Guaranteeing Solution Quality for SAS Optimization Problems by being Greedy

Ulrike Stege
University of Victoria
ustege@uvic.ca

Preliminary results of this in: S. Balasubramanian, R.J. Desmarais, H.A. Müller, U. Stege, S. Venkatesh. *Characterizing problems for realizing policies in self-adaptive and self-managing systems*. SEAMS 2011: 70-79

Optimization problems

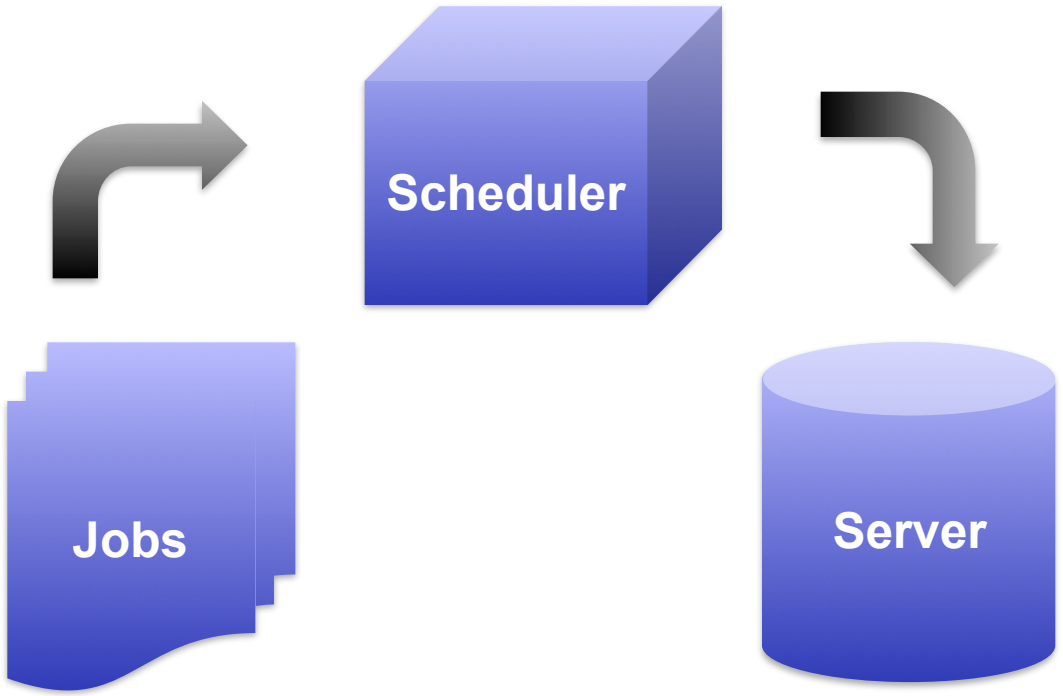
- **Given** Constraints, objective function
- **Goal** Whatever optimizes the objective function

Outline

- SAS optimization problem: SCHEDULING
- Algorithms & policies
- Definition and properties of **maximization** problems
- The **greedy** algorithm
- When is the greedy algorithm a good choice?
- Solution quality: Adding structure to maximization problems
- Conclusions & future works
- **Group work**

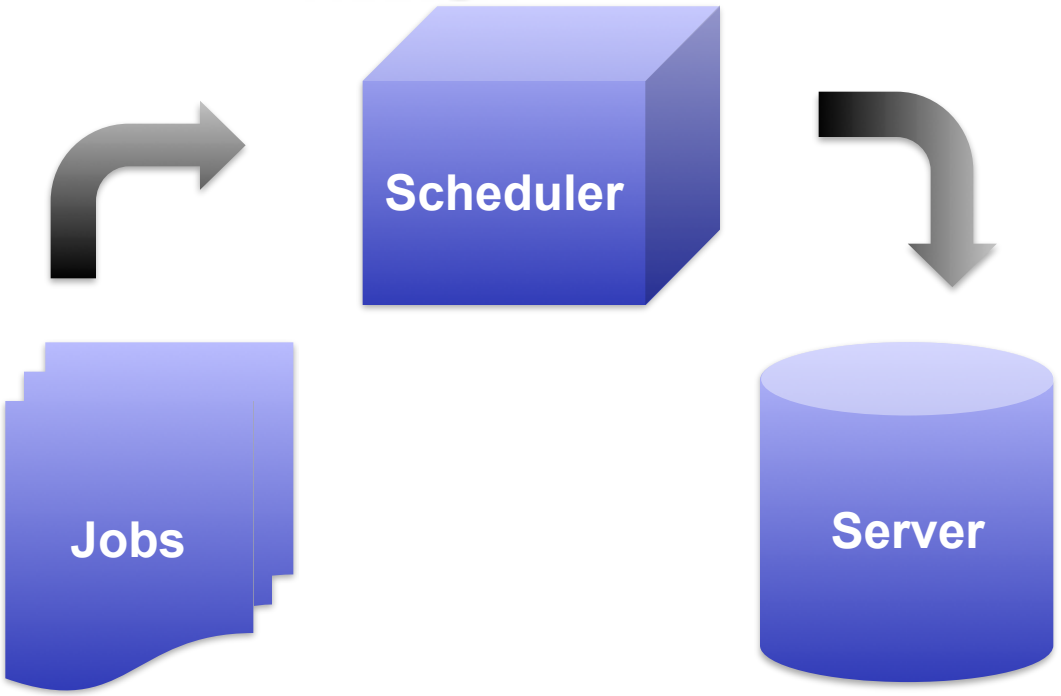
A SAS optimization problem

DATA CENTER SCHEDULING



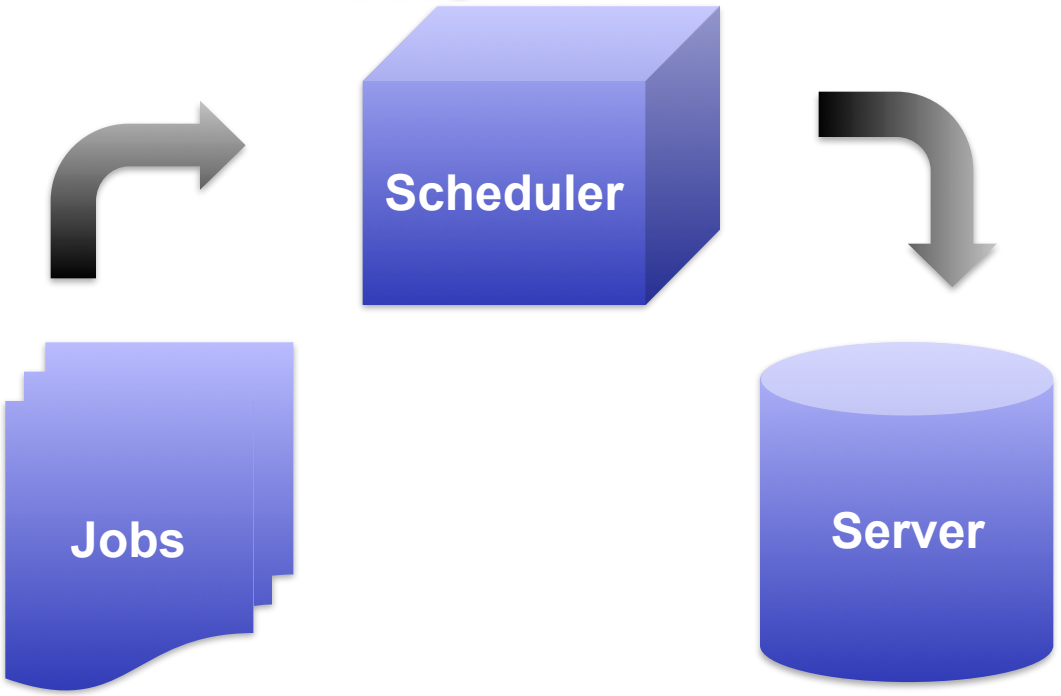
A SAS optimization problem

DATA CENTER SCHEDULING



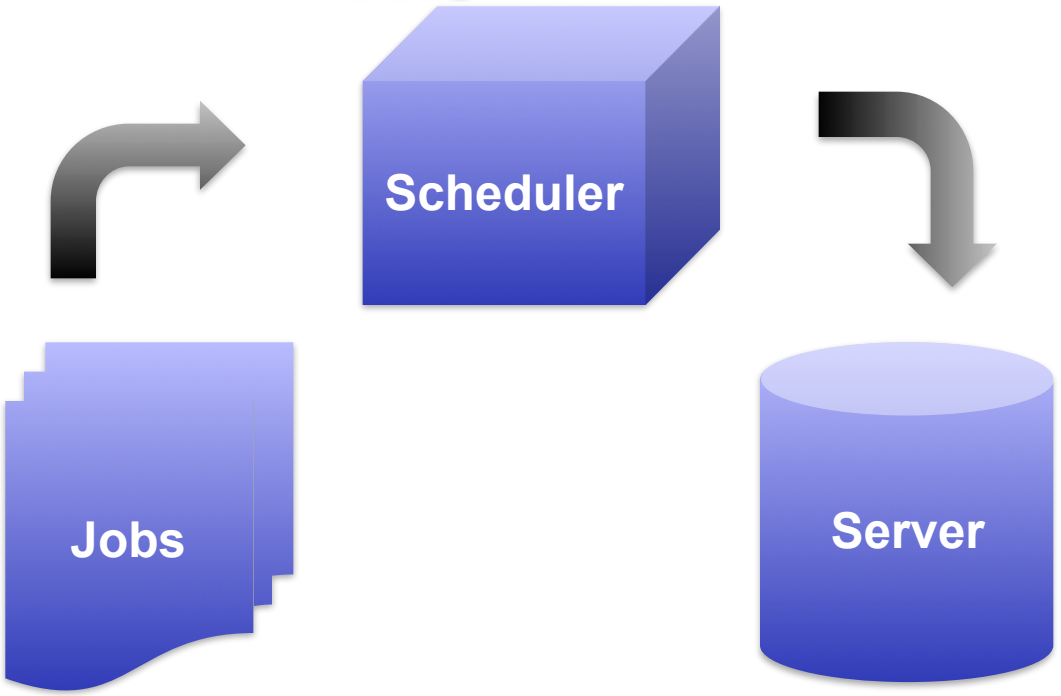
A SAS optimization problem

CLOUD SCHEDULING



A SAS optimization problem

SCHEDULING



SCHEDULING

Input Jobs $j_1, \ldots, j_n$; for each j_i the following constraints

- arrival time a_i
- deadline d_i
- processing time t_i
- revenue r_i

Output A linear schedule of jobs that maximizes their revenue, that is the sum of the revenues of all scheduled jobs

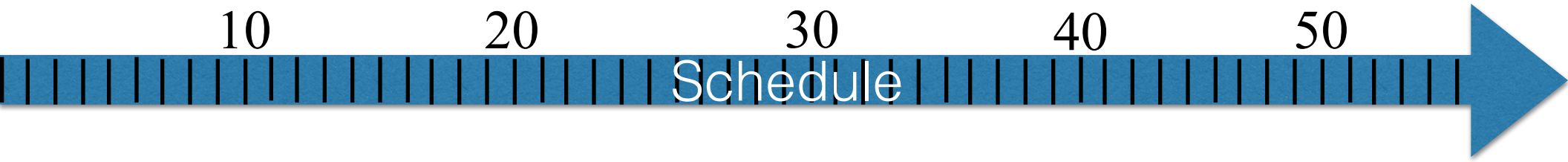
Wishlist of algorithmic properties for solving SCHEDULING

- **Fast**
- **Optimal** or **good** (the solution is feasible and satisfies a quality guarantee)

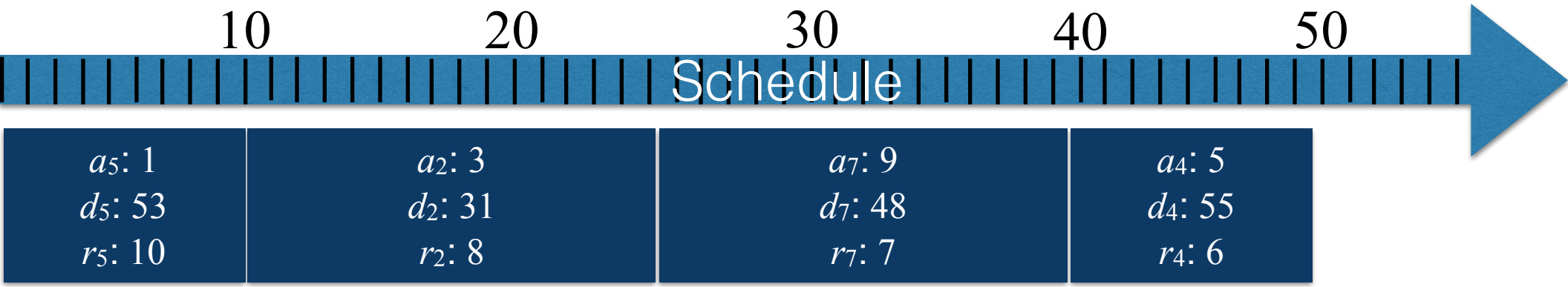
What would be really greedy?

- Sort the jobs in decreasing order according to their revenue
  - Take (and remove) first job from sorted list
  - add to schedule if feasible
  - repeat

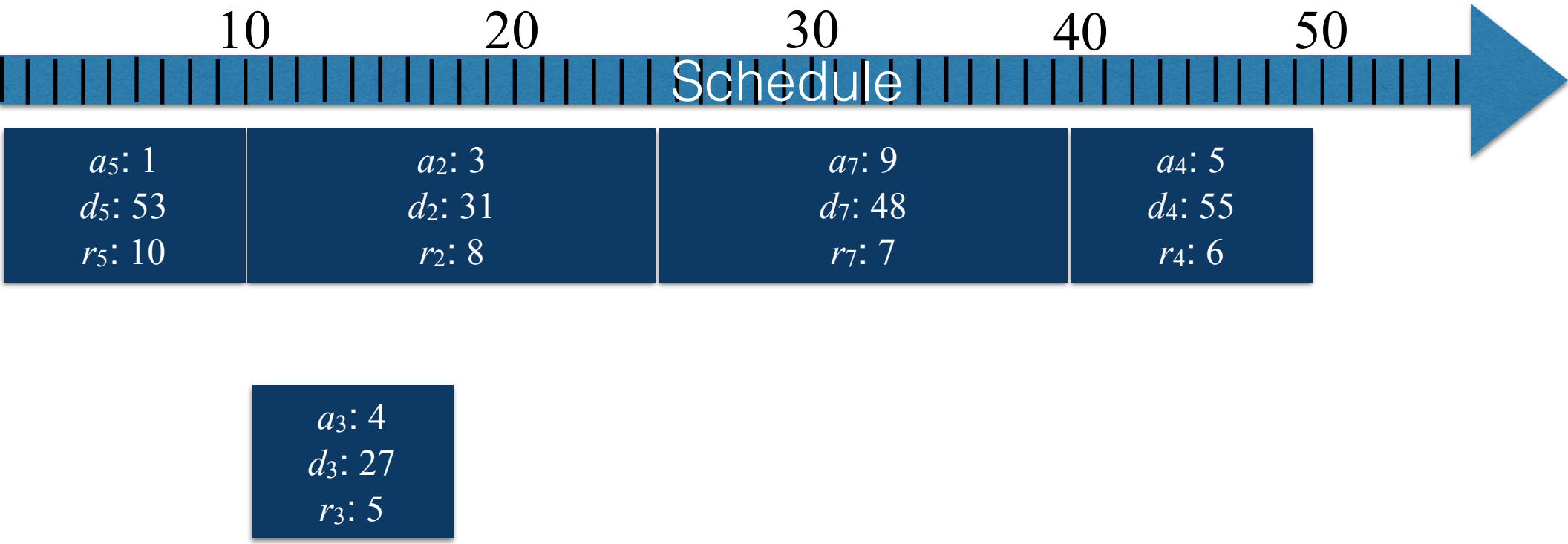
How good is greedy for SCHEDULING?



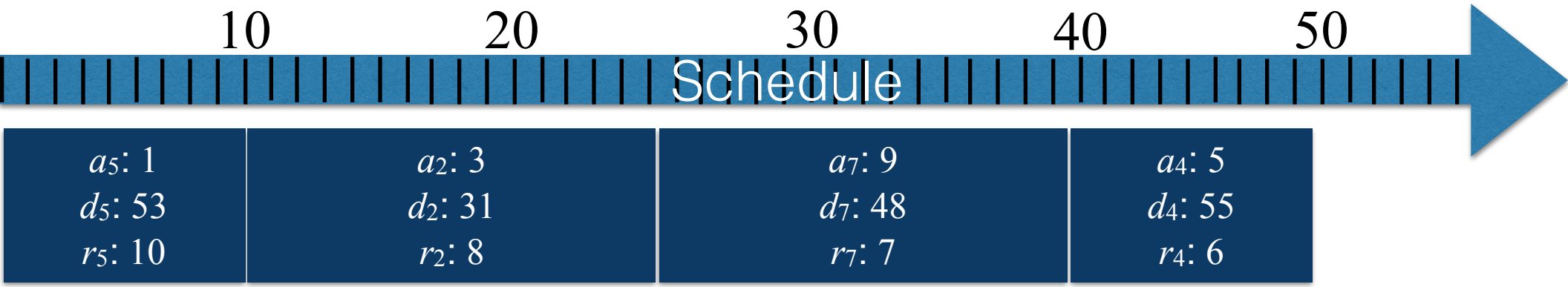
How good is greedy for SCHEDULING?



How good is greedy for SCHEDULING?



How good is greedy for SCHEDULING?

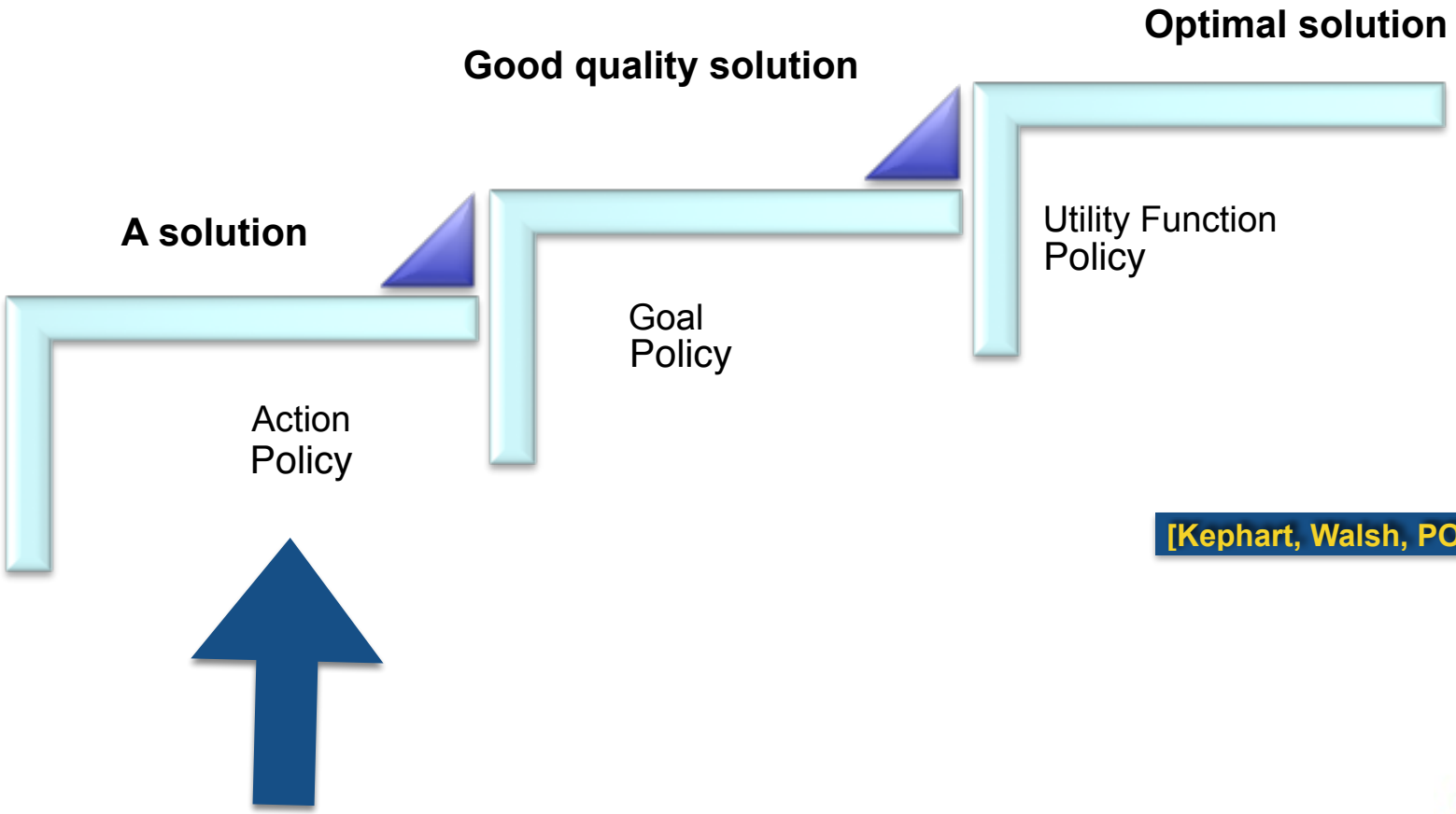


a_5: 1 d_5: 53 r_5: 10	a_2: 3 d_2: 31 r_2: 8	a_7: 9 d_7: 48 r_7: 7	a_4: 5 d_4: 55 r_4: 6

a_3: 4 d_3: 27 r_3: 5	a_1: 2 d_1: 25 r_1: 4

University of Victoria

Solution quality



[Kephart, Walsh, POLICY 2004]

What now?

- Possible approach: Come up with **better algorithm**
- Instead**: tweak** the **problem**
  - Add **problem structure** while keeping greedy algorithm

Components of an optimization problem

SCHEDULING

Input Jobs $j_1, \ldots, j_n$; for each j_i the following constraints

- arrival time a_i
- deadline d_i
- processing time t_i
- revenue r_i

Output A linear schedule of jobs that maximizes their revenue, that is the sum of the revenues of all scheduled jobs

constraints

objective function

Generic Maximization Problem

Given Set system $(U, \mathcal{F})$: U is a finite set,

$\mathcal{F} \subseteq 2^U$ satisfies a given **constraint set**;

objective function g: $2^U \rightarrow \mathrm{IR}^+$: monotone, non-decreasing

Output $S \in \mathcal{F}$ for which g has maximum value

SCHEDULING
input as set system

- D: latest deadline in job list
- Set system $(U, \mathcal{F})$:
  - $U = \{1, 2, \ldots, n\} \text{x} \{1, 2, \ldots, D\}$ with
    - $(i, j) \in U$: job i that starts at time j
  - $\mathcal{F}$: set of all feasible schedules where for each $(i, j) \in S \subseteq \mathcal{F}$, $a_i \leq j \leq d_i\text{-}t_i$

SCHEDULING
objective function

Let $S \in \mathcal{F}$ be a feasible schedule. Then

$$g(S) = \sum_{(i,j) \in S} r_i \ .$$

Before we can tweak the problem ….

Wanted: a way to figure out how good the greedy algorithm is for our maximization problem

Then: two mathematical frameworks

- Constraint-based framework
- Objective function based framework

Some definitions …

- **Objective functions**
  - linear
  - submodular
- **Constraint set**
  - matroid
  - k-extendible

Given set U, function $f: 2^U \rightarrow \mathrm{IR}^+$ ...

$$\sum_{s \in A} w(s)$$

Given set U, function $f: 2^U \rightarrow \mathbb{R}^+$ ...

- is **linear** if for any $A \subseteq U: f(A) = \sum_{s \in A} w(s)$ for some given weight function $w: U \rightarrow \mathbb{R}^+$

Given set U, function $f: 2^U \rightarrow \mathbb{R}^+$ ...

- is **linear** if for any $A \subseteq U: f(A) = \sum_{s \in A} w(s)$ for some given weight function $w: U \rightarrow \mathbb{R}^+$
- is **submodular** if $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$ for all $A, B \subseteq U$

Given set U, function $f: 2^U \rightarrow \mathrm{IR}^+$ …

- is **linear** if for any $A \subseteq U: f(A) = \sum_{s \in A} w(s)$ for some given weight function $w: U \rightarrow \mathrm{IR}^+$
- is **submodular** if $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$ for all $A, B \subseteq U$

Note: Not every submodular function is linear but every linear function is submodular.

A set system $(U, \mathcal{F})$ …

- is called a **matroid** if it satisfies the following conditions
  - $\mathcal{F}$ is **downward closed**: **if** $A \subseteq B$ and $B \in \mathcal{F}$ **then** $A \in \mathcal{F}$
  - $\mathcal{F}$ satisfies the **augmentation** property: for all $A, B \in \mathcal{F}$ with $|B| > |A|$ there exists $x \in B\text{-}A$ s.t. $A \cup \{x\} \in \mathcal{F}$

A set system $(U, \mathcal{F})$ …

- is called a ***k*-extendible** if it satisfies the following conditions
  - $\mathcal{F}$ is **downward closed**
  - $\mathcal{F}$ satisfies the **exchange** property: Let $A, B \in \mathcal{F}$ with $A \subseteq B$ and let $x \in U\text{-}B$ s.t. $A \cup \{x\} \in \mathcal{F}$. Then there exists $Y \subseteq B\text{-}A$, $|Y| \leq k$, s.t. $(B\text{-}Y) \cup \{x\} \in \mathcal{F}$.

Properties of SCHEDULING Summary

Maximization problems

Properties of SCHEDULING Summary

- Its objective function is linear and therefore submodular

Properties of SCHEDULING Summary

- Its objective function is linear and therefore submodular
- Its set system is downward closed

Properties of SCHEDULING Summary

- Its objective function is linear and therefore submodular
- Its set system is downward closed
- but does not satisfy the augmentation property (and therefore is not a matroid)

Properties of SCHEDULING Summary

- Its objective function is linear and therefore submodular
- Its set system is downward closed
- but does not satisfy the augmentation property (and therefore is not a matroid)
- Exchange property? Yes: it's a k-extendible system

SCHEDULING
downward closed

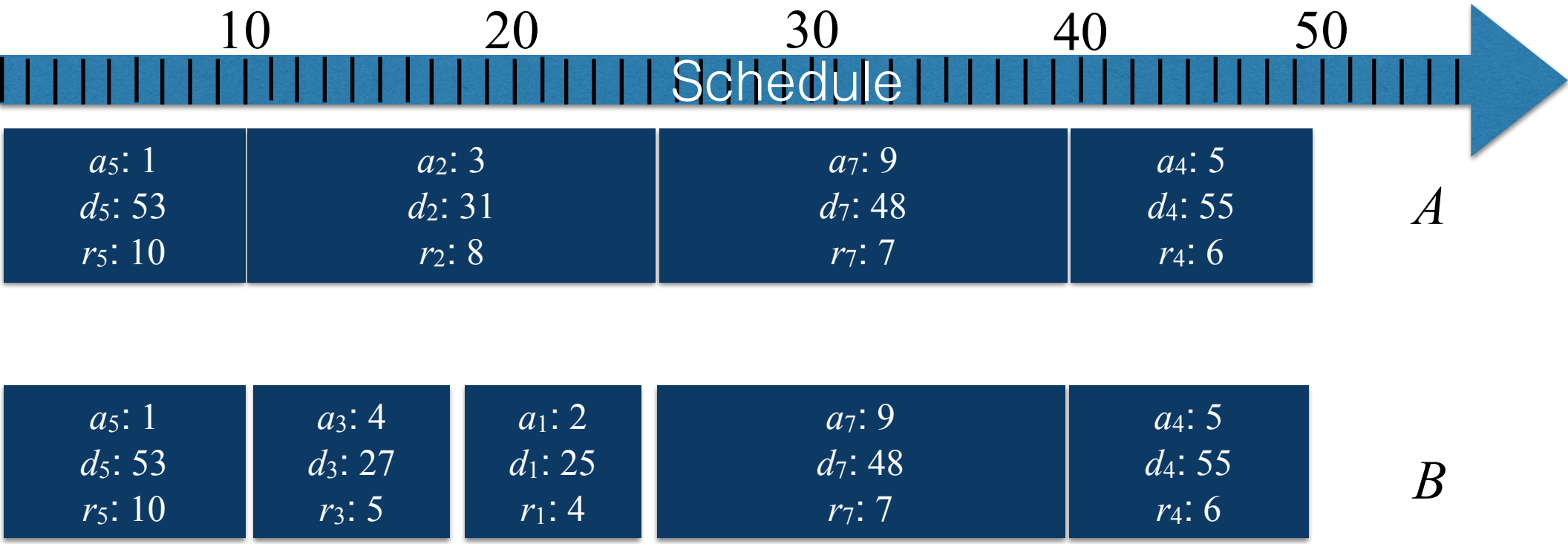
- $\mathcal{F}$: set of all feasible schedules
- To show: if $A \subseteq B$ and $B \in \mathcal{F}$ then $A \in \mathcal{F}$
- That is we need to show: a subset of a feasible schedule is also feasible itself

SCHEDULING augmentation property

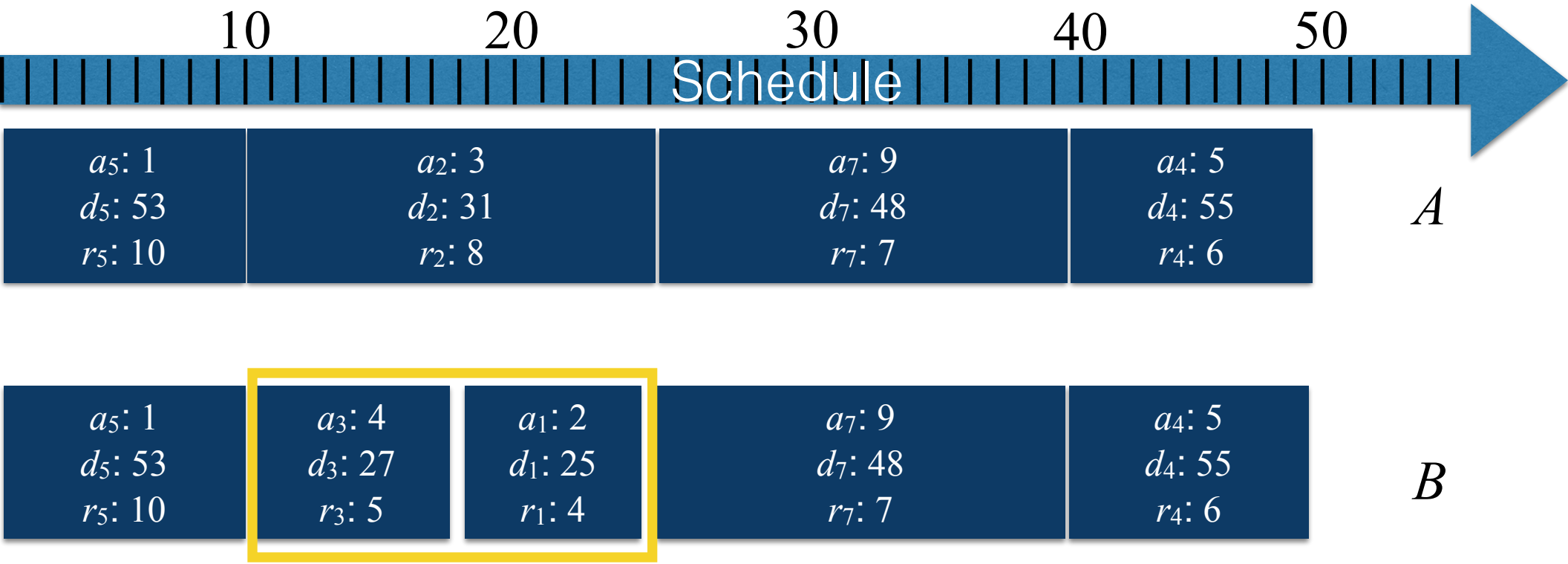
- to show: for all $A, B \in \mathcal{F}$ with $|B| > |A|$ there exists $x \in B\text{-}A$ s.t. $A \cup \{x\} \in \mathcal{F}$

- that is: given any two feasible schedules where one (B) contains more jobs than the other (A), there must be a job that is in B and not in A that can be added to A while maintaining feasibility

But:



But:



But:

a_5: 1 d_5: 53 r_5: 10	a_2: 3 d_2: 31 r_2: 8	a_7: 9 d_7: 48 r_7: 7	a_4: 5 d_4: 55 r_4: 6	*A*

a_5: 1 d_5: 53 r_5: 10	a_3: 4 d_3: 27 r_3: 5	a_1: 2 d_1: 25 r_1: 4	a_7: 9 d_7: 48 r_7: 7	a_4: 5 d_4: 55 r_4: 6	*B*

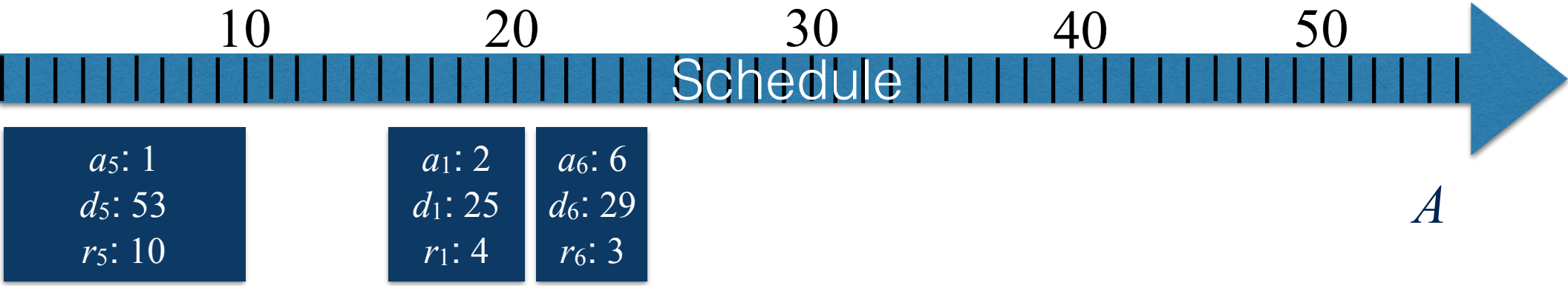
SCHEDULING: exchange property

- to show: Let $A, B \in \mathcal{F}$ with $A \subseteq B$ and let $x \in U\text{-}B$ s.t. $A \cup \{x\} \in \mathcal{F}$. Then there exists $Y \subseteq B\text{-}A$, $|Y| \leq k$, s.t. $(B\text{-}Y) \cup \{x\} \in \mathcal{F}$

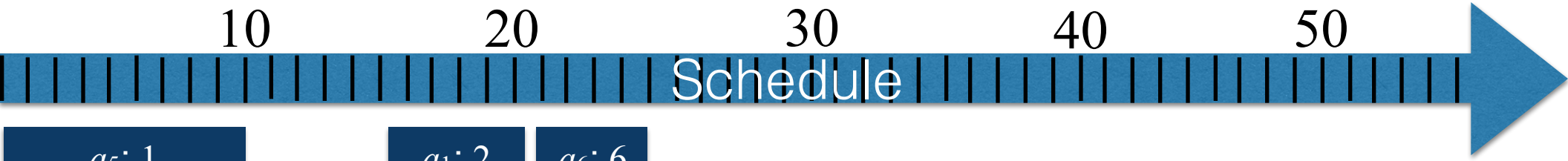
- in other words: Let A, B be feasible schedules with $A \subseteq B$. Let x be a job that is not in B that can augment A to a feasible schedule.

  - Then there must exist up to k jobs that are in B and not in A such that, once removed from B, B can be expanded to a feasible schedule by job x

Illustration



Illustration



a_5: 1
d_5: 53
r_5: 10

a_1: 2
d_1: 25
r_1: 4

a_6: 6
d_6: 29
r_6: 3

A

a_5: 1
d_5: 53
r_5: 10

a_3: 4
d_3: 27
r_3: 5

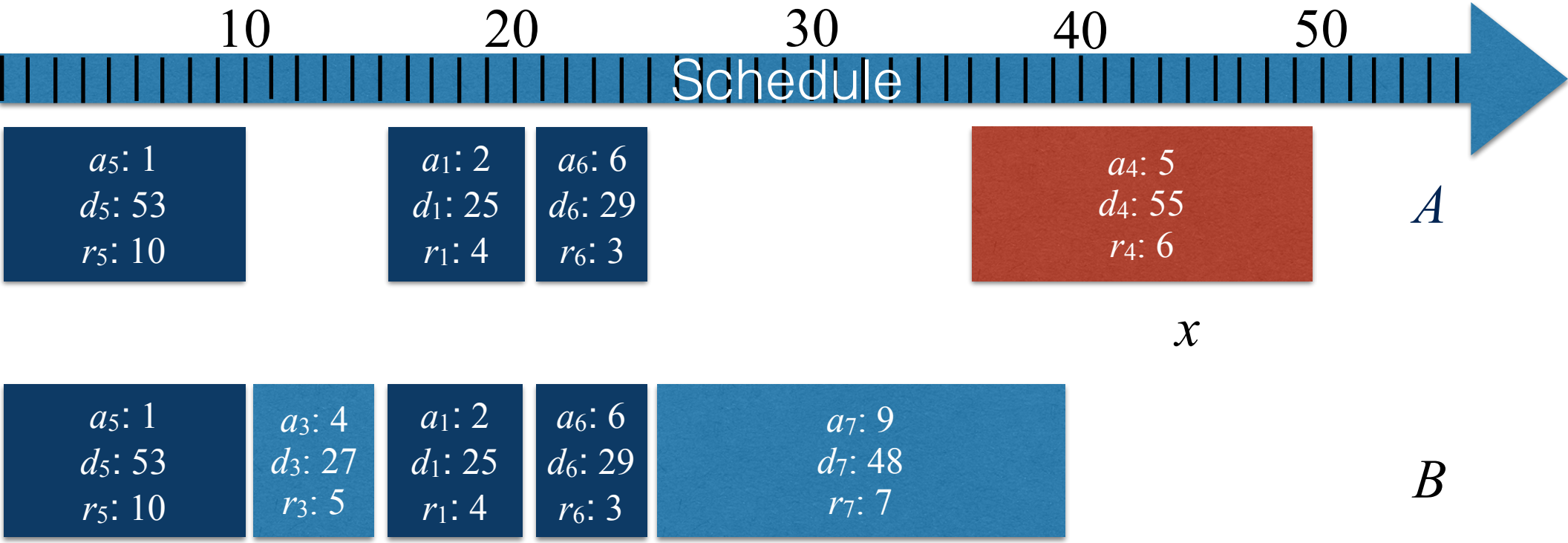
a_1: 2
d_1: 25
r_1: 4

a_6: 6
d_6: 29
r_6: 3

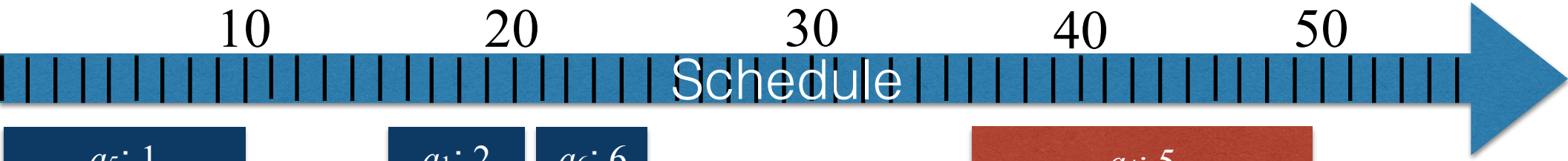
a_7: 9
d_7: 48
r_7: 7

B

Illustration



Illustration



a_5: 1
d_5: 53
r_5: 10

a_1: 2
d_1: 25
r_1: 4

a_6: 6
d_6: 29
r_6: 3

a_4: 5
d_4: 55
r_4: 6

A

x

a_5: 1
d_5: 53
r_5: 10

a_3: 4
d_3: 27
r_3: 5

a_1: 2
d_1: 25
r_1: 4

a_6: 6
d_6: 29
r_6: 3

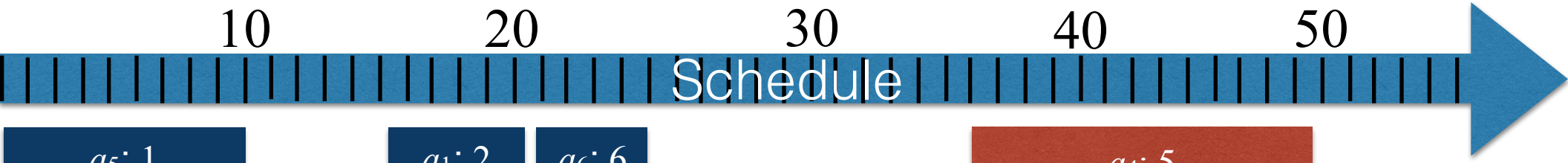
a_7: 9
d_7: 48
r_7: 7

B

a_7: 9
d_7: 48
r_7: 7

Y

Illustration



a_5: 1
d_5: 53
r_5: 10

a_1: 2
d_1: 25
r_1: 4

a_6: 6
d_6: 29
r_6: 3

a_4: 5
d_4: 55
r_4: 6

A

x

a_5: 1
d_5: 53
r_5: 10

a_3: 4
d_3: 27
r_3: 5

a_1: 2
d_1: 25
r_1: 4

a_6: 6
d_6: 29
r_6: 3

a_4: 5
d_4: 55
r_4: 6

B

a_7: 9
d_7: 48
r_7: 7

Y

Properties of SCHEDULING Summary

Maximization problems

Properties of SCHEDULING Summary

- **Objective function** is **linear** and submodular

Properties of SCHEDULING Summary

- **Objective function** is **linear** and submodular
- **Set system** is **downward closed**, does not satisfy augmentation property (and therefore is not a matroid), but satisfies **exchange property** and therefore is a ***k*-extendible system**

  [Here: k is computed by the longest job processing time decided by the shortest (which can be considered the time unit)]

What's so amazing about these properties?

- quality guarantees for greedy algorithm

$S \leftarrow \varnothing$; $A \leftarrow \varnothing$

repeat

$A \leftarrow \{e \in U \mid S \cup \{e\} \in \mathcal{F}\}$

if $A \neq \varnothing$ **then**

$$u \leftarrow \operatorname*{argmax}_{x \in A} g(S \cup \{x\})$$

$S \leftarrow S \cup \{u\}$

until $A = \varnothing$

return S

generic greedy algorithm

Quality guarantees

Theorem 1. If set system $(U, \mathcal{F})$ is a matroid and objective function g is linear, then the greedy algorithm returns the optimal solution.

[Edmonds 1971]

Theorem 2. If set system $(U, \mathcal{F})$ is a matroid and objective function g is submodular then the greedy algorithm is an $1/2$-approximation.

[Fisher *et al*, 1978]

Theorem 3. If set system $(U, \mathcal{F})$ is a k-extendible system and objective function g is linear then the greedy algorithm is an $1/k$-approximation.

[Mestre 2006]

Theorem 4. If set system $(U, \mathcal{F})$ is a k-extendible system and objective function g is submodular then the greedy algorithm is an $1/(k+1)$-approximation.

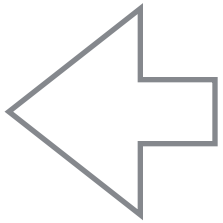
[Fisher *et al*, 1978]

When is greedy a good choice?

Definition: Approximation Algorithm

- P: maximization problem to be solved
- A: an algorithm for P
- x: an arbitrary instance for P
- $A(x)$: output of A for input x
- $\mathrm{OPT}(x)$: optimal solution for input x
- A is a ρ-**approximation algorithm** for P if

$$\frac{g(A(x))}{g(OPT(x))} \leq \rho$$



Overview of results for greedy technique

	linear function	submodular function	unrestricted
matroid	optimal	$1/2$-approximation	no guarantees
k-extendible system	$1/k$-approximation	$1/(k+1)$-approximation	no guarantees
unrestricted	no guarantees	no guarantees	no guarantees

When is greedy a good choice?

Constraint-based framework

- Objective function: **linear**
- Add structure to constraint set to satisfy **matroid** or ***k*-extendibility** property

When is greedy a good choice?

Objective function based framework

- Constraint set: **matroid**
- Add structure to objective function to make it **linear** or **submodular**

When is greedy a good choice?

Back to SCHEDULING

- Properties
  - Linear function
  - k-extendible system
- Greedy is a $1/k$-approximation
- Can we tweak the constraints to make it a matroid?
  - yes, if processing times of all jobs are equal then augmentation property is satisfied

When is greedy a good choice?

Recipe

1. Formulate maximization problem as set system and describe objective function, according to the GENERIC MAXIMIZATION PROBLEM
2. Investigate the following:
    1. Is the set system a matroid?
    2. If not: Is the set system a k-extendible system?
    3. Is the objective function linear?
    4. If not: is it submodular?
3. If no optimal solution: Can you tweak the problem? Give more structure to constraints or objective function?

When is greedy a good choice?

Conclusions: two frameworks for adding structure to maximization problems for using greedy technique

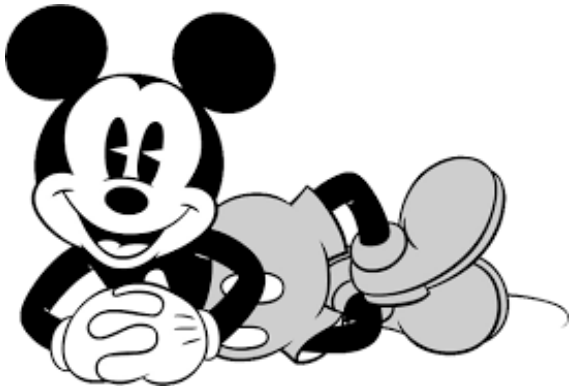
	linear function	**submodular function**	**unrestricted**
matroid	optimal	$1/2$-approximation	no guarantees
k-extendible system	$1/k$-approximation	$1/(k+1)$-approximation	no guarantees
unrestricted	no guarantees	no guarantees	no guarantees

Future Works

- Study other tractable algorithm techniques
  - P-time techniques
  - other exact algorithmic techniques
    - When can they be efficiently deployed?

Acknowledgements

Sowmya Balasubramanian, Ron Desmarais, Steven Lonergan, Hausi Müller, Venkatesh Srinivasan



Thank you!

University of Victoria

Group Work

- Pick an optimization problem that you are working on (Kostas' Meeting Scheduler or one of John's problems) and describe it to the group
- Formalize it: Input (including constraint set and objective function) and Goal/Output must be well defined
- Consider your constraints: are they all needed? Could there be more? What properties do your inputs (for your application) have?