Greedy Scheduling

https://cs.pomona.edu/classes/cs140/

Outline

Topics and Learning Objectives

- Introduce greedy algorithms
- Discuss the greedy scheduling algorithm
- Discuss exchange argument proofs

Exercise

- Greedy scheduling

Extra Resources

- Introduction to Algorithms, 3rd, chapter 16
- Algorithms Illuminated Part 3: Chapter 13

- Lots of examples: https://www.geeksforgeeks.org/greedy-algorithms/

Greedy Algorithms

- Iteratively make <u>myopic</u> (short-sighted) decisions and hope it works
- Never go back and recheck/reevaluate that you were correct

Contrasting with Divide and Conquer

- It is generally easier to create greedy algorithms (good and bad to this)
- It is typically easier to analyze greedy algorithms (e.g., no master theorem)
- It is often harder to prove/understand the correctness of greedy algorithms
- It is common for greedy algorithms to be incorrect (/ sub-optimal)

Greedy Algorithms

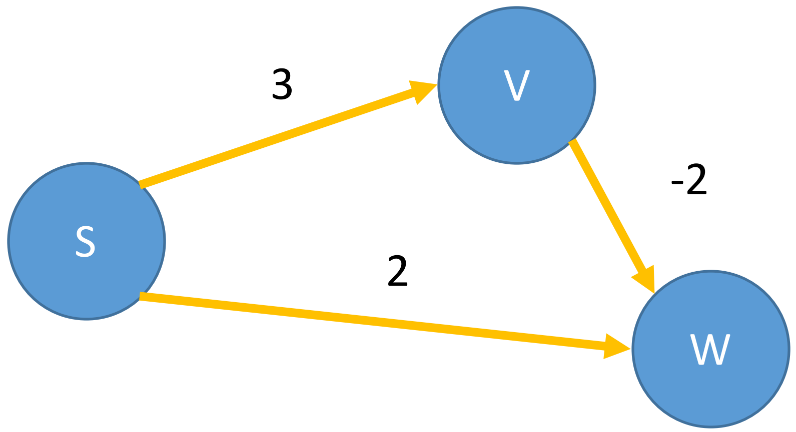
Proofs of correctness

- It can sometimes feel like more of an art than a science

1. Proof by induction on the greedy decision
2. Proof by induction on an exchange argument
    1. Either by contraction
    2. Or by exchanging with the optimal solution
3. Whatever works…

Example of a greedy algorithm

- We've seen one greedy algorithm before. What was it?



- What path length does Dijkstra's output for S → W?
- What is the correct shortest path length for S → W?

Scheduling (ignoring concurrency)

You have a shared resource

For example, a processor (or professor)

You have many jobs that need to use the resource (students at office hours)

Each job j has:

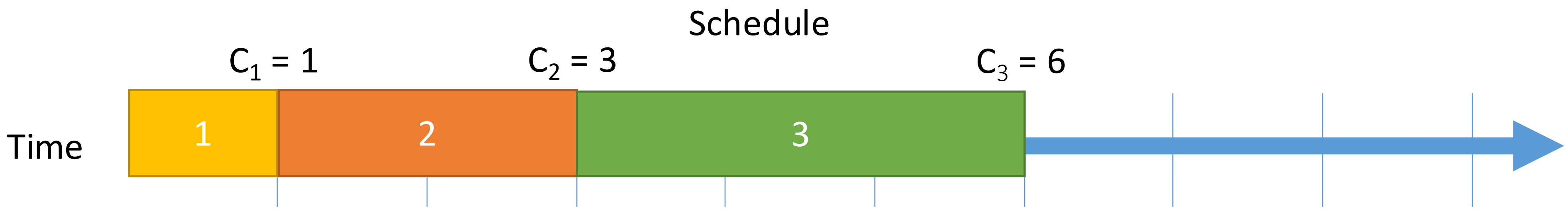
- A Priority P_j that stands for the job's importance
- A Duration D_j that stands for the length of time to run the job

In what sequence should we complete the jobs?

Scheduling (ignoring concurrency)

In what sequence should we complete the jobs?

- What is our criteria? What do we want to optimize?
- Let's start by looking at a derived property job j's **completion time** C_j
- Given three jobs: $D_1 = 1$, $D_2 = 2$, $D_3 = 3$
- What is the completion time for each if they are scheduled in order?



What is the completion time of Job 5?

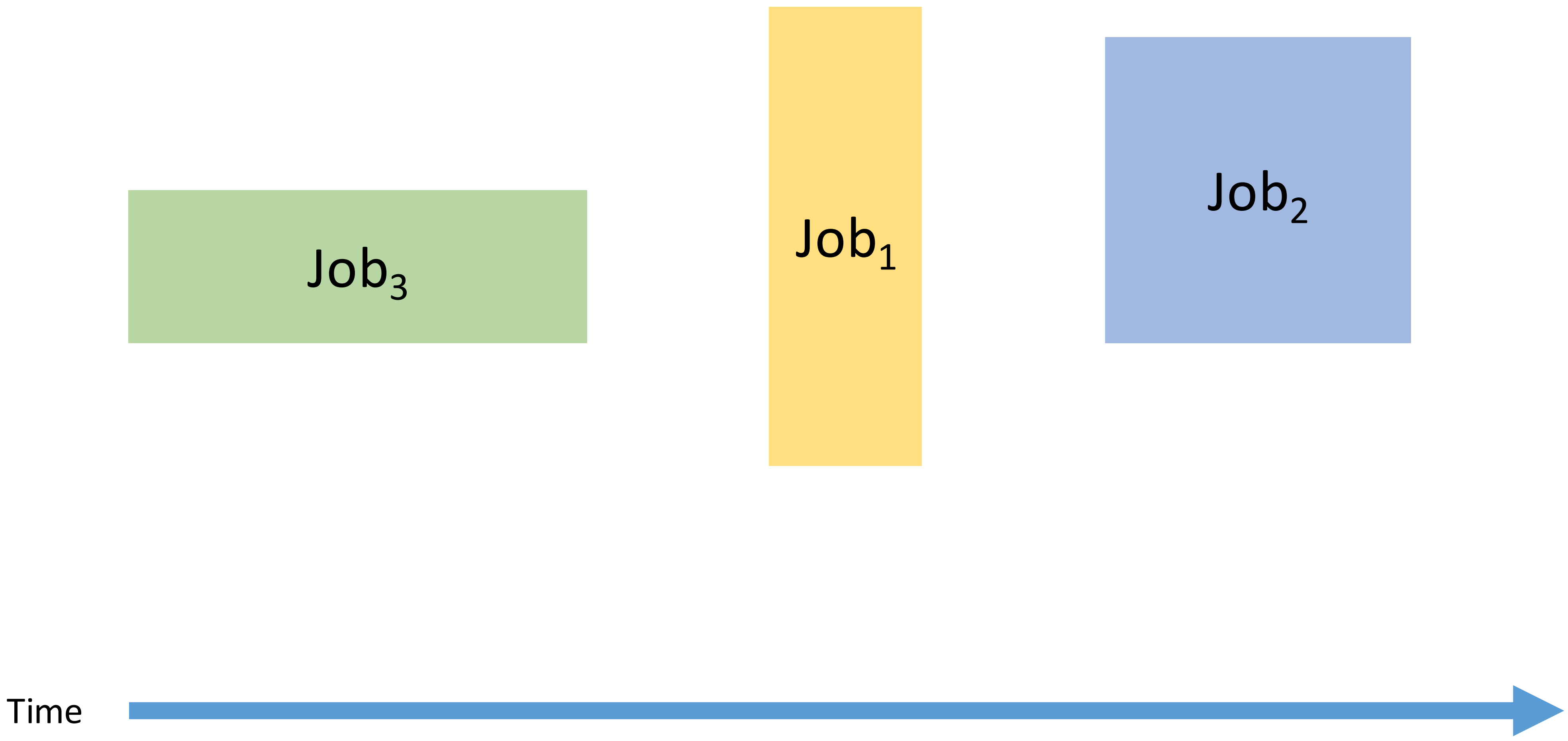
Scheduling

Optimization objective: minimize the weighted sum of completion times

$$S_{\text{cost}} = \min[\sum_{j=1}^{n} P_j C_j]$$

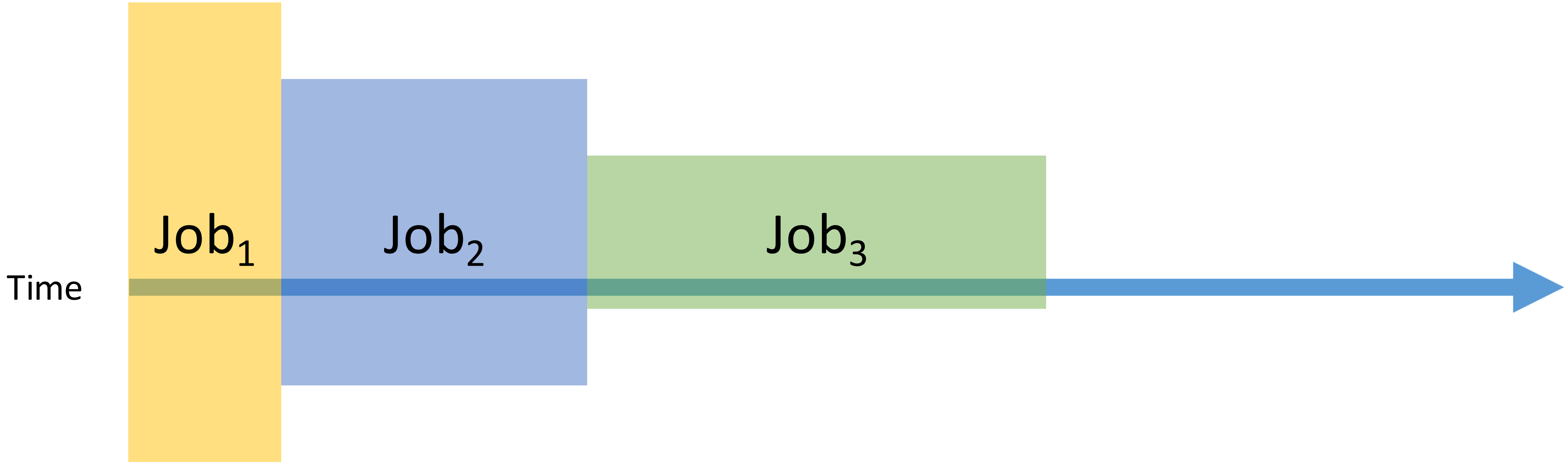
What is the weighted sum of completion times if we schedule the following jobs in order?

Job	J_1	J_2	J_3
Duration	$D_1 = 1$	$D_2 = 2$	$D_3 = 3$
Priority	$P_1 = 3$	$P_2 = 2$	$P_3 = 1$



Job3
Job1
Job2
Time

Exercise Question 1, 2, and 3



Scheduling

Calculate the weighted sum of completion times for the following jobs if they are scheduled in the order: 1, 2, 3.

Job	J_1	J_2	J_3
Duration	$D_1 = 1$	$D_2 = 2$	$D_3 = 3$
Priority	$P_1 = 3$	$P_2 = 2$	$P_3 = 1$
Completion			
Weight			

Weighted sum of completion times: ?

Greedy Scheduling

Our process for creating a greedy scheduling algorithm

1. Look at some special cases for our problem
2. Describe some possible greedy criteria
3. Compare our greedy criteria
4. Select the "best" one
5. Prove correctness if possible

Greedy Scheduling

Goal: devise a greedy algorithm to minimize the weighted sum of completion times

Why are we approaching this problem with a greedy algorithm?

- It's a pretty easy way to start.
- Compare the approach we go through in these slides with a Divide and Conquer approach

1. What are some special cases to consider?

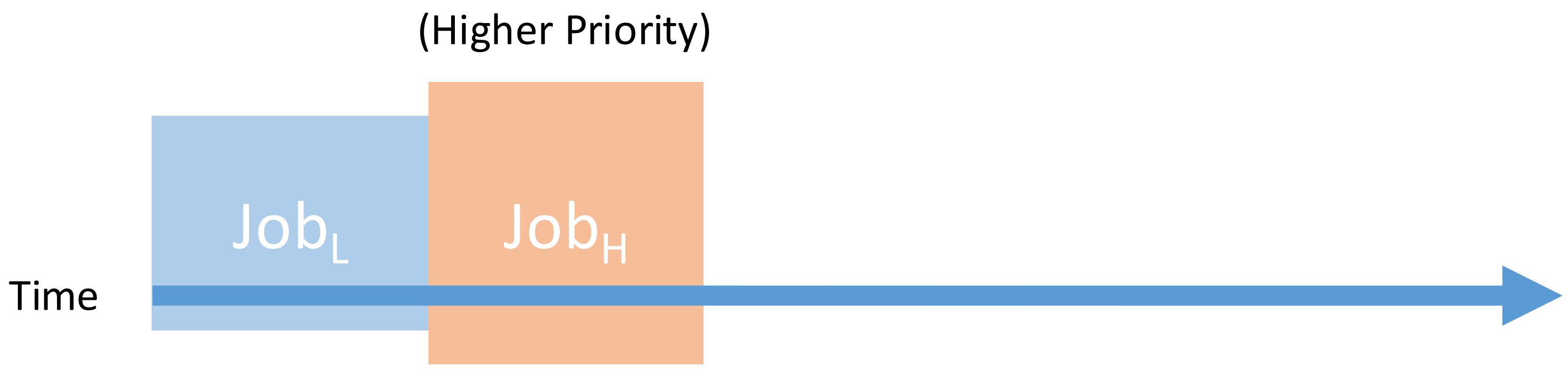
Consider <u>two</u> jobs with equal durations (D)

- These jobs have different priorities (P_H and P_L)
- **Do we schedule the lower or higher priority job first?**

1. What are some special cases to consider?

Consider <u>two</u> jobs with equal durations (D)

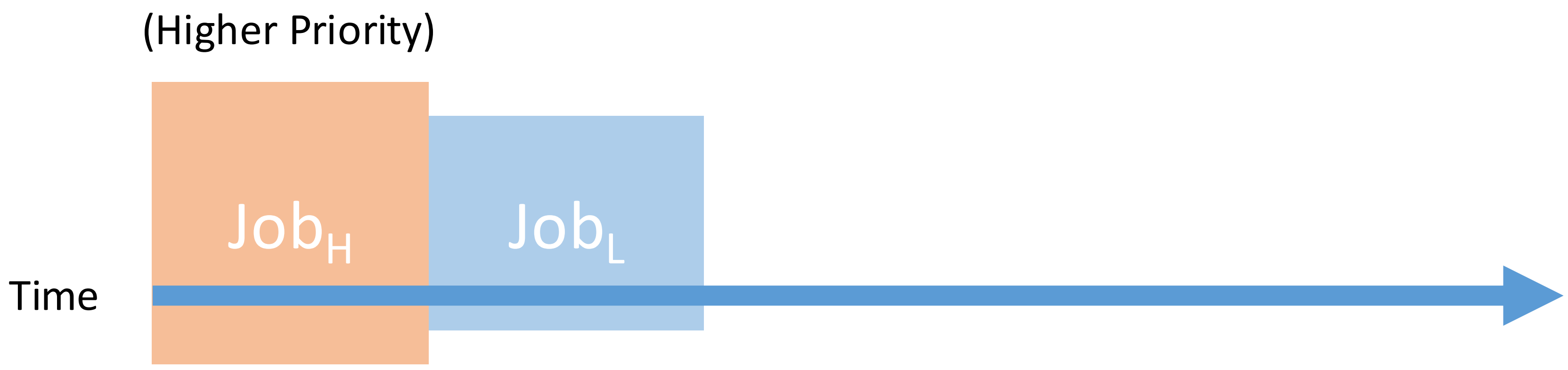
- These jobs have different priorities (P_H and P_L)
- **Do we schedule the lower or higher priority job first?**



1. What are some special cases to consider?

Consider <u>two</u> jobs with equal durations (D)

- These jobs have different priorities (P_H and P_L)
- **Do we schedule the lower or higher priority job first?**



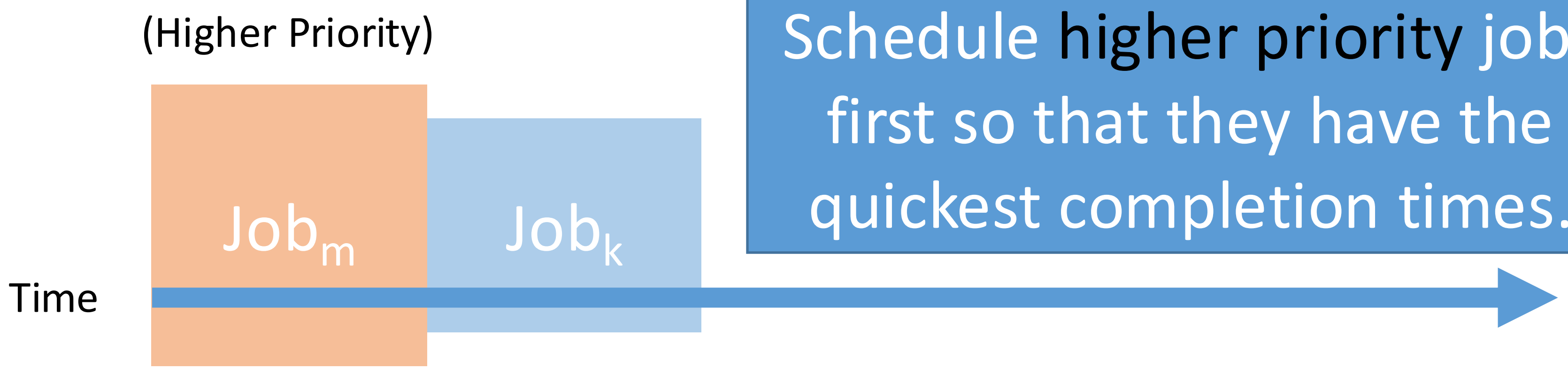
Schedule with Lower Priority First

Schedule with Higher Priority First

1. What are some special cases to consider?

Consider two jobs with equal durations (D)

- These jobs have different priorities (P_H and P_L)
- **Do we schedule the lower or higher priority job first?**

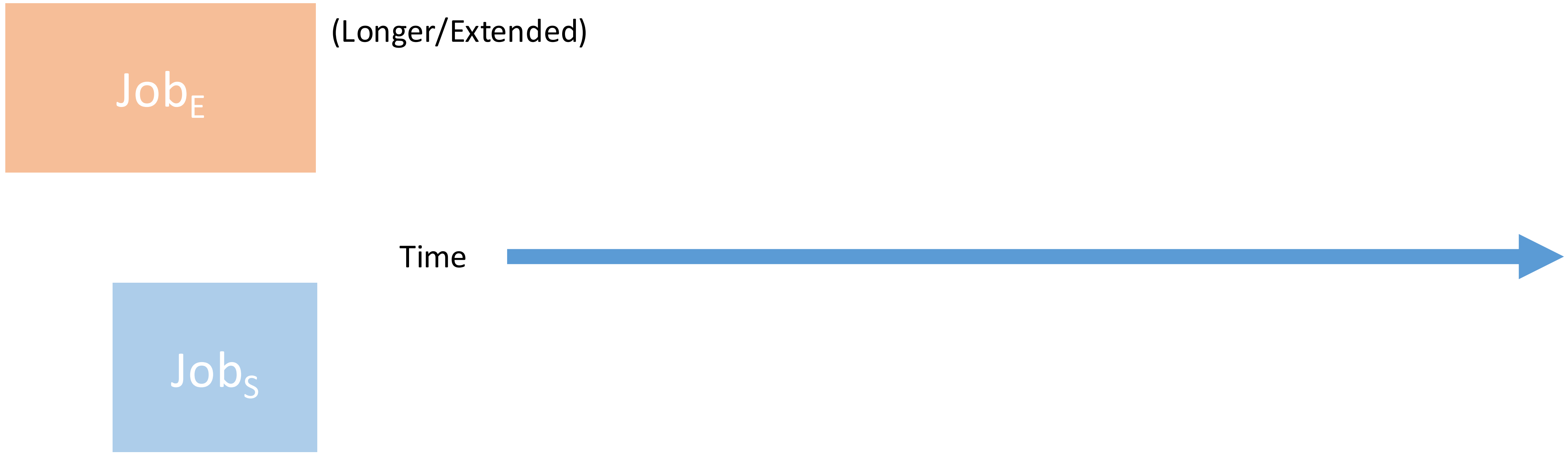


Schedule higher priority jobs first so that they have the quickest completion times.

1. What are some special cases to consider?

Consider <u>two</u> jobs with equal priorities (P)

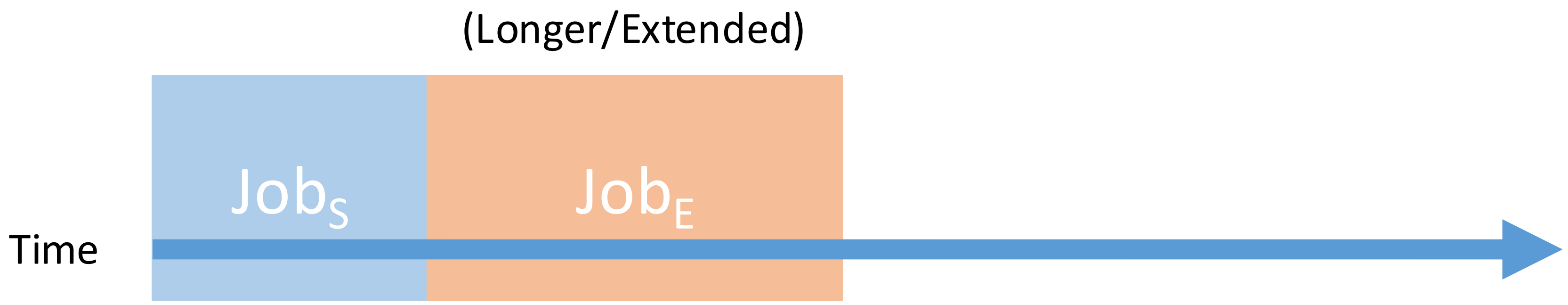
- These jobs have different durations (D_E and D_S)
- **Do we schedule the shorter or longer (Extended) job first?**



1. What are some special cases to consider?

Consider <u>two</u> jobs with equal priorities (P)

- These jobs have different durations (D_E and D_S)
- **Do we schedule the shorter or longer (Extended) job first?**



1. What are some special cases to consider?

Consider two jobs with equal priorities (P)

- These jobs have different durations (D_E and D_S)
- **Do we schedule the shorter or longer (Extended) job first?**

Schedule with Shorter Job First

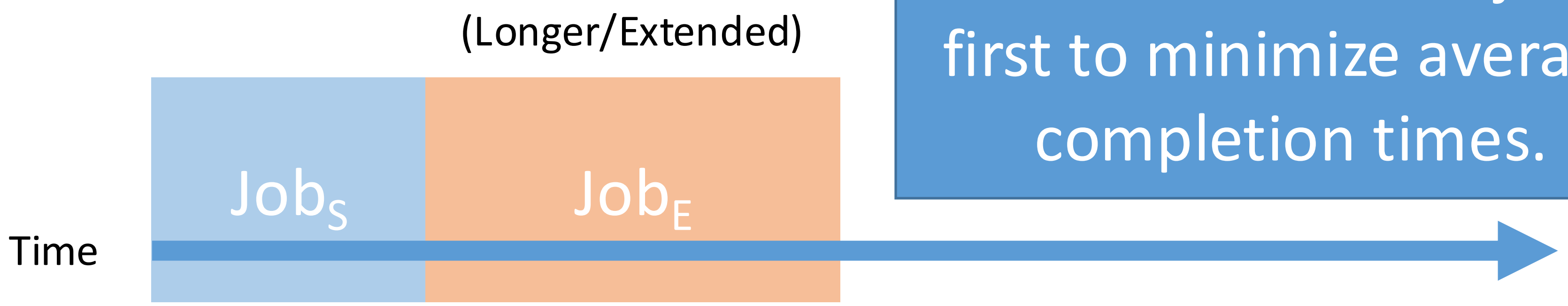
Schedule with Longer Job First

1. What are some special cases to consider?

Consider two jobs with equal priorities (P)

- These jobs have different durations (D_E and D_S)
- **Do we schedule the shorter or longer (Extended) job first?**

Schedule shortest jobs first to minimize average completion times.



2. Describe some possible greedy criteria

1. Schedule highest priority first
2. Schedule shortest duration first

What do we do when in the more general case:

$P_i > P_j \; and \; D_i > D_j$ (job i has higher priority and longer duration)

What are some simple scoring functions that *aggregate* our criteria?

We want a function for which jobs with a bigger score are scheduled first:

- Score increases for higher priorities
- Score increases for shorter times

1. Greedy Criterion 1: $P_i - D_i$ (take the difference)
2. Greedy Criterion 2: P_i / D_i (take the ratio)

3. Compare our greedy criteria

- Jobs will be ordered from biggest to smallest value

Job with same duration	Difference Metric ($P_i - D_i$)	Ratio Metric (P_i/D_i)
Job 1: P=2, D=1		
Job 2: P=5, D=1		

Which job should be scheduled first?

3. Compare our greedy criteria

- Jobs will be ordered from biggest to smallest value

	Job with same duration	Difference Metric $(P_i - D_i)$	Ratio Metric (P_i / D_i)
	Job 1: P=2, D=1	1	2
Highest priority	Job 2: P=5, D=1	4	5
	Total weighted sum		

3. Compare our greedy criteria

- Jobs will be ordered from biggest to smallest value

	Job with same duration	Difference Metric $(P_i - D_i)$	Ratio Metric (P_i/D_i)	
	Job 1: P=2, D=1	1	2	
Highest priority	Job 2: P=5, D=1	4	5	
	Total weighted sum	5*1 + 2*2 = 9	5*1 + 2*2 = 9	Same Result

Job with same priority	Difference Metric $(P_i - D_i)$	Ratio Metric (P_i/D_i)
Job 1: P=1, D=3		
Job 2: P=1, D=4		
Total weighted sum		

Which job should be scheduled first?

3. Compare our greedy criteria

- Jobs will be ordered from biggest to smallest value

Job with same duration	Difference Metric ($P_i - D_i$)	Ratio Metric (P_i / D_i)
Job 1: P=2, D=1	1	2
Job 2: P=5, D=1	4	5
Total weighted sum	5*1 + 2*2 = 9	5*1 + 2*2 = 9

Highest priority

Same Result

Job with same priority	Difference Metric ($P_i - D_i$)	Ratio Metric (P_i / D_i)
Job 1: P=1, D=3	-2	1/3
Job 2: P=1, D=4	-3	1/4
Total weighted sum		

Shortest time

Which job should be scheduled first?

3. Compare our greedy criteria

- Jobs will be ordered from biggest to smallest value

	Job with same duration	Difference Metric ($P_i - D_i$)	Ratio Metric (P_i / D_i)	
	Job 1: P=2, D=1	1	2	
Highest priority	Job 2: P=5, D=1	4	5	
	Total weighted sum	5*1 + 2*2 = 9	5*1 + 2*2 = 9	Same Result

	Job with same priority	Difference Metric ($P_i - D_i$)	Ratio Metric (P_i / D_i)	
Shortest time	Job 1: P=1, D=3	-2	1/3	
	Job 2: P=1, D=4	-3	1/4	
	Total weighted sum	1*3 + 1*7 = 10	1*3 + 1*7 = 10	Same Result

Which job should be scheduled first?

3. Compare our greedy criteria

- Let’s try to get them to disagree.
- Why does it matter if they don’t produce the same result?
- One scoring metric must be better than the other

- Apply the two greedy algorithms and calculate their weighted sum of completion times

Exercise Question 4

3. Compare our greedy criteria

- Jobs will be ordered from biggest to smallest metric value

Job	Difference Metric $(P_i - D_i)$	Ratio Metric (P_i/D_i)
Job 1: P=3, D=5		
Job 2: P=1, D=2		
Total weighted sum		

3. Compare our greedy criteria

- Jobs will be ordered from biggest to smallest metric value

Job	Difference Metric $(P_i - D_i)$	Ratio Metric (P_i/D_i)
Job 1: P=3, D=5	-2	3/5
Job 2: P=1, D=2	-1	1/2
Total weighted sum		

Which job goes first?

3. Compare our greedy criteria

- Jobs will be ordered from biggest to smallest metric value

Job	Difference Metric ($P_i - D_i$)	Ratio Metric (P_i/D_i)
Job 1: P=3, D=5	-2	3/5
Job 2: P=1, D=2	-1	1/2
Total weighted sum		

Which job goes first?

What is the priority sum?

4. Select the "best" one

- Jobs will be ordered from biggest to smallest metric value

Job	Difference Metric ($P_i - D_i$)	Ratio Metric (P_i / D_i)
Job 1: P=3, D=5	-2	3/5
Job 2: P=1, D=2	-1	1/2
Total weighted sum	1*2 + 3*7 = 23	3*5 + 1*7 = 22

Which job goes first?

What is the priority sum?

Which criteria is better?

5. Prove correctness if possible

Is criteria 2 optimal?

- We don’t know yet.

Claim: Criteria 2 is optimal for minimizing the weighted sum of completion times.

- We’re going to prove this using an exchange argument!

Exchange Arguments

- Consider your greedy solution, G

- Consider an alternative solution, A
  - A can be anything that is not G
  - Create A by changing G in some way

- Compare these solutions
  - Show that turning A into G makes A get better

Proof

- Assume that we have no ties (all P_i/D_i are distinct numbers)
- Fix an arbitrary input with n jobs
- Let's perform a proof using an exchange argument contradiction

Let G = the greedy schedule and A = the (alternative) optimal schedule

- Let's **assume** that A must be better than G (assume greedy is not optimal)
- To perform the contradiction, we must show that G is better than A, thus contradicting the purported optimality of A

Proof

Let G = the greedy schedule and A = the optimal schedule

- Assume that: $P_1/D_1 > P_2/D_2 > \ldots > P_n/D_n$
- We can just rename all jobs after we calculate their scores…
- Thus, G is just job 1 followed by job 2 etc. (1, 2, ..., n)

Job ID	Priority	Duration	Ratio	Reorder
1	1	4	0.3	4
2	8	6	1.3	3
3	6	1	6.0	1
4	1	5	0.2	5
5	1	9	0.1	6
6	7	3	2.3	2

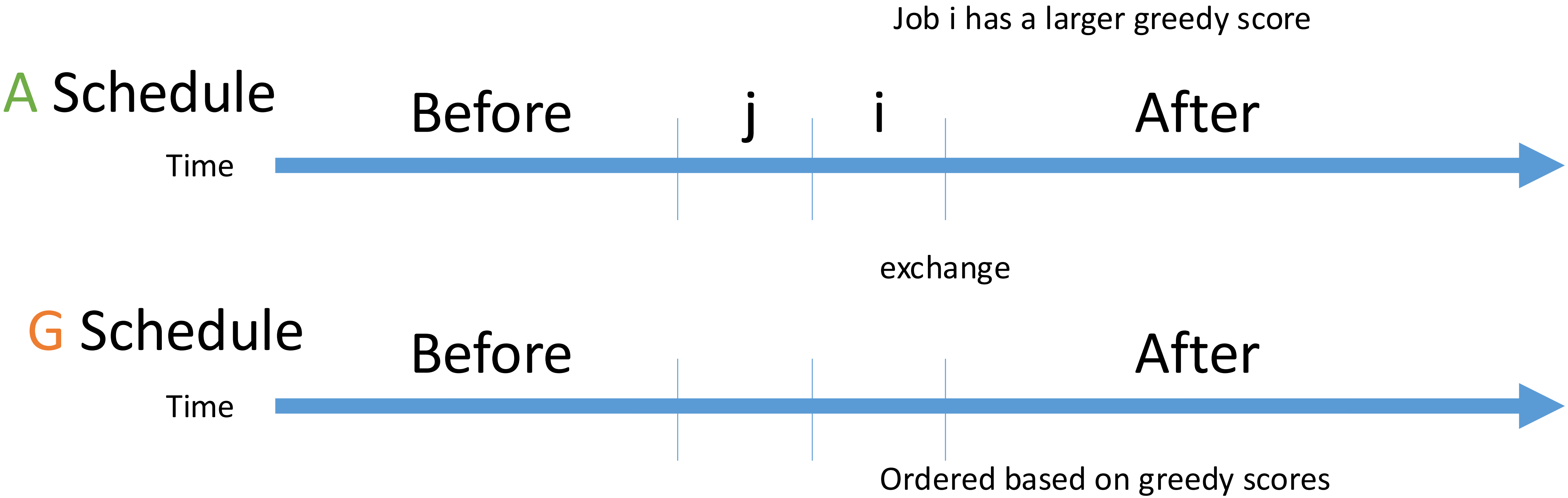
Proof

Let G = the greedy schedule and A = the optimal schedule

- Assume that: $P_1/D_1 > P_2/D_2 > \ldots > P_n/D_n$
- We can just rename all jobs after we calculate their scores…
- Thus, G is just job 1 followed by job 2 etc. (1, 2, ..., n)
- For A there must be at least two jobs that are "out of order"
    - Specifically, jobs i and j where i is scheduled after j, but $S_i > S_j$ (for example, Job_5 after Job_6)
- The greedy schedule is the only schedule where the jobs are in order

G vs A

(jobs i and j where i is scheduled after j, but $P_i/D_i > P_j/D_j$)

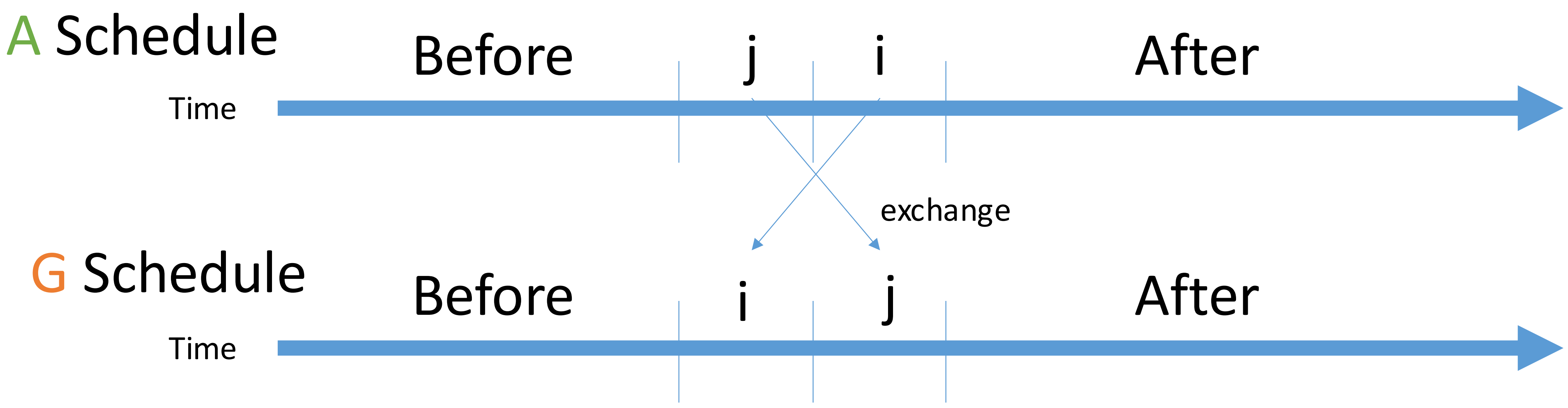


For example, i=7 and j=8

G vs A

(jobs i and j where i is scheduled after j, but $P_i/D_i > P_j/D_j$)

Job i has a larger greedy score



Ordered based on greedy scores

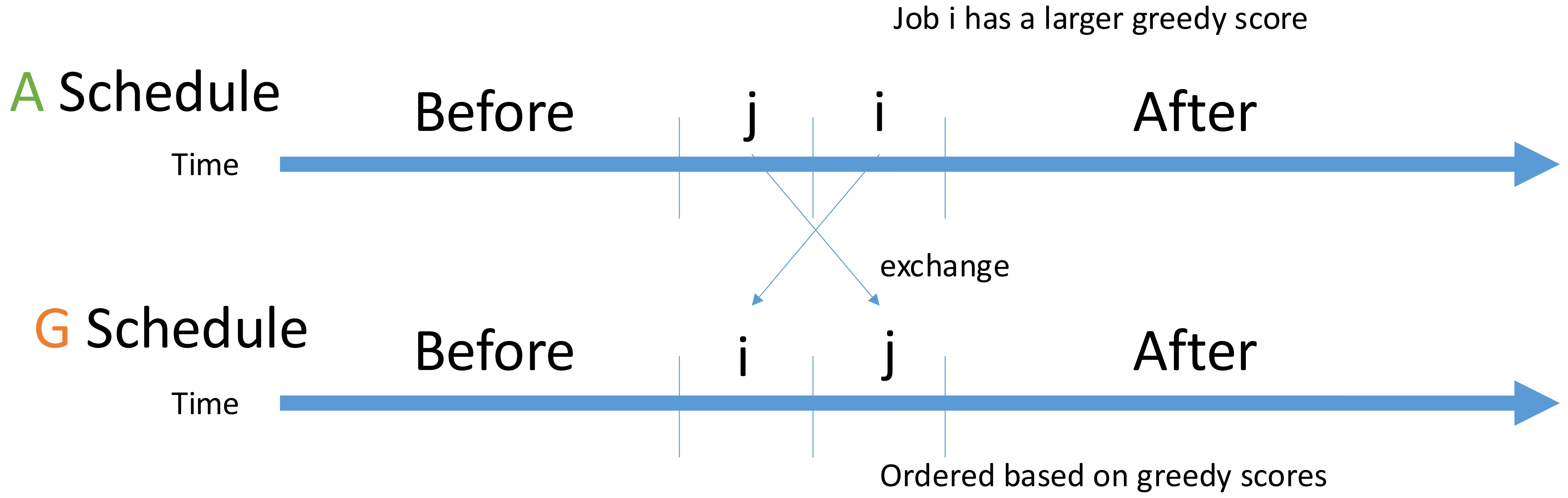
How does the exchange affect the completion time for:

1. Jobs other than i and j?
2. Job i
3. Job j

For example, i=7 and j=8

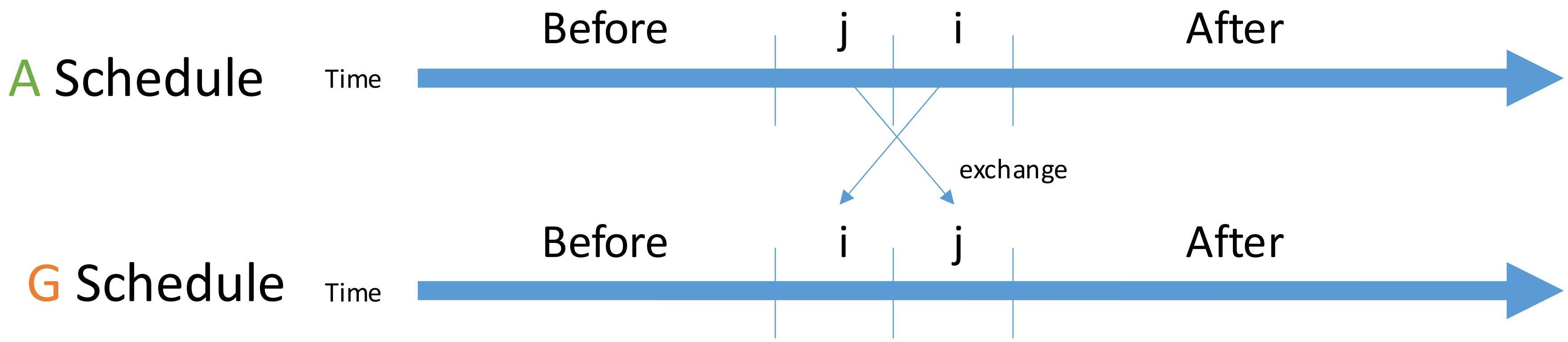
G vs A

(jobs i and j where i is scheduled after j, but $P_i/D_i > P_j/D_j$)



What is the weighted sum of completion times for each schedule?

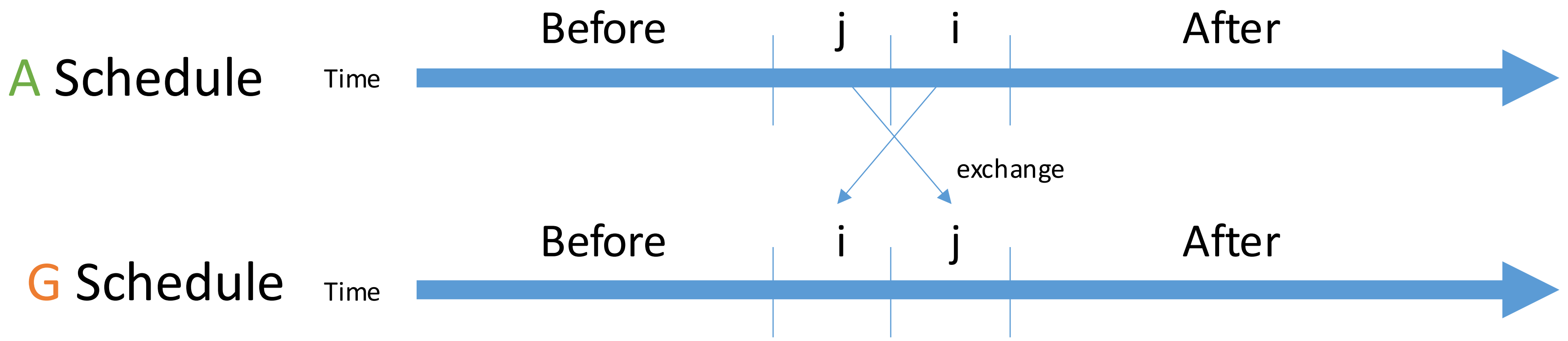
For example, i=7 and j=8




```
Cost(A) = Cost(Before) + Pj * (Tb + Dj) + Pi * (Tb + Dj + Di) + Cost(After)
Cost(G) = Cost(Before) + Pi * (Tb + Di) + Pj * (Tb + Di + Dj) + Cost(After)
```

`Cost(A) < Cost(G)` Implied by optimality of A

```
Cost(Before) + Pj * (Tb + Dj) + Pi * (Tb + Dj + Di) + Cost(After)
    < Cost(Before) + Pi * (Tb + Di) + Pj * (Tb + Di + Dj) + Cost(After)
```




```
Cost(A) = Cost(Before) + Pj * (Tb + Dj) + Pi * (Tb + Dj + Di) + Cost(After)
Cost(G) = Cost(Before) + Pi * (Tb + Di) + Pj * (Tb + Di + Dj) + Cost(After)
```

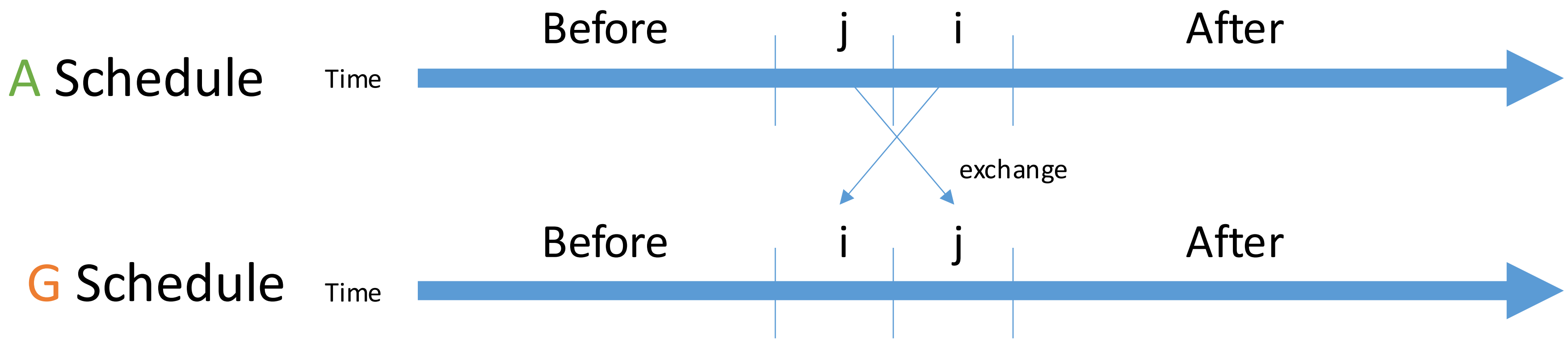
```
Cost(A) < Cost(G)
```

Implied by optimality of A

```
Cost(Before) + Pj * (Tb + Dj) + Pi * (Tb + Dj + Di) + Cost(After)
    < Cost(Before) + Pi * (Tb + Di) + Pj * (Tb + Di + Dj) + Cost(After)

Pj * (Tb + Dj) + Pi * (Tb + Dj + Di)
    < Pi * (Tb + Di) + Pj * (Tb + Di + Dj)

Pj*Tb + Pj*Dj + Pi*Tb + Pi*Dj + Pi*Di
    < Pi*Tb + Pi*Di + Pj*Tb + Pj*Di + Pj*Dj
```



Cost(A) = Cost(Before) + Pj * (Tb + Dj) + Pi * (Tb + Dj + Di) + Cost(After)
Cost(G) = Cost(Before) + Pi * (Tb + Di) + Pj * (Tb + Di + Dj) + Cost(After)

Cost(A) < Cost(G)

Implied by optimality of A

Cost(Before) + Pj * (Tb + Dj) + Pi * (Tb + Dj + Di) + Cost(After)
< Cost(Before) + Pi * (Tb + Di) + Pj * (Tb + Di + Dj) + Cost(After)

Pj * (Tb + Dj) + Pi * (Tb + Dj + Di)
< Pi * (Tb + Di) + Pj * (Tb + Di + Dj)

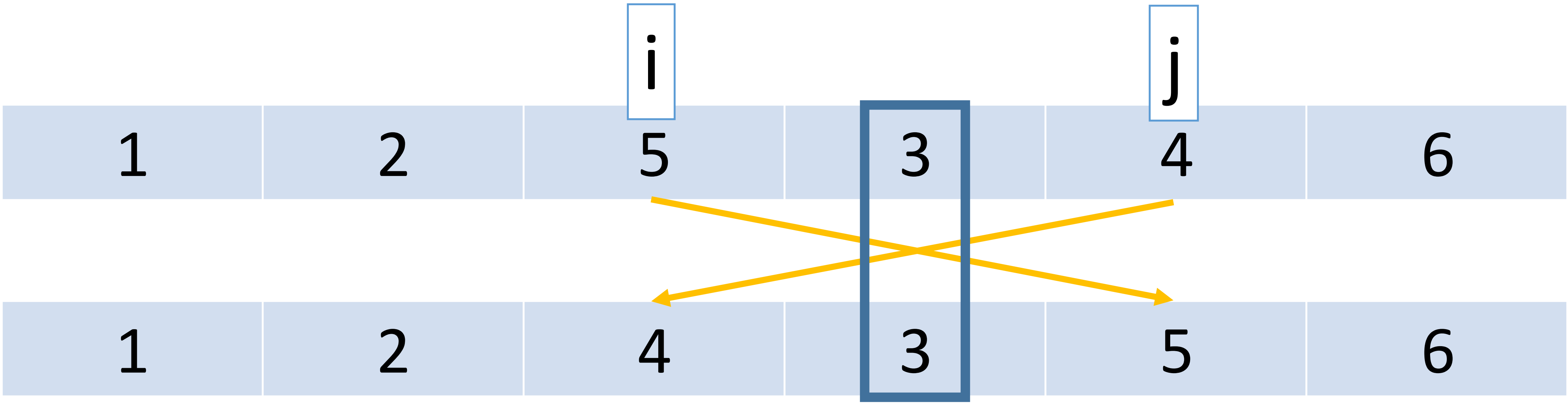
Pj*Tb + Pj*Dj + Pi*Tb + Pi*Dj + Pi*Di
< Pi*Tb + Pi*Di + Pj*Tb + Pj*Di + Pj*Dj

Pi*Dj < Pj*Di

Pi/Di < Pj/Dj

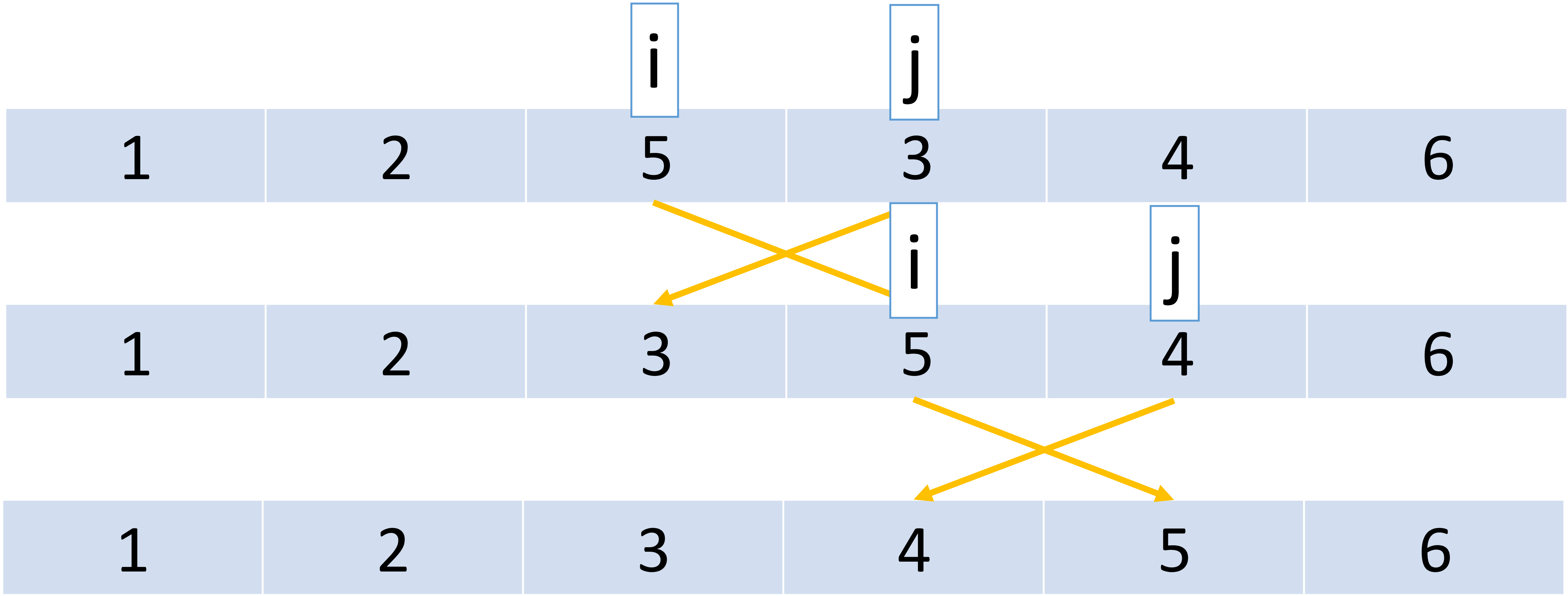
Contradiction to how they were ordered by our greedy criteria

Multiple Re-orderings



Our proof doesn’t account for this

Multiple Re-orderings



Example with Randomly Generated Jobs

					Greedy		Unoptimized	
Job ID	Priority	Duration	Ratio	Reorder	Time	Weighted	Time	Weighted
1	1	4	0.3	4	14	14	4	4
2	8	6	1.3	3	10	80	10	80
3	6	1	6.0	1	1	6	11	66
4	1	5	0.2	5	19	19	16	16
5	1	9	0.1	6	28	28	25	25
6	7	3	2.3	2	4	28	28	196
						175		**387**

Summary of Greedy Scheduling

- Given n jobs, each with a priority and a duration
- Give each job a score based on their ratio of priority to duration
- Schedule jobs in decreasing order of their score
- This gives us an optimal schedule

- What do we do if we're given more jobs while these are running?
- Any issues with this scheme?
  - Some jobs might always be postponed.