

Lecture IV - Scheduling

Programming: Everyday Decision-Making Algorithms

Dr. Tobias Vlček

Kühne Logistics University Hamburg - Winter 2024

Introduction

Today's Topic: Scheduling

Topic: Understanding scheduling problems and algorithmic solutions for optimal task ordering

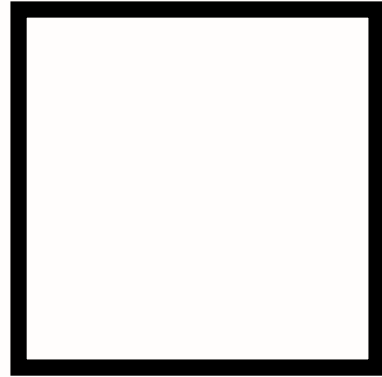
Why this matters: Every day you make scheduling decisions, from organizing your to-do list to managing projects. Today we'll learn the mathematical algorithms that can optimize these decisions and see how they connect to programming logic.

Today's Agenda

1. **Scheduling Fundamentals** - Johnson's Rule and two-machine problems
2. **Historical Context** - How scheduling theory developed
3. **Core Algorithms** - EDD for deadlines, SPT for efficiency
4. **Advanced Topics** - Dependencies, real-time scheduling, thrashing
5. **Programming Connections** - How this builds algorithmic problem-solving skills

Fundamentals

A Simple Scheduling Problem



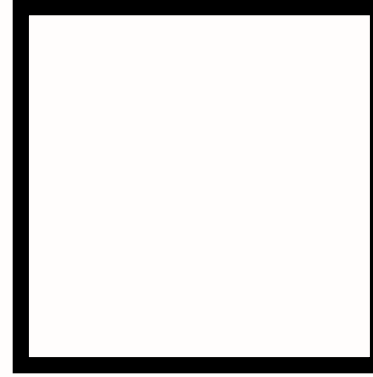
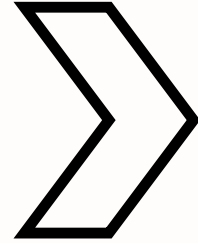
Washing

40 min

30 min

60 min

50 min



Drying

60 min

80 min

30 min

60 min

Washing Machine & Dryer

Let's solve this simple scheduling problem:

1	Task	Washing	Drying
2	A	<u>40min</u>	<u>60min</u>
3	B	<u>30min</u>	<u>80min</u>
4	C	<u>60min</u>	<u>30min</u>
5	D	<u>50min</u>	<u>60min</u>

Goal: Minimize total time for washing and drying all loads

Question: An idea how to solve this?

Johnson's Rule

Rule: To find the **optimal solution**:

1. Find the job with **shortest duration**:

- If on **Machine 1** → Schedule First
- If on **Machine 2** → Schedule Last
- If **equal** → Choose randomly

2. Remove job from list and **repeat**

Applying Johnson's Rule

1	Task	Washing	Drying	Schedule
2	A	<u>40min</u>	<u>60min</u>	
3	B	<u>30min</u>	<u>80min</u>	
4	C	<u>60min</u>	<u>30min</u>	
5	D	<u>50min</u>	<u>60min</u>	

Question: What's the first scheduled task?

Applying Johnson's Rule

1	Task	Washing	Drying	Schedule
2	A	<u>40min</u>	<u>60min</u>	
3	B	<u>30min</u>	<u>80min</u>	
4	C	<u>60min</u>	<u>30min</u>	4
5	D	<u>50min</u>	<u>60min</u>	

- In Task C, the dryer is the **shortest** task.
- It is on Machine 2 → **Schedule Last**

Question: What's the next task?

Applying Johnson's Rule

1	Task	Washing	Drying	Schedule
2	A	<u>40min</u>	<u>60min</u>	
3	B	<u>30min</u>	<u>80min</u>	1
4	C	<u>60min</u>	<u>30min</u>	4
5	D	<u>50min</u>	<u>60min</u>	

- In Task B, the washing machine is the **shortest** task.
- It is on Machine 1 → **Schedule First**

Question: What's the next task?

Applying Johnson's Rule

1	Task	Washing	Drying	Schedule
2	A	<u>40min</u>	<u>60min</u>	2
3	B	<u>30min</u>	<u>80min</u>	1
4	C	<u>60min</u>	<u>30min</u>	4
5	D	<u>50min</u>	<u>60min</u>	

- In Task A, the washing machine is the **shortest** task.
- It is on Machine 1 → **Schedule Second**

Now, we only have one task left!

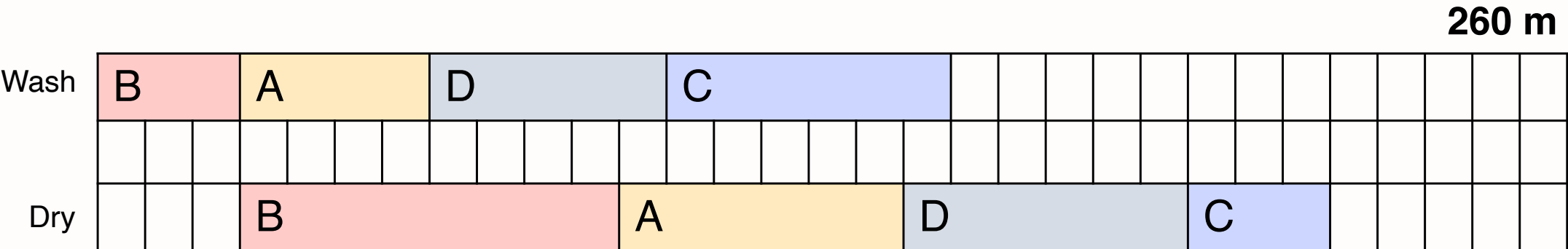
Applying Johnson's Rule

1	Task	Washing	Drying	Schedule
2	A	<u>40min</u>	<u>60min</u>	2
3	B	<u>30min</u>	<u>80min</u>	1
4	C	<u>60min</u>	<u>30min</u>	4
5	D	<u>50min</u>	<u>60min</u>	3

Final sequence: **B A D C**

Optimal Solution

Optimal Solution: **B A D C**



Total time: 4 hours 20 minutes

Question: Is there a worse solution?

Suboptimal Solution: C D A B

Lecture IV - Scheduling | Dr. Tobias Vlček | [Home](#)

History

Industrial Revolution

- First systematic visualization by Frederick Taylor
- Henry Gantt develops the Gantt Chart around 1910
- Key tool during Industrial Revolution
- But no scheduling theory yet!

Question: Who knows what a Gantt Chart is?

Modern Scheduling Theory

- RAND Corporation founded (1948)
- Selmer Johnson publishes Johnson's Rule in 1954
- Beginning of modern scheduling theory
- Many more algorithms and methods developed since

Professional Applications

Where do we see scheduling in professional practice?

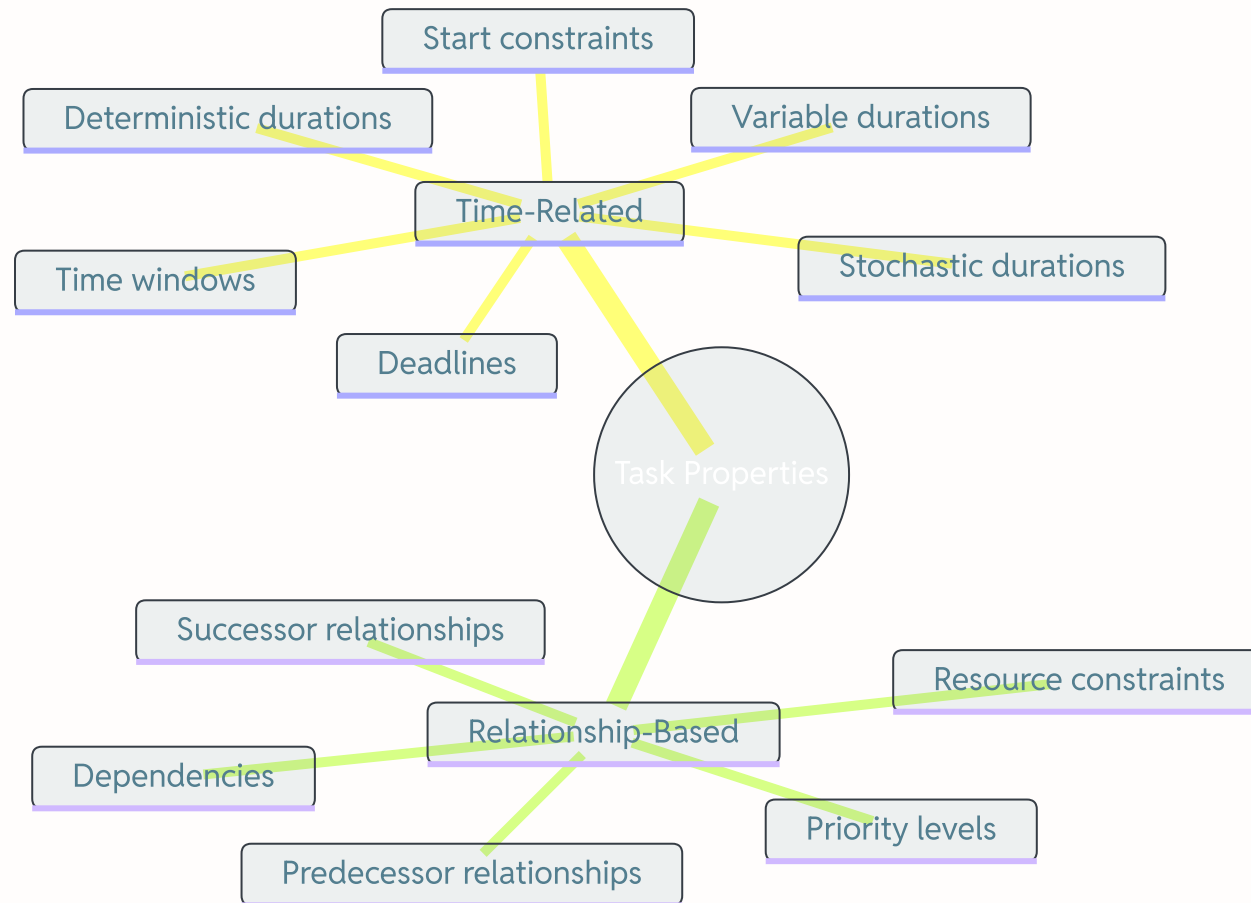
- **Project Management:** Task dependencies, resource allocation
- **Software Development:** CPU, thread management, job queues
- **Operations:** Production, supply chain optimization
- **Healthcare:** Patients, surgery planning, staff allocation
- **Transportation:** Routes, crews, maintenance

Connection to Programming: These are the same optimization problems that algorithms solve!

Scheduling Tasks

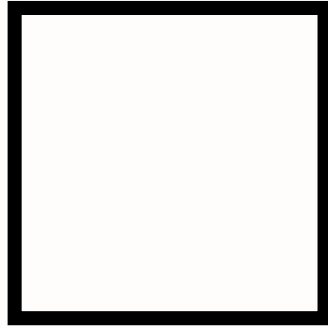
Task Classification

Question: What properties can scheduled tasks have?



Question: What types of tasks do you deal with most often?

Single Machine Scheduling



A human

A

40 min

B

30 min

C

60 min

D

50 min

Question: What is different from before?

Order is Irrelevant

Order is Irrelevant

Under simple minimization of total processing time, **order doesn't matter!**

Question: But is it that simple?

Order Matters

Order becomes crucial when we consider, **Deadlines, Priorities and Dependencies!**

Deadlines

Earliest Due Date (EDD)

Tasks with individual deadlines:

1	Task	Duration	Deadline
2	A	<u>40min</u>	<u>110min</u>
3	B	<u>30min</u>	<u>90min</u>
4	C	<u>60min</u>	<u>150min</u>
5	D	<u>50min</u>	<u>70min</u>
6	E	<u>30min</u>	<u>210min</u>

Goal: Minimize maximum deadline violation

Question: An idea how to solve this?

EDD Solution

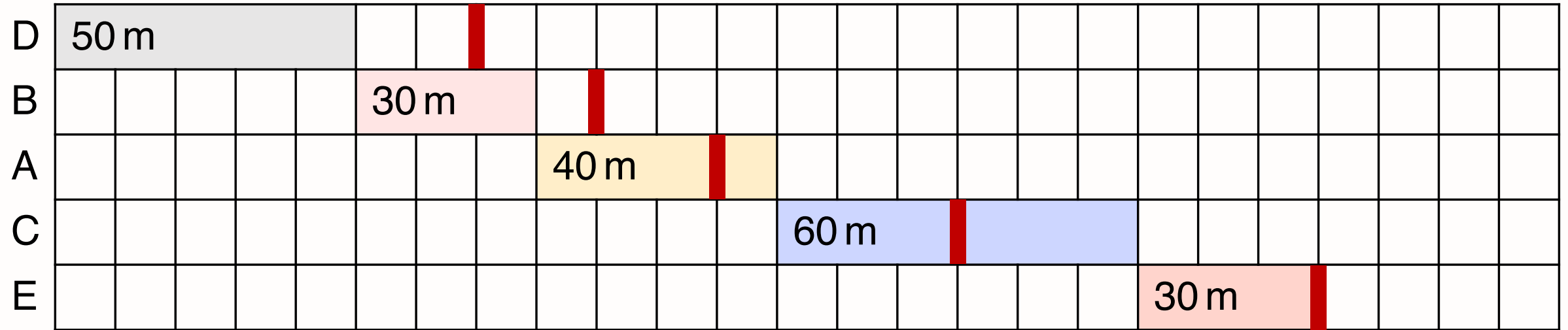
Rule: Sort the tasks by deadline.

1	Task	Duration	Deadline
2	A	<u>40min</u>	<u>110min</u>
3	B	<u>30min</u>	<u>90min</u>
4	C	<u>60min</u>	<u>150min</u>
5	D	<u>50min</u>	<u>70min</u>
6	E	<u>30min</u>	<u>210min</u>

1	Task	Duration	Deadline
2	D	<u>50min</u>	<u>70min</u>
3	B	<u>30min</u>	<u>90min</u>
4	A	<u>40min</u>	<u>110min</u>
5	C	<u>60min</u>	<u>150min</u>
6	E	<u>30min</u>	<u>210min</u>

Let's visualize this!

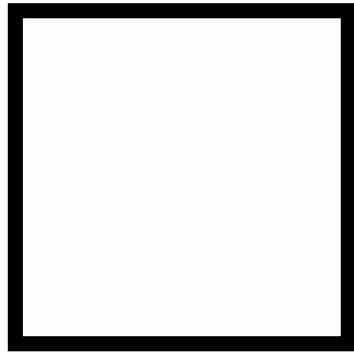
EDD Schedule



Question: What's the maximum delay here?

Processing Time

Shortest Processing Time (SPT)



A human

A

40 min

B

30 min

C

60 min

D

50 min

Instead of deadlines, we now have **processing times**.

Goal: Min. total waiting time

Question: Any ideas?

Rule: Always schedule the shortest remaining task

Shortest Processing Time Applied

Rule: Always schedule the **shortest remaining task**. Choose random if multiple tasks are tied.

1	Task	Duration	Schedule
2	A	40min	
3	B	30min	
4	C	60min	
5	D	50min	
6	E	30min	

Question: What's the order of scheduled tasks?

Shortest Processing Time Applied

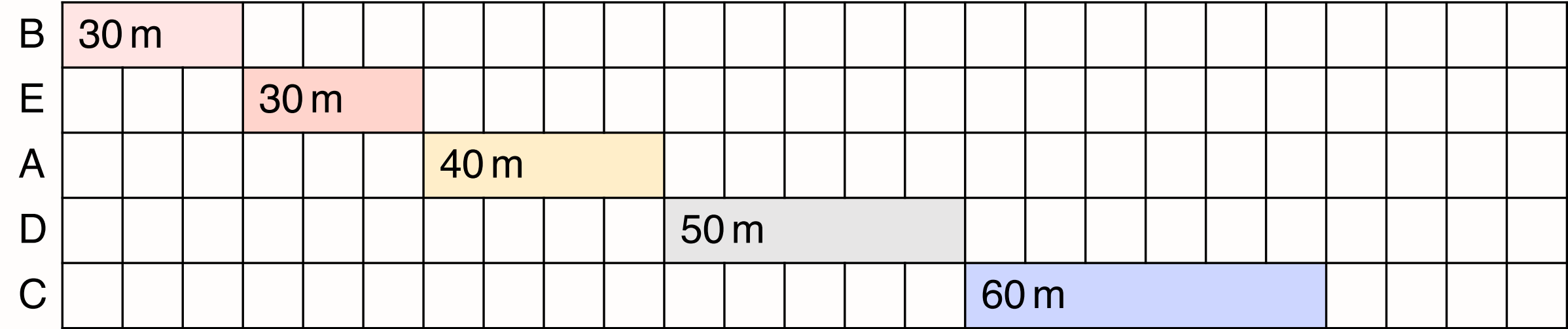
Rule: Always schedule the **shortest remaining task**. Choose random if multiple tasks are tied.

1	Task	Duration	Schedule
2	A	40min	3
3	B	30min	1
4	C	60min	5
5	D	50min	4
6	E	30min	2

Final sequence: **B E A D C** or **E B A D C**

SPT Solution

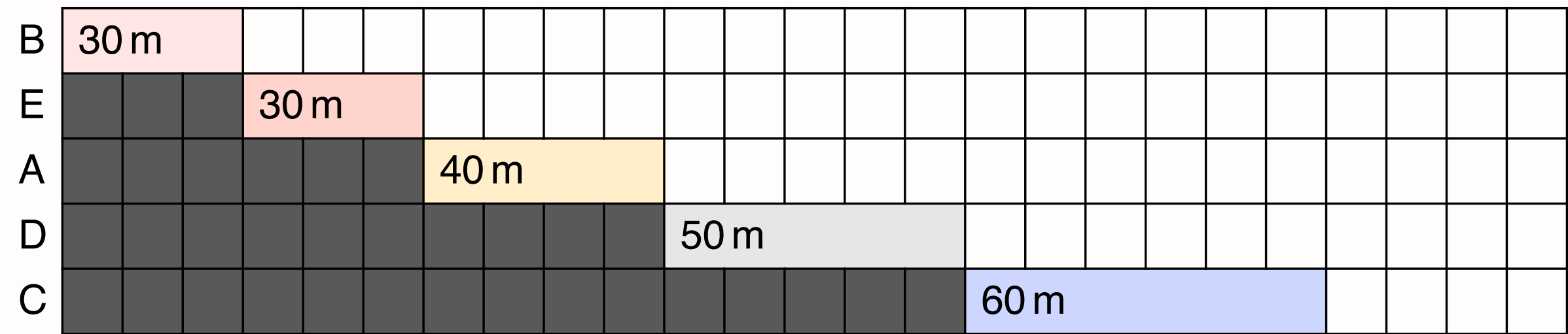
Optimal sequence:



Question: Where can we see the waiting time?

SPT Waiting Time

Total waiting time: 340 minutes



Question: Would this be applicable for your work?

Weighted SPT

- **Change:** Tasks with additional priorities
- Priorities could be, e.g., revenue if we are consultants.

1	Task	Duration	Revenue
2	A	20min	€240
3	B	30min	€200
4	C	60min	€120
5	D	50min	€70
6	E	30min	€130
7	F	40min	€120
8	G	20min	€100
9	H	70min	€110
10	I	50min	€90

Question: Any ideas how to approach this?

Gain/Revenue Per Minute

Rule: Schedule by revenue per minute (descending)

1	Task	Duration	Revenue	Revenue/Min	Schedule
2	A	20min	€240	12.0	
3	B	30min	€200	6.7	
4	C	60min	€120	2.0	
5	D	50min	€70	1.4	
6	E	30min	€130	4.3	
7	F	40min	€120	3.0	
8	G	20min	€100	5.0	
9	H	70min	€110	1.6	
10	I	50min	€90	1.8	

Question: What's the order of scheduled tasks?

Gain/Revenue Per Minute

Rule: Schedule by revenue per minute (descending)

1	Task	Duration	Revenue	Revenue/Min	Schedule
2	A	20min	€240	12.0	1
3	B	30min	€200	6.7	
4	C	60min	€120	2.0	
5	D	50min	€70	1.4	
6	E	30min	€130	4.3	
7	F	40min	€120	3.0	
8	G	20min	€100	5.0	
9	H	70min	€110	1.6	
10	I	50min	€90	1.8	

Gain/Revenue Per Minute

Rule: Schedule by revenue per minute (descending)

1	Task	Duration	Revenue	Revenue/Min	Schedule
2	A	<u>20min</u>	€240	12.0	1
3	B	<u>30min</u>	€200	6.7	2
4	C	<u>60min</u>	€120	2.0	
5	D	<u>50min</u>	€70	1.4	
6	E	<u>30min</u>	€130	4.3	
7	F	<u>40min</u>	€120	3.0	
8	G	<u>20min</u>	€100	5.0	
9	H	<u>70min</u>	€110	1.6	
10	I	<u>50min</u>	€90	1.8	

Gain/Revenue Per Minute

Rule: Schedule by revenue per minute (descending)

1	Task	Duration	Revenue	Revenue/Min	Schedule
2	A	<u>20min</u>	€240	12.0	1
3	B	<u>30min</u>	€200	6.7	2
4	C	<u>60min</u>	€120	2.0	
5	D	<u>50min</u>	€70	1.4	
6	E	<u>30min</u>	€130	4.3	
7	F	<u>40min</u>	€120	3.0	
8	G	<u>20min</u>	€100	5.0	3
9	H	<u>70min</u>	€110	1.6	
10	I	<u>50min</u>	€90	1.8	

Gain/Revenue Per Minute

Rule: Schedule by revenue per minute (descending)

1	Task	Duration	Revenue	Revenue/Min	Schedule
2	A	<u>20min</u>	€240	12.0	1
3	B	<u>30min</u>	€200	6.7	2
4	C	<u>60min</u>	€120	2.0	
5	D	<u>50min</u>	€70	1.4	
6	E	<u>30min</u>	€130	4.3	4
7	F	<u>40min</u>	€120	3.0	
8	G	<u>20min</u>	€100	5.0	3
9	H	<u>70min</u>	€110	1.6	
10	I	<u>50min</u>	€90	1.8	

Gain/Revenue Per Minute

Rule: Schedule by revenue per minute (descending)

1	Task	Duration	Revenue	Revenue/Min	Schedule
2	A	<u>20min</u>	€240	12.0	1
3	B	<u>30min</u>	€200	6.7	2
4	C	<u>60min</u>	€120	2.0	
5	D	<u>50min</u>	€70	1.4	
6	E	<u>30min</u>	€130	4.3	4
7	F	<u>40min</u>	€120	3.0	5
8	G	<u>20min</u>	€100	5.0	3
9	H	<u>70min</u>	€110	1.6	
10	I	<u>50min</u>	€90	1.8	

Gain/Revenue Per Minute

Rule: Schedule by revenue per minute (descending)

1	Task	Duration	Revenue	Revenue/Min	Schedule
2	A	<u>20min</u>	€240	12.0	1
3	B	<u>30min</u>	€200	6.7	2
4	C	<u>60min</u>	€120	2.0	6
5	D	<u>50min</u>	€70	1.4	
6	E	<u>30min</u>	€130	4.3	4
7	F	<u>40min</u>	€120	3.0	5
8	G	<u>20min</u>	€100	5.0	3
9	H	<u>70min</u>	€110	1.6	
10	I	<u>50min</u>	€90	1.8	

Gain/Revenue Per Minute

Rule: Schedule by revenue per minute (descending)

1	Task	Duration	Revenue	Revenue/Min	Schedule
2	A	20min	€240	12.0	1
3	B	30min	€200	6.7	2
4	C	60min	€120	2.0	6
5	D	50min	€70	1.4	
6	E	30min	€130	4.3	4
7	F	40min	€120	3.0	5
8	G	20min	€100	5.0	3
9	H	70min	€110	1.6	
10	I	50min	€90	1.8	7

Gain/Revenue Per Minute

Rule: Schedule by revenue per minute (descending)

1	Task	Duration	Revenue	Revenue/Min	Schedule
2	A	20min	€240	12.0	1
3	B	30min	€200	6.7	2
4	C	60min	€120	2.0	6
5	D	50min	€70	1.4	
6	E	30min	€130	4.3	4
7	F	40min	€120	3.0	5
8	G	20min	€100	5.0	3
9	H	70min	€110	1.6	8
10	I	50min	€90	1.8	7

Gain/Revenue Per Minute

Rule: Schedule by revenue per minute (descending)

1	Task	Duration	Revenue	Revenue/Min	Schedule
2	A	20min	€240	12.0	1
3	B	30min	€200	6.7	2
4	C	60min	€120	2.0	6
5	D	50min	€70	1.4	9
6	E	30min	€130	4.3	4
7	F	40min	€120	3.0	5
8	G	20min	€100	5.0	3
9	H	70min	€110	1.6	8
10	I	50min	€90	1.8	7

Metric Priorities

Without revenues, we can use the same approach with **metric priorities**!

Dependencies

Priority Inversion

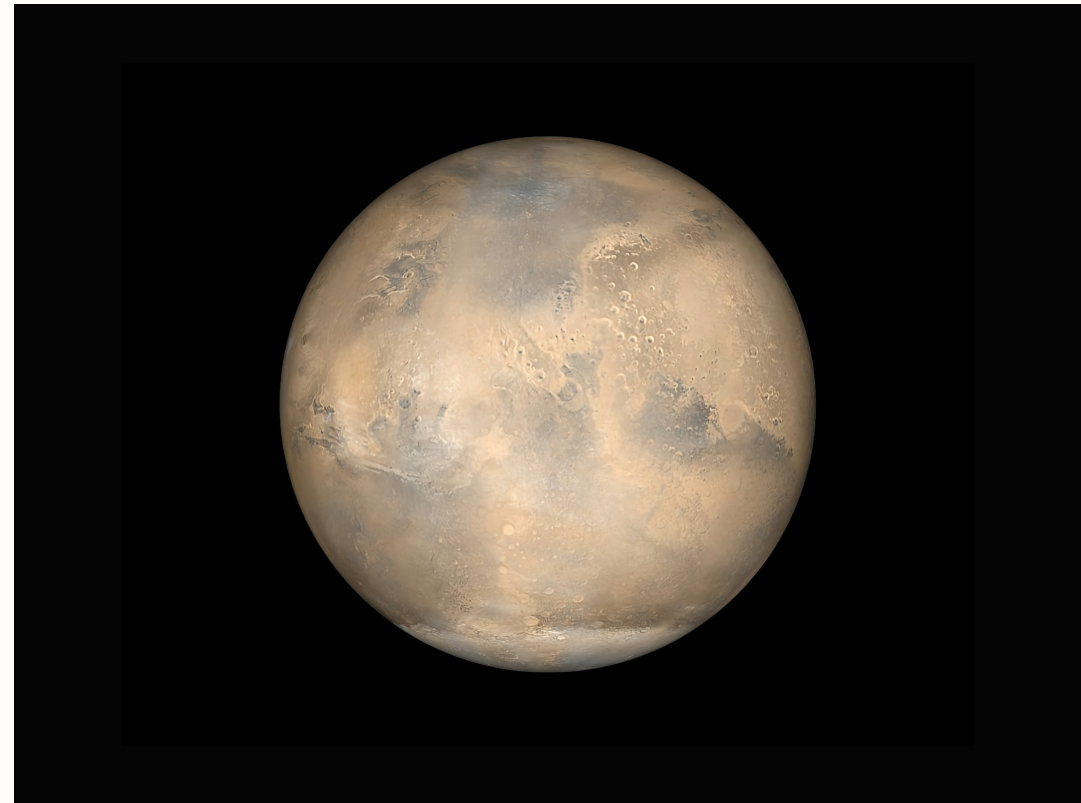
Setup:

1	Task	Duration	Priority
2	A	20min	3
3	B	30min	1
4	C	30min	2
5	D	30min	2
6	E	30min	2



Challenge: High-priority tasks depend on low-priority tasks.

Risk: Priority inversion can lead to significant delays!



Priority Inheritance

Question: How to handle with shortest processing time?

- **Rule:** Tasks inherit priority from their dependents.
- A gets the highest priority from B
- This ensures the **critical path completion**

1	Task	Duration	Priority
2	A	<u>20min</u>	1
3	B	<u>30min</u>	1
4	C	<u>30min</u>	2
5	D	<u>30min</u>	2
6	E	<u>30min</u>	2

EDD and Dependencies

Question: What's was earliest due date again?

- Sort the tasks by deadline, schedule equal tasks randomly
- Things get more complex when we add dependencies

1	Task	Duration	Deadline	Predecessor
2	A	<u>40min</u>	<u>110min</u>	None
3	B	<u>30min</u>	<u>90min</u>	D
4	C	<u>60min</u>	<u>150min</u>	A
5	D	<u>50min</u>	<u>70min</u>	None
6	E	<u>30min</u>	<u>210min</u>	C

Question: Any ideas how to solve this?

Lawler's Algorithm

Rule: We can use Lawler's Algorithm (1968)

1. Consider all tasks **without successors**
2. Choose the one with **latest deadline**
3. Schedule the task **last**
4. **Remove** it from the network and start again

Lawler's Applied

1	Task	Duration	Deadline	Predecessor	Schedule
2	A	<u>40min</u>	<u>110min</u>	None	
3	B	<u>30min</u>	<u>90min</u>	D	
4	C	<u>60min</u>	<u>150min</u>	A	
5	D	<u>50min</u>	<u>70min</u>	None	
6	E	<u>30min</u>	<u>210min</u>	C	

Question: What's the schedule?

Lawler's Applied

1	Task	Duration	Deadline	Predecessor	Schedule
2	A	<u>40min</u>	<u>110min</u>	None	
3	B	<u>30min</u>	<u>90min</u>	D	
4	C	<u>60min</u>	<u>150min</u>	A	
5	D	<u>50min</u>	<u>70min</u>	None	
6	E	<u>30min</u>	<u>210min</u>	C	5



Lawler's Applied

1	Task	Duration	Deadline	Predecessor	Schedule
2	A	<u>40min</u>	<u>110min</u>	None	
3	B	<u>30min</u>	<u>90min</u>	D	
4	C	<u>60min</u>	<u>150min</u>	A	4
5	D	<u>50min</u>	<u>70min</u>	None	
6	E	<u>30min</u>	<u>210min</u>	C	5



Lawler's Applied

1	Task	Duration	Deadline	Predecessor	Schedule
2	A	<u>40min</u>	<u>110min</u>	None	3
3	B	<u>30min</u>	<u>90min</u>	D	
4	C	<u>60min</u>	<u>150min</u>	A	4
5	D	<u>50min</u>	<u>70min</u>	None	
6	E	<u>30min</u>	<u>210min</u>	C	5



Lawler's Applied

1	Task	Duration	Deadline	Predecessor	Schedule
2	A	<u>40min</u>	<u>110min</u>	None	3
3	B	<u>30min</u>	<u>90min</u>	D	2
4	C	<u>60min</u>	<u>150min</u>	A	4
5	D	<u>50min</u>	<u>70min</u>	None	
6	E	<u>30min</u>	<u>210min</u>	C	5



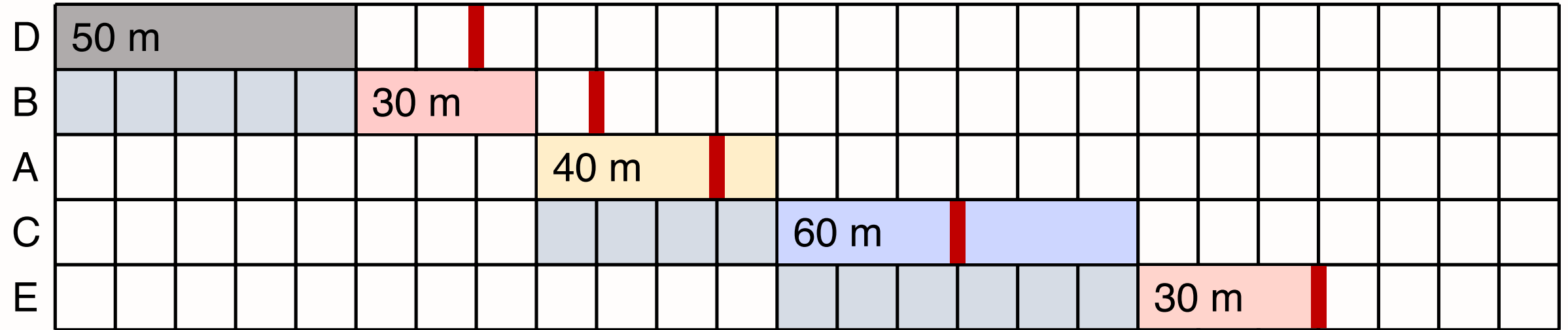
Lawler's Applied

1	Task	Duration	Deadline	Predecessor	Schedule
2	A	<u>40min</u>	<u>110min</u>	None	3
3	B	<u>30min</u>	<u>90min</u>	D	2
4	C	<u>60min</u>	<u>150min</u>	A	4
5	D	<u>50min</u>	<u>70min</u>	None	1
6	E	<u>30min</u>	<u>210min</u>	C	5

Successor Tasks

Note, how all tasks become tasks without successors at some point.

Lawler's Solution



i Predecessor Tasks

Predecessor tasks are tasks that must be completed before the current task can start. They are marked in grey in the chart.

Question: What's the maximum delay?

Scheduling Difficulty

SPT with Predecessors

- No solution in polynomial time
- **NP-hard problem** (no efficient algorithm)
- True for most scheduling problems!
- We can use heuristics, though

Question: What have we missed so far?

Real-time Scheduling

Interruptions

- In reality, we **cannot predict the future**
- We need to react to **new tasks** as they happen
- If we have a **deadline**, we might need to meet it
- Let's look at this for the **earliest due date objective**

Quick reminder

An earliest due date is a specific point in time by which a task must be completed. Under this objective, we want to minimize the **maximum delay**.

Real-time EDD

8:00-12:00 Schedule:

1 Task	Duration	Deadline
2 Email A	<u>20min</u>	9:00
3 Create PPT	<u>60min</u>	10:50
4 Investor call	<u>10min</u>	9:00
5 Email B	<u>30min</u>	10:20
6 Liquidity	<u>40min</u>	11:00
7 Email C	<u>20min</u>	11:20
8 Email D	<u>40min</u>	10:00

Question: Any ideas how to start with under the objective of the earliest due date?

EDD Rule for Real-time

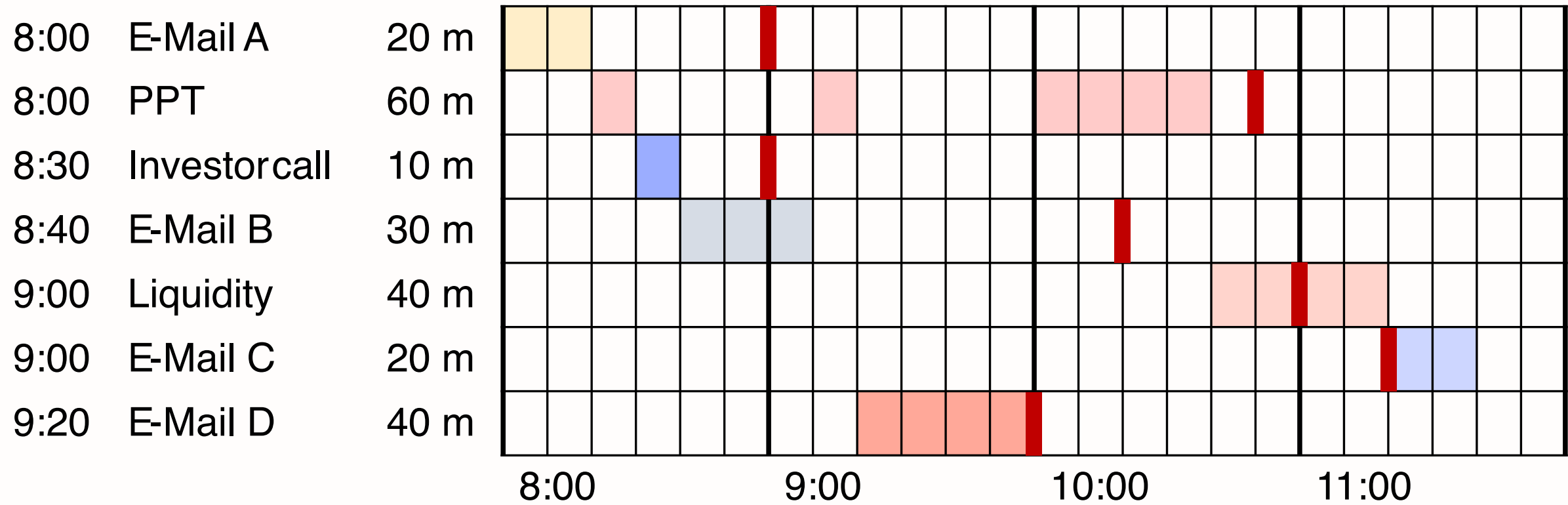
Rule:

- Always schedule the task with the **earliest deadline**
- If a new task with an earlier deadline comes in, **re-schedule**
- Otherwise, stick to the original schedule.

Equal Deadline

If a new task has the same deadline as the current task, you can choose either. But due to the cost of context switching, you might want to stick with the current task.

EDD Solution for Real-time



Question: What's the maximum delay with this schedule?

SPT for Real-time

Quick reminder

A shortest processing time is the task with the shortest duration. Under this objective, we want to minimize the **total waiting time**.

Question: Any ideas how to start here?

SPT Rule for Real-time

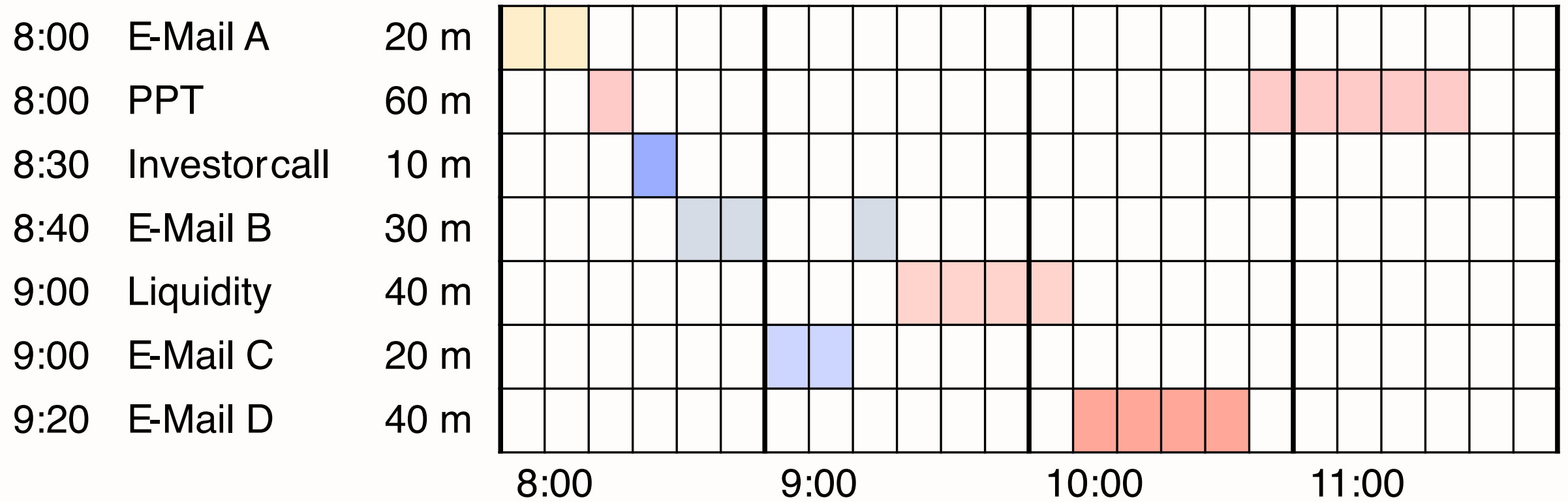
Rule:

- Always schedule the task with the **shortest duration**
- If a new task with a shorter duration comes in, **re-schedule**
- Otherwise, stick to the original schedule.

Equal Duration

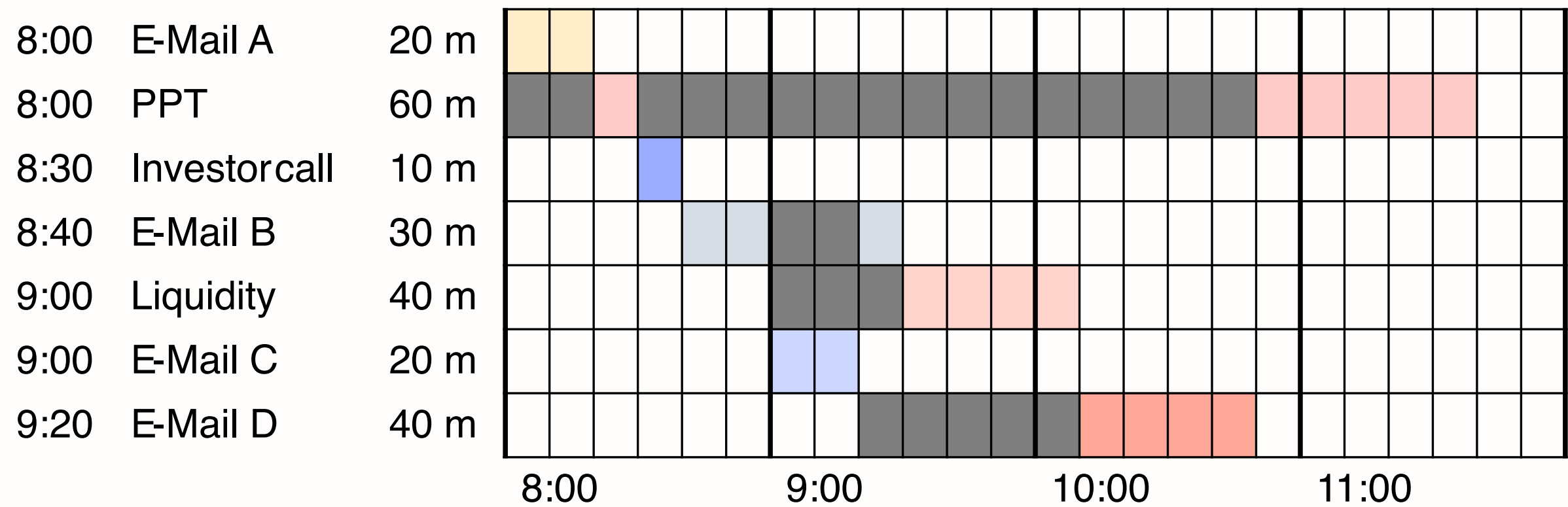
If a new task has the same duration as the current task, you can choose either. But due to the cost of context switching, you might again want to stick with the current task.

SPT Solution for Real-time



Question: Where can we see the waiting time?

SPT Solution for Real-time



Total waiting time: 260 minutes

Thrashing

What is Thrashing?

- Excessive context switching
- Organization **overhead exceeds productivity**
- Maximum activity, minimum output

Question: Have you ever experienced this?

Thrashing Warning Signs

- Constant **task switching**
- **Nothing** getting completed
- Increasing **stress levels**
- Declining **quality**

Question: Any ideas how to prevent this?

Preventing Thrashing Strategic

Strategic

Strategic solutions focus on **long-term** changes to prevent thrashing.

1. Task rejection/delegation threshold
2. Simplified organization systems
3. Minimum work period rules
4. Reduced reactivity requirements

Preventing Thrashing Tactical

1. Time blocking
2. Focus periods
3. Task batching
4. Priority freezes

Question: What strategies have worked for you?

Apply This Knowledge

Immediate Applications

How can you use these algorithms starting today?

- **Daily planning:** Use SPT for your to-do list to minimize waiting time
- **Project management:** Apply EDD when managing deadlines
- **Team coordination:** Use Johnson's Rule for two-stage processes
- **Time blocking:** Prevent thrashing with strategic scheduling

Key Takeaways

What we learned today:

1. **Scheduling algorithms** provide optimal solutions to time management problems
2. **Different objectives** require different algorithms
3. **Dependencies and constraints** add complexity but have algorithmic solutions
4. **Thrashing prevention** is crucial for productivity
5. **These principles** directly translate to programming and system design

Any questions so far?

After the break — Scheduling

- Programming session in our new notebooks
- How to translate the idea into code and experiments
- Different scheduling algorithms applied to problems

 **That's it for scheduling!**

Let's have a short break and then continue with our fourth Python programming session.

Literature

Interesting literature to start

- Christian, B., & Griffiths, T. (2016). Algorithms to live by: the computer science of human decisions. First international edition. New York, Henry Holt and Company.¹

Books on Programming

- Downey, A. B. (2024). Think Python: How to think like a computer scientist (Third edition). O'Reilly. [Here](#)
- Elter, S. (2021). Schrödinger programmiert Python: Das etwas andere Fachbuch (1. Auflage). Rheinwerk Verlag.

Questions & Discussion

- i Think Python is a great book to start with. It's available online for free. Schrödinger Programmiert Python is a great alternative for German students, as it is a very playful introduction to programming with lots of examples.

More Literature

For more interesting literature, take a look at the [literature list](#) of this course.