Args_{JI} \backslash E$.

Proof. The job-instrument constraint AF consists of $O(nt)$ arguments. To add attacks, we must iterate over S_i and SI_i for each $i \in \mathcal{O}$, taking $O(mnt)$.

To verify explanations, we only need to check for self-attacks, taking $O(nt)$. □

3.6 Summary

In this chapter, we have considered various aspects of the dataset in Section 2.6 and utilised previous theory from Čyras et al.[5] to extend the mapping for this problem. In particular, we have extended the efficiency AF to consider multiple cost variables of different weightings; it is worth noting the assumptions that are taken into account and alternative cost formulae that could be used instead, giving different results. For example, taking the product of the cost variables, rather than the sum, would provide a different set of results when a combination of variables are considered. In particular, alternative formulae relating to priority can be used to better model this optimality, as discussed in Chapter 6.

We've also considered previous theory on fixed decisions and applied this to real-world problems: the requirement constraints created by skills and instruments. By dividing these constraints into separate argumentation frameworks, we can easily identify which constraint is violated for a given schedule. Note that we could combine all these constraints into a single argumentation framework, akin to the fixed decision AF from Čyras et al., however this would not offer as detailed explanations to the user.

We end this chapter by tying together all the argumentation frameworks we have explored thus far:

Theorem 3.37. *Given $\mathcal{O}, \mathcal{A}, \mathcal{K}, \mathcal{I}$, schedules S, SI and job-instrument constraint ζ, the total time taken to construct the argumentation frameworks for the problem is $O(mn(mn + K + t))$.*

Verifying explanations for the relevant extensions can be done in $O(m(mn^2 + t))$.

Proof. For S, Lemmas 2.8, 3.20, 3.24, 3.28 and 3.36 give all constructions/explanations relating to job assignment. For SI, Lemma 2.8, 3.24, 3.36 give all constructions/explanations relating to instrument assignment. Adding all these together gives the above results. □

Chapter 4

Design and Implementation

In this chapter, we explore the implementation of the theory discussed in Chapter 3. As a basis for the tool, we use the proof-of-concept presented by Čyras et al.[29](the source code for which can be found on GitHub[6]), as shown in Figure 4.1. The final tool is shown in Figure 4.2 and is accessible at https://gitlab.doc.ic.ac.uk/jcl122/OptimalSchedulingWebApp.

4.1 Incorporating Schedule Order

One key addition to the argumentation framework theory is the idea of schedule order. The original interactive schedule explainer[6] takes an input S of type boolean NumPy array, such that the i, jth element of S is true iff job j is assigned to operator i. In Section 3.2 we define σ_i, the schedule order for an individual operator $i \in \mathcal{O}$.

There are a few ideas that could be used to translate σ_i from a textual input into a form our backend could interpret:

1. Taking a separate input σ into our model. σ gives the order of **all** jobs, which can then be interpreted by the tool into its individual σ_i.

2. Revise the parsing function for schedule S to interpret the order that jobs appear within the schedule input text, as in Figure 4.3.

Figure 4.4 shows the parsing and formatting functions which follows the latter method, and saves the order of jobs in one variable σ. For example, in the case of Figure 4.3, we would have $\sigma = \{D, C, B, A, E\}$ since we have no way of knowing the true order of jobs spread between operators. Namely, the user does not have to input the schedule more than once, in different forms (reducing the risk of user errors). Similarly, any actions that update the schedule S also update σ, such that `format_schedule_with_sigma` will give the correct updated schedule ordering.

4.2 Weighted Variables

In Section 3.2, we explore an extended cost function that accommodates a combination of processing time, distance and priority when deciding whether or not a schedule is preferred to another. We have the restriction that $\alpha + \beta = 1$; however, enforcing this restriction is not intuitive for the user.

One potential approach for implementing this would be to include three text boxes for α, β, γ values in the problem input section of the interface. However, this leads to infinite

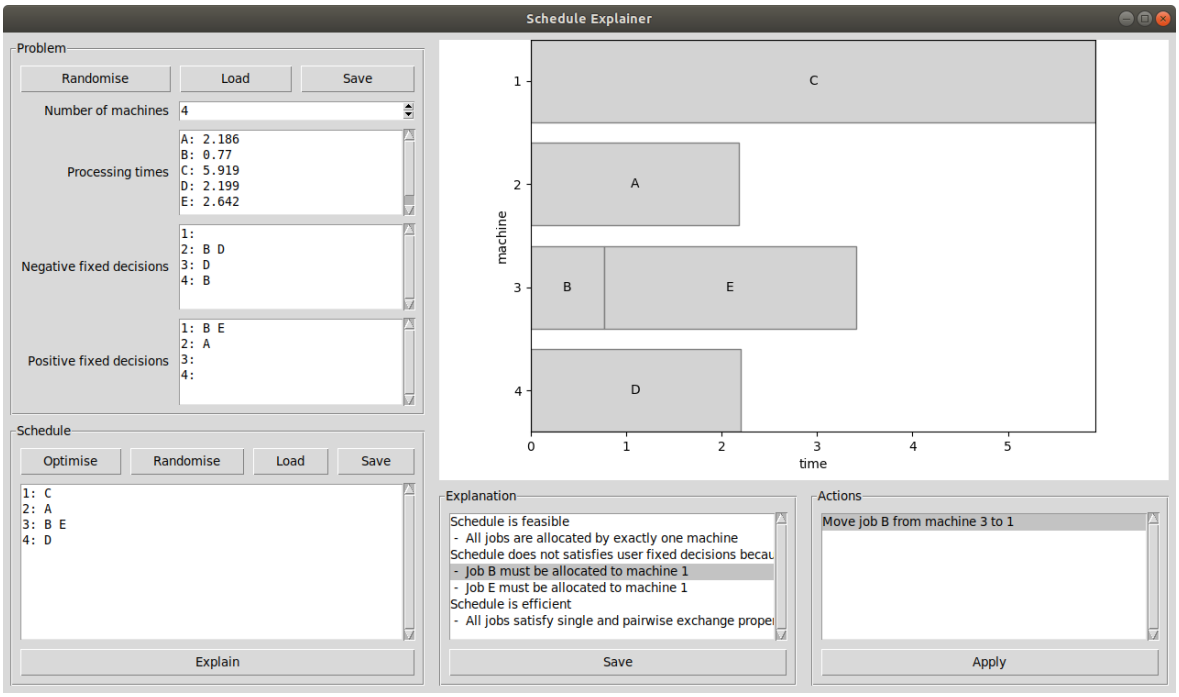


Figure 4.1: Original proof-of-concept tool[6]

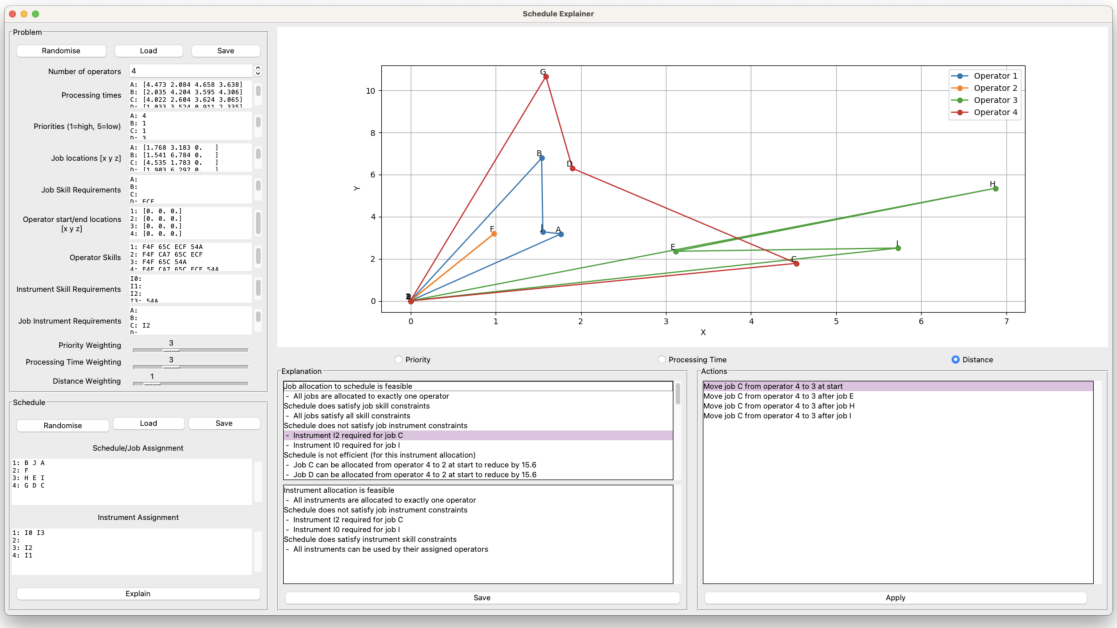


Figure 4.2: Updated tool interface

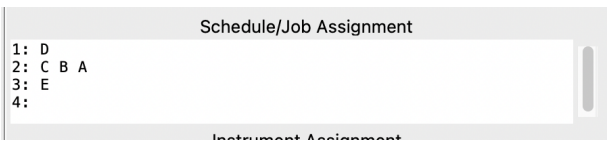


Figure 4.3: Order-sensitive schedule input

```
import numpy as np
import src.formatter as formatter

def parse_schedule(text, m, n):
    indices = [[formatter.letters_to_number(cell) for cell in row]
        for row in vectorise(text, int)]
    m_sparse = len(indices)
    m = max(m, m_sparse)
    n = max([i for row in indices for i in row] + [-1, n - 1]) + 1
    S = np.zeros((m, n), dtype=bool)
    for i in range(m_sparse):
        for j in indices[i]:
            S[i, j] = True
    return S

def format_schedule_with_sigma(sigma, S):
    m, _ = S.shape
    return ''.join('{}: {}\n'.format(i + 1,' '.join(
        [formatter.number_to_letters(j) for j in sigma if S[i, j]]
        )) for i in range(m))

```

Figure 4.4: `parse_schedule` and `format_schedule_with_sigma` for order-dependant schedule implementation.

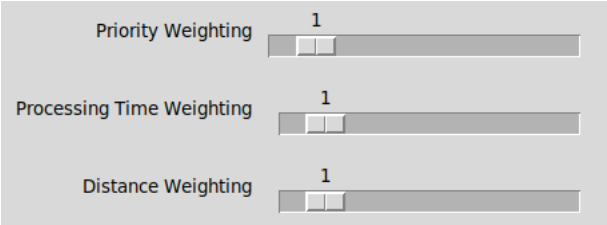


Figure 4.5: Weighted variables in tool interface

combinations and values of α, β, γ which, while valid in the context of the problem, is not the most user-friendly approach. Figure 4.5 shows our implementation of sliders in the tool, restricting the user to selecting an integer value between 0 and 10 for each cost variable of processing time, priority and distance. This allows the user sufficient flexibility in selecting ratios between the three cost variables, while keeping these possibilities finite. This also makes the interface more readable in cases where there are already multiple text boxes in the problem input section.

4.2.1 Graphical Representations

As we can see in Figure 4.1, the original tool from Karamlou[6] features a bar chart that shows the jobs allocated to each schedule with their processing time. While this is intuitive for a simple cost calculation like processing time, for a combined cost with multiple variables (including distance which is not easily demonstrated by bar chart), this simplified graphical representation provides limited useful information to the user.

The updated tool allows the user to switch between 3 different graphs, depending on whether they want to view the priority, processing time or distance distribution between operators. In particular, while the processing time and priority are represented by bar charts,

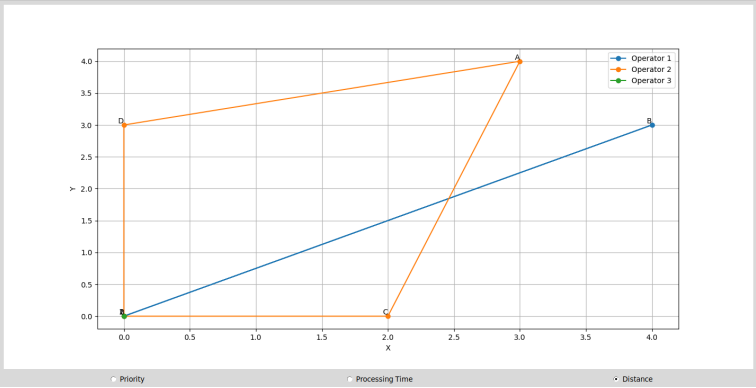


Figure 4.6: Distance graph output: operator 2 is allocated jobs C, A, D in that order, and operator 1 is allocated job B.

distance is plotted on a 2D axis (the z coordinates, assumed to be height, are ignored for readability in the graph) with the path of each operator shown by a different colour, as in Figure 4.6.

4.3 Instruments

Instrument assignment essentially requires adding another scheduling problem into the tool. Since we are considering two argumentation frameworks simultaneously, we divide the "Explanations" section into two for jobs and instruments. Selecting an actionable explanation from either box gives suggestions as to how to improve the schedule using that argumentation framework; for example, consider Figure 4.7 which gives `'Instrument I4 is required for job F"` as an explanation for both frameworks. Here, our tool gives two possible types of action, depending on which box the user clicks on:

1. For job assignment: `"Move job F from operator 2 to ..."`;
2. For instrument assignment: `"Move instrument I4 from operator _ to 2"`.

In other words, we can improve the overall assignment schedules by either updating the job or the instrument assignment. The majority of the instrument implementation utilises the original source code of Karamlou[6] as instrument allocation does not require additional variables to be considered, such as schedule order.

4.4 Limitations of Source Code

As this tool is built upon the previous proof-of-concept tool by Karamlou[6], there are some built-in features of the code that lead to limitations when extended to this much-broader use case.

1. Feasibility AF: Since order is considered in our extended implementation, a schedule which has a job j assigned to the same operator i twice – in different positions in the schedule – is infeasible (in Karamlou's tool this does not impact the model since the schedule is independent of order; if the job is applied multiple times it will only be counted once in the schedule).

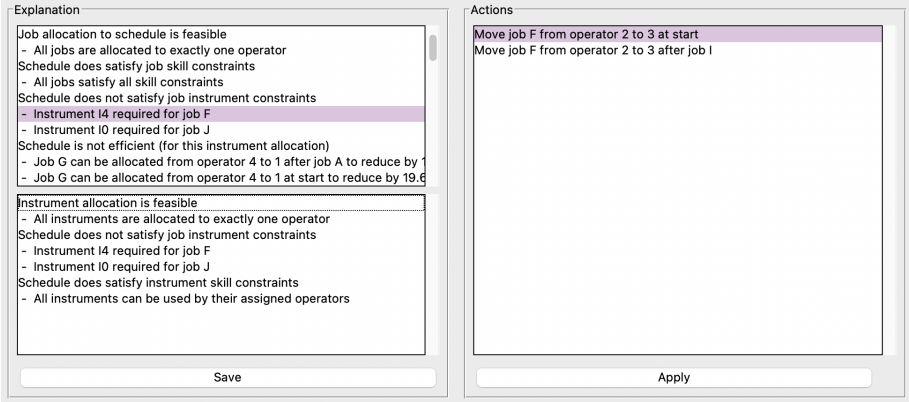


Figure 4.7: Explanations and actions output

To counteract this, in our parsing checks we specifically test if any job is counted twice within the same schedule. While this does not use an argumentation implementation, it allows the other feasibility checks (a job appears twice under *different* operators) to be tested by argumentation.

2. Optimality by Instrument Allocation: In theory, every AF is considered independently, such that efficiency AFs may suggest improvements that then contradict fixed decisions. To prevent this, the extended tool will only give efficiency improvement suggestions if the improved schedule does not conflict with any fixed decisions – namely, job skill requirements and job instrument requirements (there is no efficiency AF for instrument allocation so instrument-skill requirements do not impact this).

   In the case of job-instrument requirements, applying suggested actions will only improve the schedule for the given instrument allocation. For example, consider a schedule containing a job j and instrument I1. I1 is an instrument prerequisite for job j, and these are both allocated to operator 3. However, suppose the processing time for operator 3 is 10, compared to the processing time for operator 1 which is 2. It seems preferable therefore to move instrument I1 and job j to operator 1 (assuming I1 is not a prerequisite for any other jobs), however this suggestion will not be made by our tool since it sees the position of I1 as fixed.

Chapter 5

Evaluation

In this chapter we evaluate our tool implementation with both quantitative and qualitative tests. First, we use our tool to provide explanations for real data examples from *Terranova*[4]. While not a perfect mapping (namely due to the lack of time component, as explored in Chapter 6), the tool provides a comparison with the real-examples generated by *Terranova* using an optimisation software.

Secondly, we analyse a user study completed by *Terranova*, testing the tool and comparing the efficiency and accuracy of improvements made by the tool vs by hand. As part of this, we compare how users improve completing optimisation problems by hand as well as with the tool, as the survey progresses.

5.1 Comparison with Company-Provided Examples

5.1.1 City

We first look at the data sample given in `.json` format and translated into the appropriate `.problem` and `.schedule` files (see https://gitlab.doc.ic.ac.uk/jcl122/OptimalSchedulingWebApp/-/tree/master/terranova_sample_data). This data sample contains 30 operators, 57 jobs and 0 instruments. We use default weightings $(\alpha, \beta, \gamma) = (0, 1, 1)$ to analyse the schedule.

Importantly, this schedule has no feasible solution (since some jobs have skill requirements which no operator satisfies), with the corresponding schedule for the data sample simply omitting the infeasible jobs. Our implementation from Chapter 4 gives an error as shown in Figure 5.1.

If we were to parse the original problem for jobs without feasible allocations, it would take many comparisons to check which jobs' skill requirements were not met by any operator. In this case, since the tool gives the HEX code for each unfulfilled skill, we can easily go through the list of job skill prerequisites and remove these to get the output seen in Figure 5.2. The full text output can be seen in Appendix B.1.1.

Unsurprisingly, the tool explains this schedule is not feasible, since 5 jobs are not allocated to any operator; these are the same jobs that had infeasible skill requirements, and such we can justify this explanation and don't need to take further action. Notably, the schedule passes all other efficiency criteria and fixed decision checks, suggesting our tool and *Terranova*'s optimisation software are at least as good as each other. For any ratio of processing time and distance weightings (including $\beta = 0$ or $\gamma = 0$), we get the same optimal schedule result.

For priority weightings (any combination where $\alpha \neq 0$), we no longer have an optimal

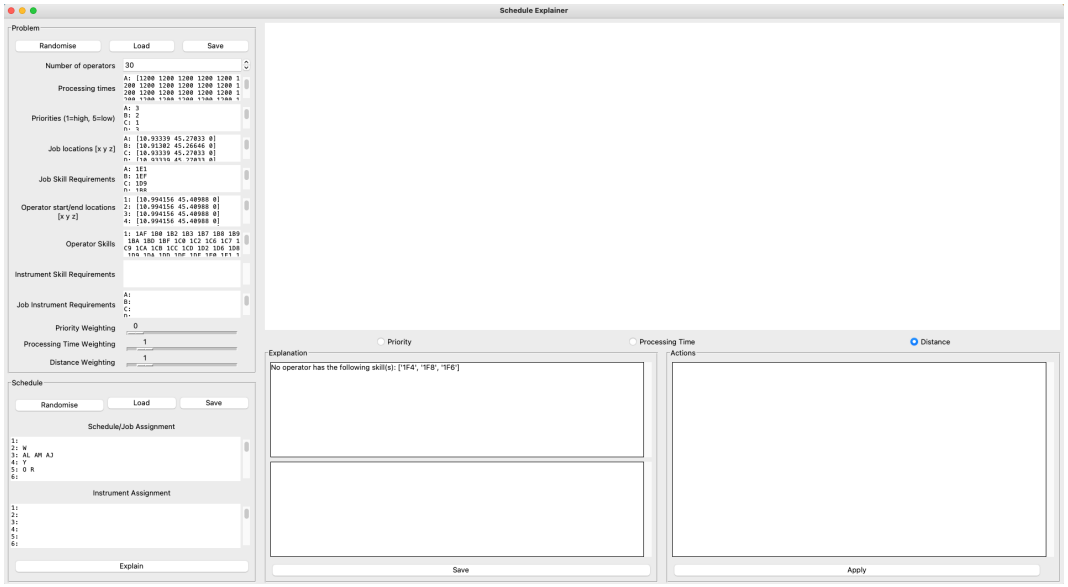


Figure 5.1: Initial output for *City* data sample

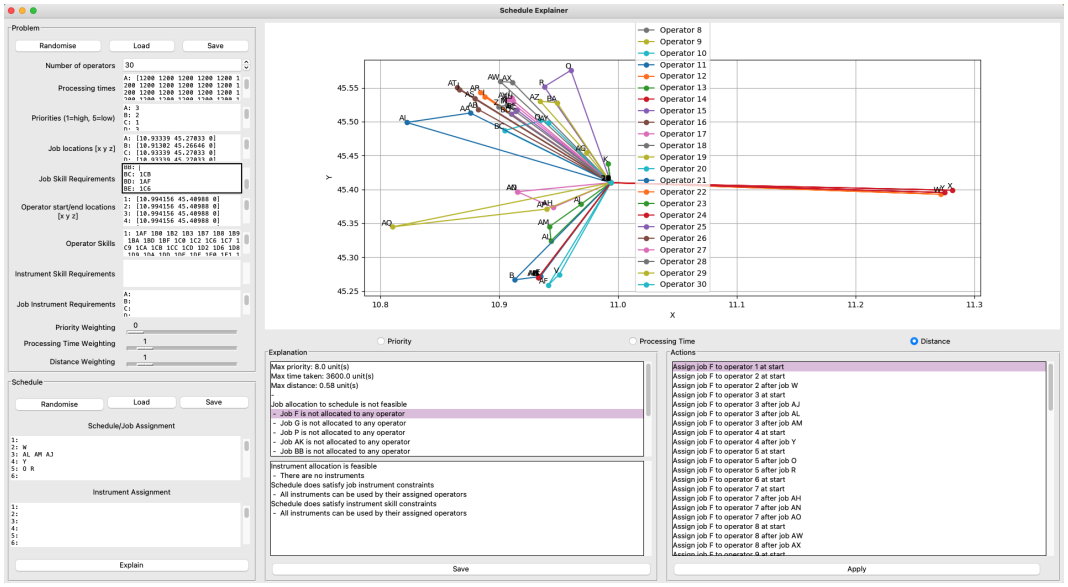


Figure 5.2: Amended output for *City* data sample

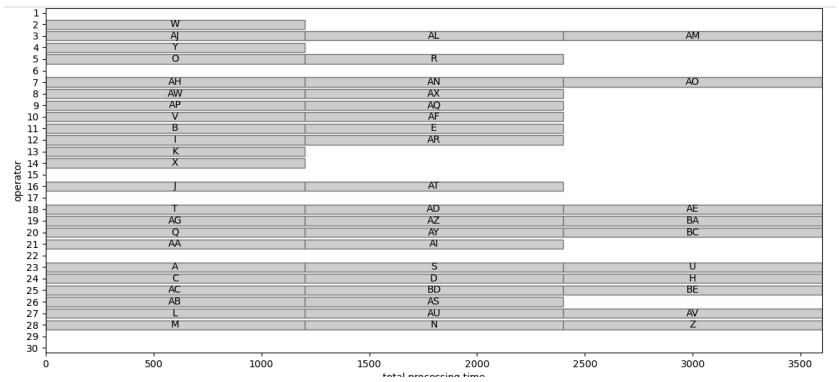


Figure 5.3: Processing time output for *City* data sample

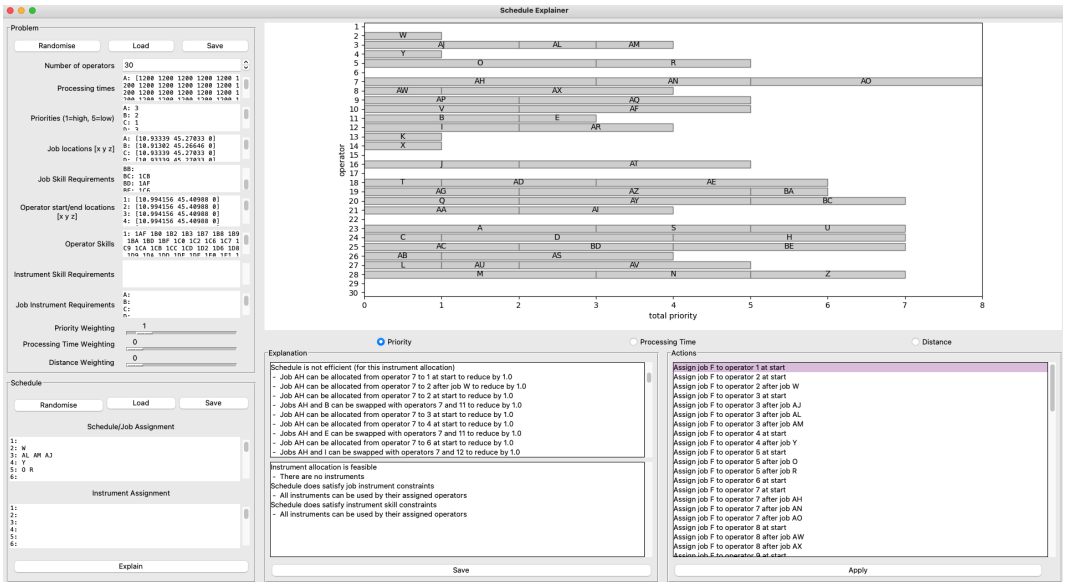


Figure 5.4: Priority output for *City* data sample

schedule, as shown in Figure 5.4. The full output text can be found in Appendix B.1.2. Considering the context of the problem and the low-emphasis on priority as an optimisation criteria, the likely conclusion from these results is either that our version of priority mapping is not sufficient for the given problem, or priority should not be weighted this heavily in the optimisation.

5.1.2 hpa

Once again, the *hpa* data sample is given in `.json` format, and translated into a suitable `.problem` and `.schedule` file for our tool to interpret. Larger than *City*, this data sample contains 30 operators, 132 jobs and 0 instruments.

Using once again our default weighting $(\alpha, \beta, \gamma) = (0, 1, 1)$, the tool states that the schedule is inefficient, as shown in Figure 5.5. The textual output is given in Appendix B.2.1. However, if we set $(\alpha, \beta, \gamma) = (0, 1, 0)$ or $(\alpha, \beta, \gamma) = (0, 0, 1)$ (see Figure 5.6, for example), we find that, similarly to *City*, our tool output states the schedule is optimal.

Our tool does give some unexpected results. For example, when processing time is the only optimisation criteria, the maximum time taken for any operator is 19200 (minutes); when we consider a dual cost involving distance and processing time (and follow the suggested improvement actions), we are able to get this schedule down to 12000 (minutes).

This is because our tool will only suggest actions **if** this will lead to a reduction in the overall maximum cost for any operator. Consider the example in Figure 5.7 where every job takes 3 units of time for every operator (with no restrictions/fixed decisions, $(\alpha, \beta, \gamma) = (0, 1, 0)$). Clearly, the optimal solution is to assign one job per operator. However, since there is not one move that will improve the schedule (the makespan will still be 6 regardless of which job is moved), no suggestions for improved efficiency will be made.

Coming back to our data sample example, since there are 2 operators with the exact same maximum total processing time, no actions are suggested to improve the schedule. On the other hand, since adding a distance metric allows us to differentiate between these jobs, our tool can now suggest actions with immediate cost improvements.

The reason for the tool otherwise giving improvements for this *Terranova* data example is

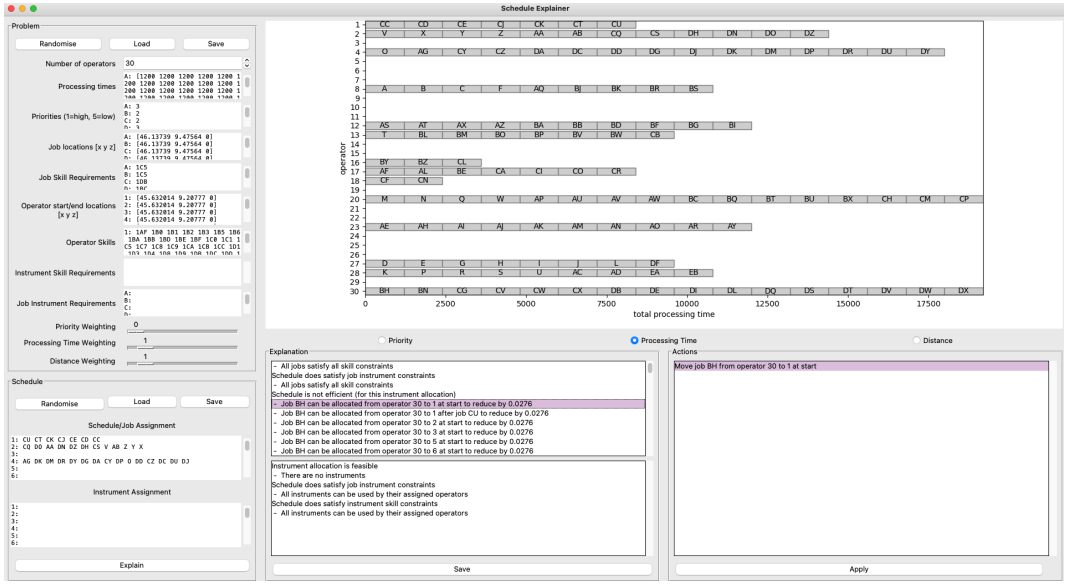


Figure 5.5: Initial output for *hpa* data sample

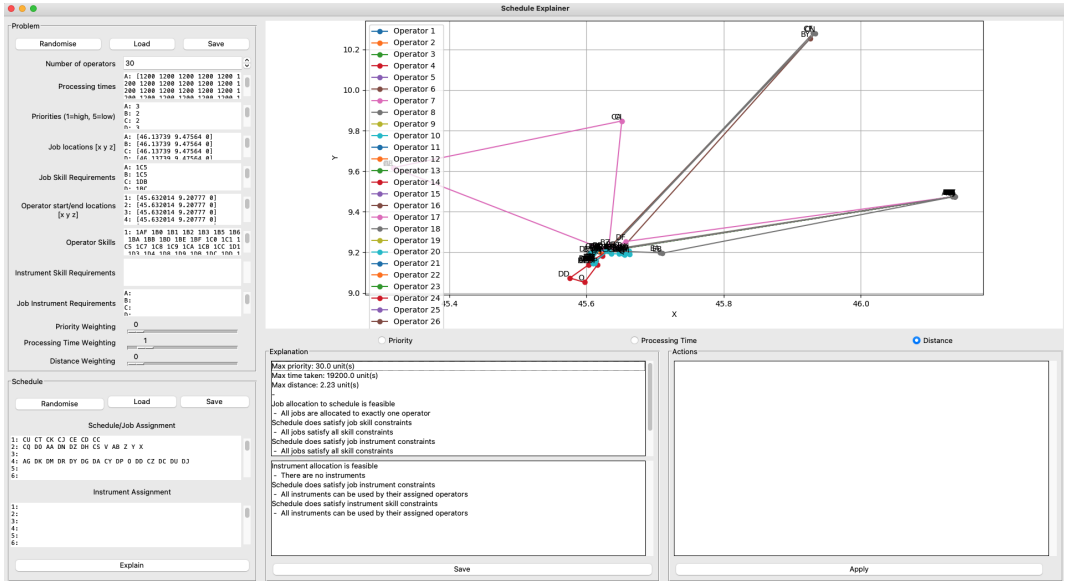


Figure 5.6: Distance output for *hpa* data sample

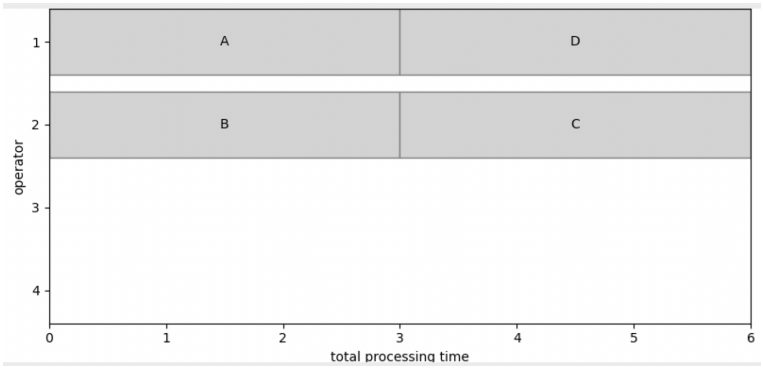


Figure 5.7: Toy example to demonstrate suboptimality

	Number of respondents
Half then half 1	7
Alternating 1	7
Half then half 2	3
Alternating 2	1

Table 5.1: Count of survey responses for each order type

likely due to the lack of time component, which forces jobs to be completed at specific times in the schedule and thus limits both the order of jobs and which operators can complete the job.

5.2 User Study

As part of the evaluation of our proof-of-concept tool, 18 employees from *Terranova* completed a study which compares the speed and accuracy of users problem solving using our tool, compared to working by hand to solve the exercises.

The format for the user study consists of 2 sets of questions and 4 distinct question orders:

- Half then half 1 (question set 1 done by hand; tool introduced; question set 2 done using tool),
- Half then half 2 (question set 2 done by hand; tool introduced; question set 1 done using tool),
- Alternating 1 (alternating questions from both sets with tool introduced at start; set 1 done by hand, set 2 done using tool),
- Alternating 2 (alternating questions from both sets with tool introduced at start; set 2 done by hand, set 1 done using tool).

The list of questions and full breakdown of orderings can be found in the GitLab repository for the project. The number of responses for each type are shown in Table 5.1.[1]

Questions in each set progressed in difficulty, in terms of optimisation criteria, choice of numbers (e.g. for processing times, integer values where all operators had same processing time for a given job; for distance, Pythagorean triples given) and additional constraints (instrument and skill requirements).

5.2.1 Outline of Questions

Appendix C.1 gives an overview of the information provided for respondents during the user study. Each question set contains 7 questions in the following order:

1. Distance (easy).
2. Time (easy).

[1]The survey order shown to each user was randomised upon refresh, set so each survey order was shown an even number of times. Hence, every refresh or view of the survey incremented the counter (even if that survey was not completed). For that reason, by randomisation, results are heavily weighted in favour of orders where question set 1 was completed by hand, and question set 2 completed using the tool.

- Scheduling with 4 jobs and some minor variation in processing time between operators.

3. Time (medium).

   - Compared to the previous time question, this question requires the respondent to schedule 8-9 jobs with completely varying processing times across operators.

4. Choice of Schedule (medium).

   - Respondents are given 3 schedules, only one of which is optimal, and must select the optimal schedule from the non-optimal schedules.

5. Time with Skills (medium).

6. Distance with Instruments (medium).

7. Combined (hard).

   - Respondents are asked to optimise the combined cost of distance and processing time, set on a 1:1 ratio (so an increase of 3 units in processing time but decrease in 3 distance units leads to no net different in cost).

All questions involved 4 operators. The questions were designed by amending randomly generated problems from the tool (e.g. rounding floats to makes numbers easier to work with). For ease, all z-coordinates for any locations were set to 0 to allow respondents to visualise the problem more easily.

In particular, the order of questions was chosen to ensure problems became progressively complex as the user became familiar with the tool and the layout of questions.

5.2.2 Completion Times

Figure 5.8 shows the average time users from each survey type spent on each question (the colour blue here denotes the combined average of half-then-half 1 and alternating 1, while blue denotes half-then-half 2 and alternating 2).[2]

The time taken to solve problems greatly fluctuates depending on the question (with some weak correlation between sets), but we can make the following conclusions from Figure 5.8:

1. For both question sets, the first distance question (meant to be a simple route optimisation question) has a high average completion time. This is likely due to lack of familiarity with the format of questions and/or the tool.

2. Time 1 & 2 are intended to be simple makespan problems, while Time 3 & 4 use much larger examples (around double the number of jobs etc.). This is reflected in the steep increase in completion time between Time 1 and 4, and Time 2 and 3, for their respective question sets, particularly for those completing the problems by hand.

[2]For question set 2, there was only one response for alternating by hand, whom did not answer Q6 or Q7. The recorded times the user spend on the page for each question were 7583.96 seconds (more than 2 hours) and 2.25 seconds respectively. These data points are omitted as outliers in the above table.

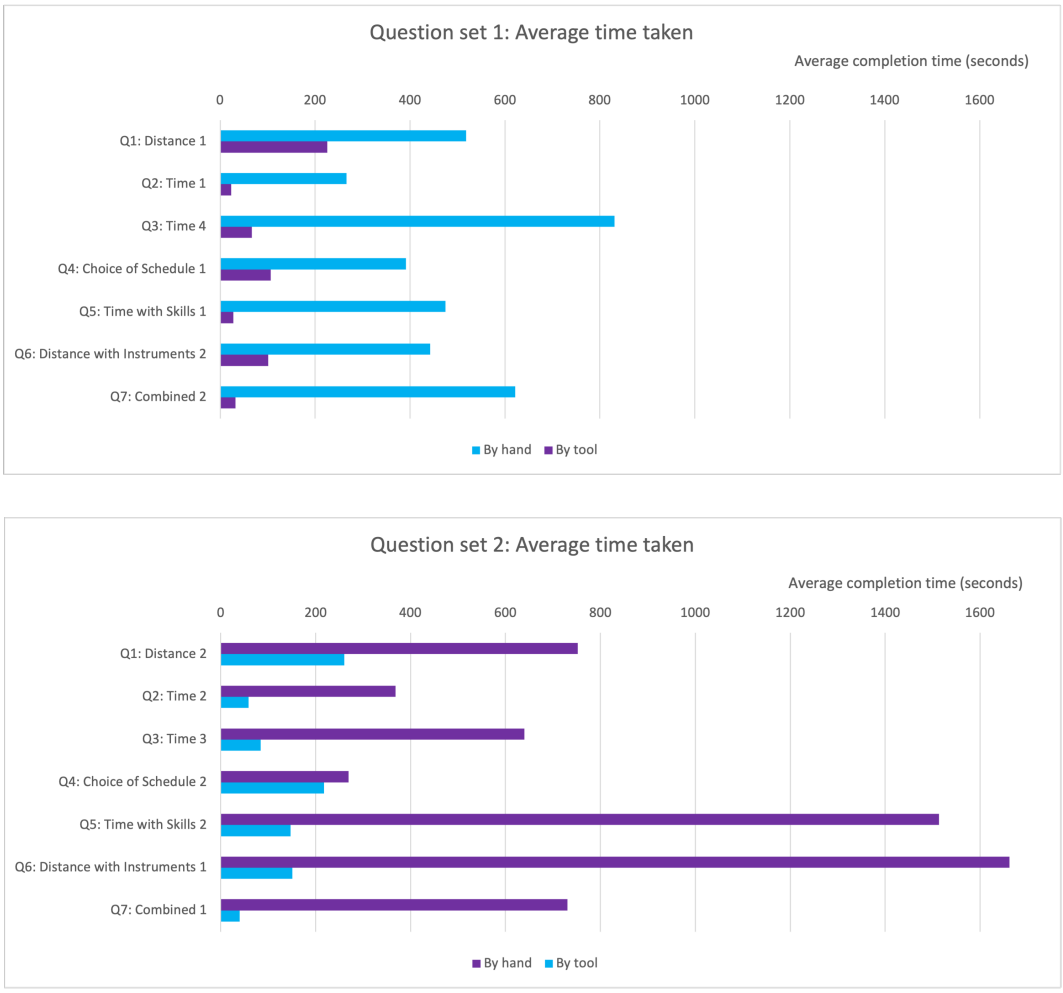


Figure 5.8: Average time taken (in seconds) for alternating (Alt) and half-then-half (HTH) groups to complete each question set.

	Average Accuracy
Alternating 1	68.2%
Half-then-half 1	65.1%
Alternating 2	36.0%
Half-then-half 2	81.2%
ALL RESPONDENTS	67.4%

Table 5.2: Accuracy of each survey group by proportion of questions correctly answered.

3. The remaining questions are intended as medium difficulty (but still challenging to solve by hand). The one exception to this is Combined 1 & 2, which we see above is particularly quickly solved by tool. This is likely due to users being particularly familiar with the tool and survey format by this point, and the question containing no constraints as in the previous 2 (skills and instruments), allowing for a faster use of the interface.

On average, the time taken for users to complete questions by hand is more than 10 times slower (1008%) than with aid of the tool for question set 1, and more than 8 times slower (823%) for question set 2.

5.2.3 Accuracy

For each question, respondents are asked to provide an improved schedule for the problem (or, for question 4, respondents should pick the optimal schedule from the options given). For questions 1-3 and 5-6, respondents should additionally include the maximum distance or makespan (depending on the question) for their optimised schedule. This is omitted from question 4 for obvious reasons, and omitted from question 7 since it utilises a combination of distance and time (making the question somewhat ambiguous).

The accuracy here is calculated as the **number of correct responses out of all users that give a response** (i.e. any questions left blank are not included in this calculation). We make the following assumptions for our accuracy calculations:

1. Schedules are counted as correct if the tool identifies them as feasible, efficient, individually efficient and all fixed decisions (skill and instrument requirements) are met.

2. The maximum cost for any operator is correct (regardless of whether the tool says the schedule given is efficient or not).

3. For questions involving instrument allocation, we assume (where not given) the respondent found a correct instrument allocation, if one exists.

Note that, since our tool can only check single exchange and pairwise exchange properties, some schedules may be recorded as correct, even if their cost values are not.

Table 5.2 shows the average accuracy for each survey group, across all answered schedule questions (not maximum cost). The overall accuracy for all respondents is about two thirds. In particular, the average accuracy for the two groups that answered question set 1 by hand and question set 2 using the tool is 66.6%; in comparison, the average accuracy for the two groups that answered question set 2 by hand and question set 1 using the tool is 69.9%. We can deduce from this that the two question sets are roughly similar in terms of difficulty, making them a suitable comparison for this user study. Note that the large variation between "Alternating" and "Half-then-half" from the second set of user study versions is likely due to

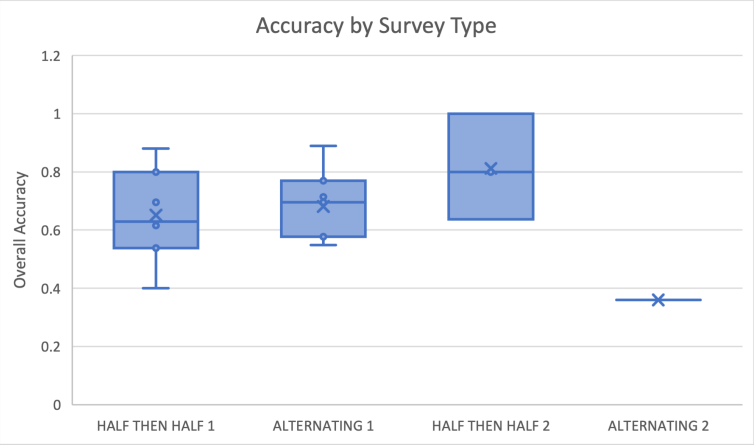


Figure 5.9: Accuracy range of respondents

the low response number distorting the results. For example, the single respondent for "Alternating 2" had the lowest number of correct answers out of all survey respondents.

Figure 5.9 shows the range of accuracy across the respondents for each schedule. Note that Half-then-half 2 and Alternating 2 contain the highest and lowest accuracy scores across all the respondents, respectively. This leads to somewhat skewed results for the corresponding question set by hand and by use of tool, compared to the other survey groups which are reasonably consistent in median and range.

Schedule Accuracy

Figure 5.10 shows the proportion of correct answers for each schedule question, split into respondents who answered by hand versus those who used the tool. We make the following observations:

1. In general, respondents who used the tool were more likely to get a question correct.

2. The anomalies to the above were as follows:

   (a) Q3 (both sets): Choice of Schedule. It is unclear why respondents struggled to use the tool to answer this question – possibly, having multiple schedules to download and compare may have caused some confusion. Indeed, in the case of both questions, the answer was relatively easy to deduce by analysing the three schedules (route optimisation/distance) visually.

   (b) Q6 (question set 2): Distance with Instruments 1. This problem introduced instrument assignments and constraints. In particular, the provided problem led to the tool creating a loop (job F required instruments I1 and I0, but these were allocated to different operators, so moving the job did not help in solving the problem). The correct answer here was to move one of two instruments, so that both instrument requirements for job F were met, then optimise.

   This was the worst answered question out of all those in the user study, with just 31% of respondents giving a correct schedule and only one user providing a corresponding instrument allocation (although this was not clearly specified in the question).

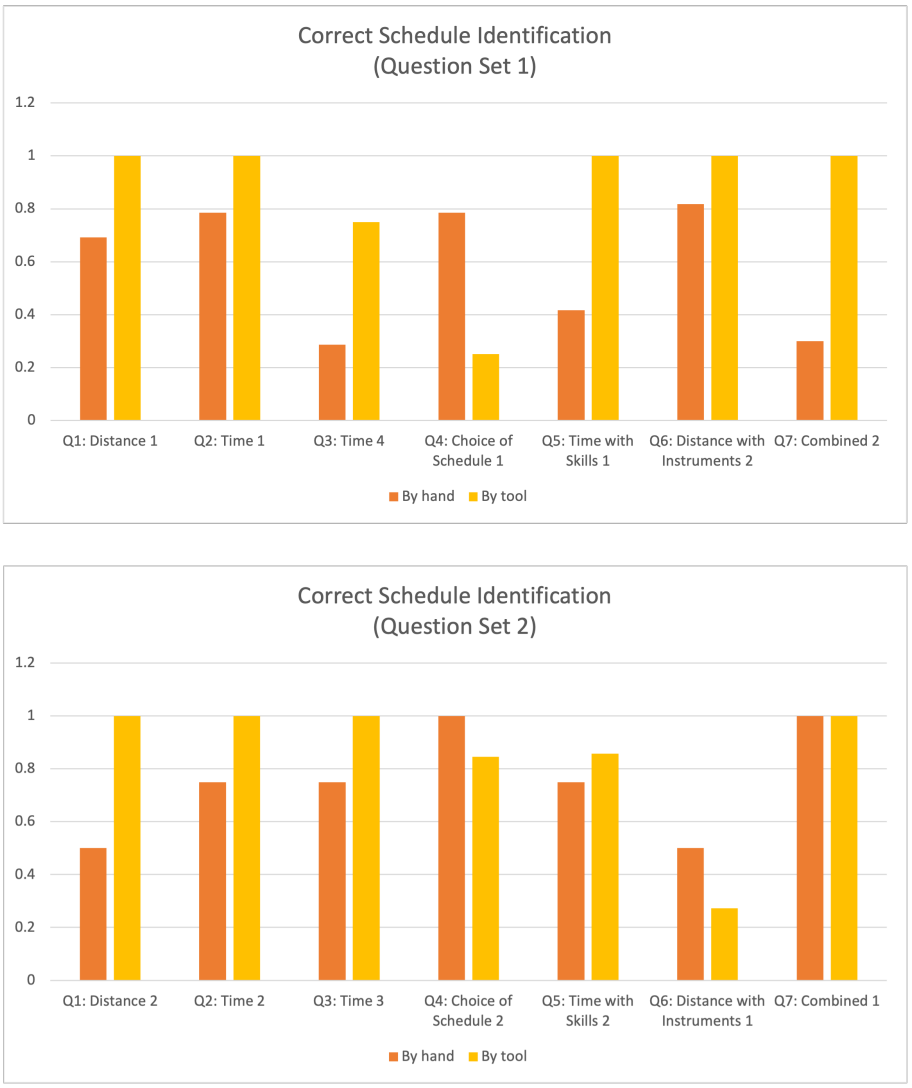


Figure 5.10: Accuracy for surveys by question set (schedule answers)

(c) Q7 (question set 2): Combined 1. It is worth noting that only 2 respondents attempted this question by hand (despite both reaching a correct answer). Thus, the 100% success rate for the question may not fully relay its complexity for those completing the question by hand.

Value Accuracy

Figure 5.11 shows the proportion of correct answers for each question with numerical input (max cost, for distance and processing time problems), split into respondents who answered by hand versus by tool. Note that for these questions, we only accepted the true optimal value (which we judged as a lowest max cost given, where the schedule answer provided passed all tests by the tool and gave the same cost value).

We make the following observations:

1. Results are generally much more variable than the schedule questions. This is likely due to the tool's sub-optimal algorithm, meaning some solutions found by hand were better than solutions immediately found by the tool.

2. Specific question anomalies:

   (a) Q1 (question set 1): Distance 1. Despite the tool giving the KPIs, half of respondents using the tool put "13" for the distance instead of "26" for the distance travelled (essentially not counting the return journey). Regardless, these respondents gave a correct schedule solution.

   (b) Q5 (question set 1): Time with Skills 1. Despite many respondents getting this question correct by hand as well as by tool, comparing to Figure 5.10, we see that most of these respondents gave an incorrect schedule. In the majority of these cases, the users ignored the skill requirements, thus invalidating their schedule (even if the numerical makespan they provided was correct).

   (c) Q3 (question set 2): Time 3. As discussed above, this is a question where the tool recommends a schedule that is sub-optimal. It is worth noting that every respondent here gave a makespan answer which is optimal by the tool's standards.

   (d) Q6 (question set 2): Distance with Instruments 1. As with the schedule response, this question had a low success rate with the tool due to lack of understanding applying changes to the instrument allocation.

3. Still, for the remaining questions, the accuracy rate using the tool was greater than that completing the questions by hand.

5.2.4 Accuracy Breakdown & Format Comparisons

In this section, we define accuracy as the percentage of correct answers across the entire survey; in other words, blank responses are counted as incorrect answers. Here, we only include schedule answers as opposed to max cost answers.

Table 5.3 compares the accuracy of Half-then-half 1 and Alternating 1 (both contain 7 responses each). From this we deduce the following:

1. The success rate using the tool is more than 70% greater when using the tool, compared to computing answers by hand.

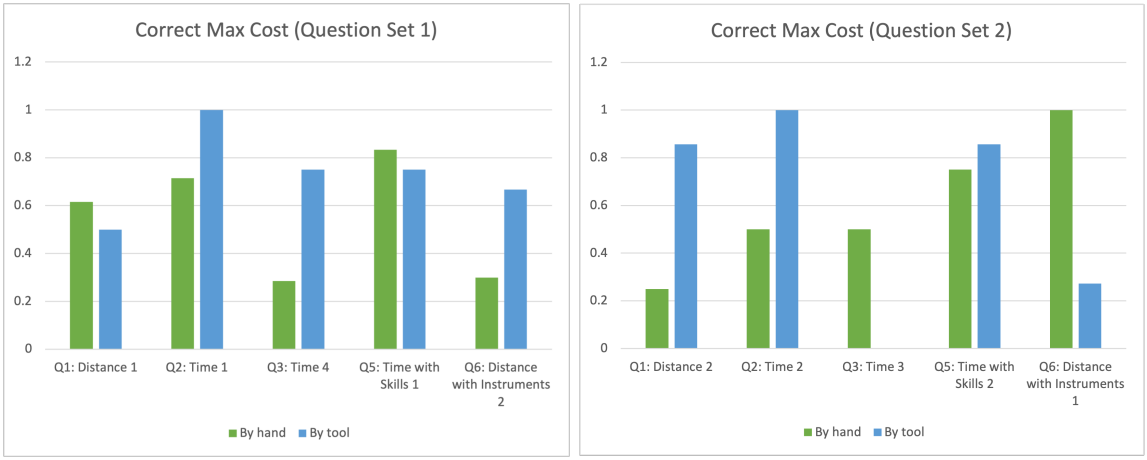


Figure 5.11: Accuracy for surveys by question set (max cost answers)

Answers by hand	**Correct Answers**	**Total Questions**	**Success Rate**
Half-then-half	19	42	45.2%
Alternating	22	42	52.4%
Total	41	84	48.8%

Answers by tool	**Correct Answers**	**Total Questions**	**Success Rate**
Half-then-half	35	42	83.3%
Alternating	36	42	85.7%
Total	71	84	84.5%

Table 5.3: Comparison of alternating question vs half-then-half (all by hand, all by tool) survey format.

2. Performance is marginally (albeit not significantly) better when questions alternate between "by hand" and "by tool", particularly for the former questions. This suggests that using the tool may improve the respondents performance when answering questions by hand.

When we include the results from the remaining survey groups, we find the **average accuracy across all respondents is 46.8% by hand and 73.8% when using the tool**. This is a clear indication that using the tool increases the accuracy of users when deciding improvements in a given schedule.

5.3 Qualitative Evaluation

All respondents were asked for feedback on the usability of the tool at the end of the survey. The results are summarised below:

1. Question: Now you have spent some time getting familiar with the tool, what features do you like?

   Top responses included:

   - Explanations,
   - Easy to apply suggested changes,
   - Simple but clear visualisation/interface.

   Other answers included the flexibility in variable control, and job locations.

2. Question: What would you improve about the tool or its interface?

   The top response was to equalise the X-Y scale for the distance plot to improve readability. Other responses included improving the look of the interface (more dynamic, adding colour to highlight KPIs).

3. Question: Any other comments?

   The majority of respondents left this question blank.

   - One respondent noted that the tool did not always give the most optimal result.
   - Another respondent stated that since the tool only considers the maximum cost of any operator, other operators are sometimes left with too much free time.

Chapter 6

Further Theory

Due to time constraints for the project, certain attributes of the dataset in Section 2.6 could not be implemented into the proof-of-concept tool.

We include here further theory on optimal scheduling using argumentation, which could be used to extend the implementation in Chapter 4. In particular, this chapter discusses adding a time component to the framework, offering another dimension of modelling to the tool. We also present some novel alternative approaches to optimising a priority factor.

6.1 Time Constraints & Extended AFs

We consider an extension to the above framework that accommodates date-time restrictions relating to both job constraints and operator shifts. For example, suppose operator 1 has a scheduled shift from 9:00 to 18:00, but the total processing time of all jobs in its individual schedule is 10 hours – how can we illustrate that this constraint is violated?

We assume in this section that a feasible schedule must exist (i.e. there is a schedule such that all jobs can be completed within the shifts of all operators). Assuming otherwise would impact the feasibility AF from previous chapters, which depends on all jobs being allocated to exactly one operator.

Definition 6.1. *We define λ_j and μ_j as the feasible lower and upper bounds for a job $j \in \mathcal{A}$ to be completed.*

Similarly, we define λ_i and μ_i as the start and end (date-)time for an operator i's shift, for $i \in \mathcal{O}$.

Note this should not be confused with s_i and f_i as defined in Definition 3.10, which refers to the schedule order of jobs rather than job or operator constraints.

When accounting for the start time of jobs in our schedule, we must consider the following:

1. Is it possible for the job to be completed within the operator's shift time (say, a job must begin at 10:00 but the operator's shift only begins at 12:00)?
2. If the above is possible, then are the allocated start times for each job (and end times) within the operator's shift times?
3. Do any jobs overlap in the given schedule?
4. If jobs/operators are not located at the same location, is there sufficient time for the operator to get between jobs and to the correct end location for its shift?

Note that the first point implies the second, so we only need to worry about the last three points.

6.1.1 Interval Scheduling

We first consider the case where there are no distance attributes associated with jobs (i.e. they are all located in the same place and travel does not contribute to the overall time).

Definition 6.2. *Consider a schedule S such that $x_{i,j} = 1$. We define $\tilde{\lambda_{i,j}}$ as the time operator i begins job j, according to schedule S.*

Definition 6.3. *The **schedule bounds AF** $(Args_{D_B}, \rightsquigarrow_{D_B})$ is defined as:*

- $Args_{D_B} = Args_F$,
- $\rightsquigarrow_{D_B} = (\rightsquigarrow_F \cup \{(a_{i,j}, a_{i,j}) : \tilde{\lambda_{i,j}} + p_{i,j} > \mu_i\} \cup \{(a_{i,j}, a_{i,j}) : \tilde{\lambda_{i,j}} < \lambda_i\}$
  $\cup \{(a_{i,j}, a_{i,j}) : \tilde{\lambda_{i,j}} < \lambda_j\} \cup \{(a_{i,j}, a_{i,j}) : \tilde{\lambda_{i,j}} + p_{i,j} > \mu_j\})$.

In other words, the schedule is violating if the assigned start time of a job is outside either the job bounds or corresponding operator's shift times.

Lemma 6.4. *The following are subsets of D^-:*

1. $\{(i,j) \in \mathcal{O} \times \mathcal{A} : \tilde{\lambda_{i,j}} + p_{i,j} > \mu_i\}$,
2. $\{(i,j) \in \mathcal{O} \times \mathcal{A} : \tilde{\lambda_{i,j}} < \lambda_i\}$,
3. $\{(i,j) \in \mathcal{O} \times \mathcal{A} : \tilde{\lambda_{i,j}} < \lambda_j\}$,
4. $\{(i,j) \in \mathcal{O} \times \mathcal{A} : \tilde{\lambda_{i,j}} + p_{i,j} > \mu_j\}$.

Proof. We omit this proof as it follows the same structure as 3.23. □

Definition 6.5. *For bounds $\lambda_i, \mu_i, \lambda_j, \mu_j$ for each $(i,j) \in \mathcal{O} \times \mathcal{A}$ and schedule S with $\tilde{\lambda_{i,j}}$ for $x_{i,j} = 1$, $(Args, \rightsquigarrow) \in \{(Args_F, \rightsquigarrow_F), (Args_{D_B}, \rightsquigarrow_{D_B})\}$, for an attack $a \rightsquigarrow b, a, b \in E$:*

- $(a,b) \in \rightsquigarrow_{D_B} \setminus \rightsquigarrow_F \implies$ *S violates operator shifts and/or job bounds.*

Example 6.1. *Consider a problem consisting of 2 operators and 3 jobs. The processing time of all jobs for each operator is 60 minutes. The shift for operator 1 is 12:00-17:00. The shift for operator 2 is 9:00-14:00.*

Job 1 can be completed at any time; 2 must be completed from 15:00-16:00 and 3 must be completed from 12:00-13:00. Consider the schedule

$$S = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

with $\tilde{\lambda_{1,3}}$ =12:00, $\tilde{\lambda_{1,1}}$ =13:30 and $\tilde{\lambda_{2,2}}$ =15:00.

Figure 6.1 shows the resulting argumentation framework. While job 2 fits within its job bounds, its scheduled time is after operator 2's shift has ended.

Lemma 6.6. *Given a schedule S, the schedule bounds AF can be constructed in $O(n^2m^2)$ time.*

Verifying whether an extension $E \subseteq Args_S$ such that $E \approx S$, is stable can be done in $O(n^2m^2)$ time.

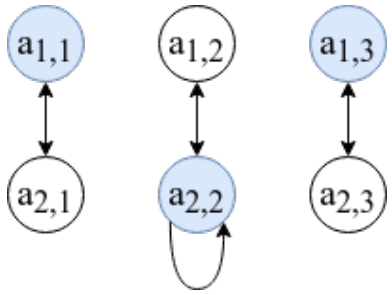


Figure 6.1: Example 6.1 schedule bounds AF

Proof. To get from skills to negative fixed decisions, we must iterate over each job and operator $(j, i) \in \mathcal{A} \times \mathcal{O}$ and check the 4 inequalities as in Definition 6.3. This takes $O(mn)$. There are no positive fixed decisions so we can ignore this array completely, or for every $(i, j) \in \mathcal{A} \times \mathcal{O}$ set the fixed decision as false (taking $O(mn)$ time).

By Lemma 2.13, we have that (from these fixed decisions) the fixed decision AF can be constructed in $O(n^2m^2)$ time. As such we have the overall construction time is $O(n^2m^2)$.

Verification follows as in Lemma 3.24 from Lemma 2.13. □

While the schedule bounds AF can give information as to if the job can be allocated to a given operator or not, it is unable to check the overlap between jobs: e.g. what if operator 1 is assigned a 2-hour job at 15:00, but has another job allocated at 16:00?

Definition 6.7. *Consider a schedule S with schedule order σ_i for each $i \in \mathcal{O}$. Recall j^+ as in Definition 3.13.*

*The **overlap bounds AF** $(Args_{D_O}, \rightsquigarrow_{D_O})$ is defined as:*

- $Args_{D_O} = Args_F$,
- $\rightsquigarrow_{D_O} = (\rightsquigarrow_F \cup \{(a_{i,j}, a_{i,j^+}) : \tilde{\lambda_{i,j}} + p_{i,j} > \tilde{\lambda_{i,j^+}}\})$.

In other words, if the next job in an individual schedule order begins before the previous ends, this violates our overlap constraint.

Lemma 6.8. $\{(i, j) \in \mathcal{O} \times \mathcal{A} : \tilde{\lambda_{i,j}} + p_{i,j} > \tilde{\lambda_{i,j^+}}\} \subseteq D^-$

Proof. We omit this proof as it follows the same structure as 3.23. □

Definition 6.9. *For schedule S with schedule order $\sigma_i, i \in \mathcal{O}$ and $\tilde{\lambda_{i,j}}$ for each $x_{i,j} = 1$, $(Args, \rightsquigarrow) \in \{(Args_F, \rightsquigarrow_F), (Args_{D_B}, \rightsquigarrow_{D_B}), (Args_{D_O}, \rightsquigarrow_{D_O})\}$, for an attack $a \rightsquigarrow b, a, b \in E$:*

- $(a, b) \in \rightsquigarrow_{D_O} \setminus (\rightsquigarrow_F \cup \rightsquigarrow_{D_B}) \implies$ *S has an overlap of jobs within the schedule.*

We only consider set of attacks $\rightsquigarrow_{D_O} \setminus (\rightsquigarrow_F \cup \rightsquigarrow_{D_B})$ since an overlap in jobs does not make sense unless the start times of jobs are feasible within the individual operator schedules (similarly to Section 2.3 when an inefficient schedule does not make sense unless the schedule is at least feasible).

Lemma 6.10. *Given a schedule S, the overlap bounds AF can be constructed in $O(n^2m^2)$ time.*

Verifying whether an extension $E \subseteq Args_S$ such that $E \approx S$, is stable can be done in $O(n^2m^2)$ time.

Proof. As in Lemma 6.6. □

6.1.2 Interval Scheduling with Distance

Definition 6.11. *We define v as the estimated speed of travel between jobs.*

Here, v acts as a simple estimate in our model to translate distance metrics into time, for the purpose of calculating the entire length of an operator's shift. While it may be possible to input real travel time data (indeed, this exists in *Terranova*[4] output schedules, shown in Section 2.6), this would make any later checks impossible if we were to implement suggested improvements to our schedule.

We update the AFs outlined in Subsection 6.1.1 to accommodate these distance parameters.

Definition 6.12. *The **schedule bounds with distance AF** $(Args_{D_{B+}}, \rightsquigarrow_{D_{B+}})$ is defined as:*

- $Args_{D_{B+}} = Args_F$,
- $\rightsquigarrow_{D_{B+}} = (\rightsquigarrow_F \cup \{(a_{i,j}, a_{i,j}) : \tilde{\lambda_{i,j}} + p_{i,j} > \mu_i - v \times \delta(j, f_i)\}$
  $\cup \{(a_{i,j}, a_{i,j}) : \tilde{\lambda_{i,j}} < \lambda_i + v \times \delta(s_i, j)\}$
  $\cup \{(a_{i,j}, a_{i,j}) : \tilde{\lambda_{i,j}} < \lambda_j\} \cup \{(a_{i,j}, a_{i,j}) : \tilde{\lambda_{i,j}} + p_{i,j} > \mu_j\})$.

In other words, a job cannot be scheduled before the operator has sufficient time to travel from its starting position, to where the job takes place (and similar for its finishing position).

Definition 6.13. *The **overlap bounds AF with distance** $(Args_{D_{O+}}, \rightsquigarrow_{D_{O+}})$ is defined as:*

- $Args_{D_{O+}} = Args_F$,
- $\rightsquigarrow_{D_{O+}} = (\rightsquigarrow_F \cup \{(a_{i,j}, a_{i,j^+}) : \tilde{\lambda_{i,j}} + p_{i,j} + v \times \delta(j, j^+) > \tilde{\lambda_{i,j^+}}\})$.

Lemma 6.4, 6.6, 6.8 and 6.10 can be extended similarly to accommodate for distance. These are omitted here for conciseness but follow the same format and proof method.

Definition 6.14. *For bounds $\lambda_i, \mu_i, \lambda_j, \mu_j$ for each $(i, j) \in \mathcal{O} \times \mathcal{A}$ and schedule S with $\tilde{\lambda_{i,j}}$ for $x_{i,j} = 1$, $(Args, \rightsquigarrow) \in \{(Args_F, \rightsquigarrow_F), (Args_{D_{B+}}, \rightsquigarrow_{D_{B+}}), (Args_{D_{O+}}, \rightsquigarrow_{D_{O+}})\}$, for an attack $a \rightsquigarrow b, a, b \in E$:*

- $(a, b) \in \rightsquigarrow_{D_{B+}} \setminus \rightsquigarrow_F \implies$ *S violates operator shifts and/or job bounds.*
- $(a, b) \in \rightsquigarrow_{D_{O+}} \setminus (\rightsquigarrow_F \cup \rightsquigarrow_{D_{B+}}) \implies$ *S has an overlap of jobs within the schedule (including travel time).*

Example 6.2. *Continuing Example 6.1, suppose jobs $1 - 3$ are located at the following coordinates:*

1. $(3, 4, 0)$
2. $(12, 5, 0)$
3. $(5, 12, 0)$

with both operators beginning and ending their shifts at the origin $(0, 0, 0)$. Setting $v = 5$ distance units/minute, our new schedule bounds with distance AF can be shown in Figure 6.2.

We have $a_{2,2} \rightsquigarrow_{D_{B+}} a_{2,2}$ for the same reason as Example 6.2. Since we must now consider the travel time for jobs, $\lambda_{1,3} + 5 \times \delta(s_1, 2)$ =12:00+1:05=13:05> $\tilde{\lambda_{1,3}}$ so $a_{1,3} \rightsquigarrow_{D_{B+}} a_{1,3}$. Similarly, the time between jobs 1 and 3 are no longer sufficient for travel, so we also get $a_{1,1} \rightsquigarrow_{D_{B+}} a_{1,1}$.

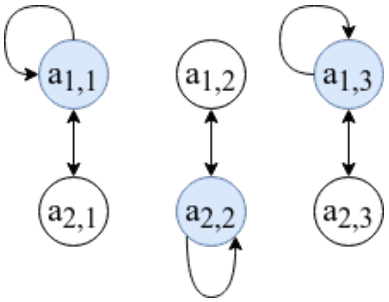


Figure 6.2: Example 6.2 schedule bounds with distance AF

6.1.3 Prerequisite Job Requirements

Suppose we have a job j' which is a prerequisite to j, such that the start time of j must be after the completion of j', even if j and j' are assigned to different operators.

Definition 6.15. *We define $\mathcal{Z}_j$ as the set of prerequisite jobs for a job $j \in \mathcal{A}$.*

Definition 6.16. *The **prerequisite job AF** $(Args_{D_Z}, \rightsquigarrow_{D_Z})$ is defined as:*

- $Args_{D_Z} = Args_F$,
- $\rightsquigarrow_{D_Z} = (\rightsquigarrow_F \cup \{(a_{i,j'}, a_{i,j}) : \tilde{\lambda_{i,j}} < \tilde{\lambda_{i',j'}} + p_{i',j'}, j' \in \mathcal{Z}_j\})$,

where $x_{i,j} = x_{i',j'} = 1$ and i, i' may or may not be equal.

Note that the prerequisite job AF is independent of the distance between jobs, since j and j' may be completed by different operators.

Lemma 6.17. $\{(i,j) \in \mathcal{O} \times \mathcal{A} : \tilde{\lambda_{i,j}} < \tilde{\lambda_{i',j'}} + p_{i',j'}, j' \in \mathcal{Z}_j\} \subseteq D^-$

Proof. Again, we omit this proof as it follows the same structure as 3.23. □

Definition 6.18. *For schedule S and $\tilde{\lambda_{i,j}}$ for each $x_{i,j} = 1$, $(Args, \rightsquigarrow) \in \{(Args_F, \rightsquigarrow_F), (Args_{D_Z}, \rightsquigarrow_{D_Z})\}$, for an attack $a \rightsquigarrow b, a, b \in E$:*

- $(a,b) \in \rightsquigarrow_{D_Z} \setminus \rightsquigarrow_F \implies$ *S violates prerequisite job requirements.*

6.2 Schedule-Dependant Individual Priority

To round off our further theory in extending argumentation frameworks for optimal scheduling, we return to the priority variable $\boldsymbol{q}$, as introduced in Chapter 3. Recall that $\boldsymbol{q}$ is a vector where $q_j = q \in \{1,2,3,4,5\}$, representing the level of priority of job j (with 1 being highest priority and 5 the least prioritised). As explored in Section 2.4, much of prior implementation of priority in scheduling algorithms have been related to a job completion deadline. We focus on some novel explorations in this section as to how a numerical-value priority can ultimately be interpreted by our argumentation framework.

Recall from Definition 3.4 that individual cost $c_{i,j} = \alpha q_j + \beta p_{i,j}$. Importantly, the individual cost can be seen as the non-schedule-dependent part of the overall operator cost C_i. However, when we consider how we prioritise tasks, often the assignment of jobs does play a part; in this section, we look briefly at two different approaches to calculating priority, and how this impacts the efficiency AF introduced in Definition 3.17.

6.2.1 Variance of Priority

Suppose we wanted to ensure the jobs are evenly distributed, such that all operators are assigned roughly an equal set of high priority and low priority. We can use the expected value E to favour schedules with a more even distribution of prioritised jobs.

Definition 6.19. *Let $\tilde{q}$ be the mean average for all jobs $j \in \mathcal{A}$.*

*The **extended cost** C_i **with priority variance and schedule ordering** σ_i is defined as*

$$C_i = \alpha(\mathrm{E}_{x_{i,j}=1}[q_j] - \tilde{q})^2 + \beta \sum_{j=1}^{N} x_{i,j} p_{i,j} + \gamma \left(\delta(s_i, \sigma_i[1]) + [\sum_{k=1}^{k_i - 1} \delta(\sigma_i[k], \sigma_i[k+1])] + \delta(\sigma_i[k_i], f_i) \right)$$

where E *is the mean average (or expected value) and δ is the distance metric.*

The purpose of $(\mathrm{E}_{x_{i,j}=1}[q_j] - \tilde{q})^2$ is to calculate how far from the expected mean priority each operator's assigned set of jobs are. If we have a distribution of jobs such that multiple low-priority jobs are assigned to one operator, and multiple high priority to another, then this will be reflected with a higher value of $(\mathrm{E}_{x_{i,j}=1}[q_j] - \tilde{q})^2$.

Example 6.3. *Consider 3 jobs with priorities $1, 3, 5$ respectively. Hence $\tilde{q} = \mathrm{E}[\{1, 3, 5\}] = 9/3 = 3$. Suppose we have schedule*

$$S = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Then for operator 1, $(\mathrm{E}_{x_{1,j}=1}[q_j] - \tilde{q})^2 = (2-3)^2 = 1$ and for operator 2, $(\mathrm{E}_{x_{2,j}=1}[q_j] - \tilde{q})^2 = (5-3)^2 = 4$.

On the other hand, for

$$S' = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix},$$

we have $(\mathrm{E}_{x_{1,j}=1}[q_j] - \tilde{q})^2 = (\mathrm{E}_{x_{2,j}=1}[q_j] - \tilde{q})^2 = (3-3)^2 = 0$.

6.2.2 Priority in Schedule Order

An alternative approach to priority might be to penalise schedule orders that place high-priority jobs later in the schedule.

Definition 6.20. *The **extended cost** C_i **with priority ordering and schedule ordering** σ_i is defined as*

$$C_i = \alpha \sum_{k=1}^{k_i} k(6 - q_{\sigma[k]}) + \beta \sum_{j=1}^{N} x_{i,j} p_{i,j} + \gamma \left(\delta(s_i, \sigma_i[1]) + [\sum_{k=1}^{k_i - 1} \delta(\sigma_i[k], \sigma_i[k+1])] + \delta(\sigma_i[k_i], f_i) \right)$$

where $q_{\sigma[k]}$ is the priority for job $\sigma_i[k]$ (i is omitted for tidiness) and δ is the distance metric. k_i is the number of jobs assigned to operator i in schedule S.

Note that since a lower priority is indicated by a higher value of q (and we want to minimise the overall cost of the schedule), we transform $q'_j = 6 - q_j$ so high priority jobs are penalised more heavily for later scheduling. The individual order σ_i is crucial here in calculating the priority cost value: the (transformed) priority of each job is multiplied by its place in the operator i's individual order.

Remark 6.21. *For a heavier penalisation, consider instead the exponential penalty*

$$\sum_{k=1}^{k_i} 2^k (6 - q_{\sigma[k]}).$$

Chapter 7

Conclusions

In this project, we have considered the current scope of optimisation problems and the sub-optimal algorithms that are used to solve such problems (Chapter 2). We have utilised previous argumentation applications and extended them to a more complex problem with direct real-world applications provided by a company, theorising new argumentation frameworks which allow us to model specific scenarios relating to route optimisation, broader optimisation criteria and specific cases of fixed decisions (Chapters 3).

We then modelled a subset of this theory as a proof-of-concept tool, allowing us to test this tool against both toy examples and real examples provided by the company (Chapter 4). We evaluated the tool against these real examples and conducted a user study to obtain both quantitative and qualitative results regarding the usability and accuracy of the tool (Chapter 5). Finally, we considered some further theory to fully model problems according to the company ask (interval scheduling), along with considerations of some improvements to our previously defined theory (priority variance) (Chapter 6).

Recall our objectives as outlined in Section 1.2 along with our corresponding contributions:

1. To extend the scheduling theory of Čyras et al.[5] for the problem case provided by *Terranova* (Chapters 3 and 6):
    - We provided extensive theory, new frameworks and time complexity proofs to map the *Terranova* problem as closely as possible.
    - Waste disposal variables (e.g. instrument capacity, compression ratio) were omitted, due to the large difference between the workforce management problem and the waste disposal/transport problem. Waste disposal was considered beyond the scope of the project.
    - Full theory was provided for all workforce management-specific aspects of the problem.
2. To extend the original tool designed by Karamlou[6] to accommodate the extended argumentation theory for our optimisation problem (accessible via GitLab, user guide can be found here):
    - All theory from Chapter 3 was implemented in the tool.
    - The tool was tested by multiple users and described by user study respondents as "easy to use".
    - Due to time constraints, theory from Chapter 6 (i.e. time components) was omitted from the tool.

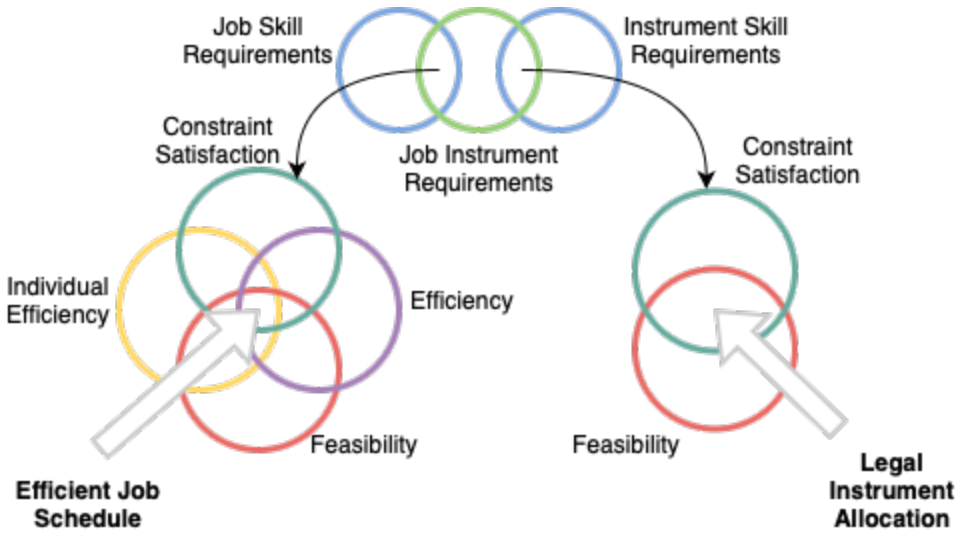


Figure 7.1: Properties for job scheduling and instrument allocation

3. To evaluate the overall improvement the tool can provide to users aiming to improve a near-optimal schedule (Chapter 5):

    - Using the tool was found to be (on average) 9 times faster than computing schedule improvements by hand.
    - The tool was found to improve accuracy of users' response rate by 57% (see Subsection 5.2.4) when completing the user study.

7.1 Limitations of Argumentation in Scheduling

Figure 7.1 gives a visual representation of the framework developed in Chapter 3 and implemented in Chapter 4. In particular, the constraint satisfaction for both job scheduling and instrument allocation is made up of many argumentation frameworks, including an overlap for job-instrument requirements.

It is worth noting that our tool links efficiency and the constraint satisfaction requirements (or fixed user decisions) by defining fixed-decision aware exchange properties. In other words, the efficiency AF will not suggest any changes that violate fixed decisions. While this is intuitive for most of these "hard" constraints, we ran into problems when we have "soft" constraints like the job instrument allocation (where instruments can move instead of jobs and vice versa). This meant our tool only finds efficient schedules for whatever the current instrument allocation is.

A similar limitation is present when it comes to the efficiency and individual efficiency argumentation frameworks: since we are limited to single and pairwise exchange checks, our tool can only search for more optimal schedules in its local area. For example, consider a problem with three jobs and 3 operators, with processing times as in Table 7.1. Suppose we have schedule

$$S = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

	Operator 1	Operator 2	Operator 3
Job A	1	2	4
Job B	4	1	2
Job C	2	4	1

Table 7.1: Processing times for non-optimal schedule example

so every operator takes 2 units of time and our is 2. However, clearly the optimal schedule is

$$S' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

If we try to move a single job (single exchange), we will increase the makespan. Suppose instead we try swapping a pair of jobs (pairwise exchange): if we want to assign job A to operator 1, we can swap this with job C. However, moving job C to operator 2 gives a processing time of 4, so this will also increase the makespan – this is the same for any swap. As such, there are no possible single or pairwise exchange moves to improve the schedule, so our tool cannot find the optimal solution.

This is a major limitation of the tool for schedules that are not close to an optimal solution. However, providing a better optimality criteria is beyond the scope of the current project. It is worth noting that there is currently no truly optimal algorithm to identify an optimal schedule within reasonable computational means.

7.2 Future Work

In no particular order, we suggest possible avenues for future work on the topic:

- **Implementation of interval scheduling (and remaining further theory) in tool**: We have shown that it is possible to apply abstract argumentation to explain interval scheduling problems that enforce job completion at specific times or involve limitations with operator shift lengths. Due to the time constraints of this project, we were unable to implement the full extent of theory with sufficient time for user testing and evaluation.

- **Travelling salesman problem implementation using argumentation**: As discussed briefly in Sections 2.5 and 3.4, the idea of individual efficiency is largely linked to TSP, another NP-hard optimisation problem. In this project, we have utilised ideas from the efficiency AF discussed by Čyras et al.[5] and applied this for the specific individual schedule case. However, this is not equivalent to the 2-opt theory as defined in Definition 2.16.

  The major limitation with current argumentation applications is the lack of order associated with the schedule S. In Definition 3.27 we avoid this problem by using self-attacks to indicate if the individual position of arguments (and thus job assignments in the schedule) is inefficient. However, in the case of a single exchange, this doesn't tell us where might be a more optimal position for the job, meaning we must recurse all options again to check where exactly would be a preferable move for the job (thus the argumentation framework isn't necessarily enough to offer a suggested improvement).

To be able to wholly map TSP to an argumentation framework, one would need to design a system such that the order is tracked and whole sets of arguments can be easily swapped between operators (see Figure 7.2).

With various applications of TSP that require users to improve schedules and routing on the job, argumentation based on similar approaches to those explored in the project should allow users sufficient flexibility to gain natural language explanations and improvements in real time.

- **Assumption based argumentation applications**: Due to time constraints, this project focuses wholly on applying abstract argumentation (AA) to solve the problem provided by *Terranova*. However, due to the less constricted nature of the theory and bounded nature of some constraints in the problem (particularly relating to time), parts of this implementation may be more suited to an assumption-based argumentation (ABA) application[29][30].

7.3 Summary

- Utilising our argumentation tool improves user accuracy by 57% and time efficiency by more than 900%.
- It is possible to model complex, multi-variable optimisation problems using argumentation, however it may not be the best suited method of approach to maximise optimally, nor compute results (time or space-)efficiently.

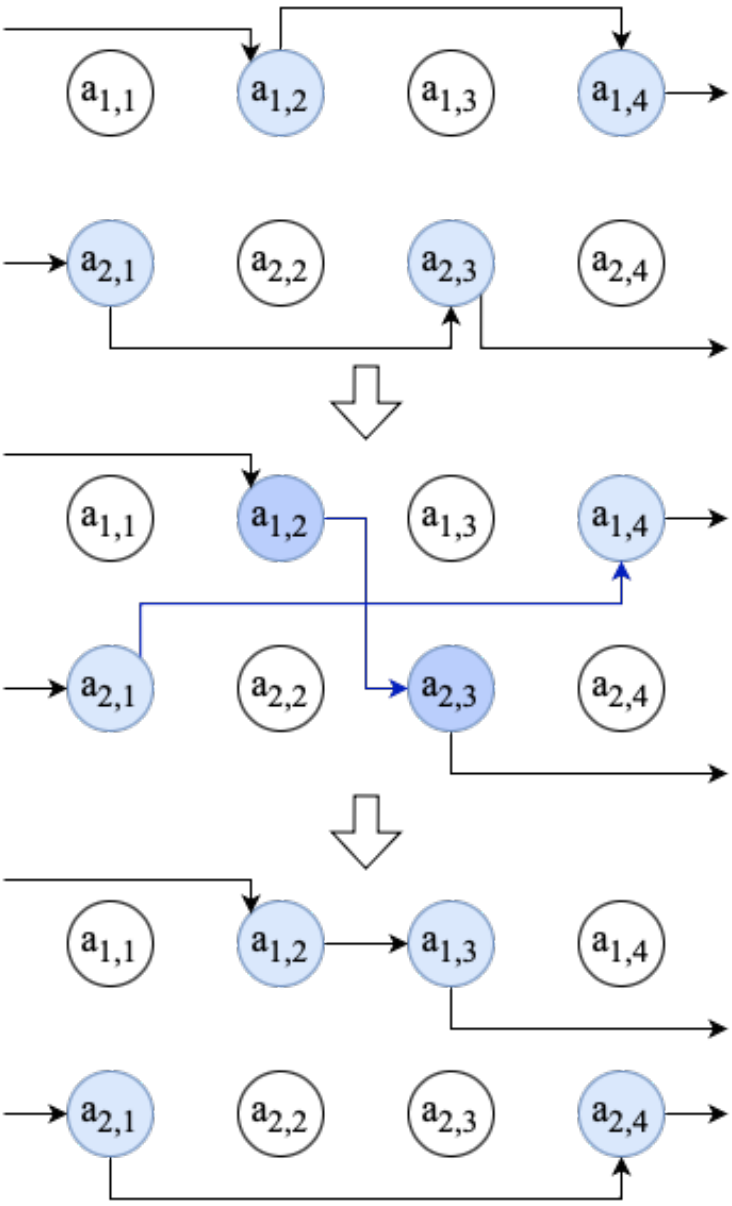


Figure 7.2: TSL swapping process illustrated with arguments and attacks

Bibliography

[1] Warner DM, Prawda J. A mathematical programming model for scheduling nursing personnel in a hospital. Manage Sci. 1972;19:411-22.

[2] Lin F, Shoham Y. Argument systems: A uniform basis for nonmonotonic reasoning. 1st International Conference on Knowledge Representation and Reasoning. 1989;89:245-55.

[3] Dung PM. Negations as Hypotheses: An Abductive Foundation for Logic Programming. In: 8th International Conference on Logic Programming; 1991. p. 3-17.

[4] Terranova. Terranova Smart Network. Terranova; 2023. Accessed 23/8/2023. Available from: https://www.terranovasoftware.eu/en/solutions/terranova-smart-network.

[5] Čyras K, Letsios D, Misener R, Toni F. Argumentation for explainable scheduling. In: 33rd AAAI Conference on Artificial Intelligence. vol. 33; 2019. p. 2752-9.

[6] Karamlou A. Interactive Schedule Explainer for Nurse Rostering. GitHub; 2020. https://github.com/AminKaramlou/AESWebApp.

[7] Vasileiou SL, Yeoh W, Son TC, Kumar A, Cashmore M, Magazzeni D. A logic-based explanation generation framework for classical and hybrid planning problems. J Artif Intell Res. 2022;73:1473-534.

[8] Sukkerd R, Simmons R, Garlan D. Tradeoff-Focused Contrastive Explanation for MDP Planning. In: 29th IEEE International Conference on Robot and Human Interactive Communication; 2020. p. 1041-8.

[9] Graham RL. Bounds on multiprocessing timing anomalies. SIAM J Appl Math. 1969;17(2):416-29.

[10] Dung PM. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif Intell. 1995;77(2):321-57.

[11] García AJ, Chesñevar CI, Rotstein ND, Simari GR. Formalizing dialectical explanation support for argument-based reasoning in knowledge-based systems. Expert Syst Appl. 2013;40(8):3233-47.

[12] Fan X, Toni F. On computing explanations in argumentation. In: 29th AAAI Conference on Artificial Intelligence; 2015. p. 1496-502.

[13] Ulbricht M, Wallner JP. Strong explanations in abstract argumentation. In: 35th AAAI Conference on Artificial Intelligence; 2021. p. 6496-504.

[14] Bench-Capon TJM, Dunne PE. Argumentation in artificial intelligence. Artif Intell. 2007;171(10):619-41.

[15] Ludwig J, Kalton A, Stottler R. Explaining Complex Scheduling Decisions. In: IUI Workshops; 2018. .

[16] Gatta VL, Moscato V, Postiglione M, Sperlì G. CASTLE: Cluster-aided space transformation for local explanations. Expert Syst Appl. 2021;179:115045.

[17] Coffman EG, Denning PJ. Operating Systems Theory. vol. 973. Prentice-Hall Englewood Cliffs, NJ; 1973.

[18] Audsley NC, Burns A, Davis RI, Tindell KW, Wellings AJ. Fixed priority pre-emptive scheduling: An historical perspective. Real-Time Syst. 1995;8(2-3):173-98.

[19] Pazzaglia P, Biondi A, Natale MD. Simple and General Methods for Fixed-Priority Schedulability in Optimization Problems. In: International Conference on Design, Automation & Test in Europe; 2019. p. 1543-8.

[20] Liu C, Layland J. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. J ACM. 1973;20(1):46-61.

[21] Jin X, Yu L. Research and implementation of high priority scheduling algorithm based on intelligent storage of power materials. Energy Rep. 2022;8:398-405.

[22] Gutin G, Punnen AP. The traveling salesman problem and its variations. Kluwer Academic Publishers; 2002.

[23] Finke G, Claus A, Gunn E. A two-commodity network flow approach to the traveling salesman problem. Congress Num. 1984 01;41:167-78.

[24] Padberg M, Sung TY. An analytical comparison of different formulations of the travelling salesman problem. Math Program. 1991;52(1-3):315-57.

[25] Lenstra JK, Kan AHGR. Some Simple Applications of the Travelling Salesman Problem. Oper Res Q. 1975;26(4):717-33.

[26] Little JDC, Murty KG, Sweeney DW, Karel C. An Algorithm for the Traveling Salesman Problem. Oper Res. 1963;11(6):972-89.

[27] Lin S. Computer solutions of the traveling salesman problem. Bell Syst Tech J. 1965;44(10):2245-69.

[28] Croes GA. A Method for Solving Traveling-Salesman Problems. Oper Res. 1958;6(6):791-812.

[29] Čyras K, Letsios D, Misener R, Toni F. Explainable Interactive Scheduling via Assumption-Based Argumentation with Preferences; N.D. Unpublished.

[30] Čyras K, Toni F. ABA : Assumption-Based Argumentation with Preferences. arXiv preprint arXiv:161003024. 2016.

Appendix A

Terranova JSON Input Format

Below is the full content of `JSON_REQUEST_DEF.json` provided by *Terranova*, which illustrates the format of their datasets.

```
{
    "request_id": 4269189,
    "company_id": "00E2",
    "command": "OPTIMIZE",
    "application": "WFM",                /* Caller ID: WFM, AMBIENTE */
    "algorithm_id": "ALGCW",             /* Algorithm to be used (ID) */
    "startdate": "2010-07-12",           /* Starting date for the scheduling
    */
    "numdays": 1,                        /* Days to schedule (for now always
    set to 1) */
    "back_to_deposit_period": 1,         /* Days before going back to deposit;
     values between 1 and 5 (for now always set to 1) */
    "include_routes": 1,                 /* If 1 we need the detailed routing
    */
    "distribution_type" : "uniform",     /* "uniform" = equally distributes
    the activities through the operators, "minimize" = minimise the number of
     required operators */
    "unbalance_parameters": {            /* Penalty parameters (used only in
    distribution_type = uniform case) */
        "distance": null,        /* 0-100: how much we penalise long distances
     (higher means more penalty) (for now always null)*/
        "time": 20,              /* 0-100: how much we penalise the long tasks
     for the operators  (higher means more penalty) (for now always null) */
        "priority": {            /* 0-100: Regulate the proprity weights */
            "1": 100,
            "2": 75,
            "3": 50,
            "4": 25,
            "5": 0
        }
    },
    "locations": [
        {
            "location_id": "LOC0000001",
            "location_type": "DEP",                 /*DEP = Deposit, ULP =
    Unloading point (only for AMBIENTE) */
            "X": 11.1028363,                         /* If SRID = WGS84, X =
    Longitude, Y = Latitude in degrees with up to 6 decimals */
            "Y": 42.7856349,
            "Z": null,
            "SRID": "WGS84",
```

```
31          "capacity": 999999999,                  /* capacity in liters,
    null means NA */
32          "compression_ratio": 0.01               /* (0-1] for example 0.3
    means used capacity = capacity*0.3, null means NA */
33      },
34      {
35          "location_id": "ULP0000001",
36          "location_type": "ULP",             /*DEP = Deposit, ULP =
    Unloading point */
37          "X": 11.8856349,
38          "Y": 42.1038363,
39          "Z": null,
40          "SRID": "WGS84",
41          "capacity": 10000,                  /* capacity in liters, null
    otherwise */
42          "compression_ratio": 0.5                /* (0-1] for example 0.3
    means used capacity = capacity*0.3 */
43      },
44      {
45          "location_id": "ULP0000002",
46          "location_type": "ULP",             /*DEP = Deposit, ULP =
    Unloading point */
47          "X": 11.3856349,
48          "Y": 42.1128363,
49          "Z": null,
50          "SRID": "WGS84",
51          "capacity": 10000,                  /* capacity in liters, null
    otherwise */
52          "compression_ratio": 0.5                /* (0-1] for example 0.3
    means used capacity = capacity*0.3 */
53      }
54  ],
55  "activities": [                 /* 1 or more items */
56      {
57          "activity_id": "WFM00000001",   /* external unique id */
58          "X": 11.7856349,
59          "Y": 42.1028363,
60          "Z": null,
61          "SRID": "WGS84",
62          "duration": {
63              "default": 600, /* default duration (sec) that applies to the
     operators not listed in "specific". If null, all the operators must be
    listed */
64              "specific": [
65                  {
66                      "operator_id": "00000123",
67                      "value": 300    /* Specific duration related to the
    operator */
68                  },
69                  {
70                      "operator_id": "00000323",
71                      "value": 480
72                  }
73              ]
74          },
75          "activity_type": "MeterInstallation",       /* string identifying
     the activity type */
76          "start_timestamp" : "2010-07-12 08:30:00",      /* First possible
     time (after ...) */
```

```
77          "end_timestamp" : "2010-07-12 08:50:00",        /* Last possible
    time (before ...) */
78          "priority": 3,                                  /* [1 - 5], 1 =
    max priority */
79          "must_start_after": null,                       /* ID code of the
     activity that is required before this one (if any) */
80          "needed_capacity": 300,                         /* Possible
    required capacity */
81          "needed_skills" : [             /* 0 or more items */
82              "BLAB"
83          ],
84          "needed_instruments" : [            /* 0 or more items */
85              "VEH",
86              "MAZ"
87          ]
88      },
89      {
90          "activity_id": "WFM00000002",   /* external unique id */
91          "X": 11.7856349,
92          "Y": 42.1028363,
93          "Z": null,
94          "SRID": "WGS84",
95          "duration": {
96              "default": 300, /* default duration (sec) that applies to the
     operators not listed in "specific". If null, all the operators must be
    listed */
97              "specific": null
98          },
99          "activity_type": "MeterReplacement",        /* string identifying
     the activity type */
100         "start_timestamp" : null,                   /* First possible
    time (after ...) */
101         "end_timestamp" : null,                     /* Last possible time
     (before ...) */
102         "priority": 5,                                  /* [1 - 5], 1 =
    max priority */
103         "must_start_after": "WFM00000001",              /* ID code of the
     activity that is required before this one (if any) */
104         "needed_capacity": null,                         /* Possible
    required capacity */
105         "needed_skills" : [             /* 0 or more items */
106             "XFRE"
107         ],
108         "needed_instruments" : [            /* 0 or more items */
109         ]
110     },
111     {
112         "activity_id": "WFM00000003",   /* external unique id */
113         "X": 11.7856349,
114         "Y": 42.1028363,
115         "Z": null,
116         "SRID": "WGS84",
117         "duration": {
118             "default": 900, /* default duration (sec) that applies to the
     operators not listed in "specific". If null, all the operators must be
    listed */
119             "specific": [
120                 {
121                     "operator_id": "00000123",
```

```
122                         "value": 300    /* Specific duration related to the
    operator */
123                     }
124                 ]
125             },
126             "activity_type": "MeterDiagnosis",       /* string identifying
    the activity type */
127             "start_timestamp" : null,               /* First possible time (
    after ...) */
128             "end_timestamp" : null,                 /* Last possible time (
    before ...) */
129             "priority": 5,                                  /* [1 - 5], 1 =
    max priority */
130             "must_start_after": null,                       /* ID code of the
     activity that is required before this one (if any) */
131             "needed_capacity": null,                        /* Possible
    required capacity */
132             "needed_skills" : [             /* 0 or more items */
133                 "XMEN"
134             ],
135             "needed_instruments" : [             /* 0 or more items */
136             ]
137         }
138     ],
139     "operators": [                 /* 1 or more items */
140         {
141             "operator_id": "00000123",
142             "start_position": {                 /* Starting point (it can be
    one of the locations listed or another place). It cannot be null */
143                 "location_id": "LOC0000001",
144                 "X": null,
145                 "Y": null,
146                 "Z": null,
147                 "SRID": null
148             },
149             "returning_position": {       /* Ending point (it can be one of
    the locations listed or another place). It cannot be null  */
150                 "location_id": "LOC0000001",
151                 "X": null,
152                 "Y": null,
153                 "Z": null,
154                 "SRID": null
155             },
156             "skills" : [            /* 0 or more items */
157                 "PATA",
158                 "PESX",
159                 "BLAB",
160                 "XFRE"
161             ],
162             "working_shift" : [     /* it includes holidays and leave */
163                 {
164                     "start_timestamp" : "2010-07-12 08:00:00",
165                     "end_timestamp" : "2010-07-12 12:00:00"
166                 },
167                 {
168                     "start_timestamp" : "2010-07-12 13:00:00",
169                     "end_timestamp" : "2010-07-12 17:00:00"
170                 }
171             ]
172         },
```

```
        {
            "operator_id": "00000323",
            "start_position": {                 /* Starting point (it can be
    one of the locations listed or another place). It cannot be null */
                "location_id": "LOC0000001",
                "X": null,
                "Y": null,
                "Z": null,
                "SRID": null
            },
            "returning_position": {       /* Ending point (it can be one of
    the locations listed or another place). It cannot be null  */
                "location_id": null,
                "X": 11.7856349,
                "Y": 42.1028363,
                "Z": null,
                "SRID": "WGS84"
            },
            "skills" : [            /* 0 or more items */
                "PATA"
            ],
            "working_shift" : [     /* it includes holidays and leave */
                {
                    "start_timestamp" : "2010-07-12 08:00:00",
                    "end_timestamp" : "2010-07-12 12:00:00"
                },
                {
                    "start_timestamp" : "2010-07-12 13:00:00",
                    "end_timestamp" : "2010-07-12 15:00:00"
                }
            ]
        }
    ],
    "instruments": [                /* 1 or more items */
        {
            "instrument_id": "000002131",
            "instrument_type": "VEH",       /* "VE*" = Vehicle. In this case
    it needs the field capacity. Otherwise another 3-characters code */
            "capacity": 30000,              /* capacity in liters (if VEH),
    null otherwise */
            "needed_skills" : [             /* 0 or more items */
                "PATA",
                "PESX"
            ]
        },
        {
            "instrument_id": "000002141",
            "instrument_type": "MAZ",       /* "VE*" = Vehicle. In this case
    it needs the field capacity. Otherwise another 3-characters code */
            "capacity": null,              /* capacity in liters (if VEH),
    null otherwise */
            "needed_skills" : [             /* 0 or more items */
            ]
        },
        {
            "instrument_id": "000002241",
            "instrument_type": "FEV",       /* "VE*" = Vehicle. In this case
    it needs the field capacity. Otherwise another 3-characters code */
            "capacity": null,              /* capacity in liters (if VEH),
    null otherwise */
```

```
            "needed_skills" : [             /* 0 or more items */
                "DRFE"
            ]
        }
    ]
}
```

Appendix B

Tool Outputs for Company-Provided Data

B.1 City

B.1.1 Text Output

The following text output is for weighting $\alpha = 0, \beta = \gamma = 1$. In other words, the optimisation criteria is equally weighted between processing time and distance.

```
Max priority: 8.0 unit(s)
Max time taken: 3600.0 unit(s)
Max distance: 0.58 unit(s)
-
Job allocation to schedule is not feasible
 -  Job F is not allocated to any operator
 -  Job G is not allocated to any operator
 -  Job P is not allocated to any operator
 -  Job AK is not allocated to any operator
 -  Job BB is not allocated to any operator
Schedule does satisfy job skill constraints
 -  All jobs satisfy all skill constraints
Schedule does satisfy job instrument constraints
 -  All jobs satisfy all skill constraints
Schedule is efficient (for this instrument allocation)
 -  All jobs satisfy single and pairwise exchange properties
Individual schedule(s) are efficient
 -  All jobs satisfy single and pairwise exchange properties;
Instrument allocation is feasible
 -  There are no instruments
Schedule does satisfy job instrument constraints
 -  All instruments can be used by their assigned operators
Schedule does satisfy instrument skill constraints
 -  All instruments can be used by their assigned operators
```

B.1.2 Text Output for Priority Optimisation

The following text output is for weighting $\alpha = 1, \beta = \gamma = 0$. In other words, the only variable that is optimised is the priority.

```
Max priority: 8.0 unit(s)
Max time taken: 3600.0 unit(s)
Max distance: 0.58 unit(s)
-
Job allocation to schedule is not feasible
 -  Job F is not allocated to any operator
 -  Job G is not allocated to any operator
 -  Job P is not allocated to any operator
 -  Job AK is not allocated to any operator
 -  Job BB is not allocated to any operator
Schedule does satisfy job skill constraints
 -  All jobs satisfy all skill constraints
Schedule does satisfy job instrument constraints
 -  All jobs satisfy all skill constraints
Schedule is not efficient (for this instrument allocation)
 -  Job AH can be allocated from operator 7 to 1 at start to reduce by 1.0
 -  Job AH can be allocated from operator 7 to 2 after job W to reduce by 1.0
 -  Job AH can be allocated from operator 7 to 2 at start to reduce by 1.0
 -  Jobs AH and B can be swapped with operators 7 and 11 to reduce by 1.0
 -  Job AH can be allocated from operator 7 to 3 at start to reduce by 1.0
 -  Job AH can be allocated from operator 7 to 4 at start to reduce by 1.0
 -  Jobs AH and E can be swapped with operators 7 and 11 to reduce by 1.0
 -  Job AH can be allocated from operator 7 to 6 at start to reduce by 1.0
 -  Jobs AH and I can be swapped with operators 7 and 12 to reduce by 1.0
 -  Jobs AH and J can be swapped with operators 7 and 16 to reduce by 1.0
 -  Job AH can be allocated from operator 7 to 11 at start to reduce by 1.0
 -  Job AH can be allocated from operator 7 to 12 at start to reduce by 1.0
 -  Jobs AH and L can be swapped with operators 7 and 27 to reduce by 1.0
 -  Job AH can be allocated from operator 7 to 13 at start to reduce by 1.0
 -  Job AH can be allocated from operator 7 to 14 at start to reduce by 1.0
 -  Job AH can be allocated from operator 7 to 15 at start to reduce by 1.0
 -  Job AH can be allocated from operator 7 to 17 at start to reduce by 1.0
 -  Jobs AH and R can be swapped with operators 7 and 5 to reduce by 1.0
 -  Job AH can be allocated from operator 7 to 21 at start to reduce by 1.0
 -  Jobs AH and V can be swapped with operators 7 and 10 to reduce by 1.0
 -  Jobs AH and W can be swapped with operators 7 and 2 to reduce by 1.0
 -  Jobs AH and X can be swapped with operators 7 and 14 to reduce by 1.0
 -  Jobs AH and Y can be swapped with operators 7 and 4 to reduce by 1.0
 -  Job AH can be allocated from operator 7 to 26 at start to reduce by 1.0
 -  Jobs AH and AA can be swapped with operators 7 and 21 to reduce by 1.0
 -  Jobs AH and AB can be swapped with operators 7 and 26 to reduce by 1.0
 -  Job AH can be allocated from operator 7 to 30 at start to reduce by 1.0
 -  Jobs AH and AD can be swapped with operators 7 and 18 to reduce by 1.0
 -  Jobs AH and AG can be swapped with operators 7 and 19 to reduce by 1.0
 -  Jobs AH and AI can be swapped with operators 7 and 21 to reduce by 1.0
```

```
- Jobs AH and AJ can be swapped with operators 7 and 3 to reduce by 1.0
- Jobs AH and AL can be swapped with operators 7 and 3 to reduce by 1.0
- Jobs AH and AM can be swapped with operators 7 and 3 to reduce by 1.0
- Jobs AH and AP can be swapped with operators 7 and 9 to reduce by 1.0
- Jobs AH and AR can be swapped with operators 7 and 12 to reduce by 1.0
- Jobs AH and AU can be swapped with operators 7 and 27 to reduce by 1.0
- Job AN can be allocated from operator 7 to 1 at start to reduce by 1.0
- Job AN can be allocated from operator 7 to 2 after job W to reduce by 1.0
- Job AN can be allocated from operator 7 to 2 at start to reduce by 1.0
- Job AN can be allocated from operator 7 to 3 at start to reduce by 1.0
- Job AN can be allocated from operator 7 to 4 at start to reduce by 1.0
- Job AN can be allocated from operator 7 to 5 at start to reduce by 1.0
- Jobs AN and E can be swapped with operators 7 and 11 to reduce by 1.0
- Job AN can be allocated from operator 7 to 6 at start to reduce by 1.0
- Job AN can be allocated from operator 7 to 9 at start to reduce by 1.0
- Job AN can be allocated from operator 7 to 10 at start to reduce by 1.0
- Job AN can be allocated from operator 7 to 11 at start to reduce by 1.0
- Job AN can be allocated from operator 7 to 12 at start to reduce by 1.0
- Jobs AN and L can be swapped with operators 7 and 27 to reduce by 1.0
- Job AN can be allocated from operator 7 to 13 at start to reduce by 1.0
- Job AN can be allocated from operator 7 to 14 at start to reduce by 1.0
- Job AN can be allocated from operator 7 to 15 at start to reduce by 1.0
- Job AN can be allocated from operator 7 to 16 at start to reduce by 1.0
- Job AN can be allocated from operator 7 to 17 at start to reduce by 1.0
- Jobs AN and T can be swapped with operators 7 and 18 to reduce by 1.0
- Job AN can be allocated from operator 7 to 21 at start to reduce by 1.0
- Jobs AN and W can be swapped with operators 7 and 2 to reduce by 1.0
- Jobs AN and X can be swapped with operators 7 and 14 to reduce by 1.0
- Jobs AN and Y can be swapped with operators 7 and 4 to reduce by 1.0
- Job AN can be allocated from operator 7 to 26 at start to reduce by 1.0
- Job AN can be allocated from operator 7 to 27 at start to reduce by 1.0
- Jobs AN and AB can be swapped with operators 7 and 26 to reduce by 1.0
- Job AN can be allocated from operator 7 to 30 at start to reduce by 1.0
- Jobs AN and AL can be swapped with operators 7 and 3 to reduce by 1.0
- Jobs AN and AM can be swapped with operators 7 and 3 to reduce by 1.0
- Jobs AN and AU can be swapped with operators 7 and 27 to reduce by 1.0
- Jobs AN and BA can be swapped with operators 7 and 19 to reduce by 1.0
- Job AO can be allocated from operator 7 to 1 at start to reduce by 1.0
- Job AO can be allocated from operator 7 to 2 after job W to reduce by 1.0
- Job AO can be allocated from operator 7 to 2 at start to reduce by 1.0
- Jobs AO and B can be swapped with operators 7 and 11 to reduce by 1.0
- Job AO can be allocated from operator 7 to 3 at start to reduce by 1.0
- Job AO can be allocated from operator 7 to 4 at start to reduce by 1.0
- Jobs AO and E can be swapped with operators 7 and 11 to reduce by 1.0
- Job AO can be allocated from operator 7 to 6 at start to reduce by 1.0
- Job AO can be allocated from operator 7 to 8 at start to reduce by 1.0
- Jobs AO and I can be swapped with operators 7 and 12 to reduce by 1.0
- Jobs AO and J can be swapped with operators 7 and 16 to reduce by 1.0
- Job AO can be allocated from operator 7 to 11 at start to reduce by 1.0
```

```
 -  Job AO can be allocated from operator 7 to 12 at start to reduce by 1.0
 -  Jobs AO and L can be swapped with operators 7 and 27 to reduce by 1.0
 -  Job AO can be allocated from operator 7 to 13 at start to reduce by 1.0
 -  Job AO can be allocated from operator 7 to 14 at start to reduce by 1.0
 -  Job AO can be allocated from operator 7 to 15 at start to reduce by 1.0
 -  Job AO can be allocated from operator 7 to 17 at start to reduce by 1.0
 -  Jobs AO and R can be swapped with operators 7 and 5 to reduce by 1.0
 -  Job AO can be allocated from operator 7 to 21 at start to reduce by 1.0
 -  Jobs AO and V can be swapped with operators 7 and 10 to reduce by 1.0
 -  Jobs AO and W can be swapped with operators 7 and 2 to reduce by 1.0
 -  Jobs AO and X can be swapped with operators 7 and 14 to reduce by 1.0
 -  Jobs AO and Y can be swapped with operators 7 and 4 to reduce by 1.0
 -  Job AO can be allocated from operator 7 to 26 at start to reduce by 1.0
 -  Jobs AO and AA can be swapped with operators 7 and 21 to reduce by 1.0
 -  Jobs AO and AB can be swapped with operators 7 and 26 to reduce by 1.0
 -  Job AO can be allocated from operator 7 to 30 at start to reduce by 1.0
 -  Jobs AO and AD can be swapped with operators 7 and 18 to reduce by 1.0
 -  Jobs AO and AG can be swapped with operators 7 and 19 to reduce by 1.0
 -  Jobs AO and AI can be swapped with operators 7 and 21 to reduce by 1.0
 -  Jobs AO and AJ can be swapped with operators 7 and 3 to reduce by 1.0
 -  Jobs AO and AL can be swapped with operators 7 and 3 to reduce by 1.0
 -  Jobs AO and AM can be swapped with operators 7 and 3 to reduce by 1.0
 -  Jobs AO and AP can be swapped with operators 7 and 9 to reduce by 1.0
 -  Jobs AO and AR can be swapped with operators 7 and 12 to reduce by 1.0
 -  Jobs AO and AU can be swapped with operators 7 and 27 to reduce by 1.0
 -  Jobs AO and AW can be swapped with operators 7 and 8 to reduce by 1.0
Individual schedule(s) are efficient
 -  All jobs satisfy single and pairwise exchange properties;
Instrument allocation is feasible
 -  There are no instruments
Schedule does satisfy job instrument constraints
 -  All instruments can be used by their assigned operators
Schedule does satisfy instrument skill constraints
 -  All instruments can be used by their assigned operators
```

B.2 hpa

B.2.1 Text Output

The following text output is for weighting $\alpha = 0, \beta = \gamma = 1$. In other words, the optimisation criteria is equally weighted between processing time and distance.

```
Max priority: 30.0 unit(s)
Max time taken: 19200.0 unit(s)
Max distance: 2.23 unit(s)
-
Job allocation to schedule is feasible
 -  All jobs are allocated to exactly one operator
Schedule does satisfy job skill constraints
```

```
 -  All jobs satisfy all skill constraints
Schedule does satisfy job instrument constraints
 -  All jobs satisfy all skill constraints
Schedule is not efficient (for this instrument allocation)
 -  Job BH can be allocated from operator 30 to 1 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 1 after job CU to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 2 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 3 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 5 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 6 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 7 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 8 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 9 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 10 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 11 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 12 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 13 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 14 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 15 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 16 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 17 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 18 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 19 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 21 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 23 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 24 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 25 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 26 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 27 at start to reduce by 0.276
 -  Job BH can be allocated from operator 30 to 28 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 1 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 1 after job CU to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 2 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 3 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 5 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 6 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 7 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 8 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 9 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 10 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 11 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 12 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 13 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 14 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 15 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 16 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 17 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 18 at start to reduce by 0.276
 -  Job BN can be allocated from operator 30 to 19 at start to reduce by 0.276
```

```
- Job BN can be allocated from operator 30 to 21 at start to reduce by 0.276
- Job BN can be allocated from operator 30 to 23 at start to reduce by 0.276
- Job BN can be allocated from operator 30 to 24 at start to reduce by 0.276
- Job BN can be allocated from operator 30 to 25 at start to reduce by 0.276
- Job BN can be allocated from operator 30 to 26 at start to reduce by 0.276
- Job BN can be allocated from operator 30 to 27 at start to reduce by 0.276
- Job BN can be allocated from operator 30 to 28 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 1 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 1 after job CU to reduce by 0.276
- Job CG can be allocated from operator 30 to 2 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 3 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 5 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 6 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 7 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 8 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 9 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 10 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 11 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 12 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 13 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 14 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 15 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 16 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 17 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 18 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 19 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 21 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 23 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 24 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 25 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 26 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 27 at start to reduce by 0.276
- Job CG can be allocated from operator 30 to 28 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 1 after job CU to reduce by 0.276
- Job CV can be allocated from operator 30 to 1 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 2 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 3 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 5 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 6 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 7 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 9 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 10 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 11 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 12 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 13 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 14 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 15 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 16 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 17 at start to reduce by 0.276
```

```
- Job CV can be allocated from operator 30 to 18 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 19 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 21 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 22 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 23 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 24 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 25 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 26 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 27 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 28 at start to reduce by 0.276
- Job CV can be allocated from operator 30 to 29 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 1 after job CU to reduce by 0.276
- Job CW can be allocated from operator 30 to 1 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 2 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 3 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 5 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 6 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 7 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 9 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 10 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 11 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 12 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 13 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 14 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 15 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 16 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 17 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 18 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 19 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 21 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 22 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 23 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 24 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 25 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 26 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 27 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 28 at start to reduce by 0.276
- Job CW can be allocated from operator 30 to 29 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 1 after job CU to reduce by 0.276
- Job CX can be allocated from operator 30 to 1 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 2 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 3 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 5 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 6 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 7 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 8 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 9 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 10 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 11 at start to reduce by 0.276
```

```
- Job CX can be allocated from operator 30 to 12 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 13 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 14 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 15 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 16 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 17 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 18 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 19 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 21 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 23 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 24 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 25 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 26 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 27 at start to reduce by 0.276
- Job CX can be allocated from operator 30 to 28 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 1 after job CU to reduce by 0.276
- Job DB can be allocated from operator 30 to 1 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 2 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 3 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 5 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 6 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 7 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 8 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 9 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 10 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 11 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 12 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 13 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 14 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 15 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 16 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 17 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 18 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 19 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 21 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 23 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 24 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 25 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 26 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 27 at start to reduce by 0.276
- Job DB can be allocated from operator 30 to 28 at start to reduce by 0.276
- Job DE can be allocated from operator 30 to 1 after job CU to reduce by 0.276
- Job DE can be allocated from operator 30 to 1 at start to reduce by 0.276
- Job DE can be allocated from operator 30 to 2 at start to reduce by 0.276
- Job DE can be allocated from operator 30 to 3 at start to reduce by 0.276
- Job DE can be allocated from operator 30 to 5 at start to reduce by 0.276
- Job DE can be allocated from operator 30 to 7 at start to reduce by 0.276
- Job DE can be allocated from operator 30 to 11 at start to reduce by 0.276
- Job DE can be allocated from operator 30 to 15 at start to reduce by 0.276
```

```
- Job DE can be allocated from operator 30 to 19 at start to reduce by 0.276
- Job DE can be allocated from operator 30 to 25 at start to reduce by 0.276
- Job DE can be allocated from operator 30 to 26 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 1 after job CU to reduce by 0.276
- Job DI can be allocated from operator 30 to 1 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 2 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 3 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 5 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 6 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 7 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 9 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 10 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 11 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 12 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 13 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 14 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 15 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 16 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 17 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 18 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 19 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 21 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 22 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 23 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 24 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 25 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 26 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 27 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 28 at start to reduce by 0.276
- Job DI can be allocated from operator 30 to 29 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 1 after job CU to reduce by 0.276
- Job DL can be allocated from operator 30 to 1 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 2 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 3 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 5 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 6 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 7 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 9 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 10 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 11 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 12 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 13 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 14 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 15 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 16 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 17 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 18 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 19 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 21 at start to reduce by 0.276
```

```
- Job DL can be allocated from operator 30 to 22 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 23 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 24 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 25 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 26 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 27 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 28 at start to reduce by 0.276
- Job DL can be allocated from operator 30 to 29 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 1 after job CU to reduce by 0.276
- Job DQ can be allocated from operator 30 to 1 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 2 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 3 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 5 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 6 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 7 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 9 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 10 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 11 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 12 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 13 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 14 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 15 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 16 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 17 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 18 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 19 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 21 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 22 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 23 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 24 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 25 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 26 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 27 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 28 at start to reduce by 0.276
- Job DQ can be allocated from operator 30 to 29 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 1 after job CU to reduce by 0.276
- Job DS can be allocated from operator 30 to 1 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 2 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 3 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 5 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 6 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 7 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 8 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 9 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 10 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 11 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 12 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 13 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 14 at start to reduce by 0.276
```

```
- Job DS can be allocated from operator 30 to 15 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 16 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 17 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 18 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 19 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 21 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 23 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 24 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 25 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 26 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 27 at start to reduce by 0.276
- Job DS can be allocated from operator 30 to 28 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 1 after job CU to reduce by 0.276
- Job DT can be allocated from operator 30 to 1 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 2 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 3 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 5 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 6 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 7 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 8 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 9 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 10 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 11 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 12 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 13 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 14 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 15 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 16 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 17 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 18 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 19 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 21 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 23 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 24 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 25 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 26 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 27 at start to reduce by 0.276
- Job DT can be allocated from operator 30 to 28 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 1 after job CU to reduce by 0.276
- Job DV can be allocated from operator 30 to 1 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 2 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 3 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 5 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 6 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 7 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 8 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 9 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 10 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 11 at start to reduce by 0.276
```

```
- Job DV can be allocated from operator 30 to 12 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 13 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 14 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 15 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 16 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 17 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 18 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 19 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 21 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 23 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 24 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 25 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 26 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 27 at start to reduce by 0.276
- Job DV can be allocated from operator 30 to 28 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 1 after job CU to reduce by 0.276
- Job DW can be allocated from operator 30 to 1 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 2 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 3 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 5 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 6 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 7 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 8 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 9 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 10 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 11 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 12 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 13 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 14 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 15 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 16 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 17 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 18 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 19 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 21 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 23 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 24 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 25 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 26 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 27 at start to reduce by 0.276
- Job DW can be allocated from operator 30 to 28 at start to reduce by 0.276
- Job DX can be allocated from operator 30 to 1 after job CU to reduce by 0.276
- Job DX can be allocated from operator 30 to 1 at start to reduce by 0.276
- Job DX can be allocated from operator 30 to 2 at start to reduce by 0.276
- Job DX can be allocated from operator 30 to 3 at start to reduce by 0.276
- Job DX can be allocated from operator 30 to 5 at start to reduce by 0.276
- Job DX can be allocated from operator 30 to 7 at start to reduce by 0.276
- Job DX can be allocated from operator 30 to 11 at start to reduce by 0.276
- Job DX can be allocated from operator 30 to 15 at start to reduce by 0.276
```

```
 -  Job DX can be allocated from operator 30 to 19 at start to reduce by 0.276
 -  Job DX can be allocated from operator 30 to 25 at start to reduce by 0.276
 -  Job DX can be allocated from operator 30 to 26 at start to reduce by 0.276
 -  Jobs BN and DJ can be swapped with operators 30 and 4 to reduce by 0.0319
 -  Jobs CG and DN can be swapped with operators 30 and 2 to reduce by 0.0286
 -  Jobs CG and CQ can be swapped with operators 30 and 2 to reduce by 0.0238
 -  Jobs CG and DO can be swapped with operators 30 and 2 to reduce by 0.0238
 -  Jobs BN and DM can be swapped with operators 30 and 4 to reduce by 0.0222
 -  Jobs BN and DR can be swapped with operators 30 and 4 to reduce by 0.0222
 -  Jobs BN and DK can be swapped with operators 30 and 4 to reduce by 0.0216
 -  Jobs BN and DY can be swapped with operators 30 and 4 to reduce by 0.0216
 -  Jobs CG and V can be swapped with operators 30 and 2 to reduce by 0.0188
 -  Jobs CG and X can be swapped with operators 30 and 2 to reduce by 0.0188
 -  Jobs CG and Y can be swapped with operators 30 and 2 to reduce by 0.0188
 -  Jobs CG and Z can be swapped with operators 30 and 2 to reduce by 0.0188
 -  Jobs CG and AB can be swapped with operators 30 and 2 to reduce by 0.0188
 -  Jobs CG and CS can be swapped with operators 30 and 2 to reduce by 0.0188
 -  Jobs CG and DH can be swapped with operators 30 and 2 to reduce by 0.0188
 -  Jobs CG and DZ can be swapped with operators 30 and 2 to reduce by 0.0188
 -  Jobs CG and AA can be swapped with operators 30 and 2 to reduce by 0.0163
 -  Jobs BN and DU can be swapped with operators 30 and 4 to reduce by 0.0121
Individual schedule(s) are efficient
 -  All jobs satisfy single and pairwise exchange properties;
Instrument allocation is feasible
 -  There are no instruments
Schedule does satisfy job instrument constraints
 -  All instruments can be used by their assigned operators
Schedule does satisfy instrument skill constraints
 -  All instruments can be used by their assigned operators
```

Appendix C

User Study Data

C.1 User Study Layout

We include the following pages from the user study:

1. Introduction to the user study (for the half-then-half format);
2. Formatting guide to answer questions;
3. Question examples:
   (a) Distance 2;
   (b) Choice of Schedule 2;
4. Introduction to optimal scheduling tool.

The full user study can be found at https://gitlab.doc.ic.ac.uk/jcl122/OptimalSchedulingWebApp/-/tree/master/user_study.

Imperial College London

Introduction (HALF THEN HALF)

Optimal Scheduling User Study

Format

This exercise will judge the general ability to understand and explain various schedules (including makespan schedules).

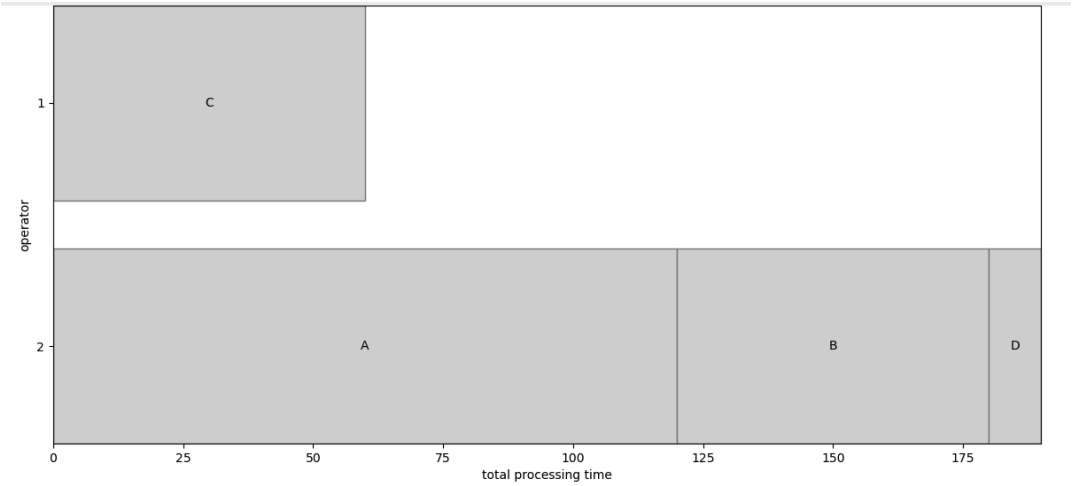
- Schedules consist of *m* operators and *n* jobs.
- Every job must be assigned to exactly one operator for the schedule to be feasible.
- The objective is to minimise the longest collective processing time and/or distance traveled by any operator.
- Some jobs have constraints (skills) which the corresponding operators must satisfy.

Operators are denoted by integers 1, 2, 3, etc.
Jobs are denoted by letters A, B, C, etc.
Skills (if applicable) are denoted by 3-digit HEX codes, e.g. AB3, B04
Instruments (if applicable) are denoted by "I" plus integers (beginning at 0), e.g. I0, I1, I2, etc.

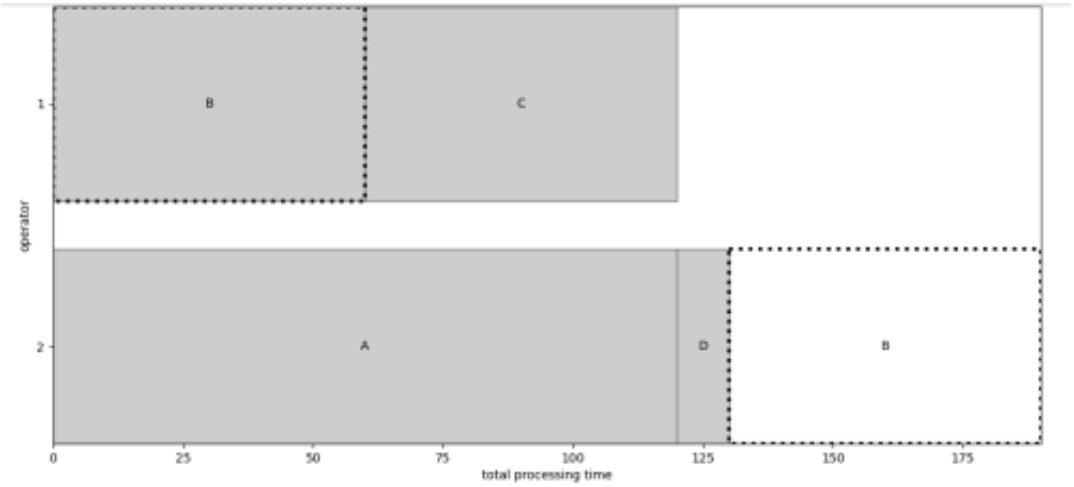
For simplicity, working shifts and travelling times have not been taken into account

Example

In this example, we want to minimise the makespan. Here, all jobs have the same processing time, regardless of the operator it is assigned to.



This schedule is not optimal since jobs A, B or D could be moved to operator 1.
Suppose we choose to move job B from operator 2 to operator 1 (shown below):



This schedule is optimal since no different assignments could be made which would improve the overall makespan cost.

Further Information

This exercise consists of 14 questions. The first half will require you to solve the problems by hand. You will then have access an optimal scheduling tool, which should help you complete the remaining half of questions.

Please note some of these questions are difficult, particularly solving by hand. If you have still not got an answer after a few minutes, take a guess or leave blank, and move on.

Expected duration: 1 hour.
This exercise should be completed in one sitting.

Formatting

Formatting

For questions asking for an **improved schedule**, answers should be in the form:

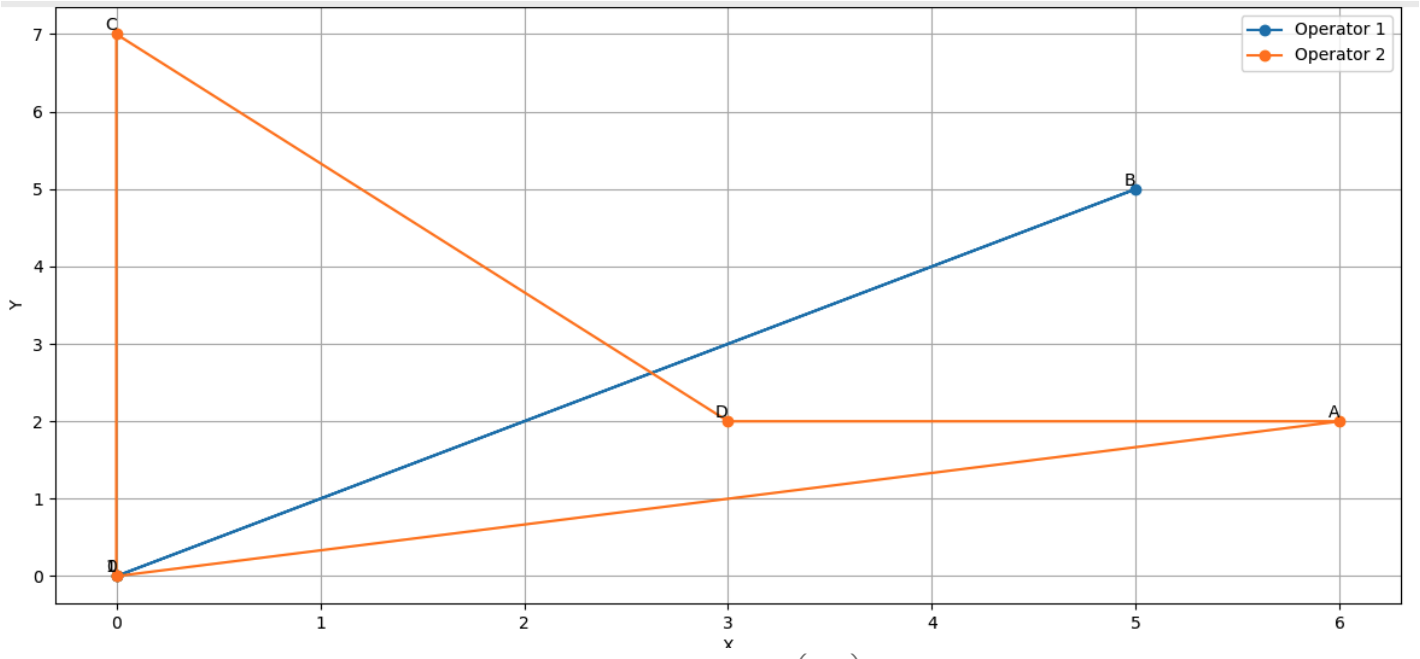
```
1: A B C
2: D
```

For questions asking for the **maximum distance/processing time** an operator takes, answers should be given to 2 decimal places, and only for the operator who travels the furthest/takes the most time.
e.g. If operator 1 travels 2.34 units and operator 2 travels 3.12 units, the answer would be 3.12

Distance 2 - by hand READING

(TO BE COMPLETED BY HAND)
In this question, we want to **minimise** the **maximum distance travelled** by any operator.

The graph below illustrates the location of each job and each operator's start/end position.



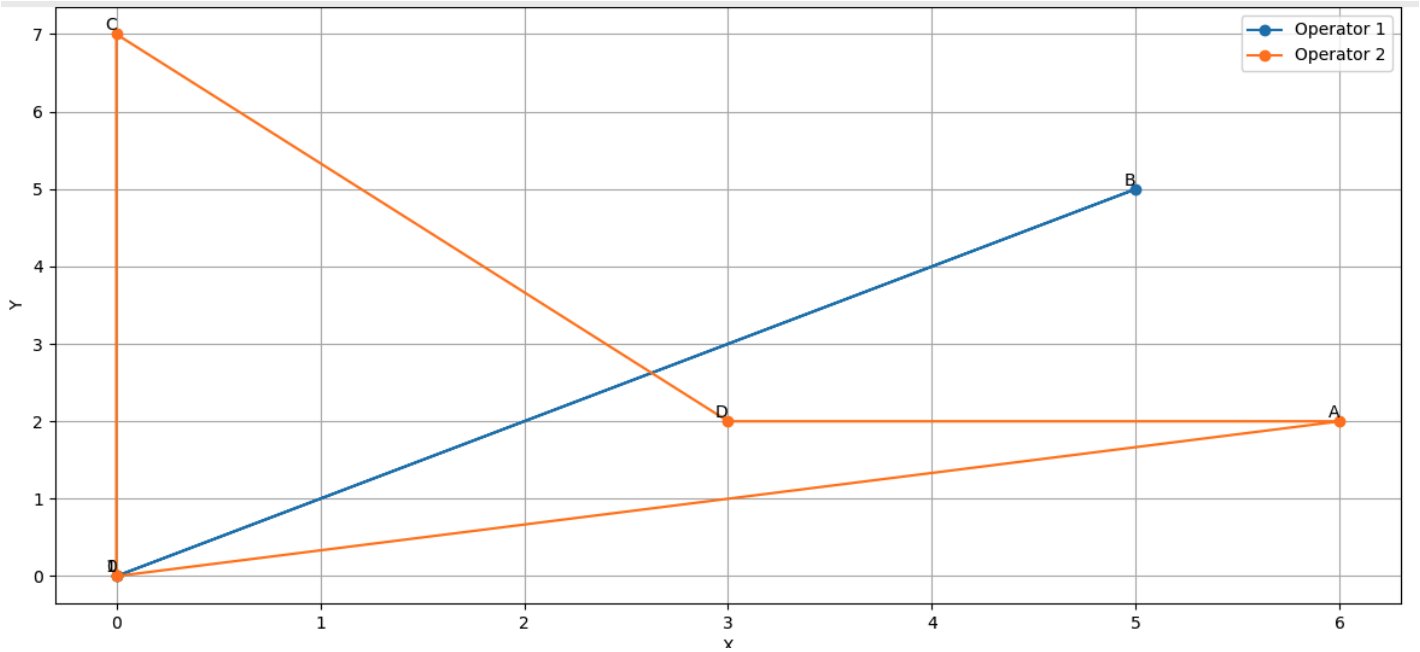
- All operators must begin and end their shift at the origin (0,0)
- Job A is located at (6,2)
- Job B is located at (5, 5)
- Job C is located at (0,7)
- Job D is located at (3,2)

Read the information above, then click "Next" to begin the question.

Distance 2

In this question, we want to **minimise** the **maximum distance travelled** by any operator.

The graph below illustrates the location of each job and each operator's start/end position.



- All operators must begin and end their shift at the origin (0,0)
- Job A is located at (6,2)
- Job B is located at (5, 5)
- Job C is located at (0,7)
- Job D is located at (3,2)

The below schedule is inefficient:

```
1: B
2: C D A
```

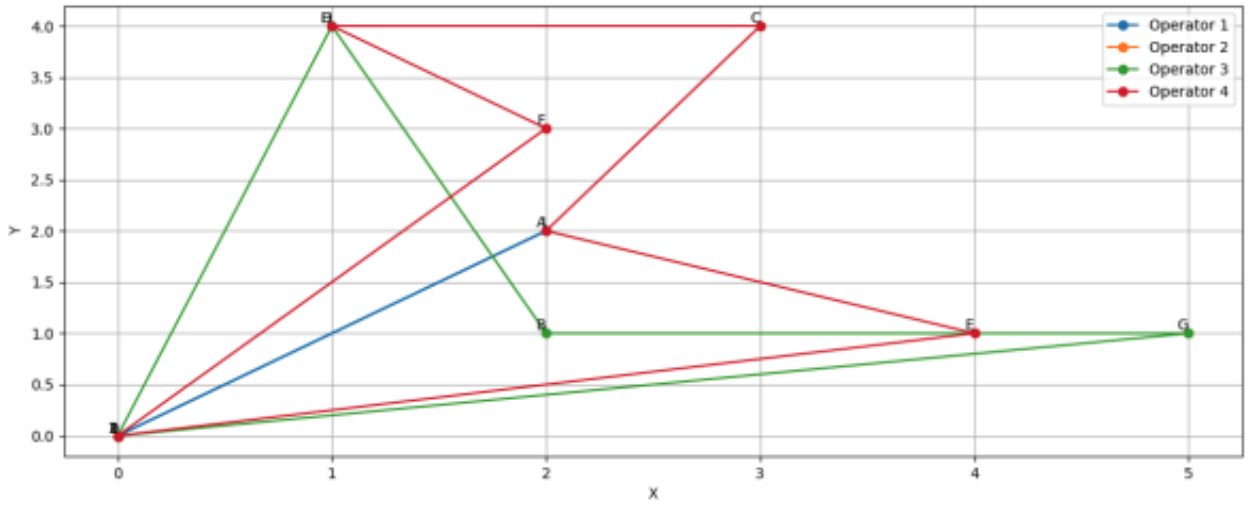
Using the above information about distance, suggest an improved schedule for this problem

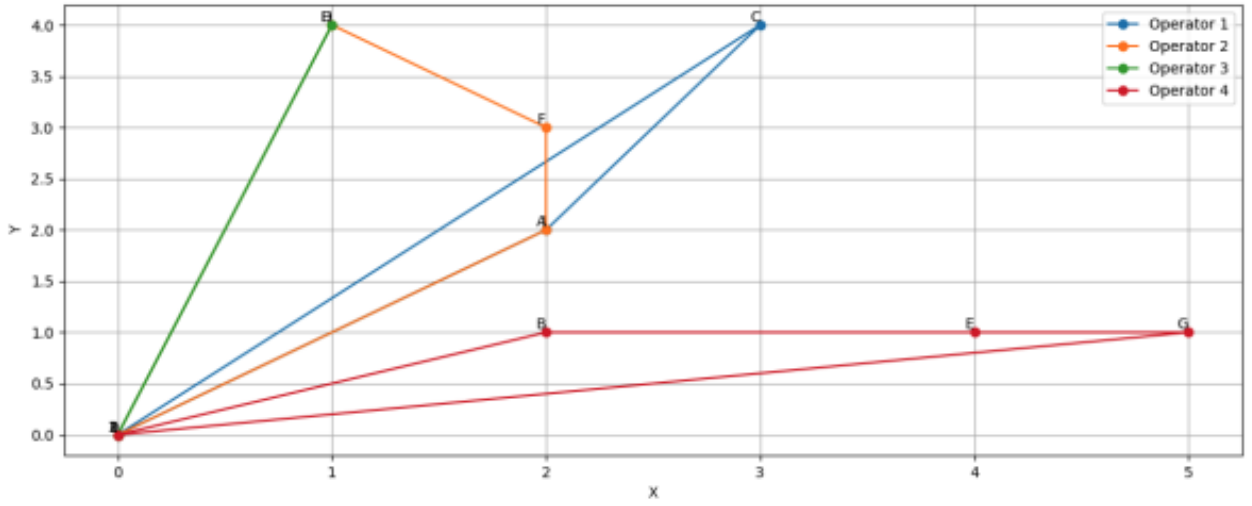
What is the maximum distance that any operator must travel in your improved schedule?

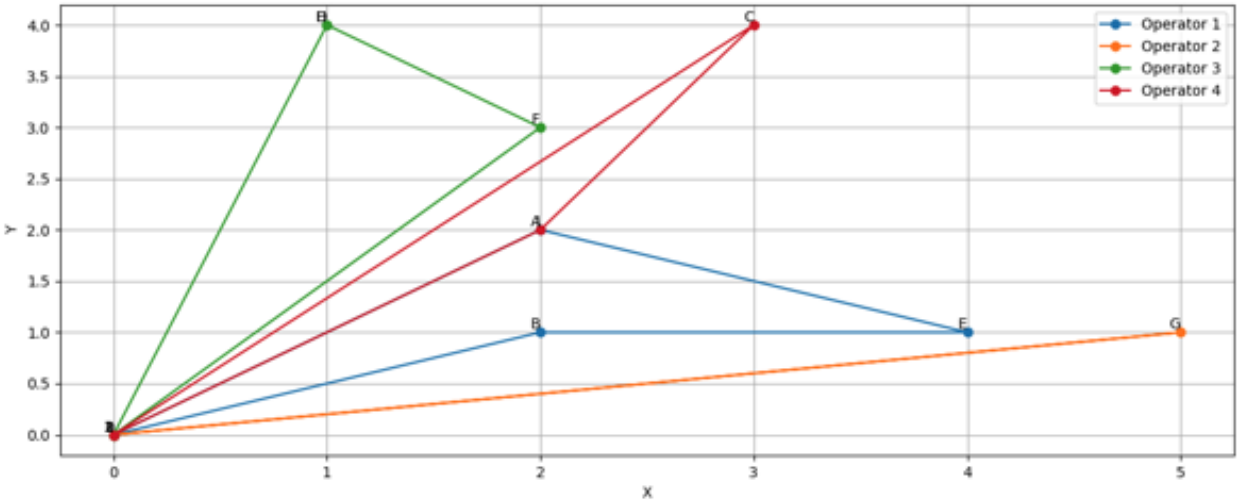
Choice of Schedule 2 - by hand

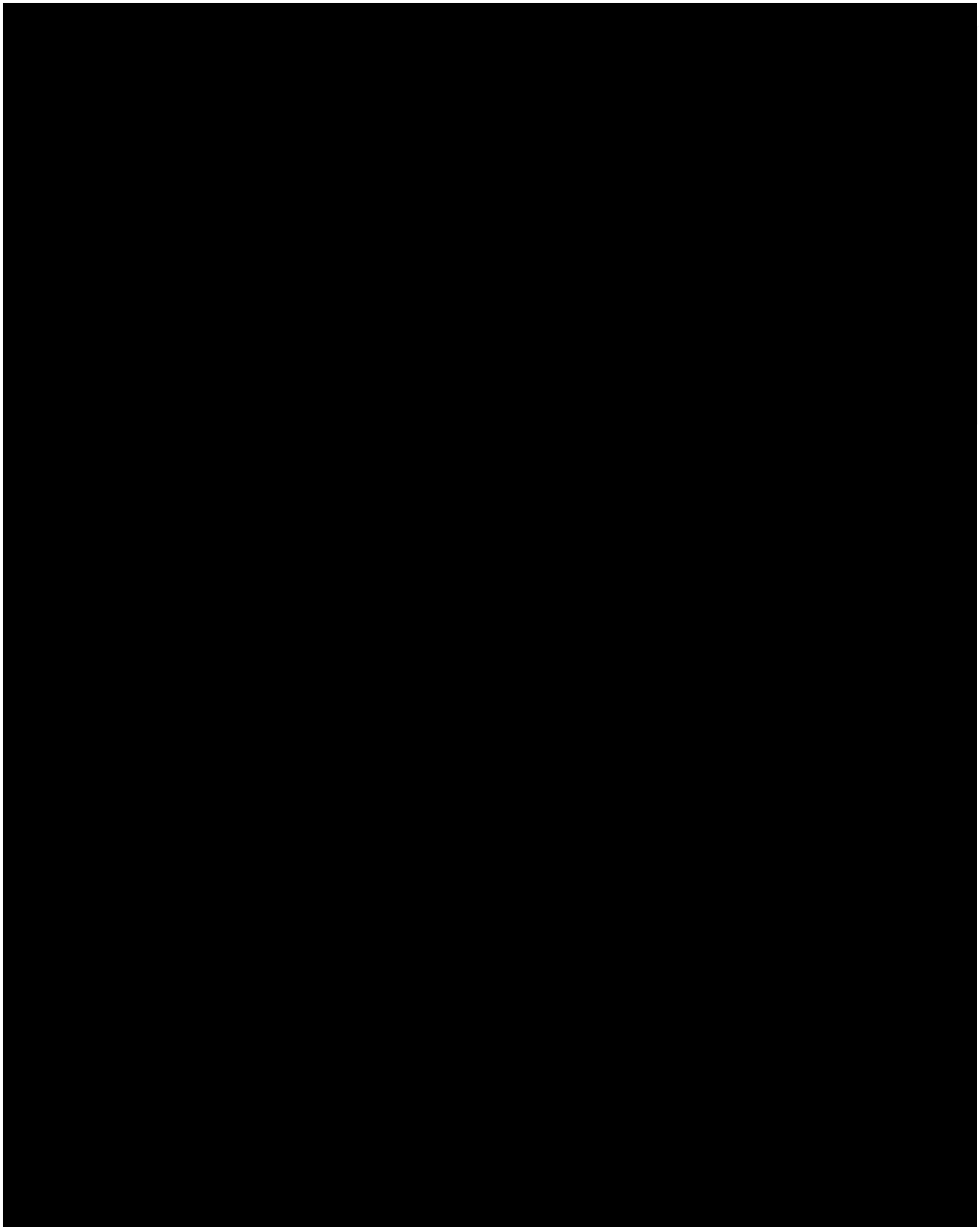
(TO BE COMPLETED BY HAND)
In this question, we want to **minimise** the **maximum distance travelled** by any operator.

Select the optimal schedule.






Operator 1
Operator 2
Operator 3
Operator 4
X
Y



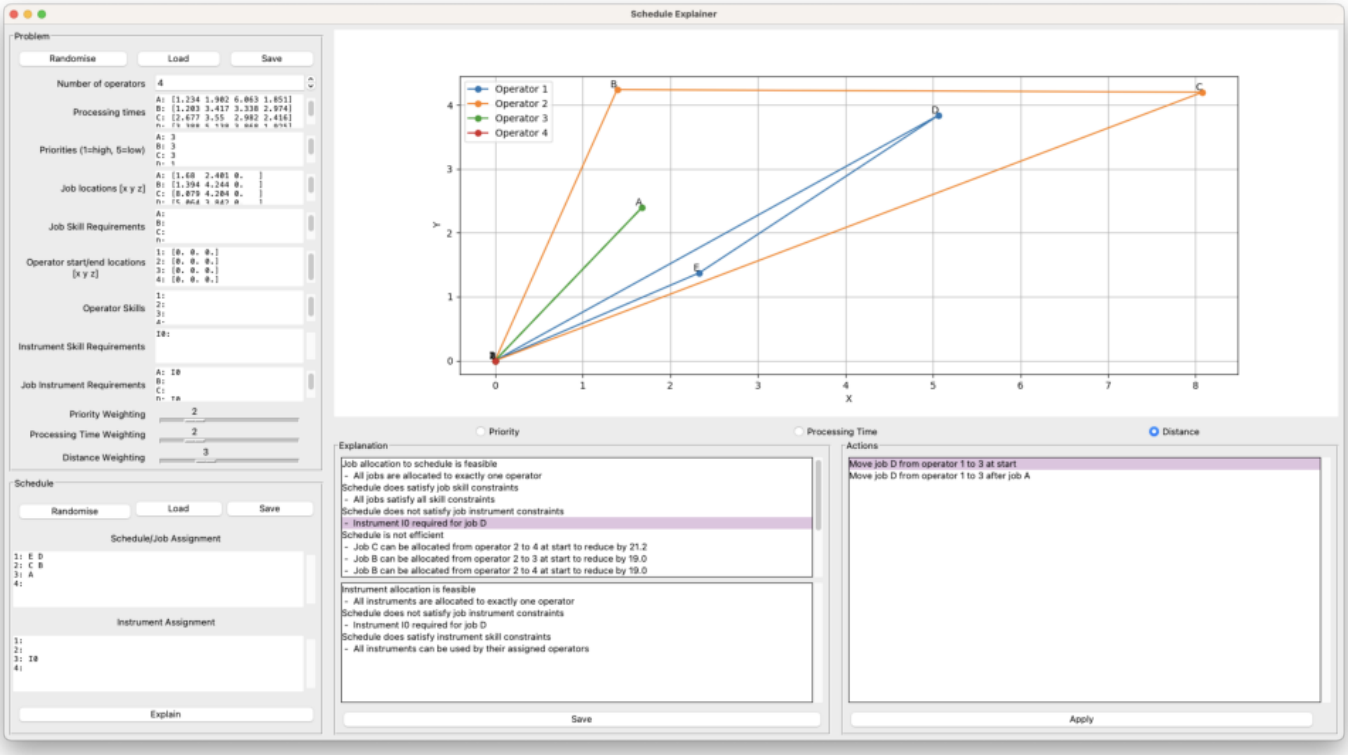
Introduction to Optimal Scheduling Model

Optimal Scheduling Tool

Please read [this guide] under 'How to use the tool' for an introduction to the tool and how to use it.

For Windows, you can download the tool here. Download the zip folder containing all 3 files to your computer (run_app.bat, requirements.txt and main.exe).

To run the tool, go into this folder and click `run_app` (`.bat` in Windows). (If your module requirements are already met, you can remove this line from the file.)



Alternate Installation

Download the source code from this link.

This assumes a Ubuntu 22.04 or MacOS Ventura 13.4 system. The following packages are required and can be installed from the terminal:

- `pip install matplotlib`
- `pip install numpy`
- `pip install tkinter-tooltip`

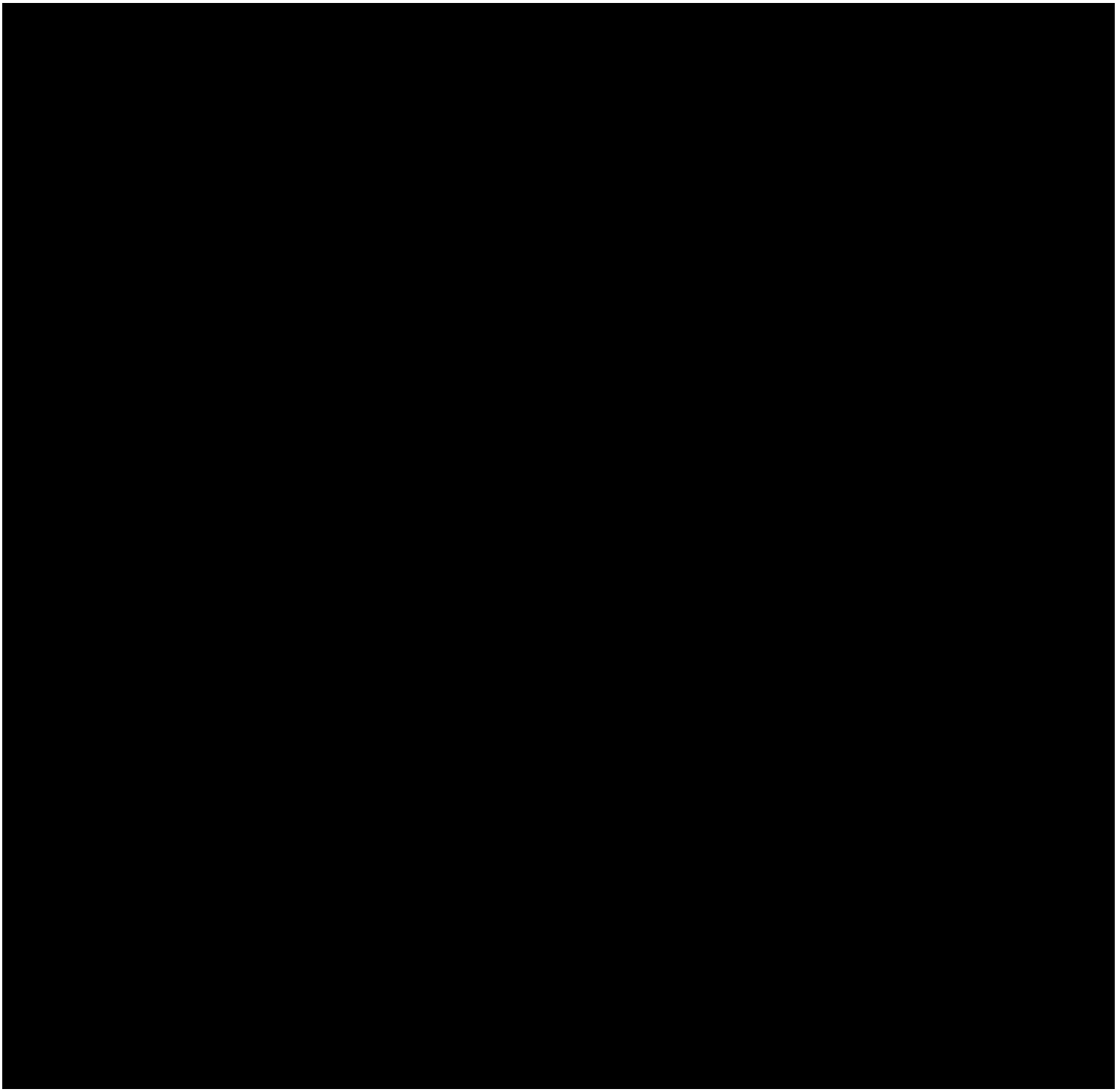
To start the tool, run `python main.py -g` in the `src` directory supplied in the repository.

Next Questions

For the remainder of exercises, you should use the optimal scheduling tool to calculate your answers.

1. Each question will include a `.problem` and a `.schedule` file which represents the problem and schedule described (ignore any numbers associated with the exercises – the order is semi-random).
2. Download the relevant files at the start of each question, and use the `Load` button in the relevant sections of the tool to upload this information to the app.

Please take a couple minutes to get familiar with the tool before moving to the next section.



	By hand			**By tool**		
Set 1	**Alt**	**HTH**	**Total (by hand)**	**Alt**	**HTH**	**Total (by tool)**
Q1	630.42	406.13	**518.27**	476.65	142.39	**225.95**
Q2	257.09	274.97	**266.03**	25.87	22.55	**23.38**
Q3	1026.20	635.47	**830.84**	14.37	84.24	**66.78**
Q4	274.98	508.12	**391.55**	130.87	97.98	**106.20**
Q5	437.49	511.83	**474.66**	9.35	33.81	**27.70**
Q6	409.33	475.44	**442.38**	125.70	93.12	**101.27**
Q7	438.24	805.62	**621.93**	1.53	42.49	**32.25**

	By hand			**By tool**		
Set 2	**Alt**	**HTH**	**Total (by hand)**	**Alt**	**HTH**	**Total (by tool)**
Q1	1452.87	518.04	**752.24**	418.48	102.43	**260.46**
Q2	703.03	256.38	**368.04**	69.14	48.31	**58.72**
Q3	1214.38	448.07	**639.64**	112.29	56.61	**84.44**
Q4	78.17	333.22	**269.46**	138.06	296.88	**217.47**
Q5	1647.76	1468.62	**1513.40**	235.14	58.89	**147.02**
Q6	-	1661.55	**1661.55**	232.13	69.71	**150.92**
Q7	-	730.47	**730.47**	53.25	26.56	**39.90**

Table C.1: Average time taken (in seconds) for alternating (Alt) and half-then-half (HTH) groups to complete each question set.

C.2 Average Question Completion Time

Table C.1 gives the average time each user spent on a given question before clicking submit.