BSc report APPLIED MATHEMATICS

"Parallel machine scheduling under uncertainty"

FOS VAN DER MEER

Technische Universiteit Delft

Mentor

K. Postek

Committee members

K. Postek

B. van den Dries

May, 2021

Delft

Contents

1 Introduction

Consider the following problem: We have three phones that need charging, and two chargers. For now, we will assume the phones have identical charging time, say, two hours. The goal is to charge all phones as quickly as possible. To achieve this, we can divide them over the chargers as shown in figure 1.

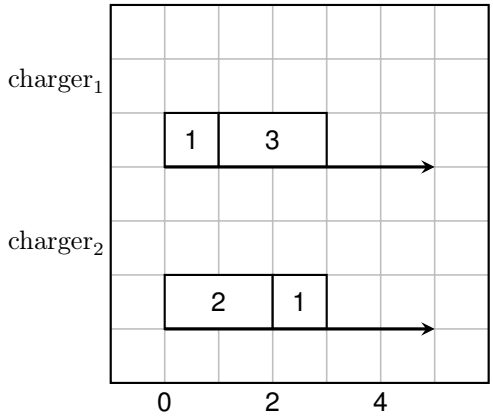

Figure 1: Charging three phones on two chargers

We start out with phone 1 and phone 2, then, when phone 1 is charged halfway, we replace it with phone 3. After phone 2 is done charging, we continue with phone 1 until all phones are fully charged. This problem is an example of a *scheduling problem*, which this report revolves around.

1.1 Literature review

An in-depth introduction to scheduling problems can be found in Pinedo, 2008. In general, scheduling problems can be described in terms of *tasks* (charging phones) that need to be executed on *machines* (chargers). Scheduling problems can have many restrictions and conditions. The scheduling problems in this report share these properties:

- We want to minimize the *total makespan*, which is the time needed to finish all tasks,
- there are two machines, m_1 and m_2,
- there are a finite number of tasks to be scheduled,
- tasks can not be *preempted* (i.e. put on hold), for example task/phone 1 in figure 1,
- tasks must be executed one after another, without a gap in between,

- the task durations have a lower and upper bound, but are not known beforehand.

This report revolves around the last property, which tries to model reality by introducing uncertainty.

1.2 Goal of the report

In this report, we will be looking at ways to solve scheduling problems, where the task durations are not known beforehand. We will investigate the performance of some *heuristics*, following Cohen et al., 2021, which typically will not give the optimal solution, and finding the best heuristic comes down to some speculation.

2 Modeling uncertainty

To model uncertainty, we will use the lower and upper bounds of the task durations to make a *budgeted uncertainty set*. This type of uncertainty set has been inspired by Li and Floudas, 2011, and Cohen et al., 2021. We have n tasks, where for each $i \in [n]$, we have

$$\hat{d}_i \leq d_i \leq \hat{d}_i + \hat{z}_i.$$

where d_i is the duration of task i, in an arbitrary time unit, and the lower and upper bounds are given by $\hat{d}_i$ and $\hat{d}_i + \hat{z}_i$ respectively. In other words,

$$d_i = \hat{d}_i + r_i, \quad 0 \leq r_i \leq \hat{z}_i, \tag{1}$$

where r_i describes the part of d_i we are uncertain about. We need some more constraints on r_i to make some meaningful observations. Suppose the weighted sum of r_i can not exceed a maximum value.

$$\sum_{i=1}^{n} w_i r_i \leq M, \quad w_i > 0, M \geq 0. \tag{2}$$

When we make M smaller, it limits the possibilities for values of r_i. There is no point in making $M > \sum_{i=1}^{n} w_i \hat{z}_i$, since then for any r_i, the condition in equation (2) will be met. Let us now define M in the following way.

$$M_\alpha = \alpha \sum_{i=1}^{n} w_i \hat{z}_i, \quad \alpha \in [0, 1].$$

The *budget* M_α is parametrized by α. Substituting M_α in equation (2) yields

$$\sum_{i=1}^{n} w_i r_i \leq \alpha \sum_{i=1}^{n} w_i \hat{z}_i, \quad \alpha \in [0, 1]. \tag{3}$$

If $\alpha = 1$, this equation will be true for all r_i, since $0 \leq r_i \leq \hat{z}_i$.
If $0 < \alpha < 1$, it will limit the possibilities for values of r_i.

If $\alpha = 0$, then all $r_i = 0$.
We can write equations (1) and (3) in vector notation.

$$\mathbf{d} = \hat{\mathbf{d}} + \mathbf{r} \tag{4}$$
$$\mathbf{0} \leq \mathbf{r} \leq \hat{\mathbf{z}} \tag{5}$$
$$\mathbf{wr} \leq \alpha \mathbf{w}\hat{\mathbf{z}}, \quad \alpha \in [0, 1] \tag{6}$$

where $\mathbf{r}$, $\mathbf{w}$ and $\hat{\mathbf{z}}$ represent the vectors containing all r_i, w_i and $\hat{z}_i$ respectively.
We can use these equations to make our budgeted uncertainty set.

2.1 Budgeted uncertainty set and some simplifications

To generate a budgeted uncertainty set, we will use equations (5) and (6) from section 2. Let us first, for a given $\mathbf{w} > \mathbf{0}$, $\alpha \in [0, 1]$ and $\hat{\mathbf{z}} > \mathbf{0}$, define the set

$$Y = \{\mathbf{r} \in \mathbb{R}^n : \quad 0 \leq \mathbf{r} \leq \hat{\mathbf{z}}, \quad \mathbf{wr} \leq \alpha \mathbf{w}\hat{\mathbf{z}}\}.$$

It describes a polytope in $\mathbb{R}^n$, where α gives the location of the cutting plane with normal vector $\mathbf{w}$.
Next, we will make two simplifications.

1. Of the polytope described by Y, we will only look at the vertices.
2. Of these vertices, we will only consider the *non-dominated* vertices.

For the first simplification, define a set $\hat{Y}$, containing only vertices of the polytope described by Y.
For the second simplification, we want to have the *non-dominated* vertices. Which comes down to vertices $\mathbf{r}$ with the highest value for $\mathbf{wr}$. We take the set $\hat{Y}$, and define

$$R = \{\mathbf{r} \in \hat{Y} : \quad \mathbf{wr} \geq \mathbf{wy}, \quad y \in \hat{Y}\}.$$

This set R contains values for $\mathbf{r}$, and since $\mathbf{d} = \hat{\mathbf{d}} + \mathbf{r}$, these will be used to make our uncertainty set U:

$$U = \{\hat{\mathbf{d}} + \mathbf{r} : \quad \mathbf{r} \in R\}.$$

This will be the uncertainty set we will be using for our simulations.
From this point on, we will be representing U as a matrix in the following way:

$$U = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_s \end{bmatrix},$$

with the elements of U as vectors $\mathbf{u}_j$.

3 Description of the system state

During scheduling, we keep track of the system state using the following information:

Symbol	Description
n, s	Number of tasks, number of scenarios respectively.
$[n]$	$\{1, 2, 3, \cdots, n\}$ These are the indices for the tasks.
i	Almost always signifies the index of a task, $i \in [n]$.
P, S, F	Sets containing indices of *planned* tasks, *started* tasks. and *finished* tasks, respectively. $\{P, S, F\}$ forms a partition on $[n]$.
$\mathbf{p}$, $\mathbf{s}$, $\mathbf{f}$	Vectors in $\{0, 1\}^n$ such that $p_i = 1 \Leftrightarrow i \in P$. Idem for $\mathbf{s}$ and $\mathbf{f}$. the set and vector representation are used for convenience.
M_1, M_2	Sets containing indices of tasks (started or finished) on *machine 1* or *machine 2* respectively. $\{M_1, M_2\}$ forms a partition on $S \cup F$.
$\mathbf{m}_1$, $\mathbf{m}_2$	Vectors in $\{0, 1\}^n$ such that $\mathbf{m}_{ki} = 1 \Leftrightarrow i \in M_k$, $k \in \{1, 2\}$.
U	Matrix representation of the uncertainty set, $U = \begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_s\end{bmatrix}$. Duration of task i in scenario $j = u_{ij}$
$\mathbf{d}$	Vector containing observed *durations*, d_i is the observed duration of task i.
$U(\mathbf{d})$	Uncertainty set containing only feasible scenarios given $\mathbf{d}$. See below.
t	A specific point in time after start of scheduling. Depends on context

We have some more definitions, which will be important later on.

3.1 Feasible scenarios $U(\mathbf{d})$.

Given observed durations $\mathbf{d}$, and uncertainty set U, $U(\mathbf{d})$ is defined as follows:

$$U(\mathbf{d}) = \{\mathbf{u} \in U : \quad d_i < u_i, i \in S, \quad d_j = u_j, j \in F\}.$$

In other words, $U(\mathbf{d})$ contains only scenarios $\mathbf{u}$, where the duration of running tasks is less than the duration given in $\mathbf{u}$, and where the duration of the finished tasks is exactly equal to $\mathbf{u}$.

3.2 Makespan

The makespan is the amount of time needed to finish all tasks. Suppose we have a system that has finished all tasks. The makespan is defined as follows.

$$t = \max(\mathbf{m}_1\mathbf{d}, \mathbf{m}_2\mathbf{d}),$$

where $\mathbf{m}_k\mathbf{d}$ (using shorthand dot product notation) is the total time machine k has been active.

3.3 True scenario

At the start of scheduling, we have an uncertainty set U. From this set, one scenario $\hat{\mathbf{u}}$ will be the *true scenario*, meaning that any task i will have its execution time $d_i = \hat{u}_i$. The scheduler will not know which scenario will be the true scenario.

4 Lower bound

In section 5, we will be looking at heuristics and comparing their performance. We can determine a lower bound for the total makespan of a scheduling problem. This lower bound will help us in determining an *upper* bound for the makespan of some specific scheduling problems.

Lemma 4.1. *For any scenario $\boldsymbol{u} \in U$,*

1. *the lower bound t_{min} of the makespan $t = \max(\boldsymbol{m}_1\boldsymbol{d}, \boldsymbol{m}_2\boldsymbol{d})$ is given by:*

$$t_{min} = \frac{1}{2}||\boldsymbol{u}||_1,$$

   where $||\boldsymbol{u}||_1 = \sum_{i\in[n]} u_i$ is the sum of all execution times in $\boldsymbol{u}$;

2. *$t = t_{min}$ if and only if $\boldsymbol{m}_1\boldsymbol{d} = \boldsymbol{m}_2\boldsymbol{d}$.*

Proof. Pick a true scenario $\hat{\mathbf{u}} \in U$. Its makespan is given by

$$t = \max(\mathbf{m}_1\mathbf{d}, \mathbf{m}_2\mathbf{d}).$$

1. Since we picked the true scenario ourselves, we know for any finished task i, $d_i = \hat{u}_i$. So t can be described by

$$t = \max(\mathbf{m}_1\hat{\mathbf{u}}, \mathbf{m}_2\hat{\mathbf{u}}).$$

   Since all tasks are finished after t. we have $F = [n]$. that means $\{M_1, M_2\}$ is a partition over $[n]$. From this it follows that $\mathbf{m}_1 + \mathbf{m}_2 = \mathbf{1} \in \{0,1\}^n$. We now rewrite $\max(a, b)$.

$$\max(a,b) = \frac{a+b+|a-b|}{2} \geq \frac{a+b}{2}.$$

   We substitute $\mathbf{m}_1\hat{\mathbf{u}}$ and $\mathbf{m}_2\hat{\mathbf{u}}$:

$$t = \max(\mathbf{m}_1\hat{\mathbf{u}}, \mathbf{m}_2\hat{\mathbf{u}}) \geq \frac{(\mathbf{m}_1+\mathbf{m}_2)\hat{\mathbf{u}}}{2} = \frac{1}{2}||\hat{\mathbf{u}}||_1, \tag{7}$$

   using $\mathbf{1}\hat{\mathbf{u}} = ||\hat{\mathbf{u}}||_1$. From this we get

$$t \geq \frac{1}{2}||\hat{\mathbf{u}}||_1 = t_{\min}.$$

2. We have seen that $\mathbf{d} = \hat{\mathbf{u}}$. Now following from equation 7 we get:

$$\begin{aligned} t &= t_{\min} && \Leftrightarrow \\ \frac{(\mathbf{m}_1 + \mathbf{m}_2)\hat{\mathbf{u}} + |\mathbf{m}_1\hat{\mathbf{u}} - \mathbf{m}_2\hat{\mathbf{u}}|}{2} &= \frac{(\mathbf{m}_1 + \mathbf{m}_2)\hat{\mathbf{u}}}{2} && \Leftrightarrow \\ |\mathbf{m}_1\hat{\mathbf{u}} - \mathbf{m}_2\hat{\mathbf{u}}| &= 0 && \Leftrightarrow \\ \mathbf{m}_1\hat{\mathbf{u}} &= \mathbf{m}_2\hat{\mathbf{u}} && \Leftrightarrow \\ \mathbf{m}_1\mathbf{d} &= \mathbf{m}_2\mathbf{d}. \end{aligned}$$

□

The lower bound $t_{\min}$ can be attained, take for example

$$\hat{\mathbf{u}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{m}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Then

$$\begin{aligned} t_{\min} &= \frac{1}{2}||\hat{\mathbf{u}}||_1 = \frac{1}{2} \times 2 = 1, \\ t &= \max(\mathbf{m}_1\hat{\mathbf{u}}, \mathbf{m}_2\hat{\mathbf{u}}) = \max(1, 1) = 1. \end{aligned}$$

5 Heuristics

Typically, these kinds of scheduling problems, where durations of tasks are not known, will be solved using integer linear programming, and every time a new task needs to be chosen, the worst case scenario will be assumed to be true. In Cohen et al., 2021, the approach was to recursively go through all different task orders, and minimize the worst case scenario. Due to the uncertainty present in this kind of scheduling problem, linear programming alone will not be able to find the best solution in most cases. Our goal is to find a *heuristic*, whose performance exceeds linear programming. The heuristics we will be using can be described as a function that maps the state to a task index.

$$\mathscr{H}(U, \mathbf{d}, \mathbf{p}, \mathbf{s}, \mathbf{f}) = h,$$

where

U : Set of all scenarios,
$\mathbf{d}$: Vector of durations of all tasks,
$\mathbf{p}$: Vector of planned tasks,
$\mathbf{s}$: Vector of started tasks,
$\mathbf{f}$: Vector of finished tasks,
h : Index of next task chosen by the heuristic.

The heuristics we will be looking at, will all be implemented in the same way. When scheduling begins, the scheduler asks the heuristic function which task to schedule, and provides it with the system state. The task returned by this heuristic will be started.
For the upcoming heuristics, we will be using the uncertainty set in table 1.

	sc. 1	sc. 2	sc. 3	sc. 4
task 1	8	8	8	7
task 2	3	2	3	2
task 3	6	6	4	4
task 4	7	9	9	10

Table 1: Uncertainty set for the heuristic examples.

5.1 Blind heuristic

This heuristic will not be tested, but is useful for finding an upper bound for the total makespan. It starts the tasks based on their index in the uncertainty set.

$$h(U, \mathbf{d}, \mathbf{p}, \mathbf{s}, \mathbf{f}) = \min\{i : \quad \mathbf{p}_i = 1\}$$

This heuristic would choose task 1 from the example uncertainty set.

5.1.1 Upper bound for Blind heuristic

We already know, from lemma 4.1, that for any scenario $\hat{\mathbf{u}} \in U$,

$$t \geq \frac{1}{2}||\hat{\mathbf{u}}||_1 = t_{\min}$$

is a lower bound. We now try to find an upper bound for the makespan given any scenario in U.

Theorem 5.1. *For any scenario $\boldsymbol{u} \in \mathbb{R}^n_{>0}$, if $\boldsymbol{u}$ meets the condition*

$$\frac{u_i}{2} \leq u_{i+1}, \quad \forall i \in [n-1],$$

then

$$t \leq \frac{||\boldsymbol{u}||_1}{2} + \frac{u_n}{2}. \tag{8}$$

Proof. Let $\mathbf{u}$ be an arbitrary element of $\mathbb{R}^n_{>0}$, satisfying the condition

$$\frac{u_i}{2} \leq u_{i+1}, \quad \forall i \in [n-1],$$

where we also define

$$\mathbf{u}^k = \begin{bmatrix} u_1 & u_2 & \cdots & u_k \end{bmatrix}^\top, \quad k \leq n.$$

The proof will be done using induction. We use the induction hypothesis

$$
\begin{aligned}
P(k) : &\text{ If } k = 1, \\
&\text{or } k \geq 2, \text{ and } \frac{u_i^k}{2} \leq u_{i+1}^k, \quad i \in [k-1], \\
&\text{then, for the makespan we have } t \leq \frac{||\mathbf{u}^k||_1}{2} + \frac{u_k^k}{2}.
\end{aligned}
$$

We will first prove P for $k = 1$. In this case $\mathbf{u}^1 = u_1 \Rightarrow ||\mathbf{u}^1||_1 = u_1$. There is only one task, so when this finishes, $t = u_1$. According to $P(1)$,

$$t \leq \frac{||\mathbf{u}^1||_1}{2} + \frac{u_1^1}{2} = \frac{u_1}{2} + \frac{u_1}{2} = u_1.$$

So this shows $P(1)$ is true.
For the inductive step, suppose $P(p)$ is true for a particular $p < n$. We now need to show $P(p+1)$ is true as well.
From $P(p)$, we know that for $\mathbf{u}^p$, $t \leq \frac{||\mathbf{u}^p||_1}{2} + \frac{u_p}{2}$. We also know

$$\mathbf{m}_1^p \mathbf{u}^p + \mathbf{m}_2^p \mathbf{u}^p = ||\mathbf{u}^p||_1, \tag{9}$$

$$\frac{||\mathbf{u}^p||_1}{2} \leq t. \quad \text{(fom lemma 4.1)} \tag{10}$$

Suppose w.l.o.g., that $\mathbf{m}_1^p \mathbf{u}^p \geq \mathbf{m}_2^p \mathbf{u}^p$. This means that $\mathbf{m}_1^p \mathbf{u}^p = t$, since $t = \max(\mathbf{m}_1^p \mathbf{u}^p, \mathbf{m}_2^p \mathbf{u}^p)$. From this it follows that

$$\frac{||\mathbf{u}^p||_1}{2} \leq \mathbf{m}_1^p \mathbf{u}^p \leq \frac{||\mathbf{u}^p||_1}{2} + \frac{u_p}{2}, \quad \text{(these are the bounds for } t) \tag{11}$$

$$\frac{||\mathbf{u}^p||_1}{2} \geq \mathbf{m}_2^p \mathbf{u}^p \geq \frac{||\mathbf{u}^p||_1}{2} - \frac{u_p}{2}. \quad (\text{since } \mathbf{m}_2^p \mathbf{u}^p = ||\mathbf{u}^p||_1 - \mathbf{m}_1^p \mathbf{u}^p) \tag{12}$$

Next, we start the new task. It starts on machine 2, since

$$\mathbf{m}_2^p \mathbf{u}^p \leq \mathbf{m}_1^p \mathbf{u}^p,$$

which means machine 2 has had tasks that have a total duration shorter than those of machine 1. Starting the next task $p+1$ on machine 2 has no effect on machine 1:

$$
\begin{aligned}
\mathbf{m}_1^{p+1} \mathbf{u}^{p+1} &= \mathbf{m}_1^p \mathbf{u}^p \\
\mathbf{m}_2^{p+1} \mathbf{u}^{p+1} &= \mathbf{m}_2^p \mathbf{u}^p + u_{p+1}
\end{aligned}
$$

We now update the equations (11) and (12).

$$\frac{||\mathbf{u}^p||_1}{2} \leq \mathbf{m}_1^{p+1} \mathbf{u}^{p+1} \leq \frac{||\mathbf{u}^p||_1}{2} + \frac{u_p}{2}, \tag{13}$$

$$\frac{||\mathbf{u}^p||_1}{2} + u_{p+1} \geq \mathbf{m}_2^{p+1} \mathbf{u}^{p+1} \geq \frac{||\mathbf{u}^p||_1}{2} - \frac{u_p}{2} + u_{p+1}. \tag{14}$$

We now calculate the difference:

$$-u_{p+1} \leq \mathbf{m}_1^{p+1}\mathbf{u}^{p+1} - \mathbf{m}_2^{p+1}\mathbf{u}^{p+1} \leq u_p - u_{p+1}. \tag{15}$$

On the right side, we notice the expression $u_p - u_{p+1}$. By our assumption that $P(p)$ is true, we have $\frac{u_p}{2} \leq u_{p+1}$. In other words: $u_p \leq 2u_{p+1}$. So now $u_p - u_{p+1} \leq 2u_{p+1} - u_{p+1} = u_{p+1}$. Equation (15) becomes

$$-u_{p+1} \leq \mathbf{m}_1^{p+1}\mathbf{u}^{p+1} - \mathbf{m}_2^{p+1}\mathbf{u}^{p+1} \leq u_{p+1}.$$

Or equivalently

$$|\mathbf{m}_1^{p+1}\mathbf{u}^{p+1} - \mathbf{m}_2^{p+1}\mathbf{u}^{p+1}| \leq u_{p+1} \tag{16}$$

As seen before, we can write $t = \max(\mathbf{m}_1^{p+1}\mathbf{u}^{p+1}, \mathbf{m}_2^{p+1}\mathbf{u}^{p+1})$ as

$$t = \frac{\mathbf{m}_1^{p+1}\mathbf{u}^{p+1} + \mathbf{m}_2^{p+1}\mathbf{u}^{p+1} + |\mathbf{m}_1^{p+1}\mathbf{u}^{p+1} - \mathbf{m}_2^{p+1}\mathbf{u}^{p+1}|}{2}$$

Using equation (9) and (16) we get

$$t \leq \frac{||\mathbf{u}^{p+1}||_1 + u_{p+1}}{2}$$

Which is equivalent to

$$t \leq \frac{||\mathbf{u}^{p+1}||_1}{2} + \frac{u_{p+1}}{2}$$

This shows that $P(p+1)$ holds. □

This means, that if we can find a sequence (a_i) of the tasks, such that $u_{a_{i+1},j} \geq \frac{1}{2}u_{a_i,j}$, for all $i \in [n-1]$, $j \in [s]$, we can conclude that, given a scenario $\hat{\mathbf{u}} \in U$,

$$\frac{||\hat{\mathbf{u}}||_1}{2} \leq t \leq \frac{||\hat{\mathbf{u}}||_1}{2} + \frac{\hat{u}_{a_n}}{2}.$$

Ideally, we want to have a heuristic which chooses the shortest task last. since u_{a_n} determines the upper bound for t.

5.2 Longest first

Following the findings of section 5.1.1, we introduce the *longest first* heuristic. Whenever a task needs to be scheduled, the heuristic considers all possible scenarios, and chooses the task which has the highest possible duration. In the uncertainty set given in table 1, the first task to be scheduled would be task 4, after that comes task 1.
This heuristic can be described using the system state as follows:

$$h_{\mathrm{LF}}(U, \mathbf{d}, \mathbf{p}, \mathbf{s}, \mathbf{f}) = h \in \mathbf{p} : \max(U(\mathbf{d})_h) \geq \max(U(\mathbf{d})_i), i \in \mathbf{p}.$$

$\max(U(\mathbf{d})_i)$ describes the maximum duration across all feasible scenarios of task i.

5.3 Decisiveness-based heuristics

When a task finishes it reveals information about which scenarios are feasible and which are not, and this can be used to our advantage to narrow down the amount of possible scenarios to consider. Some tasks give us more information about possible remaining scenarios than others. This property will be referred to as *decisiveness*.
The point of trying to schedule more decisive tasks, is to reduce the number of feasible scenarios to 1, and then switching to linear programming, since this will definitely be the optimal solution from that point on. The sooner we are left with only one feasible scenario, the more options the the linear program has for optimizing. How to determine which task reveals most information? There are a few different ways.

5.3.1 Decisive first I: Maximum different outcomes

The first approach is to count for each task individually, how many different duration times are possible, and choose the task where this is maximised. In the example uncertainty set this would be $(2, 2, 2, 3)$ for the respective tasks, only task 4 has three different possible durations, this heuristic would choose task 4.

5.3.2 Decisive first II: Minimum leftover scenarios

This heuristic will first calculate for every task the maximum amount of scenarios left after each task has finished, and then chooses the task where this number is lowest. In the example this would give $(3, 2, 2, 2)$ for the respective tasks. In this case it chooses the first occurrence of the minimum, which is task 2.

5.3.3 Decisive first III: Minimum expected value of leftover scenarios

This heuristic functions very similar to the previous one, except this one finds for every task the expected value of leftover scenarios (assuming uniform probability) and takes the minimum. In the example this would give $(2.5, 2, 2, 1.5)$ for the respective scenarios. The heuristic would choose task 4.

6 Comparison of heuristics

In this section we will have a look at the performance of the heuristics above. They have been tested using an uncertainty set U, given by the following pa-

rameters, and obtained as described in section 2 and 2.1:

$$\begin{aligned}
\hat{\mathbf{d}} &= \begin{bmatrix} 5 & 5 & 6 & 6 & 5 & 6 & 7 & 5 & 6 & 8 \end{bmatrix}, \\
\mathbf{w} &= \begin{bmatrix} 4 & 1 & 1 & 2 & 5 & 2 & 2 & 3 & 4 & 1 \end{bmatrix}, \\
\hat{\mathbf{z}} &= \begin{bmatrix} 3 & 4 & 5 & 7 & 2 & 3 & 6 & 4 & 1 & 1 \end{bmatrix}, \\
\alpha &= 0.55.
\end{aligned}$$

The resulting uncertainty set U contains 1007 scenarios. Every scenario has been given to the heuristics, which did not know the scenario beforehand. Their performance has been compared to a linear program, which *did* know the scenario, and this will give the optimal solution, which serves as the lower bound for the makespan t of a given scenario $\hat{\mathbf{u}}$ and will be referred to as the *answer*.

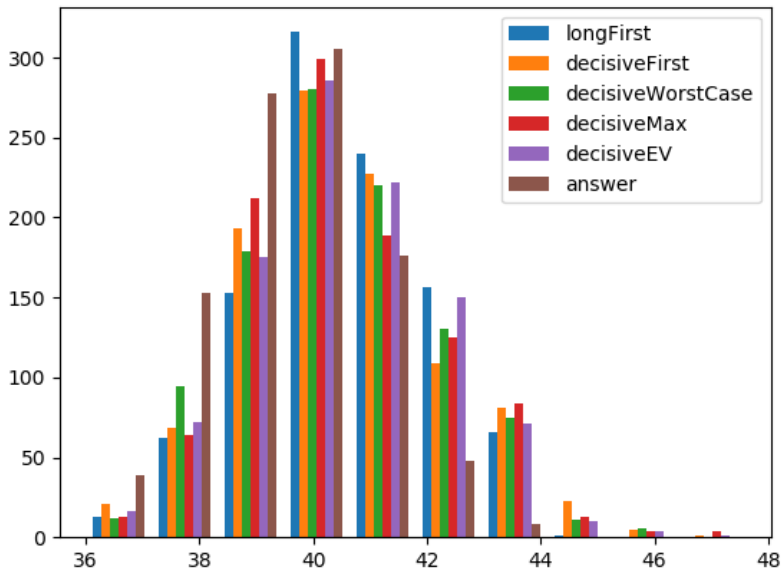

Figure 2: Histogram of heuristics, compared with the *answer* (opimal solution).

In figure 2 we can see the performance of the heuristics, with U as given before. On the horizontal axis we have the makespan t, on the vertical axis we have frequency. From this plot, it seems that *decisiveFirst* gets closest to the answer most of the time, since we see it has the highest frequency of the first bin ($36 \leq t \leq 37$).

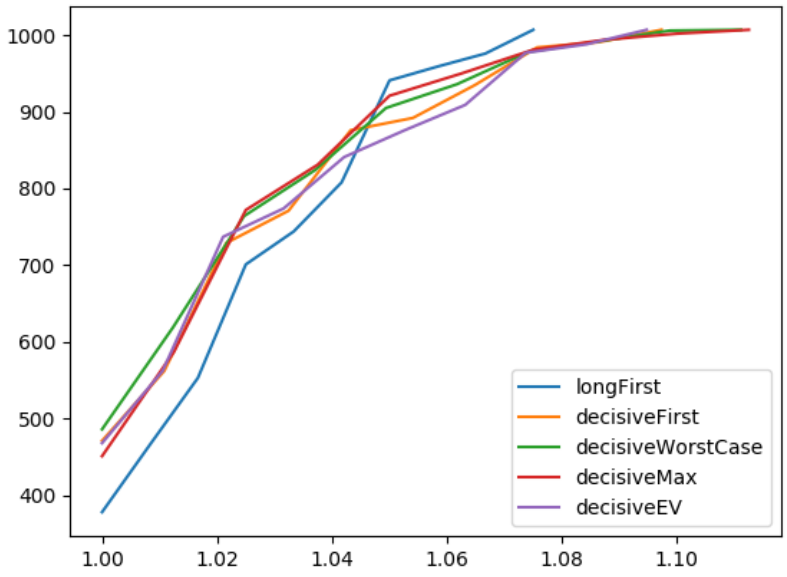

Figure 3: Same data as image 2, cumulative plot.

In figure 3, we have the same data as in figure 2, this time in a cumulative plot. We see that *longFirst* starts out as the worse heuristic, and then ends up on top. This means that the worst case scenarios for *longFirst* give a better makespan than the worst case scenarios for the other heuristics. This probably because the *longFirst* eliminates all tasks with the longest duration first. So in the end, when there are only a few tasks left, it is not such a problem when it picks the wrong task (as concluded in section 5.1.1). The other heuristics all have the same issue that they can keep the task with the longest duration until the very end of the scheduling, and because of this the makespan ends up a lot higher.

7 Conclusion

In conclusion, we have seen that there many different kinds of heuristics to try and minimize the total makespan of scheduling problems with uncertainty. The simplest heuristic, *longestFirst*, performed the best for worst case scenarios, with the given uncertainty set. This is probably a consequence of the fact that this heuristic puts the tasks with shorter duration last, and this minimizes the risk for making a mistake in scheduling. Also, if the condition in theorem 5.1 is

met, we can say that for any given scenario $\hat{\mathbf{u}}$,

$$\frac{||\hat{\mathbf{u}}||_1}{2} \leq t \leq \frac{||\hat{\mathbf{u}}||_1}{2} + \frac{\hat{u}_{a_n}}{2},$$

where $\hat{u}_{a_n}$ is the duration fo the task chosen last.

8 Discussion

Other possibilities of heuristics that can still be investigated include more diversifications or 'mash-ups' of heuristics mentioned in this paper. For example, start out using one particular heuristic, and switching to another halfway. We have already seen this in the decisive heuristics, which switch to linear programming when there is only one scenario left.
Theorem 5.1 can be generalized to:
If

$$\forall i \in [n-1], \exists m \in [n-i] : \sum_{j=1}^{m} u_{i+j} \geq \frac{m}{m+1} u_i,$$

then

$$\frac{||\mathbf{u}||_1}{2} \leq t \leq \frac{||\mathbf{u}||_1}{2} + \frac{u_n}{2}.$$

This proof is a bit more involved than the proof for the special case $m = 1$, given in theorem 5.1. However when generalized, it is much stronger.

References

Cohen, I., Postek, K., & Shtern, S. (2021). An adaptive robust optimization model for parallel machine scheduling. https://arxiv.org/abs/2102.08677

Li, Z., & Floudas, C. A. (2011). Robust counterpart optimization: Uncertainty sets, formulations and probabilistic guarantees. http://focapo.cheme.cmu.edu/2012/proceedings/data/papers/030.pdf

Pinedo, M. L. (2008). *Scheduling: Theory, algorithms and systems*. Springer.