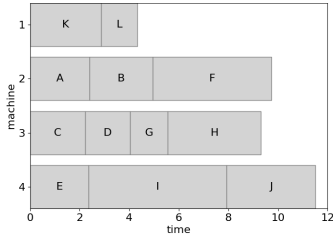


# Explaining Makespan Schedules

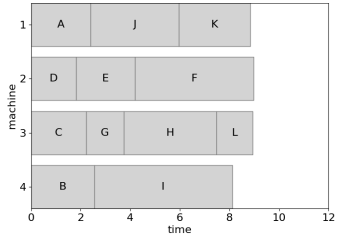
Myles Lee

25<sup>th</sup> June 2019

# Introduction

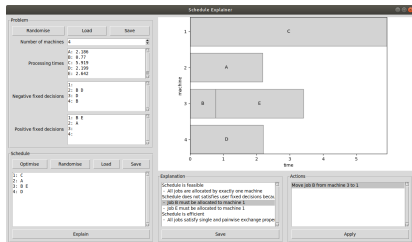


optimise  
→

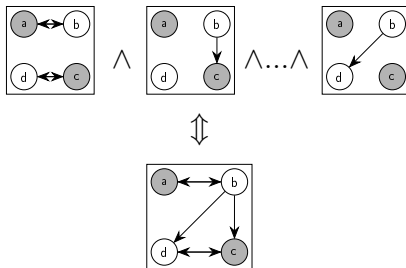


- Solvers and schedules are difficult to understand
- Apply argumentation to makespan schedules to generate explanations

# Contributions

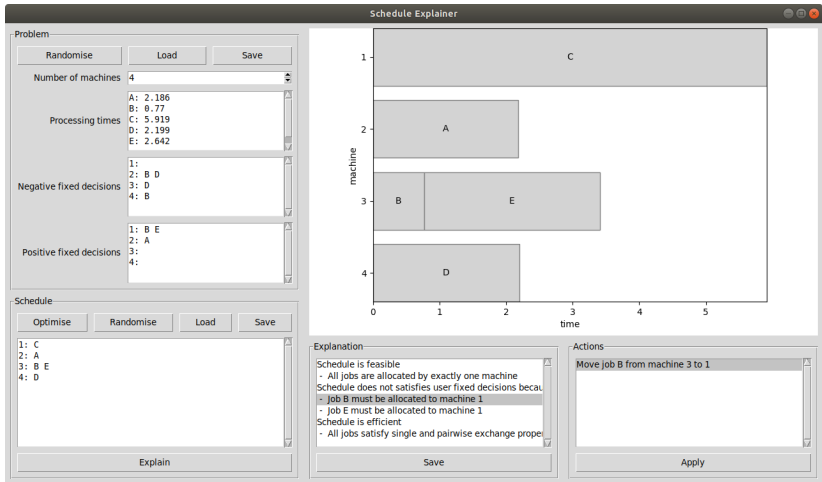


- Interactive tool
- Algorithms



- Theoretical extensions
- Discussion

# Demonstration



# Abstract Argumentation

- An abstract argumentation framework is a directed graph  $\langle Args, \rightsquigarrow \rangle$ .
- Extension  $E$  is a subset of  $Args$ .

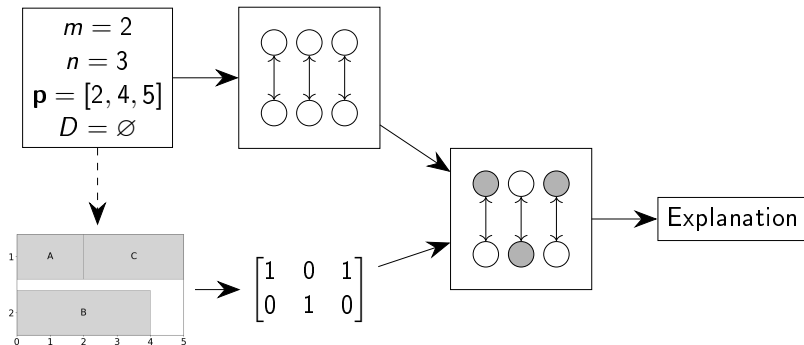
## Definition

$E$  is conflict-free on  $\langle Args, \rightsquigarrow \rangle$  iff  $\forall a, b \in E. a \not\rightsquigarrow b$ .

## Definition

$E$  is stable on  $\langle Args, \rightsquigarrow \rangle$  iff  $E$  is conflict-free on  $\langle Args, \rightsquigarrow \rangle$  and  $\forall a \in Args \setminus E. \exists e \in E. e \rightsquigarrow a$ .

# Pipeline

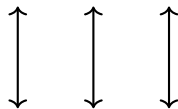


# Feasibility Framework Construction

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\forall j \in \mathcal{J}. \sum_{i \in \mathcal{M}} x_{ij} = 1$$

$\langle 1, a \rangle \quad \langle 1, b \rangle \quad \langle 1, c \rangle$



$\langle 2, a \rangle \quad \langle 2, b \rangle \quad \langle 2, c \rangle$

$$\text{Args}_F = \mathcal{M} \times \mathcal{J}$$

$$\langle i, j \rangle \rightsquigarrow_F \langle i', j' \rangle \text{ iff } i \neq i' \wedge j = j'$$

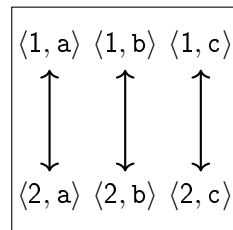
# Framework Modelling

## Definition

Framework  $\langle \text{Args}, \rightsquigarrow \rangle$  stability-models a property  $P$  iff  
 $\forall E \subseteq \text{Args}. E \text{ is stable on } \langle \text{Args}, \rightsquigarrow \rangle \Leftrightarrow P$

$$P_F \equiv \forall j \in \mathcal{J}. \sum_{i \in \mathcal{M}} x_{i,j} = 1$$

$\langle \text{Args}_F, \rightsquigarrow_F \rangle$



## Theorem

$\langle \text{Args}_F, \rightsquigarrow_F \rangle$  stability-models  $P_F$



# Algorithm Notation

$$\mathbf{0}^{2 \times 2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\ominus \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \oslash \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \oslash \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

# Construct-Feasibility Algorithm

---

## Algorithm

---

```
1: function Construct-Feasibility( $m, n$ )
2:    $\rightarrow_F \leftarrow \mathbf{0}^{(m \times n)^2}$ 
3:   for  $i_1, i_2 \in \mathcal{M}, j \in \mathcal{J}$  do
4:     if  $i_1 \neq i_2$  then
5:        $\rightarrow_{F_{i_1 j, i_2 j}} \leftarrow 1$ 
6:     end if
7:   end for
8:   return  $\rightarrow_F$ 
9: end function
```

---

# Explain-Stability Algorithm

---

## Algorithm

---

```

1: function Explain-Stability( $\mathbf{x}$ ,  $\Rightarrow$ ,  $\bar{\mathbf{u}}$ ,  $\bar{\mathbf{c}}$ )
2:    $\mathbf{u} \leftarrow \neg \mathbf{x}$ 
3:    $\mathbf{c} \leftarrow \mathbf{0}^{(m \times n)^2}$ 
4:   for  $i \in \mathcal{M}, j \in \mathcal{J}$  do
5:     if  $x_{i,j} = 1$  then
6:        $\mathbf{u} \leftarrow \mathbf{u} \wedge \neg \Rightarrow_{i,j}$ 
7:        $c_{i,j} \leftarrow \mathbf{x} \wedge \Rightarrow_{i,j}$ 
8:     end if
9:   end for
10:   $\mathbf{u} \leftarrow \mathbf{u} \wedge \neg \bar{\mathbf{u}}$ 
11:   $\mathbf{c} \leftarrow \mathbf{c} \wedge \neg \bar{\mathbf{c}}$ 
12:  return  $\langle \mathbf{u}, \mathbf{c} \rangle$ 
13: end function

```

# Explain-Feasibility Algorithm

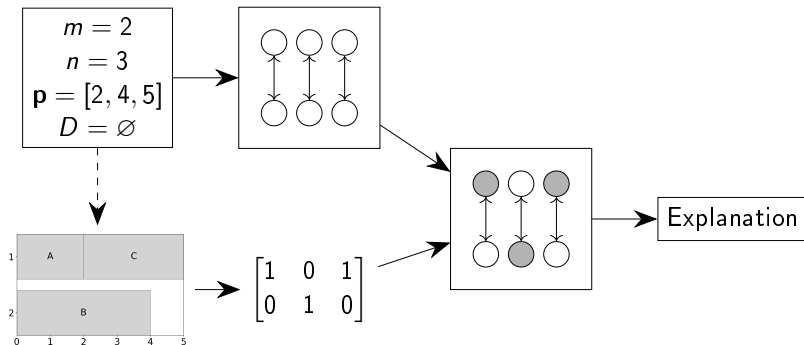
## Algorithm

```

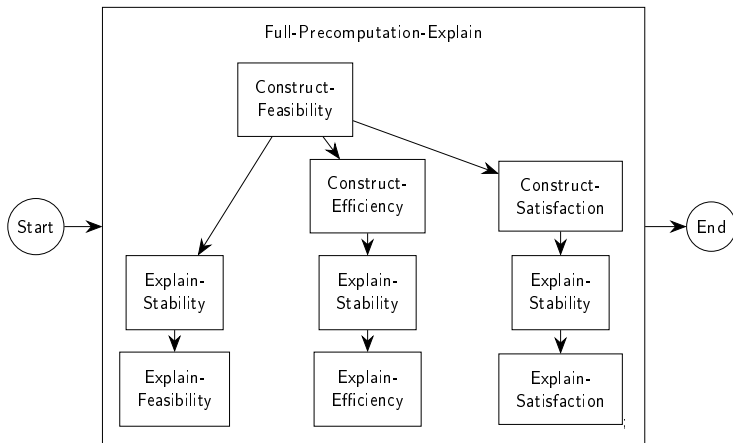
1: function Explain-Feasibility( $u, c$ )
2:   if  $m = 0$  then
3:     if  $n = 0$  then
4:       There are no jobs, so the schedule is trivially feasible.
5:     else
6:       There are no machines to allocate to jobs.
7:     end if
8:   else
9:     if  $u = 0 \wedge c = 0$  then
10:      All jobs are allocated to exactly one machine.
11:    else
12:      for  $j \in \mathcal{J}$  do
13:        if  $u_{:,j} = 1$  then
14:          Job  $j$  is not allocated to any machine.
15:        end if
16:        if  $c_{:,j,:j} \neq 0$  then
17:          Job  $j$  is over-allocated to machines  $\{i \mid i \in \mathcal{M}, x_{i,j} = 1\}$ .
18:        end if
19:      end for
20:    end if
21:  end if
22: end function

```

# Pipeline

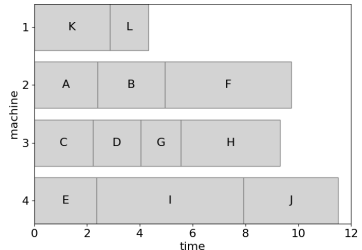


# Algorithms Overview



# Generating Efficiency Explanations

- Single and pair-wise exchange properties
- Fixed decision awareness
- Explanations are:
  - superfluous
  - local
  - expensive



# Generating Fixed Decisions Explanations

---

## Algorithm

---

```

1: function Explain-Satisfaction( $D$ ,  $u$ ,  $c$ )
2:   for  $j \in \mathcal{J}$  do
3:     if  $\exists i \in \mathcal{M} \langle i, j \rangle \notin D^-$  then
4:       Job  $j$  cannot be allocated to any machine.
5:     end if
6:     if  $D^-$  and  $D^+$  are not disjoint then
7:       Job  $j$  has conflicting negative and positive fixed decisions.
8:     end if
9:     if  $|\{i \in \mathcal{M} \mid \langle i, j \rangle \in D^+\}| > 1$  then
10:      Job  $j$  cannot be allocated to multiple machines.
11:    end if
12:  end for
13:  ...
14: end function

```

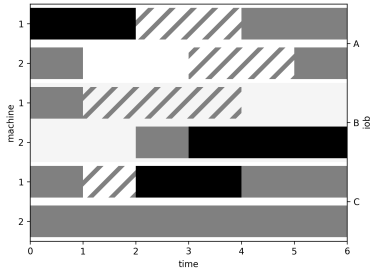
---



# Partial-Precomputation Optimisation

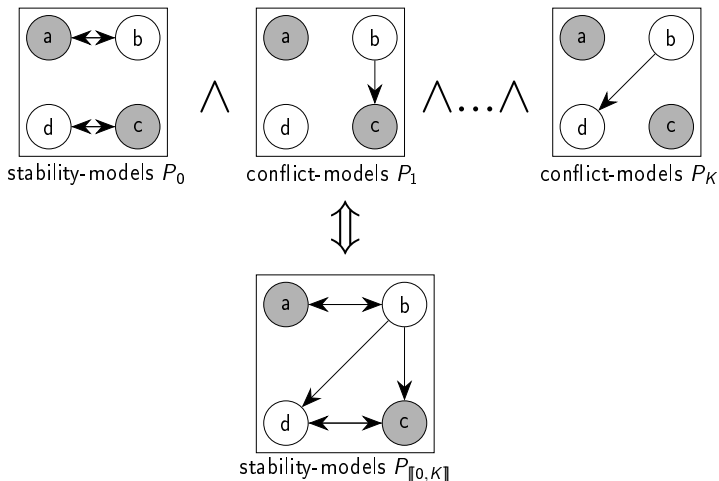
- $\rightarrow$  construction has  $\mathcal{O}(m^2 n^2)$  memory usage.
  - Inline stability and framework construction algorithms
  - Operate on sub-graphs  $\rightarrow_{i,j}$
- 
- Full-Precomputation complexity:  $\mathcal{O}(m^2 n^2)$
  - Partial-Precomputation complexity:  $\mathcal{O}(mn^2)$

# Time-indexed Interval Scheduling



- Single-allocation of jobs
- No overlapping jobs
- Start and finish times
- Fixed decisions

# Union of Modelling Frameworks Theorem



# Theoretical Applications

## Definition

Property  $P$  is stability-modellable iff  $\exists \langle \text{Args}, \rightsquigarrow \rangle. \langle \text{Args}, \rightsquigarrow \rangle$  stability models  $P$ .

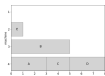
## Theorem

Interval scheduling feasibility is stability-modellable.

# Questionnaire

## Makespan Schedule Explanation Questionnaire

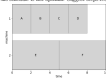
This questionnaire should take less than ten minutes. This will judge the general ability to understand and explain makespan schedules. Makespan schedules consist of  $m$  machines and  $n$  jobs. Every job is assigned to exactly one machine for the schedule to be feasible. Each job has a processing time. The objective is to minimize the longest collective processing time. Machines and jobs are denoted by integers 1, 2, 3, ... and by letters A, B, C, ... respectively. A schedule is optimal when the longest collective processing time cannot be reduced.



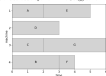
Schedule is not optimal because jobs A, C, D can be moved to machine 1. Schedule is optimal because no assignment can be improved.

Also to spend at most one minute for each question. Some questions are difficult. Write your answers in the boxes.

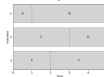
1. This schedule is not optimal. Suggest steps required to optimize this schedule.



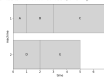
2. Is this schedule optimal? If not, suggest how it could be improved immediately.



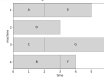
3. This schedule is optimal. Job A cannot be assigned to machine 1 or 2 anymore. How could the schedule be modified to agree with this constraint?



4. Either job C and D, or job B and E must be assigned to the same machine. Which constraint results in a better schedule, and why?

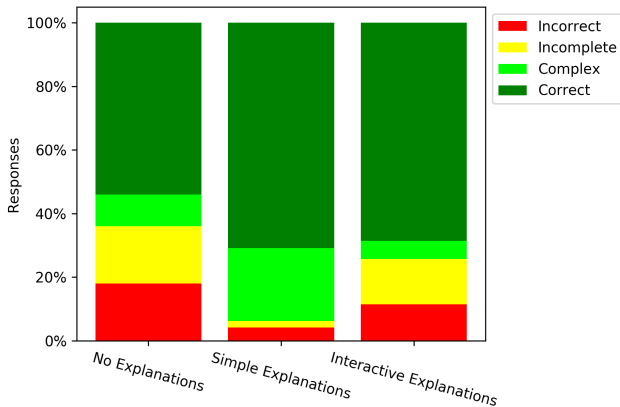


5. A new job H of 5 time units needs to be scheduled. How could the schedule be modified to best accommodate this job?



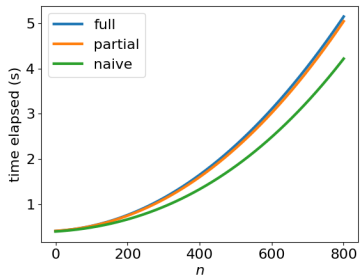
Thank you for your time. Please send your answers to [myles.lee@imperial.ac.uk](mailto:myles.lee@imperial.ac.uk).

# Questionnaire Results

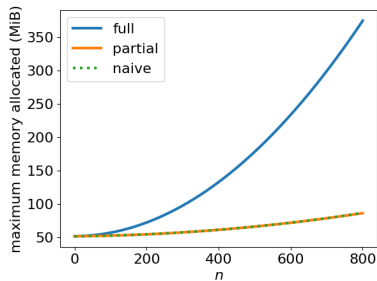


# Profiling

■  $m = 10$



CPU Performance



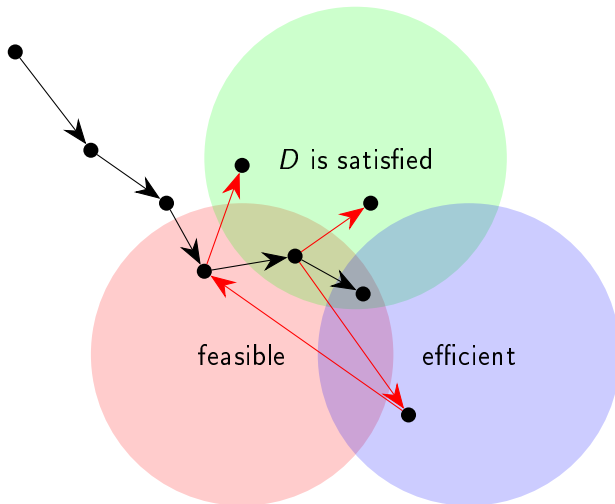
Memory Performance

# Limitations with Argumentation

- Memory performance
- Computational performance
- Abstracted interface
- Functional equivalence
- Implementation Complexity



# Practicability of Argumentation



# Summary

Explaining makespan schedules using argumentation is practically possible but not practically suitable.