Imperial College London

Undergraduate Research Opportunities Programme Final Report

# Properties of Makespan Scheduling in Argumentation

*Author:*
Yuk Lun Chiu

*Supervisors:*
Dr. Kristijonas Čyras
Prof. Francesca Toni

## Contents

**1 Introduction** **2**

**2 Problem Formulation** **3**

2.1 Makespan Scheduling Problem . . . 3

2.2 Abstract Argumentation . . . 3

2.3 Assumption-Based Argumentation . . . 5

**3 Common Constraints and their Properties** **6**

3.1 Scheduling Additional Constraints . . . 6

3.2 Properties of Finitary and Binary Constraints . . . 9

**4 Dynamics of Makespan Scheduling** **14**

4.1 Exposing Feasible Schedules to Change . . . 14

4.2 Minimum Swaps under Change . . . 22

4.3 Bounding Swaps under Common Constraints . . . 24

**5 Discussion and Future Work** **32**

**6 References** **36**

## 1 Introduction

Scheduling is well-established problem in computer science, with Garey (1976) proving the job shop variation of the makespan scheduling problem to be NP-complete for instances with more than 2 machines in [4]. Although deterministic algorithms have been developed to handle scheduling, it is difficult to reason why a certain schedule returned by the algorithm is truly optimal or even if a schedule is feasible, which is conversely where the field of Argumentation flourishes.

In this report, by coupling the makespan scheduling problem with computational argumentation, we investigate the properties of the argumentation framework modelling the makespan problem when applying common constraints to the makespan scheduling problem. The crux of the report, however, is an exploration on obtaining feasible schedules upon removing or adding a machine and job. We extend this further by giving algorithms which correctly reschedules a previously feasible schedule (upon removing or adding a machine or job) while bounding the number of disruptions made to a schedule. In the final section, we attempt to bound the number of disruptions when adding a job under constraints.

To summarise the main results in the report, we show that it is impossible to map finitary constraints in inequality form to binary constraints in inequality under the makespan framework. With regards to dynamics, we give explicit formulas for feasible schedules upon removal or addition of a job or machine. We also show that in most cases, it is possible to amend a schedule such that the number of disruptions to the schedule is much lower than the number of jobs.

# 2 Problem Formulation

## 2.1 Makespan Scheduling Problem

We first begin by describing the problem. The makespan scheduling problem is a common variant of scheduling problems in computer science; let $\mathcal{J} = \{J_1, ..., J_n\}$ be independent tasks with corresponding running times $\mathcal{P} = \{p_1, ..., p_n\}$ which need to be executed by $m$ machines $\mathcal{M} = \{M_1, ..., M_m\}$.

For an arbitrary schedule $S$, denote $C_i(S)$ to be the time it takes to execute all tasks scheduled for machine $M_i$. The value $C_{\max}(S) := \max_{1\leq i\leq m}\{C_i(S)\}$ is called the *makespan* of the schedule. The *makespan scheduling problem (MSP)* is to find $\min_{S\in\mathbb{S}}\{C_{\max}(S)\}$, where $\mathbb{S}$ is the space of all possible schedules.

To formalize this, let us denote $x_{i,j}$ to be 1 if job $J_j$ is executed by machine $M_i$, and 0 otherwise. Then we have the following expressions and equations:

$$\text{Objective: } \min_{S\in\mathbb{S}}\{C_{\max}(S)\} \tag{1}$$

$$\sum_{i=1}^{m} x_{i,j} = 1 \tag{2}$$

$$C_i = \sum_{j=1}^{n} x_{i,j}p_j \tag{3}$$

## 2.2 Abstract Argumentation

To capture the makespan problem in Argumentation, we apply Dung's abstract argumentation (AA) framework. Below are some basic definitions adapted from [3]:

**Definition 2.1. (AAF)** An abstract argumentation framework is a pair $< \mathcal{A}, \mathcal{R} >$, where $\mathcal{A}$ is the set of arguments and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ is binary relation on $\mathcal{A}$. Here $\mathcal{R}$ is called set of attacks.

**Definition 2.2. (Conflict-freeness and defence)** Let $AF =< \mathcal{A}, \mathcal{R} >$ be given.

1. We say $E \to a$ if $\exists b \in E$ such that $(b, a) \in \mathcal{R}$.
2. We say $a \to E$ if $\exists b \in E$ such that $(a, b) \in \mathcal{R}$.
3. We say $E \subseteq \mathcal{A}$ is conflict-free if and only if $\nexists a,\, b \in E$ such that $(a, b) \in E$.
4. We say $E$ defends $a$ if and only if, for each $(b, a) \in \mathcal{R}$, $\exists c \in E$ such that $(c, b) \in \mathcal{R}$.

In this report, we refer exclusively to stable and preferred extensions, which are subsets of $\mathcal{A}$ that satisfy the following conditions:

**Definition 2.3. (Stable and preferred extensions)** Let $AF =< \mathcal{A}, \mathcal{R} >$ be given.

1. If $E \subseteq \mathcal{A}$ is conflict-free and attacks each element belonging to $\mathcal{A} \setminus E$, then $E$ is a stable extension. We denote $\mathcal{E}_{\mathcal{ST}}(AF)$ to be the set of all stable extensions.

2. If $E \subseteq \mathcal{A}$ is conflict-free and defends all its elements, then $E$ is admissible. If $E \subseteq \mathcal{A}$ is admissible and maximal with respect to set inclusion, then $E$ is a preferred extension. We denote $\mathcal{E}_{\mathcal{PR}}(AF)$ to be the set of all preferred extensions.

We follow the model of basic feasibility for the makespan problem given in [2]. To capture the notion of time, allot each machine $r$ slots in which jobs can be executed. Each slot in a machine can only be assigned a job if the previous job has been completed; for example, if $J_1$ with processing time $p_1$ is assigned to $M_1$ on slot 1, then another job can only be assigned to $M_1$ on slot $1 + p_1$ or greater. In this sense, minimising the makespan equates to minimising the time variable $s$ across all machines.

**Definition 2.4. (Basic feasibility in AA, with slots)** The basic (standard) feasibility model with slots is given by the framework $AF =< \mathcal{A}, \mathcal{R} >$, where

$$\mathcal{A} = \{a_{i,j,s} : x_{i,j} \text{ is an assignment variable}, \ s \in \{1, ..., r\}\}$$
$$\mathcal{R} = \{(a_{i,j,s}, a_{k,l,t}) : (i \neq k \wedge j = l) \vee (i = k \wedge j = l \wedge s \neq t) \vee (i = k \wedge j \neq l \wedge s = t)\}$$

It should be noted now that for the remainder of this paper, we set $p_j = 1$ for the sake of simplicity. We believe it is possible to extend most of the results below to cases where $p_j > 1$ for some job $J_j$ under more complex conditions; for example, we would have to take $AF =< \mathcal{A}, \mathcal{R} \cup R_j >$ where

$$R_j = \{(a_{i,j,s}, a_{i,l,t}) : j \neq l, \ s \leq t < s + p_j\}$$

This model captures that the feasibility space of a makespan schedule is exactly the set of extensions of the standard makespan argumentation framework, proved in [2]:

**Lemma 2.5.** Let $E \in \mathcal{E}_{\mathcal{S}}(AF)$ for $\mathcal{S} \in \{\mathcal{ST}, \mathcal{PR}\}$. Put $x_{i,j,s} = 1$ if $a_{i,j,s} \in E$. Then

$$\sum_{i=1}^{m} \sum_{s=1}^{r} x_{i,j,s} = 1$$

for any $j \in \{1, ..., n\}$. In other words, every stable or preferred extension in $AF$ is a feasible makespan schedule.

Note that in some cases, it suffices to consider a simpler case where machines only have one slot, and $p_j = 1$ for all $j \in \{1, ...n\}$. In this case we drop the third component, leading to a much simpler AA framework defined below:

**Definition 2.6. (Basic feasibility in AA, no slots)** The basic feasibility model with no slots is given by the framework $AF =< \mathcal{A}, \mathcal{R} >$, where

$$\mathcal{A} = \{a_{i,j} : x_{i,j} \text{ is an assignment variable}\}$$
$$\mathcal{R} = \{(a_{i,j}, a_{k,l}) : i \neq k \wedge j = l\}$$

We will ensure it is clear which definition is used in each section of the report, if distinction is required.

### 2.3 Assumption-Based Argumentation

We will give some basic definitions adapted from [5] below:

**Definition 2.7. (ABA framework)** An ABA framework is a tuple $< \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\phantom{a}} >$, where

1. $\mathcal{L}$ is the language.
2. $\mathcal{R}$ is a set of rules of the form $r \leftarrow r_1, r_2, ..., r_m$.
3. $\mathcal{A}$ is the set of assumptions $\mathcal{A} \subseteq \mathcal{L}$.
4. $\overline{\phantom{a}}$ is a total mapping from $\mathcal{A} \rightarrow \mathcal{L}$, where $\overline{a}$ is the contrary of $a$.

**Definition 2.8. (Deductions, arguments and attacks)**

1. A deduction for $\sigma \in \mathcal{L}$ supported by $S \subseteq \mathcal{L}$ and $R \subseteq \mathcal{R}$, denoted $S \vdash \sigma$, is a (finite) tree with nodes labeled by sentences in $\mathcal{L}$ or by $\tau$ he root labeled by $\sigma$, leaves either $\tau$ or sentences in $S$, non-leaves $\sigma'$ with, as children, the elements of the body of some rule in $\mathcal{R}$ with head $\sigma'$, and $R$ the set of all such rules.
2. An argument for $\sigma \in \mathcal{L}$ supported by $S \subseteq \mathcal{A}$, written $S \vdash \sigma$ is a deduction for $\sigma$ supported by $S$.
3. An argument $S_1 \vdash \sigma_1$ attacks an argument $S_2 \vdash \sigma_2$ if and only if $\sigma_1$ is the contrary of one of the assumptions in $S_2$.

**Definition 2.9. (Stable and preferred semantics)**

1. A set of arguments $E$ is a stable extension if and only if it does not attack itself and it attacks all arguments it does not contain.
2. A set of arguments $E$ is admissible if and only if it does not attack itself and it attacks all arguments that attack it. $E$ is preferred if and only if it is maximally admissible with respect to set inclusion.

We note that from [2], it is possible to preserve a semantic correspondence between AA and ABA by defining a map from AA to ABA.

**Lemma 2.10.** Let $AF =< \mathcal{A}, \mathcal{R} >$ be given. Take $\mathcal{F} =< \mathcal{L}, \mathcal{R}_f, \mathcal{A}_f, \overline{\phantom{a}} >$, where

$$\mathcal{A}_f = \mathcal{A}$$
$$\mathcal{R}_f = \{\overline{b} \leftarrow a : (a, b) \in \mathcal{R}\}$$

Then $E$ is a stable/preferred extension in $AF$ if and only if it is a stable/preferred extension in $\mathcal{F}$.

It is now easy to see that the following definition of basic feasibility in ABA gives the same extensions as the framework in AA:

**Definition 2.11. (Basic feasibility in ABA, with slots)** Let $\mathcal{F} =< \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\phantom{a}} >$ be an ABA framework such that

$$\mathcal{A} = \{a_{i,j,s} : x_{i,j} \text{ is an assignment variable, } s \in \{1, ..., r\}\}$$
$$\mathcal{R} = \{\overline{a_{i,j,s}} \leftarrow a_{k,l,t} : (i \neq k \wedge j = l) \vee (i = k \wedge j = l \wedge s \neq t) \vee (i = k \wedge j \neq l \wedge s = t)\}$$

# 3 Common Constraints and their Properties

## 3.1 Scheduling Additional Constraints

In this section, we expand on the standard makespan framework defined in [2] by imposing common makespan constraints in to the standard model. The constraints we model are:

1. **Concurrent execution** - these constraints ensure that jobs are executed on the same time slot $t$. For example, two organ donors may need to be anaesthetised simultaneously.
2. **Restricted assignment** - these constraints ensure that certain jobs can only be assigned to certain machines; for example, an anesthesiologist should not be be performing heart surgery.
3. **Precedence relations** - these constraints impose an order in which jobs must be completed. For example, a patient needs to be anesthetised before surgery.
4. **Forbidden sets** - these constraints model that jobs should not be executed in parallel.

While AA is adept at capturing binary attacks between arguments, common makespan constraints often rely on logical disjunctions rather than conjunctions, which makes these constraints difficult to capture in AA. ABA is thus a better choice to model these constraints. Additionally, to avoid confusion with constraints, we shall denote

$$\mathcal{R}_{MSP} = \{\overline{a_{i,j,s}} \leftarrow a_{k,l,t} : (i \neq k \wedge j = l) \vee (i = k \wedge j = l \wedge s \neq t) \vee (i = k \wedge j \neq l \wedge s = t)\}$$

the standard makespan rules. Let us begin with concurrent execution:

**Definition 3.1. (Concurrent execution)** Denote $[J_p, J_{p+1}, ..., \ J_q]_{CE}$ as jobs to be concurrently executed. Given some time slot $t$ where jobs $\{J_p, J_{p+1}, ..., \ J_q\}$ should be executed in parallel (on different machines), in which $t$ can be randomly chosen if no restraint is given, we represent concurrent execution by

$$\sum_{k=1}^{m} \sum_{l=p}^{q} x_{k,l,s} = \begin{cases} q-p & s = t \\ 0 & s \neq t \end{cases}$$

Now we show that the addition of $\mathcal{R}_{CE} := \{\overline{a_{i,j,s}} \longleftarrow a_{k,l,t} \mid j,l \in \{p,...,q\}, \ s \neq t\}$ to the standard makespan rules allows the ABA framework $\mathcal{F}$ to satisfy the concurrent execution constraint.

**Lemma 3.2.** Suppose $[J_p, J_{p+1}, ..., \ J_q]_{CE}$. Let $\mathcal{F} =< \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\phantom{a}} >$ be an ABA framework such that $\mathcal{R} \supseteq \mathcal{R}_{MSP} \cup \mathcal{R}_{CE}$, where

$$\mathcal{R}_{CE} := \{\overline{a_{i,j,s}} \longleftarrow a_{k,l,t} \mid j,l \in \{p,...,q\}, \ s \neq t\}$$

Then $\forall E \in \mathcal{E}_S(\mathcal{F})$ such that $\mathcal{S} \in \{\mathcal{ST}, \mathcal{PR}\}$, $E$ satisfies

$$\sum_{k=1}^{m} \sum_{l=p}^{q} x_{k,l,s} = \begin{cases} q-p & s = t \\ 0 & s \neq t \end{cases}$$

*Proof.* Lemma 2.5 guarantees that any $E \in \mathcal{E}_{\mathcal{S}}(\mathcal{F})$ satisfies

$$\sum_{k=1}^{m}\sum_{l=p}^{q}\sum_{z=1}^{r} x_{k,l,z} = q - p$$

and hence it follows that

$$\sum_{k=1}^{m}\sum_{l=p}^{q} x_{k,l,s} \leq q - p$$

for any $1 \leq s \leq r$. Now by the new rules, $\overline{a_{i,j,s}} \longleftarrow a_{k,l,t}$ for $s \neq t$, so if $x_{k,l,t} = 1$ then $x_{i,j,s} \neq 1$ for $s \neq t$ and any $k, l, i, s$, as otherwise, $E$ is not conflict-free. Furthermore, lemma 2.5 ensures that $x_{i,j,s} = 1$ for some $s$ and $i$, and hence it must be the case that $x_{i,j,t} = 1$ for some $i$. The result follows. □

Similarly, we consider the restricted assignment, precedence relations and forbidden sets constraints:

**Definition 3.3. (Restricted assignment)** Let $[\{M_x, ..., M_y\}, \{J_p, ..., J_q\}]_{RA}$ denote jobs $p$ to $q$ are only to be executed on machines $x$ to $y$. The restricted assignment constraint is represented by

$$\sum_{k=x}^{y}\sum_{l=p}^{q}\sum_{z=1}^{r} x_{k,l,z} = q - p$$

**Lemma 3.4.** Suppose $[\{M_x, ..., M_y\}, \{J_p, ..., J_q\}]_{RA}$. Let $\mathcal{F} =< \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\phantom{x}} >$ be an ABA framework such that $\mathcal{R} \supseteq \mathcal{R}_{MSP} \cup \mathcal{R}_{RA}$, where

$$\mathcal{R}_{RA} = \{\overline{a_{i,j,s}} \longleftarrow a_{i,j,s} \mid i \in \{M_x, ..., \ M_y\}, j \notin \{J_p, ..., \ J_q\} \ \vee \ i \notin \{M_x, ..., \ M_y\}, j \in \{J_p, ..., \ J_q\}\}$$

Then $\forall E \in \mathcal{E}_{\mathcal{S}}(\mathcal{F})$ such that $\mathcal{S} \in \{\mathcal{ST}, \mathcal{PR}\}$, $E$ satisfies

$$\sum_{k=x}^{y}\sum_{l=p}^{q}\sum_{z=1}^{r} x_{k,l,z} = q - p$$

*Proof.* Lemma 2.5 guarantees that any $E \in \mathcal{E}_{\mathcal{S}}(\mathcal{F})$ satisfies

$$\sum_{k=1}^{m}\sum_{l=p}^{q}\sum_{z=1}^{r} x_{k,l,z} = q - p$$

We now note that $a_{i,j,s} \notin E$ for $i \in \{M_x, ..., \ M_y\}$, $j \notin \{J_p, ..., \ J_q\}$ or $i \notin \{M_x, ..., \ M_y\}$, $j \in \{J_p, ..., \ J_q\}\}$, otherwise $E$ is no longer conflict-free as $\overline{a_{i,j,s}} \leftarrow a_{i,j,s}$. Furthermore, let $h \in \{p, ...q\}$. Lemma 2.5 ensures that $x_{i,h,s} = 1$ for some $i, s$, and hence it must be the case that $x_{i,h,s} = 1$ for some $i \in \{x, ..., y\}$ and some $s$. □

**Definition 3.5. (Precedence relations)** Denote $[J_p, J_{p+1}, ..., J_q]_{PR}$ as a precedent tuple, so pairwise, $J_j$ executed before $J_{j+1}$ for all $j \in \{p, ..., q-1\}$. Set index$(J_q) = q$. We make sure to relabel so index of jobs are in order, i.e. index$(J_q) := q > ... >$ index$(J_{p+1}) >$ index$(J_p)$. The precedence relation constraint satisfies $p - q$ binary constraints

$$x_{i,j,s_1} + x_{k,l,s_2} < 2$$

for index$(j) <$ index$(l) \wedge s_1 > s_2$.

**Lemma 3.6.** Suppose $[J_p, J_{p+1}, ..., J_q]_{PR}$. Let $\mathcal{F} =< \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\phantom{a}} >$ be an ABA framework such that $\mathcal{R} \supseteq \mathcal{R}_{MSP} \cup \mathcal{R}_{PR}$, where

$$\mathcal{R}_{PR} := \{\overline{a_{i,j,s}} \longleftarrow a_{k,l,t} \mid j, l \in \{p, ..., q\} \wedge \text{ index}(j) < \text{index}(l) \wedge s \geq t\}$$

Then $\forall E \in \mathcal{E}_S(\mathcal{F})$ such that $\mathcal{S} \in \{\mathcal{ST}, \mathcal{PR}\}$, $E$ satisfies $p - q$ binary constraints

$$x_{i,j,s_1} + x_{k,l,s_2} < 2$$

for index$(j) <$ index$(l) \wedge s_1 > s_2$.

*Proof.* Suppose $a_{i,j,s_1} \in E$ and $a_{k,l,s_2} \in E$, with index$(j) <$ index$(l) \wedge s_1 > s_2$. But $\overline{a_{i,j,s_1}} \leftarrow a_{k,l,s_2}$, a contradiction to the conflict-freeness of $E$. So $x_{i,j,s_1} + x_{k,l,s_2} < 2$. □

**Definition 3.7. (Forbidden sets)** Denote a forbidden tuple by $[J_p, J_{p+1}, ..., J_q]_{FS}$ such that for all pairs of jobs $j, l \in \{p, ..., q\}$, $J_j$ and $J_l$ cannot be executed on the same slot of different machines. This is represented by $r$ finitary constraints

$$\sum_{k=1}^{a} \sum_{l=p}^{q} x_{k,l,s} \ \leq 1 \ \forall s \in \{1, ..., r\}$$

**Lemma 3.8.** Suppose $[J_p, J_{p+1}, ..., J_q]_{FS}$. Let $\mathcal{F} =< \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\phantom{a}} >$ be an ABA framework such that $\mathcal{R} \supseteq \mathcal{R}_{MSP} \cup \mathcal{R}_{FS}$, where

$$\mathcal{R}_{FS} = \{\overline{a_{i,j,s}} \longleftarrow a_{k,l,t} \mid j, l \in \{p, ..., q\} \wedge s = t\}$$

Then $\forall E \in \mathcal{E}_S(\mathcal{F})$ such that $\mathcal{S} \in \{\mathcal{ST}, \mathcal{PR}\}$, $E$ satisfies $r$ binary constraints

$$\sum_{k=1}^{a} \sum_{l=p}^{q} x_{k,l,s} \ \leq 1 \ \forall s \in \{1, ..., r\}$$

*Proof.* Suppose $a_{i,j,s} \in E$ and $a_{k,l,s} \in E$ for $j, l \in \{J_p, ..., \ J_q\}$. Then $\overline{a_{i,j,s}} \leftarrow a_{k,l,s}$, contradicting the conflict-freeness of $E$. Furthermore, let $h \in \{p, ...q\}$. Lemma 2.5 ensures that $x_{i,h,s} = 1$ for some $i, s$, and hence it must be the case that $x_{i,h,s} = 1$ for some $i \in \{x, ..., y\}$ and some $s \neq t$. The result follows. □

The differentiation of different constraints allows us to couple attacks with labels. This is useful, as equivalent attacks (where multiple constraints could possibly) lead to the same attack can be distinguished. In particular, our approach allows us to obtain attacks of the form $(a_{i,j,s}, a_{k,l,r}, I)$, where $I$ is a set containing all the identifiers for the attack.

More formally, we shall define a set of identifiers $\mathcal{I} = \{i_1, ..., i_e\}$, and a function $l : \mathcal{R} \to 2^{\{i_1,...,i_e\}} \setminus \emptyset$ which maps attacks between arguments to a subset of identifiers which exemplify the reasons the attacks were put into place.

**Definition 3.9. (AA and ABA with Identifiers)**

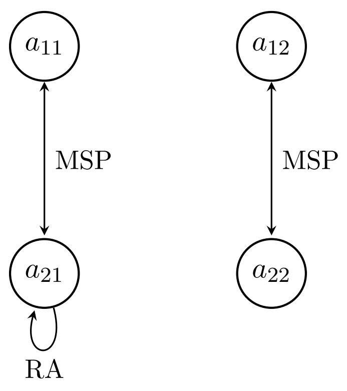
- Define the AA with Identifiers using a tuple $AF =< \mathcal{A}, \mathcal{R}, \mathcal{I} >$ where $\mathcal{I} = \{i_1, ..., i_e\}$, and $< \mathcal{A}, \mathcal{R} >$ is an AA framework. Additionally for each attack $a_{i,j,s} \rightarrow a_{k,l,t}$, define $l$ such that $l(a_{i,j,s} \rightarrow a_{k,l,t}) = \{i_p, ..., i_q\}$, where $\{i_p, ..., i_q\} \subseteq \{i_1, ..., i_e\}$.
  Moreover, $\mathcal{R} := \{(a_{i,j,s}, a_{k,l,t}, \{i_p, ..., i_q\})\}$.

Given a set which violates the schedule constraints, it is possible to return all reasons why the constraints are violated. A naive algorithm runs in $O(|E|^2)$ time by iterating through all pairs of arguments in $E$.

**Algorithm 1** Violated constraint algorithm

```
1: procedure VCALG($< \mathcal{A}, \mathcal{R}, \mathcal{I} >, E$)
2:     $\mathcal{R}_E \leftarrow \{(a_{i,j,s}, a_{k,l,r}, I) \in \mathcal{R} : a_{i,j,s} \wedge a_{k,l,r} \in E\}$        ▷ Restriction of attacks to $E$
3:     for $r \in \mathcal{R}_E$ do
4:         $c[x] \leftarrow I$        ▷ $c[i]$ is the array storing violated labels
5:         $x \leftarrow x + 1$
6:     return $c[x]$
```

**Example 3.10.** We given an example in the AA formulation. Let $|\mathcal{M}| = |\mathcal{J}| = 2$, $r = 1$ and $[\{M_1\}, \{J_1\}]_{RA}$. Then $\mathcal{I} = \{MSP, RA\}$, and $l(a_{i,j} \rightarrow a_{k,j}) = \{MSP\}$, and $l(a_{21} \rightarrow a_{21}) = \{RA\}$.



## 3.2 Properties of Finitary and Binary Constraints

In the section above, some common makespan schedule constraints were imposed in ABA. An interesting extension would be to explore if it would be possible to impose the same restrictions in AA frameworks. In particular, since AA frameworks rely on binary attacks, this is equivalent to exploring correspondences between binary constraints and so called finitary constraints which take the form $\sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_2} \sum_{s=z_1}^{z_2} x_{i,j,s} < \tau$ or $\sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_2} \sum_{s=z_1}^{z_2} x_{i,j,s} = \tau$. In this section, we show that:

1. There are finitary constraints (in the inequality form) that cannot be mapped to any set of binary constraints in inequality form such that extensions of an argumentation framework under the finitary constraints are equal to the extensions of the argumentation framework under the set of binary constraints.

2. In the case of a single finitary constraint in equality form (which we call $F$, say), this is undetermined. We are however able to give a mapping from a set of binary constraints, called $B$, constructed from $F$, to $F$ such that the feasible schedules under the set $B$ is included in the feasible schedules under $F$.

Let us begin with 1 by defining finitary and binary constraints in inequality form:

**Definition 3.11. (IFC and IBC)**

- Define binary constraints in inequality form (IBC) by $x_{i,j,s} + x_{k,l,r} < \gamma$ where $\gamma \in \{1,2\}$.
- Define finitary constraints in inequality form (IFC) by $\sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_2} \sum_{s=z_1}^{z_2} x_{i,j,s} < \tau$ with $0 < \tau \leq |\mathcal{J}|$.

We drop the third component when considering machines with no slots.

As it is unclear how finitary constraints should be represented in AA, we define the application of constraints to frameworks by taking the existing extensions in a framework that satisfy the constraints.

**Definition 3.12. (Extensions under IFC and IBC)** Let $AF =< \mathcal{A}, \mathcal{R} >$ given as in definition 2.4 or 2.6. Let $F$ be a single IFC, and $B$ be a set of IBC $b_i$. Let $\mathcal{S} \in \{\mathcal{ST}, \mathcal{PR}\}$. We define

$$\mathcal{E}_{\mathcal{S}}(AF_{IFC}) := \{E \in \mathcal{E}_{\mathcal{S}}(AF) : E \text{ satisfies } F\}$$
$$\mathcal{E}_{\mathcal{S}}(AF_{IBC}) := \{E \in \mathcal{E}_{\mathcal{S}}(AF) : E \text{ satisfies every } b_i \in B\}$$

We say that a finitary constraint $F$ (resp. a set of binary constraints $B$) is compatible with $AF$ if $\forall E \in \mathcal{E}_{\mathcal{S}}(AF_{IFC})$ (resp. $\forall E \in \mathcal{E}_{\mathcal{S}}(AF_{IBC})$), $|E| = |\mathcal{J}|$ and $\mathcal{E}_{\mathcal{S}}(AF_{IFC}) \neq \emptyset$ (resp. $\mathcal{E}_{\mathcal{S}}(AF_{IBC}) \neq \emptyset$).

We now show that there is not always a correspondence between a set of binary constraints in inequality form and finitary constraints in inequality form.

**Proposition 3.13.** Let $\mathcal{S} \in \{\mathcal{ST}, \mathcal{PR}\}$. In addition, let $|\mathcal{M}| = m \geq 2$, $|\mathcal{J}| = n \geq 3$. Finally, let $AF =< \mathcal{A}, \mathcal{R} >$ be given as in definition 2.6. Then there exists $AF$ and a IFC $F$ compatible with $AF$ such that $\mathcal{E}_{\mathcal{S}}(AF_{IFC}) \neq \mathcal{E}_{\mathcal{S}}(AF_{IBC})$ for any set $B$ containing any number of IBC.

*Proof.* Take $F$ to be the IFC

$$\sum_{j=1}^{n} x_{1,j} < n$$

We note that $\mathcal{E}_{\mathcal{S}}(AF_{IFC}) = \mathcal{E}_{\mathcal{S}}(AF) \setminus \{a_{1,1}, ..., a_{1,n}\}$. Clearly $B$ cannot be the empty set as $|\mathcal{E}_{\mathcal{S}}(AF_{IFC})| \neq |\mathcal{E}_{\mathcal{S}}(AF)|$. Now consider adding a single attack. First, a self attack $(a_{i,j}, a_{i,j})$ removes more than one extension from $AF$, since $m \geq 2$ and $\{a_{k_1,1}, ..., a_{i,j}, ..., a_{k_n,n}\}$ is an extension for every $1 \leq k \leq m$. Any additional attacks can only remove or retain the number of extensions, unless enough attacks are added such that $\exists E \in \mathcal{E}_{\mathcal{S}}(AF_{IBC})$ with $|E| \neq |\mathcal{J}|$. In either case, $\mathcal{E}_{\mathcal{S}}(AF_{IFC}) \neq \mathcal{E}_{\mathcal{S}}(AF_{IBC})$, so it is not possible that $B$ contains $x_{i,j}+x_{i,j} < 1$ or $x_{i,j}+x_{k,l} < 1$, where the latter constitutes two self attacks.

Now consider an attack from $(a_{i,j}, a_{k,l})$ for any $i, j, k, l$ such that $(a_{i,j}, a_{k,l})$ is not already an element of $\mathcal{R}$. Then $\nexists E \in \mathcal{E}_{\mathcal{S}}(AF_{IBC})$ such that $\{a_{i,j}, a_{k,l}\} \subseteq E$ but $\exists E \in \mathcal{E}_{\mathcal{S}}(AF_{IFC})$ such that $\{a_{i,j}, a_{k,l}\} \subseteq E$.

Again, any additional attacks can only remove or retain the number of extensions, unless enough attacks are added such that $\exists E \in \mathcal{E}_{\mathcal{S}}(AF_{IBC})$ with $|E| \neq |\mathcal{J}|$. Hence $B$ cannot contain an IBC $x_{i,j} + x_{k,l} < 2$ for any $i, j, k, l$. Thus there is no such $B$. $\square$

Note that conditions $|\mathcal{M}| = m \geq 2$, $|\mathcal{J}| = n \geq 3$ are necessary in the case of our proof, otherwise the finitary constraint in the proof devolves into a binary constraint.

We now move on to part 2, which requires similar definitions to the above, except in equality form:

**Definition 3.14. (FFC and FBC)**

- Define fixed binary constraints (FBC) by $x_{i,j,s} + x_{k,l,r} = \gamma$ where $\gamma \in \{0, 1, 2\}$.
- Define fixed finitary constraints (FFC) by $\sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_2} \sum_{s=z_1}^{z_2} x_{i,j,s} = \tau$ with $0 < \tau \leq |\mathcal{J}|$.

We drop the third component when considering machines with no slots.

**Definition 3.15. (Extensions under FFC and FBC)** Let $AF =< \mathcal{A}, \mathcal{R} >$ given as in definition 2.4 or 2.6. Let $F$ be a single FFC, and $B$ be a set of FBC $b_i$. Let $\mathcal{S} \in \{\mathcal{ST}, \mathcal{PR}\}$. We define

$$\mathcal{E}_{\mathcal{S}}(AF_{FFC}) := \{E \in \mathcal{E}_{\mathcal{S}}(AF) : E \text{ satisfies } F\}$$
$$\mathcal{E}_{\mathcal{S}}(AF_{FBC}) := \{E \in \mathcal{E}_{\mathcal{S}}(AF) : E \text{ satisfies every } b_i \in B\}$$

We say that a finitary constraint $F$ (resp. a set of binary constraints $B$) is compatible with $AF$ if $\forall E \in \mathcal{E}_{\mathcal{S}}(AF_{FFC})$ (resp. $\forall E \in \mathcal{E}_{\mathcal{S}}(AF_{FBC})$), $|E| = |\mathcal{J}|$ and $\mathcal{E}_{\mathcal{S}}(AF_{FFC}) \neq \emptyset$ (resp. $\mathcal{E}_{\mathcal{S}}(AF_{FBC}) \neq \emptyset$).

We now give a general mapping such that for any given FFC $F$ and $AF$ in the form of definition 2.4, we can construct a set $B$ of FBC such that $\mathcal{E}_{\mathcal{S}}(AF_{FBC}) \subseteq \mathcal{E}_{\mathcal{S}}(AF_{FFC})$.

**Proposition 3.16.** Let $F$ be a FFC compatible with $AF$. Then there exists a set $B$ of FBC with $|B| = \frac{1}{2}\left(N^2 - N\right)$ such that $\mathcal{E}_{\mathcal{S}}(AF_{FBC}) \subseteq \mathcal{E}_{\mathcal{S}}(AF_{FFC})$.

*Proof.* $F$ gives $N := (z_2 - z_1 + 1)(y_2 - y_1 + 1)(x_2 - x_1 + 1)$ possible choices of $x_{i,j,s}$ with different indices over $i$, $j$ and $s$. Denote these arguments corresponding to these choices by the set

$$C := \{a_{i,j,s} : i \in \{x_1, ..., x_2\}, j \in \{y_1, ..., y_2\}, z \in \{z_1, ..., z_2\}\}$$

Using this, we can construct $\frac{1}{2}\left(N^2 - N\right)$ distinct pairs $x_{i,j,s} + x_{k,l,r} = \gamma$. To find the value of $\gamma$ for each pair, we use the following algorithm:

Iterate through all sets $X \subseteq 2^C$ with $|X| = \tau$. For each $X$, see if $X \subseteq E$ for some $E \in \mathcal{E}_{\mathcal{S}}(AF)$. If it does, then use this $E$ to define binary constraints

$$x_{i,j,s} + x_{k,l,r} = \begin{cases} 2 & a_{i,j,s} \in E \wedge a_{k,l,r} \in E \\ 1 & a_{i,j,s} \in E \vee a_{k,l,r} \in E \\ 0 & \neg a_{i,j,s} \in E \wedge \neg a_{k,l,r} \in E \end{cases}$$

The set must be conflict-free with respect to $AF$, as we selected it from an extension of $AF$. Additionally, it must be the case that at least one $X$ exists, because $F$ is compatible with $AF$. We take $B$ to be the binary constraints defined above. Now let $S \in \mathcal{E}_{\mathcal{S}}(AF_{FBC})$. We show that $S$ satisfies $F$. Firstly, $S$ must satisfy all the FBC in $B$. Then $S$ satisfies the FBC's $\frac{1}{N-1}(x_{i,j,s} + x_{k,l,r}) = \frac{\gamma}{N-1}$, and hence must satisfy it's sum

$$\frac{1}{2(N-1)} \sum_{i \neq k,\ j \neq l,\ s \neq t} x_{i,j,s} + x_{k,l,r} = \sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_2} \sum_{s=z_1}^{z_2} x_{i,j,s} = \tau$$

as each term appears $N-1$ times in the sum, and division by 2 appears as the sum is symmetric. □

With this mapping, it is clear that there is no correspondence between $F$ and $B$. Consider $F$ to be

$$\sum_{i=1}^{2} \sum_{j=1}^{2} x_{i,j,1} = 2$$

Then we can have $\{a_{1,1,1}, a_{2,2,1}\} = E$ or $\{a_{1,2,1}, a_{2,1,1}\} = E$, giving two different collections of conflicting $B$, one containing $x_{111} + x_{221} = 2$ and the other containing $x_{121} + x_{211} = 2$. Instead, it is possible to "weaken" $B$ into a single $F$ by "allowing equivalent solutions". To do this, we need to find the value of $\tau$ which the FBC collectively satisfy, and this is done by summing the binary constraints in a following way:

**Proposition 3.17.** Given a set $B$ with FBC's of the form $x_{i,j,s} + x_{k,l,r} = \gamma_\alpha$, $\gamma_\alpha \in \{0,1,2\}$, $i,k \in \{x_1,...,x_2\}$, $j,l \in \{y_1,...,y_2\}$, $s,r \in \{z_1,...,z_2\}$ and $\neg(i = k \wedge j = l \wedge s = r)$ that are compatible with $AF$, we can recover a value

$$\tau = \frac{1}{2(N-1)} \sum_{\alpha=1}^{(N^2-N)/2} \gamma_\alpha$$

with $N := (z_2 - z_1 + 1)(y_2 - y_1 + 1)(x_2 - x_1 + 1)$ such that $AF$ is compatible with a FFC $F$ given by

$$\sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_2} \sum_{s=z_1}^{z_2} x_{i,j,s} = \tau$$

*Proof.* We first consider the $N^2$ pairs of FBC $x_{i,j,s} + x_{k,l,r}$, $i,k \in \{x_1,...,x_2\}$, $j,l \in \{y_1,...,y_2\}$, $s,r \in \{z_1,...,z_2\}$. It is easy to see that

$$\sum_{i,j,s,k,l,r} (x_{i,j,s} + x_{k,l,r})/2N = \sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_2} \sum_{s=z_1}^{z_2} x_{i,j,s} = \tau$$

as each term on the LHS appears $N$ times and the sum is symmetric. Now it suffices to show that

$$\sum_{(i \neq k) \vee (l \neq j) \vee (s \neq r)} x_{i,j,s} + x_{k,l,r} = 2\tau(N-1)$$

Note that

$$\sum_{i,j,s,k,l,r} x_{i,j,s} + x_{k,l,r} - \sum_{(i=k) \wedge (l=j) \wedge (s=r)} x_{i,j,s} + x_{k,l,r} = \sum_{(i \neq k) \vee (l \neq j) \vee (s \neq r)} x_{i,j,s} + x_{k,l,r}$$

and we have

$$\begin{aligned}\sum_{(i=k)\wedge(l=j)\wedge(s=r)} x_{i,j,s} + x_{k,l,r} &= 2\sum_{i=x_1}^{x_2}\sum_{j=y_1}^{y_2}\sum_{s=z_1}^{z_2} x_{i,j,s} \\ &= 2\tau\end{aligned}$$

so we indeed have

$$\sum_{(i\neq k)\vee(l\neq j)\vee(s\neq r)} x_{i,j,s} + x_{k,l,r} = 2\tau(N-1)$$

□

This concludes our work on the properties of makespan constraints. Some natural extensions to this include the possibility of finding a correspondence between fixed finitary and binary constraints. Also, it may be interesting to see if it is possible to map a finitary constraint to less $(N^2 - N)/2$ pairs of binary constraints, yet still retain some connection between the extensions of a makespan framework upon applying binary and finitary constraints.

# 4 Dynamics of Makespan Scheduling

The dynamics of an argumentation framework refer to the efficient re-computation of extensions of the framework when it undergoes change. In [1], the authors present a method where extensions of a framework that are exposed to change are used to generate extensions of the new framework by partitioning the framework into unaffected and affected sections. While it is possible to apply this to the standard makespan model, we propose a more elementary form of their results that is specific to the makespan model in section 4.1. In particular, let $AF'$ be the argumentation framework after a removal or addition of a machine or a job. We will first establish how to obtain $\mathcal{E}_{\mathcal{S}}(AF')$ from $\mathcal{E}_{\mathcal{S}}(AF)$. This in turn allows the construction of suitable mappings from extensions that have become unfeasible due to changes in the framework into feasible schedules in the new framework under the condition that the disruption to the machines is kept at a minimal. This is more formally defined as minimizing the number of swaps:

**Definition 4.1. (Swaps)** An assignment of job $J_l$ to machine $M_i$ is *swapped* to machine $M_k$, $i \neq k$ by taking $E' = (E \setminus \{a_{i,l,s}\}) \cup \{a_{k,l,t}\}$ for some $i, k \in \{1, ..., m\}$, $s, t \in \{1, ..., r\}$ and $E \in \mathcal{E}_{\mathcal{S}}(AF)$. For any deterministic algorithm ALGO which reschedules extensions via swaps only, we define $\mathcal{MS}(E)$ to be the number of swaps run by ALGO to ensure $E \in \mathcal{E}_{\mathcal{S}}(AF')$, after the $\mathcal{MS}(E)$ swaps.

This definition implicitly defines argument addition and deletion, as we can simply take a swap $E' = (E \setminus \emptyset) \cup \{a_{k,l,t}\}$ or $E' = (E \setminus \{a_{i,l,s}\}) \cup \emptyset$ respectively.

For any change in the standard framework and any extension $E \in \mathcal{E}_{\mathcal{S}}(AF)$, it is easy to see that any reasonable algorithm has the property $\mathcal{MS}(E) \leq |\mathcal{J}|$ if there is no regard to optimality of the extension; we can simply reassign every job as follows: assign $J_1$ to $M_1$ (slot 1), $J_2$ to $M_2$ (slot 1), ..., $J_m$ to $M_m$ (slot 1), $J_{m+1}$ to $M_1$ (slot 2), and so on. However, it is possible to do much better than this in all cases, which we show in section 4.2. In addition, it is easy to show that the algorithms we give minimise the number of swaps by finding a lower bound for the number of swaps.

A more interesting extension is to find a bound for $\mathcal{MS}(E)$ under the common constraints described in section 3, which is done in section 4.3.

## 4.1 Exposing Feasible Schedules to Change

We first begin by trying to find an explicit formula for the extensions of the makespan framework when removing $p$ machines. Let us define removing machine $M_k$ as removing all the arguments $a_{k,j,s}$ for every $j, s$, and any attacks that go to and from $a_{k,j,s}$:

**Definition 4.2.** Let $m > p$. Model removing a set of $p$ machines $\mathcal{M}_p = \{M_{i_1}, ..., M_{i_p}\}$ from $AF$ by:

$$\mathcal{A}_{\mathcal{M}_p} = \mathcal{A} \setminus \bigcup_{k=i_1}^{i_p} \{a_{k,j,s} : 1 \leq j \leq n,\ 1 \leq s \leq r\}$$

$$\mathcal{R}_{\mathcal{M}_p} = \mathcal{R} \setminus \bigcup_{k=i_1}^{i_p} \{(a_{i,j,s}, a_{k,l,t}),\ (a_{k,l,t}, a_{i,j,s}) : (i \neq k, j = l) \vee (i = k, j = l, s \neq t) \vee (i = k, j \neq l, s = t)\}$$

$$AF_{\mathcal{M}_p} =< \mathcal{A}_{\mathcal{M}_p}, \mathcal{R}_{\mathcal{M}_p} >$$

With some trial and error, it is relatively apparent that upon removal of machine $M_k$, any stable or preferred extension where a job is scheduled to machine $M_k$ can no longer be an extension in the new framework, but any extension where no job is assigned to $M_k$ remains an extension. We more formally prove this in the case of removing $p$ machines with induction. Let us begin with the case of stable semantics:

**Proposition 4.3.** Let $AF =< \mathcal{A}, \mathcal{R} >$ be given as in definition 2.4 such that $(m - p)r > n$. Then,

$$\mathcal{E}_{\mathcal{ST}}(AF_{\mathcal{M}_p}) = \mathcal{E}_{\mathcal{ST}}(AF) \setminus \bigcup_{k=i_1}^{i_p} \{E \in \mathcal{E}_{\mathcal{ST}}(AF) :\ \exists a_{k,j,s} \in E,\ 1 \leq j \leq n,\ 1 \leq s \leq r\}$$

*Proof.* Induct on $p$. Consider first the case $p = 1$, removing a single machine $M_k$. Let $S \in \mathcal{E}_{\mathcal{ST}}(AF) \setminus \{E \in \mathcal{E}_{\mathcal{ST}}(AF) : \exists a_{k,j,s} \in E,\ 1 \leq j \leq n,\ 1 \leq s \leq r\}$. Then $S$ must be conflict-free in $AF_{\{M_k\}}$ as it is conflict-free in $AF$. For the stability condition, consider an element $a \notin E$, but $a \in AF_{\{M_k\}}$. As $S \in \mathcal{E}_{\mathcal{ST}}(AF)$, $\exists b \in E$ such that $(b, a) \in \mathcal{R}$. As $a_{k,j,s} \notin S$, $(b, a) \in \mathcal{R}_{\{M_k\}}$. Hence $S \in \mathcal{E}_{\mathcal{ST}}(AF_{\{M_k\}})$.

Now let $S \in \mathcal{E}_{\mathcal{ST}}(AF_{\{M_k\}})$. Clearly $S$ is conflict-free in $AF$ as it is conflict-free in $AF_{\{M_k\}}$ and new attacks are only added to and from $a_{k,j,s}$, but $a_{k,j,s} \notin S$. As $S$ is stable in $AF_{\{M_k\}}$, we only need to consider $a_{k,j,s} \in \mathcal{A} \setminus \mathcal{A}_{-k} = \{a_{k,l,t} : 1 \leq l \leq n,\ 1 \leq t \leq r\}$. By lemma 2.5, $\exists a_{i,j,v} \in E$ for some $1 \leq i \leq m,\ 1 \leq v \leq r$, and hence $(a_{i,j,v}, a_{k,j,s}) \in \mathcal{R}$. Thus $S \in \mathcal{E}_{\mathcal{ST}}(AF)$. Clearly $a_{k,j,s} \notin S$ so $S \in \mathcal{E}_{\mathcal{ST}}(AF) \setminus \{E \in \mathcal{E}_{\mathcal{ST}}(AF) : \exists a_{k,j,s} \in E,\ 1 \leq j \leq n,\ 1 \leq s \leq r\}$. This proves the $p = 1$ case.

Now assume the proposition is true for the $p$ case. For the $p + 1$ case, by removing an additional machine $M_{p+1}$ from $AF_{\mathcal{M}_p}$, we obtain

$$\begin{aligned}\mathcal{E}_{\mathcal{ST}}(AF_{\mathcal{M}_{p+1}}) &= \mathcal{E}_{\mathcal{ST}}(AF_{\mathcal{M}_p}) \setminus \{E \in \mathcal{E}_{\mathcal{ST}}(AF_{\mathcal{M}_p}) : \exists a_{i_{p+1},j,s} \in E,\ 1 \leq j \leq n,\ 1 \leq s \leq r\} \\ &= \mathcal{E}_{\mathcal{ST}}(AF) \setminus \bigcup_{k=i_1}^{i_p} \{E \in \mathcal{E}_{\mathcal{ST}}(AF) : \exists a_{k,j,s} \in E\} \ \cup\ \{E \in \mathcal{E}_{\mathcal{ST}}(AF_{\mathcal{M}_p}) : \exists a_{i_{p+1},j,s} \in E\}\end{aligned}$$

where for the second equality, $1 \leq j \leq n$ and $1 \leq s \leq r$, omitted for space. We set

$$S_1 := \mathcal{E}_{\mathcal{ST}}(AF) \setminus \bigcup_{k=i_1}^{i_p} \{E \in \mathcal{E}_{\mathcal{ST}}(AF) : \exists a_{k,j,s} \in E\} \ \cup\ \{E \in \mathcal{E}_{\mathcal{ST}}(AF_{\mathcal{M}_p}) : \exists a_{i_{p+1},j,s} \in E\}$$

We need to show that

$$S1 = S2 := \mathcal{E}_{\mathcal{ST}}(AF) \setminus \bigcup_{k=i_1}^{i_{p+1}} \{E \in \mathcal{E}_{\mathcal{ST}}(AF) : \exists a_{k,j,s} \in E\}$$

Let us show $\mathcal{E}_{\mathcal{ST}}(AF_{\mathcal{M}_p}) \subseteq \mathcal{E}_{\mathcal{ST}}(AF)$ as an intermediate result. Let $S \in \mathcal{E}_{\mathcal{ST}}(AF_{\mathcal{M}_p})$. It is conflict-free in $AF$ as it is conflict-free in $AF_{\mathcal{M}_p}$ and new attacks are only added to and from $a_{k,j,s}$ for $k \in \mathcal{M}_p$, but $a_{k,j,s} \notin S$. For stability, if $a_{k,j,s} \in AF_{\mathcal{M}_p}$ then $\exists a_{i,j,t} \in S$ such that $(a_{i,j,t}, a_{k,j,s}) \in \mathcal{R}_{\mathcal{M}_p}$ by the stability of $S$ in $AF_{\mathcal{M}_p}$, so suppose $a_{k,j,s} \in AF \setminus AF_{\mathcal{M}_p}$. By lemma 2.5, $\exists a_{i,j,t} \in S$ with $(a_{i,j,t}, a_{k,j,s}) \in \mathcal{R} \setminus \mathcal{R}_{\mathcal{M}_p}$. Hence, $S \in \mathcal{E}_{\mathcal{ST}}(AF)$.

Hence it is clear that $S_2 \subseteq S_1$. Let $E_1 \in S_1$, and suppose $E_1 \notin S_2$. Then $\exists a_{k,j,s} \in E_1$ with $k \in \mathcal{M}_p$. But $E_1 \in \{E \in \mathcal{E}_{ST}(AF) : \exists a_{k,j,s} \in E\}$ with $k \in \mathcal{M}_p$. So $E_1 \notin S_1$. Thus $S_1 \subseteq S_2$ and the result follows. □

A very similar proof gives the same result under preferred extensions:

**Proposition 4.4.** Let $AF =< \mathcal{A}, \mathcal{R} >$ be given as in definition 2.4 such that $(m-p)r > n$. Then,

$$\mathcal{E}_{\mathcal{PR}}(AF_{\mathcal{M}_n}) = \mathcal{E}_{\mathcal{PR}}(AF) \setminus \bigcup_{k=i_1}^{i_p} \{E \in \mathcal{E}_{\mathcal{PR}}(AF) :\ \exists a_{k,j,s} \in E,\ 1 \le j \le n,\ 1 \le s \le r\}$$

*Proof.* Induct on $p$. Consider first the case $p = 1$, removing a single machine $M_k$. Let $S \in \mathcal{E}_{\mathcal{PR}}(AF_{\{M_k\}})$. It is conflict-free in $AF_{\{M_k\}}$, and so is conflict-free in $AF$ as new attacks are only obtained to and from $a_{k,j,s}$ for $1 \le j \le n$ and $1 \le s \le r$, but $a_{k,j,s} \notin S$. As $S$ is preferred in $AF_{\{M_k\}}$, $\forall b \in \mathcal{A}_{\{M_k\}}$ such that $b \to S$, $\exists c \in S$ such that $(c, b) \in \mathcal{R}_{\{M_k\}}$, so consider $a_{k,j,s} \in \mathcal{R} \setminus \mathcal{R}_{\{M_k\}} = \{a_{k,l,t} : 1 \le l \le n,\ 1 \le t \le r\}$, such that $(a_{k,j,s}, a_{p,j,t}) \in \mathcal{R}$ for some $a_{p,j,t} \in S$. But there also $\exists (a_{p,j,t}, a_{k,j,s}) \in \mathcal{R}$ by the new attacks brought about by re-adding machine $k$ to obtain $AF$, so $S$ is admissible in $AF$. Suppose now that $S$ is not maximal. Then $S \subset A$, where $A$ is some admissible set. Let $A = S \cup \{a_{i,j,s}\}$. But then $\exists a_{q,j,t} \in S \cup \{a_{i,j,s}\}$ such that $(a_{q,j,t}, a_{i,j,s}) \in \mathcal{R}$ as $|S| = |\mathcal{J}|$ by lemma 2.5, so $A$ is not conflict-free, a contradiction. Hence $S \in \{E \in \mathcal{E}_{\mathcal{PR}}(AF) :\ \nexists a_{k,j} \in E,\ 1 \le j \le n\}$.

For the reverse direction, let $S \in \mathcal{E}_{\mathcal{PR}}(AF) \setminus \{E \in \mathcal{E}_{\mathcal{PR}}(AF) : \exists a_{k,j,s} \in E,\ 1 \le j \le n,\ 1 \le s \le r\}$. $S$ is conflict-free in $AF$, and no attacks are added to obtain $AF_{\{M_k\}}$, so $S$ must be conflict-free in $AF_{\{M_k\}}$. Suppose for a contradiction that $S$ is not admissible in $AF_{\{M_k\}}$. As conflict-freeness is considered above, $\exists a_{i,j,s} \in S$ such that $\exists a_{p,j,t} \in \mathcal{A}$ with $(a_{p,j,t}, a_{i,j,s}) \in \mathcal{R}$ but $S \not\to a_{p,j,t}$. But we note that $S$ cannot contain any element $a_{k,j,t}$ for any $j, t$, and so it must also be the case that $(a_{p,j,t}, a_{i,j,s}) \in \mathcal{R}_{\{M_k\}}$ and $S \not\to a_{p,j}$ in $AF_{\{M_k\}}$, meaning $S$ is not admissible in $AF_{\{M_k\}}$, a contradiction. Maximality can be argued exactly as above. Hence, $S \in \mathcal{E}_{\mathcal{PR}}(AF_{\{M_k\}})$.

Now assume the proposition is true for the $p$ case. For the $p+1$ case, by removing an additional machine $M_{p+1}$ from $AF_{\mathcal{M}_p}$, we obtain

$$\begin{aligned}\mathcal{E}_{\mathcal{PR}}(AF_{\mathcal{M}_{p+1}}) &= \mathcal{E}_{\mathcal{PR}}(AF_{\mathcal{M}_p}) \setminus \{E \in \mathcal{E}_{\mathcal{PR}}(AF_{\mathcal{M}_p}) : \exists a_{i_{p+1},j,s} \in E,\ 1 \le j \le n,\ 1 \le s \le r\} \\ &= \mathcal{E}_{\mathcal{PR}}(AF) \setminus \bigcup_{k=i_1}^{i_p} \{E \in \mathcal{E}_{\mathcal{PR}}(AF) : \exists a_{k,j,s} \in E\} \ \cup\ \{E \in \mathcal{E}_{\mathcal{PR}}(AF_{\mathcal{M}_p}) : \exists a_{i_{p+1},j,s} \in E\}\end{aligned}$$

where for the second equality, $1 \le j \le n$ and $1 \le s \le r$, omitted for space. We set

$$S_1 := \mathcal{E}_{\mathcal{PR}}(AF) \setminus \bigcup_{k=i_1}^{i_p} \{E \in \mathcal{E}_{\mathcal{PR}}(AF) : \exists a_{k,j,s} \in E\} \ \cup\ \{E \in \mathcal{E}_{\mathcal{PR}}(AF_{\mathcal{M}_p}) : \exists a_{i_{p+1},j,s} \in E\}$$

We need to show that

$$S1 = S2 := \mathcal{E}_{\mathcal{PR}}(AF) \setminus \bigcup_{k=i_1}^{i_{p+1}} \{E \in \mathcal{E}_{\mathcal{PR}}(AF) : \exists a_{k,j,s} \in E\}$$

Let us first show $\mathcal{E}_{\mathcal{PR}}(AF_{\mathcal{M}_p}) \subseteq \mathcal{E}_{\mathcal{PR}}(AF)$ as an intermediate result. Let $S \in \mathcal{E}_{\mathcal{PR}}(AF_{\mathcal{M}_p})$. $S$ is conflict-free in $AF$ as it is conflict-free in $AF_{\mathcal{M}_p}$ and new attacks are only added to and from $a_{k,j,s}$ for $k \in \mathcal{M}_p$, but $a_{k,j,s} \notin S$. As $S$ is preferred in $AF_{\mathcal{M}_n}$, $\forall b \in \mathcal{A}_{\mathcal{M}_n}$ such that $b \to S$, $\exists c \in S$ such that $(c, b) \in \mathcal{R}_{\mathcal{M}_n}$, so consider $a_{k,j,s} \in \mathcal{R} \setminus \mathcal{R}_{\mathcal{M}_n} = \{a_{k,l,t} : k \in \{i_1, ..., i_p\},\ 1 \leq l \leq n,\ 1 \leq t \leq r\}$, such that $(a_{k,j,s}, a_{p,j,t}) \in \mathcal{R}$ for some $a_{p,j,t} \in S$. But there also $\exists (a_{p,j,t}, a_{k,j,s}) \in \mathcal{R}$ by the new attacks brought about by re-adding machines $\mathcal{M}_n$ to obtain $AF$, so $S$ is admissible in $AF$. Suppose now that $S$ is not maximal. Then $S \subset A$, where $A$ is some admissible set. Let $A = S \cup \{a_{i,j,s}\}$. But then $\exists a_{q,j} \in S \cup \{a_{i,j,s}\}$ such that $(a_{q,j,t}, a_{i,j,s}) \in \mathcal{R}$ lemma 2.5, so $A$ is not conflict-free, a contradiction. Hence $S \in \mathcal{E}_{\mathcal{PR}}(AF)$.

By the above it is clear that $S_2 \subseteq S_1$. Let $E_1 \in S_1$, and suppose $E_1 \notin S_2$. Then $\exists a_{k,j,s} \in E_1$ with $k \in \mathcal{M}_p$. But $E_1 \in \{E \in \mathcal{E}_{PR}(AF) : \exists a_{k,j,s} \in E\}$ with $k \in \mathcal{M}_p$. So $E_1 \notin S_1$. Thus $S_1 \subseteq S_2$ and the result follows. □

Given extensions of $AF$, this allows a naive algorithm which recomputes all the extensions of $AF_{\{M_k\}}$ to run in $O(r|\mathcal{J}||\mathcal{E}_{\mathcal{S}}(AF)|)$ time:

**Algorithm 2** Extension re-computation under semantics $\mathcal{ST}$, $\mathcal{PR}$ upon removing machine $M_k$

```
1: procedure MACRMV($< \mathcal{A}, \mathcal{R} >, \mathcal{E}_{\mathcal{S}}(AF)$)
2:     $\mathcal{E}_{\mathcal{S}}(AF_{\{M_k\}}) \leftarrow \emptyset$
3:     for $E \in \mathcal{E}_{\mathcal{S}}(AF)$ do
4:         if $a_{k,j,s} \notin E$ then
5:             $\mathcal{E}_{\mathcal{S}}(AF_{\{M_k\}}) \leftarrow \mathcal{E}_{\mathcal{S}}(AF_{\{M_k\}}) \cup E$
6:     return $\mathcal{E}_{\mathcal{S}}(AF_{\{M_k\}})$
```

Unsurprisingly a similar result can be achieved when considering machine addition.

**Definition 4.5.** Model adding a set of $p$ machines $\mathcal{M}_p = \{M_{i_1}, ..., M_{i_p}\}$ from $AF$ by:

$$\mathcal{A}^{\mathcal{M}_p} = \bigcup_{k=i_1}^{i_p} \{a_{k,j,s} : 1 \leq j \leq n,\ 1 \leq s \leq r\} \cup \mathcal{A}$$

$$\mathcal{R}^{\mathcal{M}_p} = \bigcup_{k=i_1}^{i_p} \{(a_{i,j,s}, a_{k,l,t}),\ (a_{k,l,t}, a_{i,j,s}) : (i \neq k, j = l) \vee (i = k, j = l, s \neq t) \vee (i = k, j \neq l, s = t)\} \cup \mathcal{R}$$

$$AF^{\mathcal{M}_p} =< \mathcal{A}^{\mathcal{M}_p}, \mathcal{R}^{\mathcal{M}_p} >$$

We note that it is relatively difficult to categorise all the extensions in the framework upon adding $p$ machines without the use of algorithms, so we first give the categorisation when adding a single machine $M_k$, then provide an algorithm which helps us do so. It follows that running the algorithm $p$ times reveals all the extensions. Let us begin with stable semantics:

**Proposition 4.6.** Let

$$S_b = \{\{a_{k,j_1,s_1}, a_{k,j_2,s_2}, ..., a_{k,j_b,s_b}, a_{i_{b+1},j_{b+1},s_{b+1}}, ..., a_{i_n,j_n,s_n}\} : \ 1 \leq i_x \leq m, \ 1 \leq j_y \leq n\}$$

for $x \in \{b+1, ..., n\}$, where exactly $b$ elements of $E$ have the first index $k$. Set

$$E_b := \{E \in S_b : \forall a_{i,j,s}, a_{k,l,t} \in E, \ j \neq l \wedge (i \neq k \vee s \neq t) \text{ pairwise}\}$$

Then,

$$\mathcal{E}_{\mathcal{ST}}(AF^{\{M_k\}}) = \bigcup_{b=1}^{n} E_b \cup \mathcal{E}_{\mathcal{ST}}(AF)$$

*Proof.* Let $S \in \mathcal{E}_{\mathcal{ST}}(AF)$. As part of proof 4.3, we showed that $\mathcal{E}_{\mathcal{ST}}(AF^{\{M_k\}}_{\mathcal{M}_p}) \subseteq \mathcal{E}_{\mathcal{ST}}(AF^{\{M_k\}})$. Taking $\mathcal{M}_p = \{M_k\}$ implies that $\mathcal{E}_{\mathcal{ST}}(AF) \subseteq \mathcal{E}_{\mathcal{ST}}(AF^{\{M_k\}})$.

Now let $S \in E_b$, $1 \leq b \leq n$. $S$ is conflict-free as the condition of $E_b$ guarantees the standard makespan attacks are avoided. It must also be stable in $AF^{\{M_k\}}$ as for some $\exists a_{i,j,s} \in E_b$ for every $1 \leq j \leq n$ for some $i, s$, so $(a_{i,j,s}, a_{q,j,t}) \in \mathcal{R}^{\{M_k\}}$ for $\neg(i = q \wedge s = t)$. Hence $E_b \in \mathcal{E}_{\mathcal{ST}}(AF^{\{M_k\}})$ for $1 \leq b \leq n$. This proves that

$$\mathcal{E}_{\mathcal{ST}}(AF^{\{M_k\}}) \supseteq \bigcup_{b=1}^{n} E_b \cup \mathcal{E}_{\mathcal{ST}}(AF)$$

For the reverse direction, let $S \in \mathcal{E}_{\mathcal{ST}}(AF^{\{M_k\}})$, and suppose $a_{k,j,s} \notin S$ for any $j$ and $s$. Then by proposition 4.3, $S \in \mathcal{E}_{\mathcal{ST}}(AF)$. Otherwise, $\exists a_{k,j,s} \in S$. We note that $j$ must be distinct for each $S$ as otherwise, $S$ would not be conflict free. Hence, $S$ takes the form $\{a_{k,j_1,s_1}, ..., a_{k,j_b,s_b}, a_{i_{b+1},j_{b+1},s_{b+1}}, ..., a_{i_n,j_n,s_n}\}$ and hence $S \in E_b$, for $1 \leq b \leq n$. □

A similar result can be done for preferred extensions:

**Proposition 4.7.** Let $E_b$ be defined as in proposition 4.6. Then

$$\mathcal{E}_{\mathcal{PR}}(AF^{\{M_k\}}) = \bigcup_{b=1}^{n} E_b \cup \mathcal{E}_{\mathcal{PR}}(AF)$$

*Proof.* Let $S \in \mathcal{E}_{\mathcal{PR}}(AF)$. Part of proof 4.4 implies that $\mathcal{E}_{\mathcal{PR}}(AF^{\{M_k\}}_{\mathcal{M}_p}) \subseteq \mathcal{E}_{\mathcal{PR}}(AF^{\{M_k\}})$. Taking $\mathcal{M}_p = \{M_k\}$ implies that $\mathcal{E}_{\mathcal{PR}}(AF) \subseteq \mathcal{E}_{\mathcal{PR}}(AF^{\{M_k\}})$.

Now let $S \in E_b$, $1 \leq b \leq n$. $S$ is conflict-free as the condition of $E_b$ guarantees that no job is scheduled to a machine which is already executing a job. Let $a_{i,j,s} \in S$ and suppose $(a_{q,j,t}, a_{i,j,s}) \in \mathcal{R}^{\{M_k\}}$ for some $a_{q,j,t} \in \mathcal{A}^{\{M_k\}}$. Then $(a_{i,j,s}, a_{q,j,t})$, so each $a_{i,j,s}$ defends itself in $S$, and so $S$ is admissible. Finally, suppose $S$ is not maximal. Then $S \subset A$, where $A := S \cup \{a_{q,j,t}\}$ is admissible. But elements in $S$ span the index of $j$ by lemma 2.5, so $\exists a_{i,j,s} \in S$ such that $(a_{i,j,s}, a_{q,j,t}) \in \mathcal{R}^{\{M_k\}}$. Hence $S$ is not conflict-free, a contradiction. This proves that

$$\mathcal{E}_{\mathcal{PR}}(AF^{\{M_k\}}) \supseteq \bigcup_{b=1}^{n} E_b \cup \mathcal{E}_{\mathcal{PR}}(AF)$$

For the reverse direction, let $S \in \mathcal{E}_{\mathcal{PR}}(AF^{\{M_k\}})$, and suppose $a_{k,j,s} \notin S$ for any $j$ and $s$. Then by proposition 4.4, $S \in \mathcal{E}_{\mathcal{PR}}(AF)$. Otherwise, $\exists a_{k,j,s} \in S$. We note that $j$ must be distinct for each $S$ as otherwise, $S$ would not be conflict free. Hence, $S$ takes the form $\{a_{k,j_1,s_1}, ..., a_{k,j_b,s_b}, a_{i_{b+1},j_{b+1},s_{b+1}}, ..., a_{i_n,j_n,s_n}\}$ and hence $S \in E_b$, for $1 \le b \le n$. □

This concludes the analysis on the removal and addition of machines on the standard makespan framework. Let us now explore job removal.

**Definition 4.8.** Let $n > p$. Model removing $p$ jobs $\mathcal{J}_p = \{J_{j_1}, ..., J_{j_p}\}$ from $AF$ by:

$$_{\mathcal{J}_p}\mathcal{A} = \mathcal{A} \setminus \bigcup_{l=j_1}^{j_p} \{a_{i,l,s} : 1 \le i \le m,\ 1 \le s \le r\}$$

$$_{\mathcal{J}_p}\mathcal{R} = \mathcal{R} \setminus \bigcup_{l=j_1}^{j_p} \{(a_{i,j,s}, a_{k,l,t}),\ (a_{k,l,t}, a_{i,j,s}) : (i \ne k, j = l) \vee (i = k, j = l, s \ne t) \vee (i = k, j \ne l, s = t)\}$$

$$_{\mathcal{J}_p}AF = <\ _{\mathcal{J}_p}\mathcal{A},\ _{\mathcal{J}_p}\mathcal{R} >$$

We now show that under stable and preferred semantics, upon removing $p$ jobs from the framework, by removing any arguments relating to the $p$ jobs from any $E \in \mathcal{E}_S(AF)$, the new set is an extension in the framework without the $p$ jobs.

**Proposition 4.9.** For $\mathcal{S} \in \{\mathcal{ST}, \mathcal{PR}\}$,

$$\mathcal{E}_{\mathcal{S}}(_{\mathcal{J}_p}AF) = \{E \setminus \{a_{i_1,j_1,s_1}, ..., a_{i_p,j_p,s_q} :\ 1 \le i_q \le m,\ 1 \le s_q \le r,\ q \in \{1, ..., p\}\} : E \in \mathcal{E}_{\mathcal{S}}(AF)\}$$

*Proof.* We induct on $p$. For $p = 1$, removing job $J_l$, we consider the different semantics separately. First consider the case of stable semantics. Let $S \in \{E \setminus \{a_{i,l,s} : 1 \le i \le m,\ 1 \le s \le r\} : E \in \mathcal{E}_{\mathcal{ST}}(AF)\}$. As $E$ is conflict-free in $AF$, it follows that $S$ is conflict-free in $_{\{J_l\}}AF$. For stability, let $a_{p,j,t} \notin S$. By lemma 2.5, $\exists a_{i,j,s} \in S$ such that $(a_{i,j,s}, a_{p,j,t}) \in\ _{\{J_l\}}\mathcal{R}$, so $S \in \mathcal{E}_{\mathcal{ST}}(_{\{J_l\}}AF)$.

Let $S \in \mathcal{E}_{\mathcal{ST}}(_{\{J_l\}}AF)$. We need to show that $S \cup \{a_{i,l,s}\} \in \mathcal{E}_{\mathcal{ST}}(AF)$, for $i, s$ such that $S \cup \{a_{i,l,s}\}$ is conflict-free. Let $S \cup \{a_{i,l,s}\}$ be conflict-free. For stability, $\forall a_{p,j,t} \in\ _{\{J_l\}}\mathcal{A}$, $\exists a_{q,j,h} \in S$ such that $(a_{q,j,h}, a_{p,j,s}) \in S$. Also, $\forall a_{p,l,s} \in \mathcal{A} \setminus\ _{\{J_l\}}\mathcal{A}$, $(a_{i,l,t}, a_{p,l,s}) \in \mathcal{R}$. Hence $S \cup \{a_{i,l,s}\} \in \mathcal{E}_{\mathcal{ST}}(AF)$.

Now let us consider the case of preferred semantics. Let $S \in \mathcal{E}_{\mathcal{PR}}(_{\{J_l\}}AF)$. We show that $S \cup \{a_{i,l,s}\} \in \mathcal{E}_{\mathcal{PR}}(AF)$, for $i$ such that $S \cup \{a_{i,l,s}\}$ is conflict-free. For admissibility, note that $S \in \mathcal{E}_{\mathcal{PR}}(AF)$ and $\forall (a_{p,l,t}, a_{i,l,s}) \in \mathcal{R}$, $\exists (a_{i,l,s}, a_{p,l,t}) \in \mathcal{R}$, i.e. $\{a_{i,l,s}\}$ defends itself, so $S \cup \{a_{i,l,s}\}$ is admissible in $AF$. Suppose now that $S \cup \{a_{i,l,s}\}$ is not maximal. Then $S \cup \{a_{i,l,s}\} \subset A$, where $A$ is some admissible set. Let $A = S \cup \{a_{i,l,s}\} \cup \{a_{p,j,t}\}$. But by lemma 2.5, $\exists a_{q,j,h} \in S \cup \{a_{i,l,s}\}$ such that $(a_{q,j,h}, a_{p,j,s}) \in \mathcal{R}$ so $A$ is not conflict-free, a contradiction. Hence $S \cup \{a_{i,l,t}\} \in \mathcal{E}_{\mathcal{PR}}(AF)$.

Conversely let $S \in \{E \setminus \{a_{i,l,s} : 1 \le i \le m,\ 1 \le s \le r\} : E \in \mathcal{E}_{\mathcal{PR}}(AF)\}$. Conflict-freeness follows from the same argument as the stable case. Suppose $a_{i,j,s} \in S$ is not acceptable with $S$. Then $E$ is not admissible in $AF$, so $S$ must be admissible in $_{\{J_l\}}AF$. For maximality, suppose $S$ is not maximal. Then $S \subset A$ for some admissible set $A$. Let $A = S \cup \{a_{i,j,s}\}$. By lemma 2.5, $\exists a_{p,j,t} \in S$ such that

$(a_{p,j,t}, a_{i,j,s}) \in {}_{\{J_l\}}\mathcal{R}$. So $A$ is not admissible, and hence $S$ is maximal. Thus $S \in \mathcal{E}_{\mathcal{PR}}({}_{\{J_l\}}AF)$.

For the $p$ case, assume true. For the $p+1$ case, by setting $S := E \setminus \{a_{i_1,j_1,s_1}, ..., a_{i_{p+1},j_{p+1},s_{p+1}}\}$,

$$\begin{aligned}\{E \setminus \{a_{i_1,j_1,s_1}, ..., a_{i_{p+1},j_{p+1},s_{p+1}}\} : E \in \mathcal{E}_{\mathcal{S}}(AF)\} &= \{S \setminus \{a_{i_{p+1},j_{p+1},s_{p+1}}\} : S \in \mathcal{E}_{\mathcal{S}}({}_{\mathcal{J}_p}AF)\} \\ &= \mathcal{E}_{\mathcal{S}}({}_{\mathcal{J}_{p+1}}AF)\end{aligned}$$

by applying the $p = 1$ case again. □

A naive algorithm which recomputes all the extensions of the new framework runs in $O(r|\mathcal{M}||\mathcal{E}_{\mathcal{S}}(AF)|)$ time:

**Algorithm 3** Extension re-computation under semantics $S \in \{\mathcal{ST}, \mathcal{PR}\}$ upon removing job $J_l$

```
1: procedure JobRmv($< \mathcal{A}, \mathcal{R} >, \mathcal{E}_{\mathcal{S}}(AF)$)
2:     $\mathcal{E}_{\mathcal{S}}({}_{\{J_l\}}AF) \leftarrow \emptyset$
3:     for $E \in \mathcal{E}_{\mathcal{S}}(AF)$ do
4:         for $1 \leq i \leq m$ do
5:             for $1 \leq s \leq r$ do
6:                 $\mathcal{E}_{\mathcal{S}}({}_{\{J_l\}}AF) \leftarrow \mathcal{E}_{\mathcal{S}}({}_{\{J_l\}}AF) \cup \{E \setminus \{a_{i,l,s}\}\}$
7:     return $\mathcal{E}_{\mathcal{S}}({}_{\{J_l\}}AF)$
```

Finally, we move onto job addition.

**Definition 4.10.** Model adding $p$ jobs $\mathcal{J}_p = \{J_{j_1}, ..., J_{j_p}\}$ from $AF$ by:

$$^{\mathcal{J}_p}\mathcal{A} = \bigcup_{l=j_1}^{j_p} \{a_{i,l,s} : 1 \leq i \leq m,\ 1 \leq s \leq r\} \cup \mathcal{A}$$

$$^{\mathcal{J}_p}\mathcal{R} = \bigcup_{l=j_1}^{j_p} \{(a_{i,j,s}, a_{k,l,t}),\ (a_{k,l,t}, a_{i,j,s}) : (i \neq k, j = l) \vee (i = k, j = l, s \neq t) \vee (i = k, j \neq l, s = t)\} \cup \mathcal{R}$$

$$^{\mathcal{J}_p}AF = <{}^{\mathcal{J}_p}\mathcal{A},\ {}^{\mathcal{J}_p}\mathcal{R}>$$

**Proposition 4.11.** Let

$$E_b = \{E \cup \{a_{i_1,j_1,s_1}, ..., a_{i_p,j_p,s_p}\} : E \in \mathcal{E}_{\mathcal{S}}(AF),\ 1 \leq i_q \leq m,\ 1 \leq j_q \leq n, 1 \leq s_q \leq r,\ q \in \{1, ..., p\}\}$$

For semantics $\mathcal{S} \in \{\mathcal{ST}, \mathcal{PR}\}$,

$$\mathcal{E}_{\mathcal{S}}({}^{\mathcal{J}_p}AF) = \{E \in E_b : \forall a_{i,j,s}, a_{k,l,t} \in E,\ j \neq l \wedge (i \neq k \vee s \neq t) \text{ pairwise}\}$$

*Proof.* Induct on $p$. For $p = 1$, consider adding a job $J_l$ in the two different semantics separately. Let us look at stable semantics first. Let $E \in \mathcal{E}_{\mathcal{ST}}(AF)$. Pick $\{a_{i,l,s}\}$ such that for every $a_{k,l,t} \in E$, $a_{i,l,s}$ satisfies $j \neq l \wedge (i \neq k \vee s \neq t)$ pairwise. Then $E$ is conflict-free. For stability, it suffices to note that $E \in \mathcal{E}_{\mathcal{ST}}(AF)$ and that $(a_{i,l,s}, a_{k,l,t})$ for $i \neq k$ and any $t, s$. Hence $E \cup \{a_{i,l,s}\} \in \mathcal{E}_{\mathcal{ST}}({}^{\{J_l\}}AF)$.

Now consider $S \in \mathcal{E}_{\mathcal{ST}}({}^{\{J_l\}}AF)$. We show that $A := S \setminus \{a_{i,l,s}\} \in \mathcal{E}_{\mathcal{ST}}(AF)$. It is clear that $A$ is conflict-free in $AF$ as $S$ is conflict-free in ${}^{\{J_l\}}AF$. By lemma 2.5, it follows that $\forall j\{1,...,n\}$, $a_{k,j,t} \in A$ for some $k, t$. Hence, any $a_{p,j,q} \notin S$ is attacked by $S$ as $(a_{k,j,t}, a_{p,j,q}) \in \mathcal{R}$. So $A \in \mathcal{E}_{\mathcal{ST}}(AF)$.

Consider now the case of preferred semantics. Let $E \in \mathcal{E}_{\mathcal{PR}}(AF)$. Pick $\{a_{i,l,s}\}$ such that for every $a_{k,l,t} \in E$, $a_{i,l,s}$ satisfies $j \neq l \wedge (i \neq k \vee s \neq t)$ pairwise. Then $E$ is conflict-free. For admissibility, note that $E \in \mathcal{E}_{\mathcal{PR}}(AF)$ and $\forall(a_{k,l,t}, a_{i,l,s}) \in {}^{\{J_l\}}\mathcal{R}$, $\exists(a_{i,l,s}, a_{k,l,t}) \in {}^{\{J_l\}}\mathcal{R}$, i.e. $\{a_{i,l,s}\}$ defends itself, so $E \cup \{a_{i,l,s}\}$ is admissible. Suppose now that $E \cup \{a_{i,r}\}$ is not maximal. Then $E \cup \{a_{i,l,s}\} \subset A$, where $A$ is some admissible set. Let $A = E \cup \{a_{i,l,s}\} \cup \{a_{p,j,q}\}$. But then $\exists a_{x,j,y} \in E \cup \{a_{i,l,s}\}$ such that $(a_{x,j,y}, a_{p,j,q}) \in {}^{\{J_l\}}\mathcal{R}$ so $A$ is not conflict-free, a contradiction. Hence $E \cup \{a_{i,l,s}\} \in \mathcal{E}_{\mathcal{PR}}({}^{\{J_l\}}AF)$.

Now consider $S \in E \in \mathcal{E}_{\mathcal{PR}}({}^{+r}AF)$. We show that $B := S \setminus \{a_{i,r}\} \in \mathcal{E}_{\mathcal{PR}}(AF)$. It is clear that $B$ is admissible in $AF$ as $S$ is admissible in ${}^{\{J_l\}}AF$. Suppose $B$ is not maximal, then $B \subset A$ for some admissible set $A$. Let $A = B \cup \{a_{p,j,s}\}$. But then $\exists a_{q,j,t} \in A$ such that $(a_{q,j,t}, a_{p,j,s}) \in \mathcal{R}$ so $A$ is not conflict-free, a contradiction. Hence $B \in \mathcal{E}_{\mathcal{PR}}(AF)$.

For $p$, assume true. For the case $p+1$, by setting $S := E \cup \{a_{i_1,j_1}, ..., a_{i_{p+1},j_{p+1}}\}$,

$$\begin{aligned}\{E \cup \{a_{i_1,j_1}, ..., a_{i_{p+1},j_{p+1}}\} : E \in \mathcal{E}_{\mathcal{S}}(AF)\} &= \{S \cup \{a_{i_{p+1},j_{p+1}}\} : S \in \mathcal{E}_{\mathcal{S}}({}^{\mathcal{J}_p}AF)\} \\ &= \mathcal{E}_{\mathcal{S}}({}^{\mathcal{J}_{p+1}}AF)\end{aligned}$$

by adding another job case the argumentation framework of $S$. □

This suggests the following algorithm running in $O(r|\mathcal{M}||\mathcal{J}||\mathcal{E}_{\mathcal{S}}(AF)|)$ time:

**Algorithm 4** Extension re-computation under semantics $S \in \{\mathcal{ST}, \mathcal{PR}\}$ upon adding job $J_l$

```
1: procedure JobRmv($< \mathcal{A}, \mathcal{R} >, \mathcal{E}_{\mathcal{S}}(AF)$)
2:     $\mathcal{E}_{\mathcal{S}}({}_{\{J_l\}}AF) \leftarrow \emptyset$
3:     for $E \in \mathcal{E}_{\mathcal{S}}(AF)$ do
4:         for $1 \leq i \leq m$ do
5:             for $1 \leq s \leq r$ do
6:                 if $j \neq l \wedge (i \neq k \vee s \neq t)$ pairwise $\forall a_{i,j,s} \in E$ then
7:                     $\mathcal{E}_{\mathcal{S}}({}^{\{J_l\}}AF) \leftarrow \mathcal{E}_{\mathcal{S}}({}^{\{J_l\}}AF) \cup \{E \cup \{a_{i,l,s}\}\}$
8:     return $\mathcal{E}_{\mathcal{S}}({}^{\{J_l\}}AF)$
```

## 4.2 Minimum Swaps under Change

Returning to the view of individual extensions, we now investigate a lower bound on the minimum number of swaps (which in turn, is related to finding the best possible algorithm in terms of minimisation of the minimum number of swaps, or the minimal disruption to a schedule). Beginning with removal of a single machine, it is easy to see that the following greedy algorithm takes swaps $n_k$ to reschedule $E \in \mathcal{E}_{\mathcal{S}}(AF)$ to $E' \in \mathcal{E}_{\mathcal{S}}(AF_{\{M_k\}})$, where $n_k$ is the number of jobs scheduled on machine $M_k$:

**Algorithm 5** Mapping $E \in \mathcal{E}_{\mathcal{S}}(AF)$ to $E' \in \mathcal{E}_{\mathcal{S}}(AF_{\{M_k\}})$

```
1: procedure MapMacRmv($E$, $< \mathcal{A}, \mathcal{R} >$)
2:     For $1 \leq i \leq m$, compute $C[i] \leftarrow (\max_{a_{i,j,s} \in E} s) + 1$ to the completion time of machine $i$.
3:     Find $C[q] \leftarrow \min_{1 \leq i \leq m} C[i]$.
4:     for $a_{i,j,s} \in E$, $i = k$ do
5:         $E \leftarrow (E \setminus \{a_{i,j,s}\}) \cup \{a_{q,j,C[q]}\}$
6:         $C[q] \leftarrow C[q] + p_j$
7:         Recompute $C[q] \leftarrow \min_{1 \leq i \leq m} C[i]$.
8:     return $E$
```

We now want to show that no other algorithm relying on swaps only to reschedule an extension can run in better than $n_k$ swaps.

**Lemma 4.12.** For any ALGO where swapping is the only rescheduling mechanism, $\mathcal{MS}(E) \geq n_k$.

*Proof.* Let jobs $J_p, ..., J_q$ be assigned to machine $M_k$, where $n_k = q - p$. As machine $M_k$ is removed from the framework, these jobs must be assigned to another machine in order for the schedule to feasible. It now suffices to note that there are $n_k$ elements of the form $a_{k,j,s}$, $j \in \{p, ..., q\}$, $s \in \{1, ...r\}$ that need to be removed from $E$, and that any swap $E \leftarrow (E \setminus \{a_{k,j,s}\}) \cup \{a_{u,j,v}\}$ can only reschedule a single job $j$ either correctly or incorrectly, so $\mathcal{MS}(E) \geq n_k$. $\square$

On the other hand, we proved that that under addition of a single machine, $\mathcal{E}_{\mathcal{S}}(AF) \subseteq \mathcal{E}_{\mathcal{S}}(AF^{\{M_k\}})$, or equivalently, every feasible schedule in the original framework remains a feasible schedule upon adding machine. It follows that $\mathcal{MS}(E) \geq 0$, i.e. an "empty algorithm" which does nothing suffices.

When removing or adding a single job, we showed that only one element of the extension needs to be removed or added. Thus it follows that $\mathcal{MS}(E) \geq 1$. We will give an example of a mapping which takes one swap in the case of job addition; the job removal case is the same with the exception that the set union operation should be a set difference operation:

**Algorithm 6** Mapping $E \in \mathcal{E}_{\mathcal{S}}(AF)$ to $E' \in \mathcal{E}_{\mathcal{S}}(^{\{J_l\}}AF)$

```
1: procedure MapJobAdd($E$, $< \mathcal{A}, \mathcal{R} >$)
2:     For $1 \leq i \leq m$, compute $C[i] \leftarrow (\max_{a_{i,j,s} \in E} s) + 1$ to the completion time of machine $i$.
3:     Find $C[q] \leftarrow \min_{1 \leq i \leq m} C[i]$.
4:     $E \leftarrow (E \setminus \emptyset) \cup \{a_{q,l,C[q]}\}$
5:     $C[q] \leftarrow C[q] + p_j$
6:     return $E$
```

In summary, the main results are listed below:

- Removing a single machine implies that $\mathcal{MS}(E) \geq n_k$ for any ALGO and any $E$.
- Adding a single machine implies that $\mathcal{MS}(E) \geq 0$ for any ALGO and any $E$.
- Removing a single job implies that $\mathcal{MS}(E) \geq 1$ for any ALGO and any $E$.
- Adding a single job implies that $\mathcal{MS}(E) \geq 1$ for any ALGO and any $E$.

## 4.3 Bounding Swaps under Common Constraints

A natural extension of the above is to explore a bound for the minimum number of swaps of the makespan framework under constraints described in section 3. Under these constraints, it is clear that adding a machine does not affect the feasibility of an extension, and similarly removing a job does not violate the additional constraint, and hence in these two cases it follows that the value of $\mathcal{MS}(E)$ does not change if we use the algorithms developed in section 4.2. The only two outstanding problems now are to explore the changes to the value of $\mathcal{MS}(E)$ upon addition of a job into a preexisting constraint (for example, adding a $J_j$ to a preexisting precedence relation $[J_p, ..., J_q]_{PR}$), or removing a machine containing jobs that are contained in a constraint. However, it is clear that removing a machine $M_k$ is equivalent to rescheduling the jobs that were scheduled onto $M_k$, which is equivalent to running the job addition algorithm $n_k$ times. Thus, we only need to consider the problem in the case job addition.

To help us visualise the remaining problems in detail, we define the *scheduling matrix* by a $|\mathcal{M}| \times r$ matrix $B = (B_{i,s})$ such that $B_{i,s} = j$ if $J_j$ is scheduled on machine $M_i$ on slot $s$. Set $B_{i,s} = 0$ if no job is scheduled on machine $M_i$ on slot $s$.

**Example 4.13.** Let $|\mathcal{M}| = |\mathcal{J}| = r = 2$ and $[J_1, J_2]_{PR}$. Then $E = \{a_{111}, a_{112}\}$ is an extension. Then

$$B_{i,s} = \begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix}$$

We first examine adding a single concurrent execution constraint on the standard framework when adding a job. More formally, let $AF =< \mathcal{A}, \mathcal{R} >$ be given and impose the concurrent execution $[J_p, ..., J_q]_{CE}$. To guarantee that it is possible to obtain a feasible schedule after job addition, we set $m \geq n + 1$ and $rm \geq n$. Now add a job $J_l$ to $AF$, leading to ${}^{\{J_l\}}AF$ and amend the constraint to $[J_p, ..., J_q, J_l]_{CE}$. We claim that the following algorithm suffices to map any extension in the old framework $E \in \mathcal{E}_{\mathcal{S}}(AF)$ to $E' \in \mathcal{E}_{\mathcal{S}}({}^{\{J_l\}}AF)$:

**Algorithm 7** Mapping $E \in \mathcal{E}_{\mathcal{S}}(AF)$ to $E' \in \mathcal{E}_{\mathcal{S}}({}^{\{J_l\}}AF)$ given $R$, a single concurrent execution chain

```
1: procedure MapJobAddCE($E$, $R = [J_p, ..., J_q, J_l]_{CE}$, $J_l$)
2:     Find $a_{i,p,t} \in E$ for some $i, t$
3:     if $\exists a_{k,h,t} \in E$, $1 \leq k \leq m$, $h \notin \{p, ..., q\}$ then
4:         $E \leftarrow (E \setminus \{a_{k,h,t}\}) \cup \{a_{k,l,t}\}$
5:         for $1 \leq u \leq m$, $1 \leq v \leq r$ do
6:             if $\nexists a_{u,j,v}$ $\forall j$ then
7:                 $E \leftarrow (E \setminus \emptyset) \cup \{a_{u,h,v}\}$
8:                 return $E$
9:     if $\nexists a_{k,h,t} \in E$, $1 \leq k \leq m$, $h \notin \{p, ..., q\}$ then
10:        $E \leftarrow (E \setminus \emptyset) \cup \{a_{h,l,t}\}$
11:        return $E$
```

To give some context on how the algorithm works, we first find an argument for job $J_p$ (which is part of the concurrent execution chain) so we know which value of $1 \leq t \leq r$ to add job $J_l$ to. As $m \geq n+1$, there must be at least one machine $M_k$ in which we can schedule $J_l$ to slot $t$. We now need to consider two cases; if machine $M_k$, slot $t$ is already occupied, and if it is empty. Suppose it is occupied by $J_h$. Then we assign $J_l$ to machine $M_k$, slot $t$ and reassign $J_h$ to some other free slot. This is done in lines 3 to 8. Otherwise, $M_k$, slot $t$ is free and we simply assign $J_l$ there (lines 9 to 11). This is perhaps better illustrated in two examples below:

**Example 4.14.** Let $|\mathcal{M}| = |\mathcal{J}| = r = 2$ and $[J_1, J_2]_{CE}$. Then

$$B_{i,s} = \begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 4 & 0 \end{bmatrix}$$

is a valid extension. Suppose now we add job $J_3$ and amend the concurrent execution constraint to $[J_1, J_2, J_3]_{CE}$. Then our algorithm swaps does the following swaps:

$$\begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 4 & 0 \end{bmatrix} \to \begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 3 & 0 \end{bmatrix} \to \begin{bmatrix} 1 & 4 \\ 2 & 0 \\ 3 & 0 \end{bmatrix}$$

This is the case in lines 3 to 8. For the case in lines 9 to 11, consider

$$B_{i,s} = \begin{bmatrix} 1 & 4 \\ 2 & 0 \\ 0 & 0 \end{bmatrix}$$

The swaps are:

$$\begin{bmatrix} 1 & 4 \\ 2 & 0 \\ 0 & 0 \end{bmatrix} \to \begin{bmatrix} 1 & 4 \\ 2 & 0 \\ 3 & 0 \end{bmatrix}$$

The correctness of this algorithm is relatively apparent, but we prove it for the sake of completeness:

**Lemma 4.15.** Let $E' =$MapJobAddCE($E$, $R = [J_p, ..., J_q]_{CE}$, $J_l$). Then $E' \in \mathcal{E}_{\mathcal{S}}(^{\{J_l\}}AF)$ and $E'$ satisfies $[J_p, ..., J_q, J_l]_{CE}$.

*Proof.* As $E \in \mathcal{E}_{\mathcal{S}}(AF)$, $J_l$ must be assigned to a machine in order for $E \in \mathcal{E}_{\mathcal{S}}(^{\{J_l\}}AF)$. In the case that $\exists a_{k,h,t} \in E$, $1 \leq k \leq m$, $h \notin \{p, ..., q\}$, the swap $E \leftarrow (E \setminus \{a_{k,h,t}\}) \cup \{a_{k,l,t}\}$ forces $E \in \mathcal{E}_{\mathcal{S}}(^{\{J_l\}}_{\{J_h\}}AF)$. Moreover, as $J_p$ is scheduled on slot $t$, $[J_p, ..., J_q, J_l]_{CE}$ is satisfied. Finally, $E \cup \{a_{u,h,v}\} \in \mathcal{E}_{\mathcal{S}}(^{\{J_l\}}AF)$, as proved in proposition 4.11. Otherwise, if $\nexists a_{k,h,t} \in E$, $1 \leq k \leq m$, $h \notin \{p, ..., q\}$, the swap $E \leftarrow (E \setminus \emptyset) \cup \{a_{h,l,t}\}$ also forces $E \cup \{a_{u,h,v}\} \in \mathcal{E}_{\mathcal{S}}(^{\{J_l\}}AF)$ by proposition 4.11. □

It is also evident that the algorithm MapJobAddCE has the property $\mathcal{MS}(E) \leq 2$, as in the first case we require two swaps and in the second only one is required.

We move onto the restricted assignment constraint; let $AF =< \mathcal{A}, \mathcal{R} >$ be given and impose the concurrent execution $[\{M_x, ..., M_y\}, \{J_p, ..., J_q\}]_{RA}$. To guarantee that it is possible to obtain a feasible schedule after job addition, we set $(y - x)r > q - p + 1$ and $rm > n$. Now, suppose we add job $J_l$ and amend $[\{M_x, ..., M_y\}, \{J_p, ..., J_q, J_l\}]_{RA}$. A simple algorithm suffices:

**Algorithm 8** Mapping $E \in \mathcal{E}_{\mathcal{S}}(AF)$ to $E' \in \mathcal{E}_{\mathcal{S}}(^{\{J_l\}}AF)$ given $R$, a single restricted assignment costraint

```
1: procedure MapJobAddRA($E$, $R = [\{M_x, ..., M_y\}, \{J_p, ..., J_q, J_l\}]_{RA}$, $J_l$)
2:     if $\exists a_{i,j,s} \in E \; \forall x \leq i \leq y, \; \forall 1 \leq s \leq r$ and some $1 \leq j \leq n$ then
3:         Pick $a_{i,j,s}$ s.t. $j \notin \{p, ..., q\}$
4:         $E \leftarrow (E \setminus \{a_{i,j,s}\}) \cup \{a_{i,l,s}\}$
5:         for $1 \leq u \leq m, \; 1 \leq v \leq r$ do
6:             if $\nexists a_{u,h,v} \; \forall h$ then
7:                 $E \leftarrow (E \setminus \emptyset) \cup \{a_{u,j,v}\}$
8:                 return $E$
9:     if $\nexists a_{i,j,s} \in E$ for some $x \leq i \leq y$, for some $1 \leq s \leq r$, $\forall 1 \leq j \leq n$ then
10:        $E \leftarrow (E \setminus \emptyset) \cup \{a_{i,l,s}\}$
11:        return $E$
```

This algorithm is similar to the one in concurrent execution; we split into two cases. If there is a job scheduled on every $M_i$, $i \in \{p, .., q\}$ and every $s$, there must be a job which is not under restricted assignment which can be swapped to another slot. Thus, we assign $J_l$ to $a_{i,l,s}$ and then swap the free argument to another slot $a_{u,j,v}$. If on the other hand there was already a free slot on $M_i$, $i \in \{p, .., q\}$ for some $s$, then we can simply assign $J_l$ to machine $M_i$, slot $s$.

Following the same argument as above, we obtain $MS(E) \leq 2$ for the algorithm MapJobAddRA. Let us now consider applying a single precedence relation to the standard makespan framework. Recall the notation from section 3, where $[J_p, ..., J_q]_{PR}$ is a precedence relation chain if $J_p$ has to be executed before $J_{p+1}$, and so on. We require some additional notation; define the *length* of a chain $[J_p, ...., J_q]_{PR}$ as $q - p$. To guarantee it is possible to obtain a feasible schedule after adding a job, we impose the conditions $r \geq n + 1$ and $rm \geq n$.

Consider adding a job to a preexisting precedence relation chain, so let $AF =< \mathcal{A}, \mathcal{R} >$ be given and impose the precedence relation constraint $[J_p, ..., J_q]_{PR}$. Now add a job $J_l$ to $AF$, leading to $^{\{J_l\}}AF$ and amend the constraint to include $J_l$. We will allow the job $J_l$ to be inserted anywhere into a chain $R = [J_p, ..., J_q]_{PR}$, which constitutes the division of the problem into four parts. Firstly, $J_l$ may have to be executed before $J_d$ but after $J_c$; we shall call this slotting $J_l$ in the middle. Alternatively, $J_l$ has to be "at the front", so $J_j$ has to be executed before any $J_d$, for all $d$. $J_l$ can also be "at the back", so $J_j$ is executed after every $J_c$, for all $c$. Finally, $J_l$ can also not be in a precedence relation. More concisely:

- If $\exists c, d \in \{p, ..., q\}$ such that $[J_c, J_j, J_d]_{PR}$ is also sub-preference relation chain of $R$.
- If $[J_j, J_d]_{PR} \; \forall d \in \{p, .., q\}$.
- If $[J_c, J_j]_{PR} \; \forall c \in \{p, ...q\}$.
- If $J_j$ is not in a precedence relation chain.

If $J_l$ is not in a precedence relation, then this simply devolves into simple job addition, which have already done above. The remaining cases are more difficult; let us begin with the third, if $[J_c, J_l]_{PR}\ \forall c \in \{p, ...q\}$.

Let us breakdown the situations which can occur in the third case. Let $J_\gamma$ be the maximal element such that $[J_c, J_\gamma]_{PR}\ \forall c \in \{p, ..q\}$, but $[J_\gamma, J_l]_{PR}$. In the case that there is $B_{i,s} = 0$ with $s > \gamma$ for any $i$, we can simply assign $J_l$ to $M_i$ on slot $s$. Let us see this in the following example:

**Example 4.16.** Let $|\mathcal{M}| = |\mathcal{J}| = 2$, $r = 3$ and $[J_1, J_2]_{PR}$. Then $E = \{a_{111}, a_{112}\}$ is an extension. Upon adding $J_3$ and amending $[J_1, J_2, J_3]_{PR}$:

$$B_{i,s} = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \end{bmatrix}$$

It is easy to see that the matrix on the right hand side is an extension and satisfies the precedence chain.

However, it can be the case that $B_{i,s} > 0$ for $s > \gamma$ and any $i$. Under the supposition that it is still possible to obtain a feasible schedule after adding a job, there must be some empty slot on a machine in an earlier position such that some $J_c$ with $[J_c, J_l]_{PR}$ can be assigned to that slot. We shift $J_c$ to the earlier slot, and then assign $J_c$ to the slot that was previously assigned to $J_c$, which gives a correct assignment. We show this in an example:

**Example 4.17.** Let $|\mathcal{M}| = 2$, $|\mathcal{J}| = 7$, $r = 4$ and $[J_1, J_2, J_3]_{PR}$. Then $E = \{a_{111}, a_{112}, a_{223}\}$ is an extension. Here we denote $X$ jobs in $\mathcal{J}$ but are not in a precedence relation to note that $B_{i,s}$ is occupied:

$$B_{i,s} = \begin{bmatrix} 1 & 2 & 0 & X \\ X & X & X & 3 \end{bmatrix}$$

Upon adding $J_4$ and amending to $[J_1, J_2, J_3, J_4]_{PR}$, there are no remaining slots at the end of $J_3$. In this case, we swap as follows:

$$\begin{bmatrix} 1 & 2 & 0 & X \\ X & X & X & 3 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 & X \\ X & X & X & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 & X \\ X & X & X & 4 \end{bmatrix}$$

which is indeed a correct assignment.

This allows us to build the ShiftLeft algorithm:

**Algorithm 9** ShiftLeft Algorithm

```
1: procedure ShiftLeft($E$, $[J_p, ..., J_q]_{PR}$ including $J_l$, $J_l$)
2:     for $\alpha$ such that $[J_\alpha, J_l]_{PR}$ in increasing order of $\alpha$ do
3:         Find $a_{j,\alpha,f} \in E$ for some $j, f$
4:         if $a_{w,h,g} \notin E$ for $g < f$ and any $h$, some $w, g$ then
5:             $E \leftarrow (E \setminus \{a_{j,e,f}\}) \cup \{a_{w,j,g}\}$
6:             $prevl \leftarrow j$
7:             $prevf \leftarrow f$
8:             break;
9:     for $\beta$ such that $[J_\beta, J_l]_{PR}$ in increasing order of $\beta$, $\beta > \alpha$ do
10:        Find $a_{\delta,\beta,\sigma} \in E$ for some $\delta, \sigma$
11:        $E \leftarrow (E \setminus \{a_{\delta,\beta,\sigma}\}) \cup \{a_{prevl,\beta,prevf}\}$
12:        $prevl \leftarrow \delta$
13:        $prevf \leftarrow \sigma$
14:    $E \leftarrow (E \setminus \emptyset) \cup \{a_{prevl,l,prevf}\}$
```

Similarly, we note that if $J_j$ is at the front, then we may need to "shift right", as exemplified in the following example:

**Example 4.18.** Let $|\mathcal{M}| = 1$, $|\mathcal{J}| = 3$, $r = 4$ and $[J_2, J_3, J_4]_{PR}$. Then $E = \{a_{121}, a_{132}, a_{143}\}$ is an extension.

$$B_{i,s} = \begin{bmatrix} 2 & 3 & 4 & 0 \end{bmatrix}$$

Upon adding $J_1$ and amending to $[J_1, J_2, J_3, J_4]_{PR}$, there are no remaining slots before $J_2$. In this case, we swap as follows:

$$\begin{bmatrix} 2 & 3 & 0 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 0 & 3 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 2 & 3 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

which is indeed a correct assignment.

The following is the corresponding ShiftRight algorithm:

**Algorithm 10** ShiftRight Algorithm

```
1: procedure ShiftRight($E$, $[J_p, ..., J_q]_{PR}$ including $J_l$, $J_l$)
2:     for $\alpha$ such that $[J_l, J_\alpha]_{PR}$ in descending order of $\alpha$ do
3:         Find $a_{j,\alpha,f} \in E$ for some $j, f$
4:         if $a_{w,h,g} \notin E$ for $g > f$ and any $h$, some $w, g$ then
5:             $E \leftarrow (E \setminus \{a_{j,e,f}\}) \cup \{a_{w,l,g}\}$
6:             $prevl \leftarrow j$
7:             $prevf \leftarrow f$
8:             break;
9:     for $\beta$ such that $[J_l, J_\beta]_{PR}$ in descending order of $\beta$, $\beta < \alpha$ do
10:        Find $a_{\delta,\beta,\sigma} \in E$ for some $\delta, \sigma$
11:        $E \leftarrow (E \setminus \{a_{\delta,\beta,\sigma}\}) \cup \{a_{prevl,\beta,prevf}\}$
12:        $prevl \leftarrow \delta$
13:        $prevf \leftarrow \sigma$
14:    $E \leftarrow (E \setminus \emptyset) \cup \{a_{prevl,l,prevf}\}$
```

This solves cases 2,3. For case 1, if $\exists c, d \in \{p, ..., q\}$ such that $[J_c, J_j, J_d]_{PR}$ is also sub-preference relation chain of $R$, then it is not entirely clear which side we should shift to. We give examples to show this:

**Example 4.19.** Let $|\mathcal{M}| = 1$, $|\mathcal{J}| = 4$, $r = 5$ and $[J_1, J_2, J_3, J_5]_{PR}$. Then $E = \{a_{111}, a_{122}, a_{133}, a_{154}\}$ is an extension.

$$B_{i,s} = \begin{bmatrix} 1 & 2 & 3 & 5 & 0 \end{bmatrix}$$

Upon adding $J_4$ and amending to $[J_1, J_2, J_3, J_4, J_5]_{PR}$, we swap as follows:

$$\begin{bmatrix} 1 & 2 & 3 & 5 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 & 0 & 5 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

which is indeed a correct assignment. If on the other hand we were given

$$B_{i,s} = \begin{bmatrix} 1 & 2 & 0 & 3 & 5 \end{bmatrix}$$

then we need to swap as follows:

$$\begin{bmatrix} 1 & 2 & 0 & 3 & 5 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 & 0 & 5 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

This allows us to build the following algorithm:

**Algorithm 11** Mapping $E \in \mathcal{E}_{\mathcal{S}}(AF)$ to $E' \in \mathcal{E}_{\mathcal{S}}(^{\{J_l\}}AF)$ given $R$, a single precedence relation

```
procedure MapJobAddPR($E$, $<\mathcal{A},\mathcal{R}>$, $R = [J_p, ..., J_q]_{PR}$ including $J_l$, $J_l$ assigned on $a_{k,l,s}$)
    Find the two *most adjacent* elements in the chain $J_c$, $J_d$ such that $[J_c, J_l, J_d]_{PR}$.
    if $\exists a_{u,c,y} \in E \wedge \exists a_{v,d,z} \in E$ for some $u, v, y, z$ then
        if $a_{i,h,x} \notin E$ for any $h$, some $y \leq x \leq z$, some $1 \leq i \leq m$ then
            $E \leftarrow (E \setminus \{a_{k,l,s}\}) \cup \{a_{i,l,x}\}$
            return $E$
        else
            $E_{copy} \leftarrow E$
            $E =$ ShiftLeft($E, [J_p, ..., J_q]_{PR}, J_l$)
            if $E$ does not satisfy $[J_p, ..., J_q]_{PR}$ then
                $E \leftarrow E_{copy}$
                $E =$ ShiftRight($E, [J_p, ..., J_q]_{PR}, J_l$)
    if $\nexists a_{u,c,y} \in E \wedge \exists a_{v,d,z} \in E$ for some $u, v, y, z$ then
        if $a_{i,h,x} \notin E$ for any $h$, some $1 \leq x \leq d$, some $1 \leq i \leq m$ then
            $E \leftarrow (E \setminus \{a_{k,l,s}\}) \cup \{a_{i,l,x}\}$
            return $E$
        else
            $E =$ ShiftRight($E, [J_p, ..., J_q]_{PR}, J_l$)
    if $\exists a_{u,c,y} \in E \wedge \nexists a_{v,d,z} \in E$ for some $u, v, y, z$ then
        if $a_{i,h,x} \notin E$ for any $h$, some $c \leq x \leq r$, some $1 \leq i \leq m$ then
            $E \leftarrow (E \setminus \{a_{k,l,s}\}) \cup \{a_{i,l,x}\}$
            return $E$
        else
            $E =$ ShiftLeft($E, [J_p, ..., J_q]_{PR}, J_l$)
    if $J_j$ is not in $R$ then
        $E =$ MapJobAdd($E, <\mathcal{A},\mathcal{R}>$)
    return $E$
```

It is now easy to prove that the number of swaps performed by MapMacRmvPR is at most the length of the precedence relation $[J_p, ...., J_q]_{PR}$:

**Lemma 4.20.** Let $[J_p, ...., J_q]_{PR}$ be a precedence relation including the new job $J_l$. Then for the algorithm MapJobAddPR, $\mathcal{MS}(E) \leq q - p$ for any $E \in \mathcal{E}_{\mathcal{S}}(AF)$.

*Proof.* We note that in lines 4-6, 14-16, 20-22, and MapJobAdd only takes one swap, if they occur. Otherwise, we either have use ShiftLeft or ShiftRight. Let us consider in the three different cases we defined above. In the first case, where we can find both $J_c$ and $J_d$ such that $[J_c, J_l, J_d]_{PR}$, suppose without loss of generality that we need to ShiftLeft. Then every $J_\alpha$ with $[J_\alpha, J_l]_{PR}$ must shift to an earlier slot. Hence in total, $j - p < q - p$ swaps are done. We can hence deduce that if ShiftRight was used instead, $q - j < q - p$ swaps are done. In the second case where $l = q$, everything before

$j$ must be pushed to the left, meaning less than or exactly $q - p$ swaps are performed. The same conclusion can be drawn in the third case where $l = p$. □

The final constraint we examine is forbidden sets. Consider adding a job to a preexisting forbidden set chain, so let $AF =< \mathcal{A}, \mathcal{R} >$ be given and impose the precedence relation constraint $[J_p, ..., J_q]_{FS}$. Now add a job $J_l$ to $AF$, leading to ${}^{\{J_l\}}AF$ and amend the constraint to include $[J_p, ..., J_q, J_l]_{FS}$. As it is relatively similar to concurrent execution, we will just give the algorithm and claim that it runs with $\mathcal{MS}(E) \leq 2$:

**Algorithm 12** Mapping $E \in \mathcal{E}_\mathcal{S}(AF)$ to $E' \in \mathcal{E}_\mathcal{S}({}^{\{J_l\}}AF)$ given $R$, a single forbidden sets costraint

```
 1: procedure MapJobAddFS($E$, $R = [J_p, ..., J_q, J_l]_{FS}$, $J_l$)
 2:     if $\exists a_{i,j,s} \in E\ \forall 1 \leq i \leq m,\ s \notin \{p, ..., q\}$ and some $1 \leq j \leq n$ then
 3:         Pick $a_{i,j,s} \in E$ s.t. $s \notin \{p, ..., q\}$
 4:         $E \leftarrow (E \setminus \{a_{i,j,s}\}) \cup \{a_{i,l,s}\}$
 5:         for $1 \leq u \leq m$, $1 \leq v \leq r$ do
 6:             if $\nexists a_{u,h,v}\ \forall h$ then
 7:                 $E \leftarrow (E \setminus \emptyset) \cup \{a_{u,j,v}\}$
 8:                 return $E$
 9:     if $\nexists a_{i,j,s} \in E$ for some $x \leq i \leq y, s \in \{p, ..., q\},\ \forall 1 \leq j \leq n$ then
10:         $E \leftarrow (E \setminus \emptyset) \cup \{a_{i,l,s}\}$
11:         return $E$
```

In summary, we gave algorithms such that:

- $\mathcal{MS}(E) \leq 2$ for all $E$ under constraints CE, RA, FS.
- $\mathcal{MS}(E) \leq q - p$ for all $E$ under constraints PR.

## 5 Discussion and Future Work

---

With regards to section 4.1, we give two interesting results, albeit using the model in definition 2.6. The first is a proposition on the number of extensions in any makespan network.

**Proposition 5.1.** Let $AF =< \mathcal{A}, \mathcal{R} >$ be given as in definition 2.6 and let $m \geq 1$, $n \geq 1$. For semantics $\mathcal{S} \in \{\mathcal{ST}, \mathcal{PR}\}$, $|\mathcal{E}_{\mathcal{S}}(AF)| = m^n$.

*Proof.* We induct on $m$ and $n$. Note that if $m = n = 1$, then $\mathcal{E}_{\mathcal{S}}(AF) = \{a_{11}\}$ is the only extension.

Now fix $n$ and suppose the result holds for $m = p$. To show that it holds for $m = p + 1$, suppose machine $M_k$ is added to $AF$, giving $AF^{\{M_k\}}$; we consider the number of extensions that are in $AF^{\{M_k\}}$, but not in $AF$. As shown above, any such extension takes the form

$$E = \{a_{k,j_1}, a_{k,j_2}, ..., a_{k,j_b}, a_{i_{b+1},j_{b+1}}, ..., a_{i_n,j_n}\}$$

where $b$ elements of $E$ are arguments of machine $M_k$, with $1 \leq b \leq n$. There are exactly $\binom{n}{b}$ choices for the $b$ elements $a_{k,j_c} \in E$. It remains to choose the elements $a_{i,j_d} \in E$. As $(a_{i,j_d}, a_{s,j_d}) \in \mathcal{R}$ for $i \neq s$, there are exactly $m^{n-b}$ choices for the $n - b$ elements $a_{i,j_d} \in E$. Hence, there are exactly $\binom{n}{b}m^{n-b}$ choices of $E$. Since $b$ can range from $1 \leq b \leq n$, the total number of new extensions is

$$\sum_{b=1}^{n} \binom{n}{b} m^{n-b}$$

Thus the total number of extensions are

$$m^l + \sum_{b=1}^{n} \binom{n}{b} m^{n-b} = \sum_{b=0}^{l} \binom{n}{b} m^{n-b} = (m+1)^n$$

Now fix $m$ and suppose the result holds for $n = q$. Let $B := {}^{\mathcal{J}_q}AF$. For $n = q + 1$, we have shown that $\mathcal{E}_{\mathcal{S}}({}^{+(q+1)}B) = \{E \cup \{a_{i,q+1}\} : E \in \mathcal{E}_{\mathcal{S}}({}^{\mathcal{J}_q}AF),\ i \in \{1, ..., m\}\}$. Hence upon adding a job, each extension $E \in \mathcal{E}_{\mathcal{S}}({}^{\mathcal{J}_q}AF)$ maps to $m$ extensions $S_{i,E} \in \mathcal{E}_{\mathcal{S}}({}^{+(q+1)}B)$. Thus the total number of extensions are $m \times m^n = m^{n+1}$. □

Four corollaries that allow us to quickly determine the number of extensions that are added or removed when a job or machine is added or removed follow directly from the proof above:

**Corollary 5.2.** For semantics $\mathcal{S} \in \{\mathcal{ST}, \mathcal{PR}\}$,

$$|\mathcal{E}_{\mathcal{S}}(AF^{\{M_k\}})| = |\mathcal{E}_{\mathcal{S}}(AF)| + \sum_{b=1}^{|\mathcal{J}|} \binom{|\mathcal{J}|}{b} |\mathcal{M}|^{|\mathcal{J}|-b}$$
$$|\mathcal{E}_{\mathcal{S}}(AF_{\{M_k\}})| = |\mathcal{E}_{\mathcal{S}}(AF)| - \sum_{b=1}^{|\mathcal{J}|} \binom{|\mathcal{J}|}{b} |\mathcal{M}|^{|\mathcal{J}|-b}$$
$$|\mathcal{E}_{\mathcal{S}}({}_{\{J_l\}}AF)| = \frac{|\mathcal{E}_{\mathcal{S}}(AF)|}{|\mathcal{M}|}$$
$$|\mathcal{E}_{\mathcal{S}}({}^{\{J_l\}}AF)| = |\mathcal{E}_{\mathcal{S}}(AF)||\mathcal{M}|$$

It would be useful to generalize this to the makespan model with slots; we believe a sensible guess is $km^n$, where $k$ is in terms of $r, m, n$.

We also present an alternative method to computing extensions under the makespan model in 2.6. This alternate method uses the fact that, upon adding a machine and given that $|\mathcal{M}| \geq |\mathcal{J}|$, it is possible to remove up to $|\mathcal{M}| + 1 - |\mathcal{J}|$ machines such that the stable (or preferred) extensions of the new argumentation framework are a subset of the stable (or preferred) extensions of $AF^{\{M_k\}}$. More precisely, let $R$ be the set of removed machines. We show that taking the union over the extensions of all possible combinations $R$ such that $|R| = q$ with $1 \leq q \leq |\mathcal{M}| + 1 - |\mathcal{J}|$ gives exactly the extensions of the $AF^{\{M_k\}}$. While this formulation forces the constraint $|\mathcal{M}| \geq |\mathcal{J}|$, we gain the advantage of allowing non-standard makespan attacks $(a_{i,j}, a_{k,l})$ for $i \neq k$ and $k \neq l$.

Let $|\mathcal{M}| = m$, $|\mathcal{J}| = n$ and $AF =< \mathcal{A}, \mathcal{R} >$ be given such that the standard makespan constraints are a subset of the attacks:

$$\mathcal{A} = \{a_{i,j} : 1 \leq i \leq m, 1 \leq j \leq n\}$$
$$\mathcal{R} \supseteq \{(a_{i,j}, a_{p,j}), (a_{p,j}, a_{i,j}) : i \neq p,\ 1 \leq j \leq n\}$$

**Proposition 5.3.** Let $|\mathcal{M}| \geq |\mathcal{J}|$. Set $p := |\mathcal{M}| + 1 - |\mathcal{J}|$. For $1 \leq q \leq p$:

$$\mathcal{E}_{\mathcal{ST}}(AF^{\{M_k\}}) = \bigcup_{\substack{R \subset \mathcal{M} \\ |R| = q}} \mathcal{E}_{\mathcal{ST}}(AF_R^{\{M_k\}})$$

*Proof.* Let $S \in \bigcup_{\substack{R \in 2^{\mathcal{A}} \\ |R| = q}} \mathcal{E}_{\mathcal{ST}}(AF_R^{\{M_k\}})$. Then $S \in \mathcal{E}_{\mathcal{ST}}(AF_R^{\{M_k\}})$ for some $R$ with $|R| = q$. By proof of 4.3, $S \in \mathcal{E}_{\mathcal{ST}}(AF^{\{M_k\}})$.

For the reverse direction, let $S \in \mathcal{E}_{\mathcal{ST}}(AF^{\{M_k\}})$. As $|M| + 1 > J$ there exists at least one $i$ such that $a_{i,j} \notin S$ for $1 \leq j \leq n$. We claim that $S \in \mathcal{E}_{\mathcal{ST}}(AF_R^{\{M_k\}})$ with $R$ chosen such that $a_{i,j} \in R$. Clearly $S$ is conflict-free in $AF_R^{\{M_k\}}$, as it is conflict-free in $AF^{\{M_k\}}$ and no attacks are added to obtain $AF_R^{\{M_k\}}$. It is also stable in $AF_R^{\{M_k\}}$ as $|S| = |\mathcal{J}|$ by lemma 2.5, so for any $a_{n,j} \notin S$, there exists $a_{p,j} \in S$ such that $(a_{p,j}, a_{n,j}) \in \mathcal{R}_R^{\{M_k\}}$. Hence $S \in \bigcup_{\substack{R \subset \mathcal{M} \\ |R| = q}} \mathcal{E}_{\mathcal{ST}}(AF_R^{\{M_k\}})$. □

**Proposition 5.4.** Let $|\mathcal{M}| \geq |\mathcal{J}|$. Set $p := |\mathcal{M}| + 1 - |\mathcal{J}|$. For $1 \leq q \leq p$:

$$\mathcal{E}_{\mathcal{PR}}(AF^{\{M_k\}}) = \bigcup_{\substack{R \subset \mathcal{M} \\ |R| = q}} \mathcal{E}_{\mathcal{PR}}(AF_R^{\{M_k\}})$$

*Proof.* Let $S \in \bigcup_{\substack{R \in 2^{\mathcal{A}} \\ |R| = q}} \mathcal{E}_{\mathcal{PR}}(AF_R^{\{M_k\}})$. Then $S \in \mathcal{E}_{\mathcal{PR}}(AF_R^{\{M_k\}})$ for some $R$ with $|R| = q$. By proof 4.4, $S \in \mathcal{E}_{\mathcal{PR}}(AF^{\{M_k\}})$.

Now consider $S \in \mathcal{E}_{\mathcal{PR}}(AF^{\{M_k\}})$. We claim that $S \in \mathcal{E}_{\mathcal{PR}}(AF_R^{\{M_k\}})$ with $R$ chosen such that $a_{i,j} \in R$. As $|M|+1 > J$ there exists at least one $i$ such that $a_{i,j} \notin S$ for $1 \leq j \leq n$. Clearly $S$ is conflict-free and admissible in $AF_R^{\{M_k\}}$, as it is conflict-free and admissible in $AF^{\{M_k\}}$ and no attacks are added to obtain $AF_R^{\{M_k\}}$. Suppose it is not maximal in $AF^{\{M_k\}}$. Then $\exists A$ with $A$ admissible such that $S \subset A$. Let $A = S \cup \{a_{q,j}\}$, $q \notin R$. As $|S| = |\mathcal{J}|$ by lemma 2.5, $\exists a_{p,j} \in S$ such that $(a_{p,j}, a_{q,j}) \in \mathcal{R}_R^{\{M_k\}}$, so $A$ is not conflict-free, a contradiction. Hence $S$ is maximal, so $S \in \mathcal{E}_{\mathcal{PR}}(AF_R^{\{M_k\}})$. $\square$

We can apply this to a more useful setting without machine addition, as exemplified in the following corollary:

**Corollary 5.5.** Let $|\mathcal{M}| \geq |\mathcal{J}|+1$. Set $p := |\mathcal{M}| - |\mathcal{J}|$. For $1 \leq q \leq p$ and semantics $\mathcal{S} \in \{\mathcal{ST}, \mathcal{PR}\}$:

$$\mathcal{E}_{\mathcal{S}}(AF) = \bigcup_{\substack{R \subset \mathcal{M} \\ |R|=q}} \mathcal{E}_{\mathcal{S}}(AF_R)$$

For example, given $|\mathcal{M}| = 4$, $|\mathcal{J}| = 2$, the following corollary allows us compute the extensions using either one $4 \times 2$ graph, or four $3 \times 2$ graphs, or six $2 \times 2$ graphs.

It may be interesting to look at the run times of while using this method; if we let $T(k,l)$ be the time taken to compute extensions of a $k \times l$ graph, note that there are exactly $\binom{|\mathcal{M}|}{q}$ choices of $R \subseteq \mathcal{M}$ with $|R| = q$, so an algorithm which computes extensions using this method has run time

$$O(T(|\mathcal{M}|-q, |\mathcal{J}|)|\mathcal{M}|^q)$$

In all the algorithms developed in section 4.3, we consider only a single instance of a single constraint. It is relatively easy to see that the algorithm we developed for concurrent execution can be extended to apply for multiple instances of different concurrent execution chains (and we believe it also can be done for restricted assignment), but for forbidden sets and precedence relations, this is more difficult as one has to ensure that shifting an argument does not conflict with other precedence relation chains, for example. Another interesting extension would be to consider overlapping precedence relation chains, i.e if $J_j$ can be in two separate precedence relations; we are unsure how the interactions between the two chains will affect the value of $\mathcal{MS}$ or if our algorithm still runs correctly.

It is also interesting to see the interactions of our algorithms with optimal schedules; it is not the case that every $E \in \mathcal{E}_{\mathcal{S}}(AF)$ gives an optimal schedule. We give an easy counter example; let $|\mathcal{M}| = |\mathcal{J}| = r = 2$ and $p_1 = p_2 = 1$. Then $E = \{a_{111}, a_{122}\}$ is a stable and preferred extension of $AF$. Note here that the makespan of $E$ is 3, but $E' = \{a_{111}, a_{221}\}$ is also a stable and preferred extension which gives a better makespan of 2. Hence, let us define optimal schedules as follows:

**Definition 5.6.** For semantics $\mathcal{S} \in \{\mathcal{ST}, \mathcal{PR}\}$, let

$$\text{Opt}_{\mathcal{S}}(AF) = \{E \in \mathcal{E}_{\mathcal{S}}(AF) : E \text{ creates an optimal schedule}\}$$

In other words, $\text{Opt}_{\mathcal{S}}(AF)$ contains the extensions of $AF$ with the minimum makespan.

It is easy to see that mapping $E \in \mathrm{Opt}_{\mathcal{S}}(AF)$ to $E' \in \mathrm{Opt}_{\mathcal{S}}(^{\{J_l\}}AF)$ only requires the following simple algorithm; find the machine with minimum completion time $t$ and assign $J_l$ to $M_i$, slot $t+1$. However, we cannot apply the same algorithm to frameworks under constraints. Perhaps an interesting extension would be to look for algorithms which give, say, a 2-approximation to the makespan problem while satisfying the additional constraint.

## 6 References

[1] Pietro Baroni, Massimiliano Giacomin, Beishui Liao, *On topology-related properties of abstract argumentation semantics. A correction and extension to Dynamics of argumentation systems: A division-based method*, Artificial Intelligence, 212:104–115, 2014.

[2] Kristijonas Čyras. *Argumentation and Makespan Scheduling*, Imperial College London. 2018.

[3] Phan Minh Dung. *On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-person Games*, Artificial Intelligence, 77:321–357, 1995.

[4] Michael Randolph Garey. *The Complexity of Flowshop and Jobshop Scheduling.* Mathematics of Operations Research. 1 (2): 117–129, 1976.

[5] Francesca Toni. *A tutorial on assumption-based argumentation*, Argument & Computation, 5:88-117