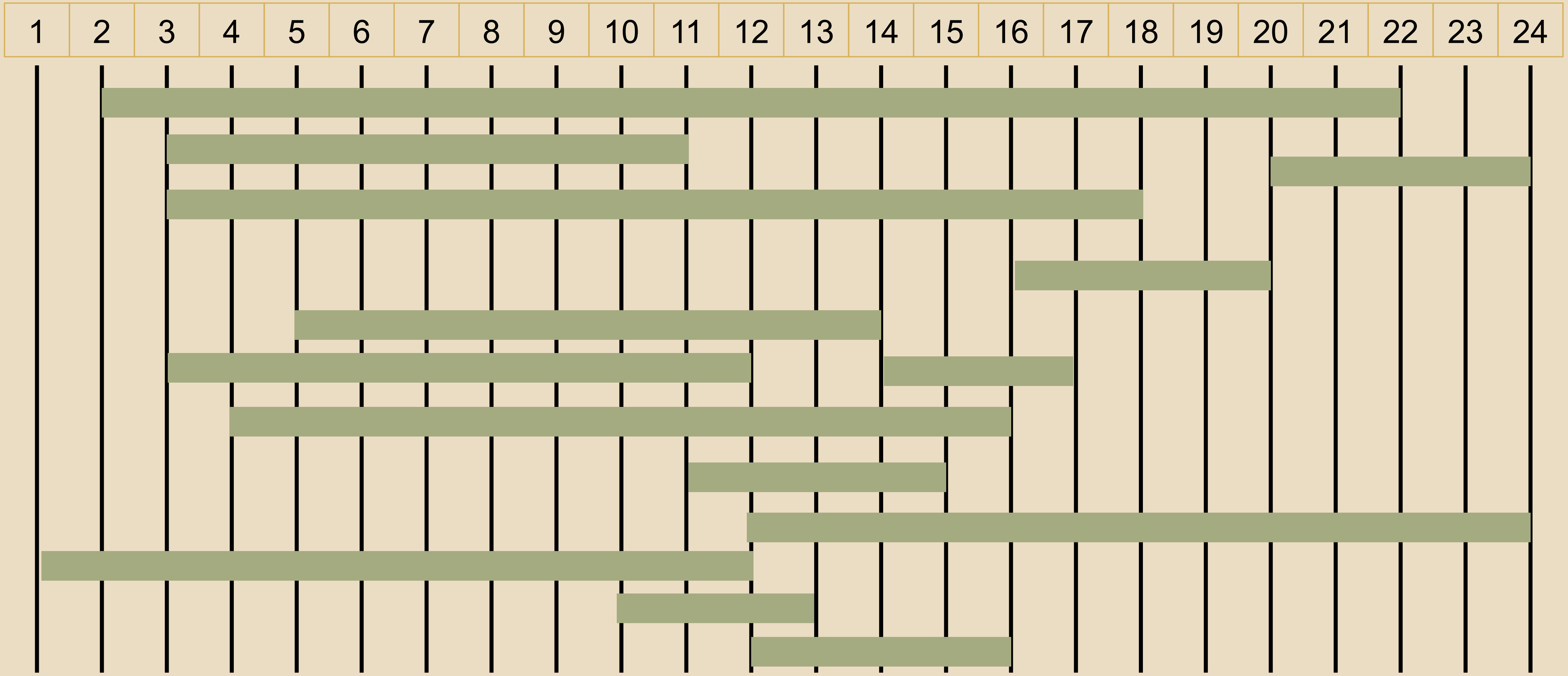
COL351: Slides for Lecture Component 13

Thanks to Miles Jones, Russell Impagliazzo, and Sanjoy Dasgupta at UCSD for these slides.

EVENT SCHEDULING

- You are running a cookie conference and you have a collection of events (or talks) that each has a start time and a finish time.
- Oh no!!! You only have one conference room!!!
- Your goal is to *schedule the most events possible that day such that no two events overlap*.

EVENT SCHEDULING



EVENT SCHEDULING SPECIFICATION

- Instance:
- Solution format:
- Constraints:
- Objective:

EVENT SCHEDULING

Your goal is to schedule the most events possible that day such that no two events overlap.

- Brute Force: Say that there are n events.
- Let’s check all possibilities. How would we do that?

EVENT SCHEDULING

- Your goal is to schedule the most events possible that day such that no two events overlap.

- Brute Force: Say that there are n events.
- Let’s check all possibilities. How would we do that?

- Go through all subsets of events. Check if it is a valid schedule, i.e., no conflicts, and count the number of events.
- Take the maximum out of all valid schedules.
- (How many subsets are there?)

EVENT SCHEDULING

- Your goal is to schedule the most events possible that day such that no two events overlap.
- Exponential is too slow. Let’s think of some greedy strategies:

EVENT SCHEDULING

- Your goal is to schedule the most events possible that day such that no two events overlap.

- Exponential is too slow. Let’s try some greedy strategies:
  - Shortest duration
  - Earliest start time
  - Fewest conflicts
  - Earliest end time

SHORTEST DURATION

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

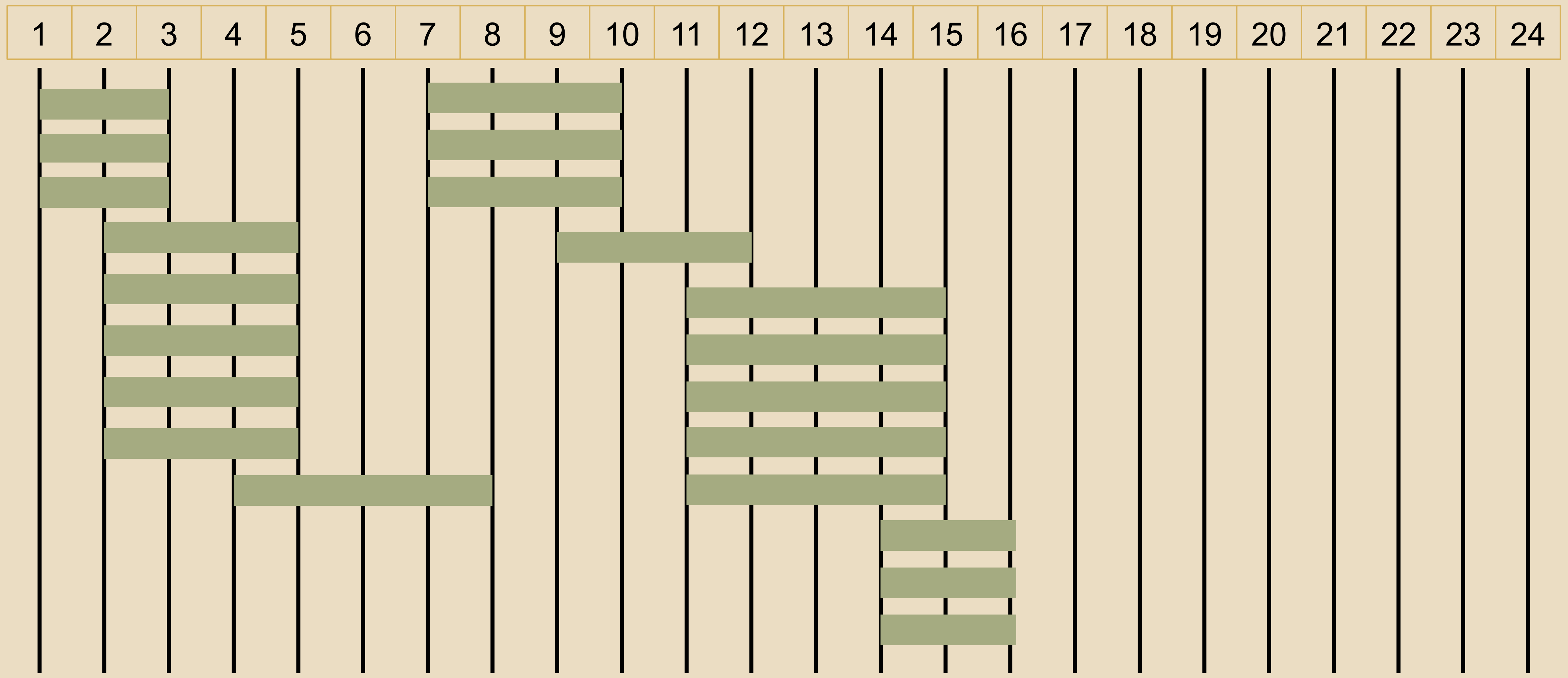
EARLIEST START TIME

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

FEWEST CONFLICTS

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

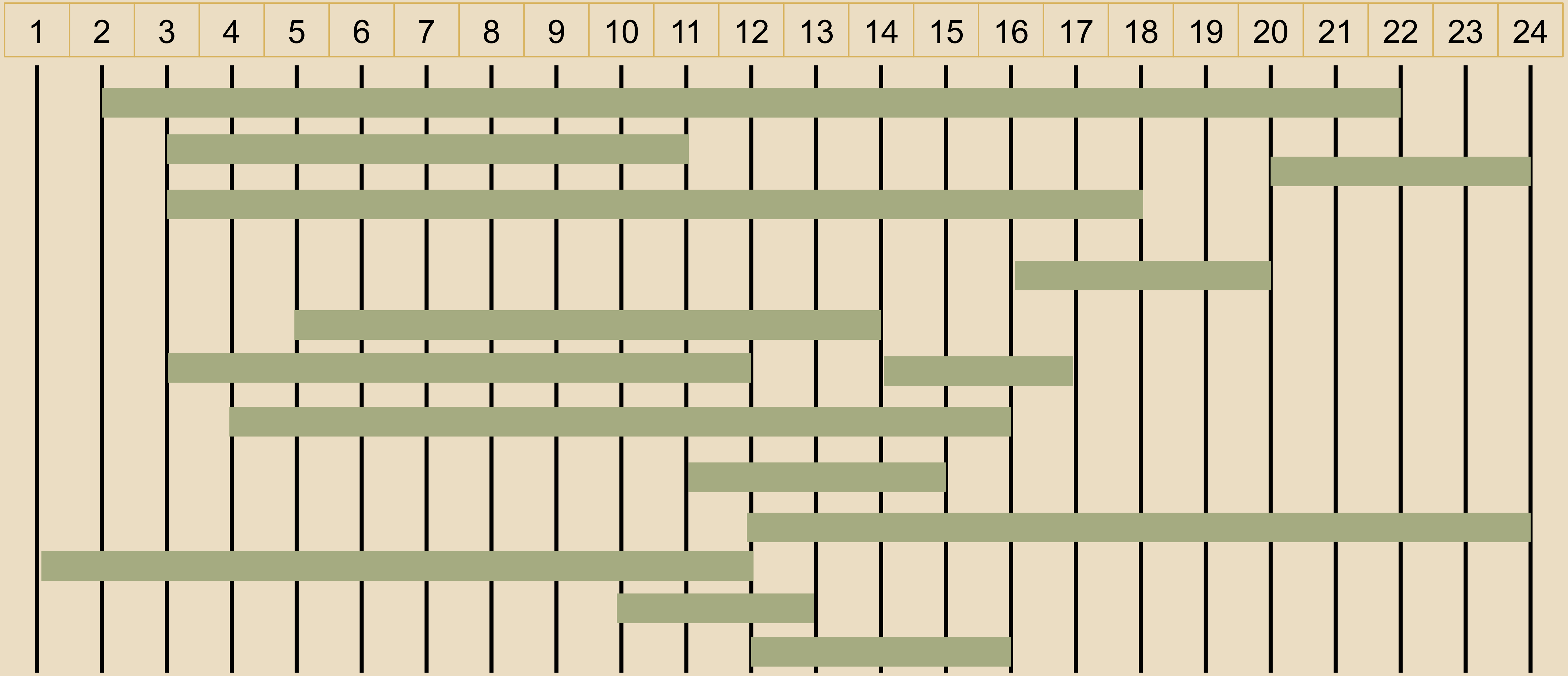
COUNTEREXAMPLE FOR FEWEST CONFLICTS



EARLIEST FINISH TIME

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

EARLIEST FINISH TIME



EVENT SCHEDULING

- Your goal is to schedule the most events possible that day such that no two events overlap.

- Exponential is too slow. Let’s try some greedy strategies:
  - ~~Shortest duration~~
  - ~~Earliest start time~~
  - ~~Fewest conflicts~~
  - Earliest end time (We can’t find a counterexample!!)

- Let’s try to prove it works!!!

PROVING OPTIMALITY

What does it mean for a greedy algorithm correctly solve a problem?

- I: problem instance
- GS: greedy solution to I
- OS: any other solution to I (for instance, an optimal solution)
- We need to show that GS is at least as good as OS.
- Tricky part: OS is an arbitrary solution. We don’t know much about it.

TECHNIQUES TO PROVE OPTIMALITY

We’ll see a number of general methods to prove optimality:

- Modify-the-solution, aka Exchange: most general
- Greedy-stays-ahead: often the most intuitive
- Greedy-achieves-the-bound: also used in approximation, LP, network flow
- Unique-local-optimum: dangerously close to a common fallacy

Which one to use is up to you.

STRATEGY: MODIFY-THE-SOLUTION

Don’t think about the entire greedy solution.
Just prove that: ***the first move of the greedy algorithm isn’t incorrect***.

General structure of modify-the-solution:

1. Prove an **Exchange/Modification Lemma**: There is an optimal solution that agrees with the greedy algorithm’s first decision.

2. Then use this as part of an inductive proof that the greedy solution is optimal.

STRATEGY: MODIFY-THE-SOLUTION

General structure of modify-the-solution:

1. Let g be the first choice the greedy algorithm makes.
2. Let OS be any solution that does not contain g.
3. Show how to transform OS into a different solution OS' that chooses g, and is at least as good as OS.
4. Use 1-3 in an inductive argument. OS_1 agrees with the first greedy choice, OS_2 the first two, and so on, until OS_t agrees with all choices, and

   Value(OS) ≤ Value(OS_1) ≤ Value(OS_2)….≤ Value($OS_t = GS$)

EARLIEST FINISH TIME

Let $E = \{E_1, \dots E_n\}$ be the set of all events with s_i, f_i the start and finish times of E_i.

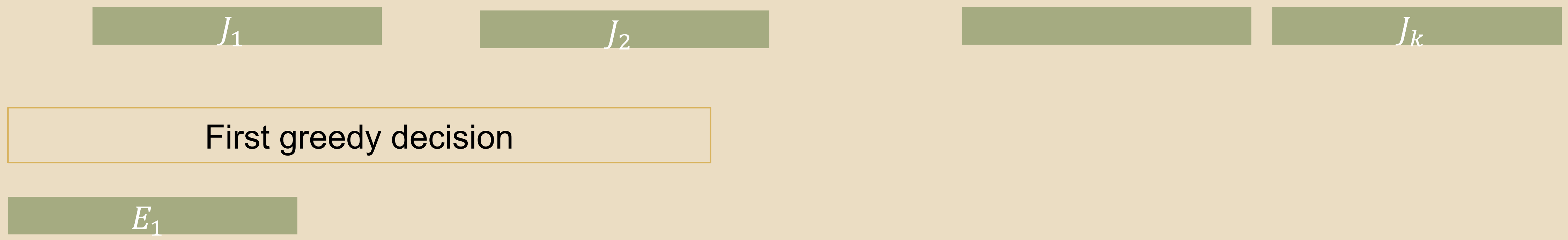
Say E_1 is the event with the earliest finish time.
The first greedy decision is to include E_1.

Modification Lemma: If OS is a legal schedule that does not include E_1 then there is a schedule OS' that does include E_1 such that $|OS'| \geq |OS|$.

- How to prove this?

MODIFY-THE-SOLUTION CONT.

OS:



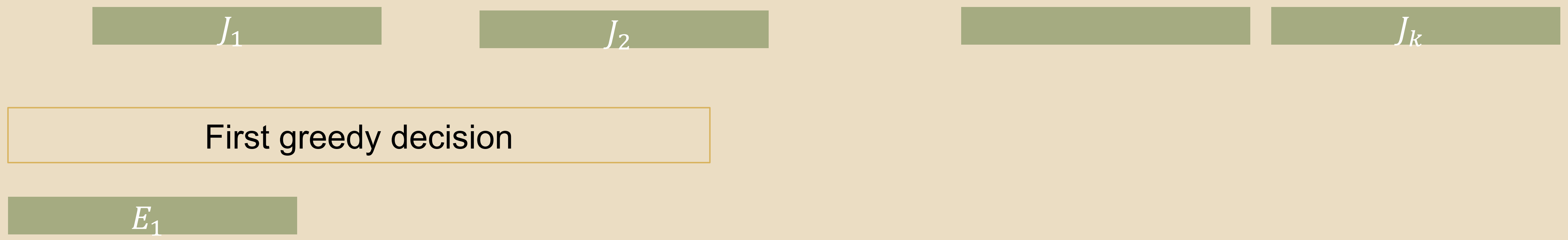
Agenda: define OS' such that

- OS' contains E_1
- OS' has no overlaps
- $|OS'| \geq |OS|$

$OS' = ???$

DEFINE OS'

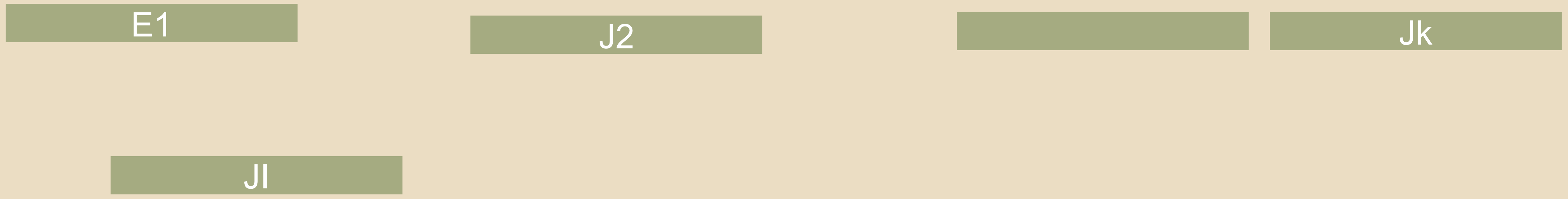
OS:



$$OS' = OS \cup \{E_1\} - \{J_1\}$$

OS' HAS NO OVERLAPS

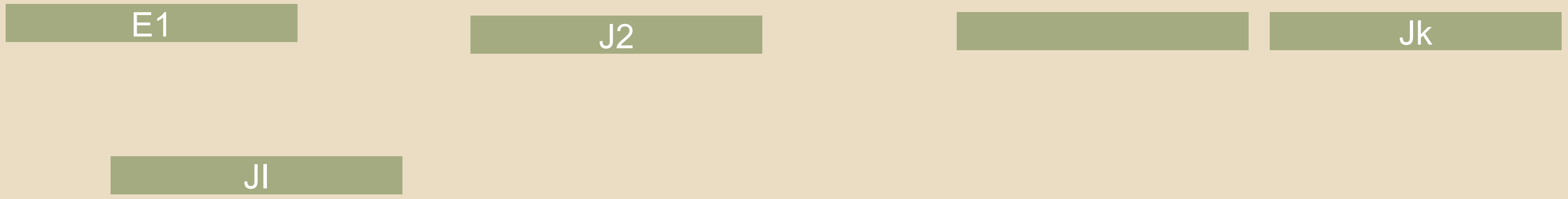
$OS' = OS \cup \{E_1\} - \{J_1\}$



Only new place for overlaps: we need to show Finish(E_1) ≤ Start(J_2)

OS' HAS NO OVERLAPS

$OS' = OS \cup \{E_1\} - \{J_1\}$

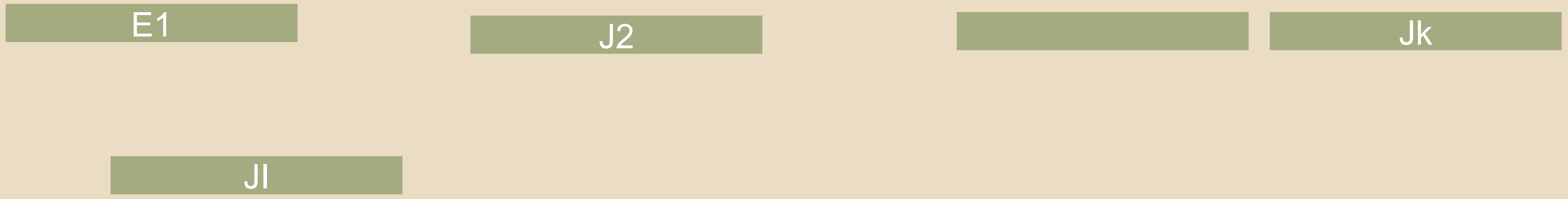


Only new place for overlaps: we need to show Finish(E_1) ≤ Start(J_2)

Finish(E_1) ≤ Finish(J_1) ≤ Start(J_2)

OS' IS AT LEAST AS GOOD AS OS

$OS' = OS \cup \{E_1\} - \{J_1\}$



$| OS' | = | OS |$

This completes the proof of the Modification Lemma: If OS is a legal schedule not containing E_1 then there is a schedule OS' containing E_1 such that $|OS'| \geq |OS|$.

INDUCTIVE PROOF OF CORRECTNESS

The greedy solution is optimal for every set of events.

Proof by strong induction on n, the number of events.

- Base Case: $n = 0$ or $n = 1$. Any choice works.

- General case: Assume greedy is optimal for any k events for $0 \leq k \leq n - 1$. Our goal is to show Greedy is optimal for any n events.

Let GS be the greedy solution. Then

GS = E_1 + GS(Events')

where Events' are the events that don't conflict with E_1.

Let OS be any other solution. Apply the Modification Lemma to OS to get OS', where

OS' = E_1 + Some solution for Events'

Applying the inductive hypothesis,

$|GS|$ = 1 + $|GS$(Events')$|$ ≥ 1 + |Some solution for Events'| = $|OS'|$ ≥ $|OS|$

GENERAL MTS TEMPLATE: MODIFICATION LEMMA

MODIFICATION LEMMA:

Let g be the first greedy decision. Let OS be any legal solution that does not pick g. Then there is a solution OS' that does pick g and OS' is at least as good as OS. (Note: we only use greedy to define g. Otherwise, GS does not directly appear).

GENERAL MTS TEMPLATE: PROOF OF LEMMA

MODIFICATION LEMMA:

Let g be the first greedy decision. Let OS be any legal solution that does not pick g. Then there is a solution OS' that does pick g and OS' is at least as good as OS.

- 1. State what we know: Definition of g. OS meets constraints.
- 2. Define OS' from OS, g
- 3. Prove that OS' meets constraints. Use 1, 2.
- 4. Compare value/cost of OS' to OS. Use 2, sometimes 1.

GENERAL MTS TEMPLATE: INDUCTION

MODIFICATION LEMMA: Let g be the first greedy decision. Let OS be any legal solution that does not pick g. Then there is a solution OS' that does pick g and S is at least as good as OS.

Using this Lemma, prove by induction on instance size that greedy is optimal.

Induction step:

- 1. Let g be first greedy decision. Let I' be the rest of problem given g.
- 2. GS = g + GS(I')
- 3. OS is any legal solution.
- 4. OS' is defined from OS by the Lemma (if OS does not include g).
- 5. OS' = g + some solution on I'.
- 6. Induction: GS(I') at least as good as some solution on I'.
- 7. GS is at least as good as OS', which is at least as good as OS.

EVENT SCHEDULING IMPLEMENTATION

Design an algorithm that uses the greedy choice of picking the next available event with the earliest finish time.

- Instance: n events each with a start and end time
- Solution format: List of events
- Constraints: Events can’t overlap
- Objective: Maximize the number of events

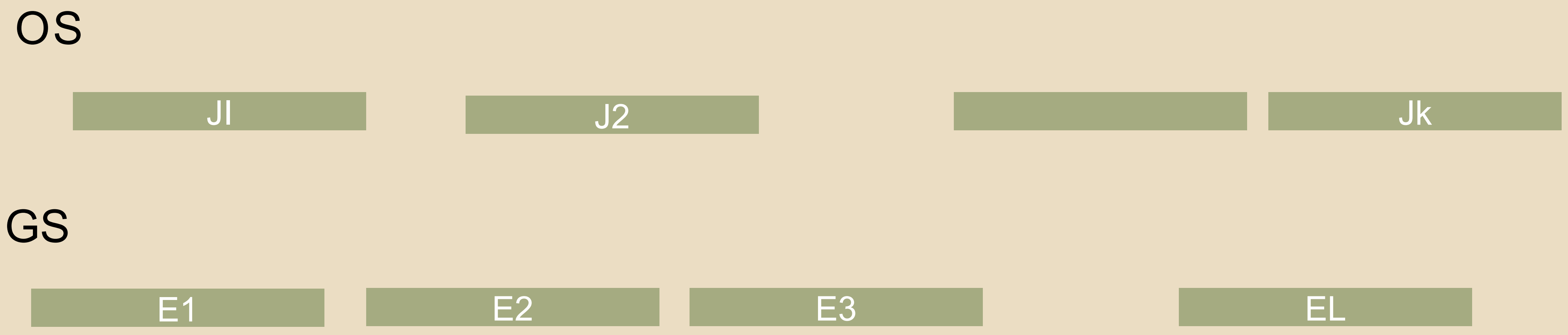
EVENT SCHEDULING

Design an algorithm that uses the greedy choice of picking the next available event with the earliest finish time.

- Initialize a Queue S
- Sort the intervals by finish time (let s_i, f_i be the start and finish times of E_i)
- Put the first event E_1 in S
- Set $F = f_1$
- For $i = 2 \ldots n$:
  - If $s_i \geq F$:
    - enqueue(E_i, S)
    - $F = f_i$
- Return S

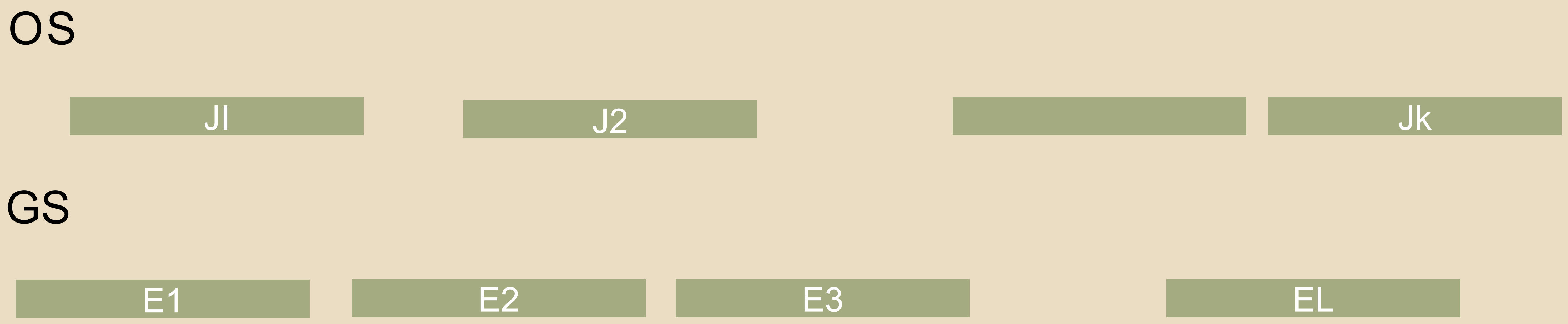
ANOTHER STRATEGY: GREEDY STAYS AHEAD

Compare all of GS to all of OS, instead of just first greedy move



Show GS is at least as good as OS, in some suitable sense, *every step of the way.*

GREEDY STAYS AHEAD



Claim: Finish(E_i) ≤ Finish(J_i)

Proof by induction on i. True for E_1, because it is the first to finish.
E_{i+1}: This is the interval starting after Finish(E_i) with the earliest end time.
J_{i+1} also begins after Finish(E_i), since Finish(J_i) ≥ Finish(E_i).
Therefore Finish(J_{i+1}) ≥ Finish(E_{i+1}).

GREEDY STAYS AHEAD: CONCLUSION

- Assume greedy weren't optimal, |GS| < |OS|.

- Let L = |GS|.
- By Lemma, Finish(E_L) ≤ Finish(J_L) ≤ Start(J_{L+1})
- Then greedy wouldn't end with E_L, contradiction.

GREEDY STAYS AHEAD: TEMPLATE

- Define a measure of progress.
- Order the decisions in OS to line up with GS.
- Prove by induction that the “progress” after the i’th decision in GS is at least as big as after the i’th decision in OS
- Conclude that GS is at least as good as OS.

COL351: Slides for Lecture Component 14

Thanks to Miles Jones, Russell Impagliazzo, and Sanjoy Dasgupta at UCSD for these slides.

EVENT SCHEDULING WITH MULTIPLE ROOMS

Suppose you have a conference to plan with n events and you have an unlimited supply of rooms. How can you assign events to rooms in such a way as to minimize the number of rooms?

Brute Force:

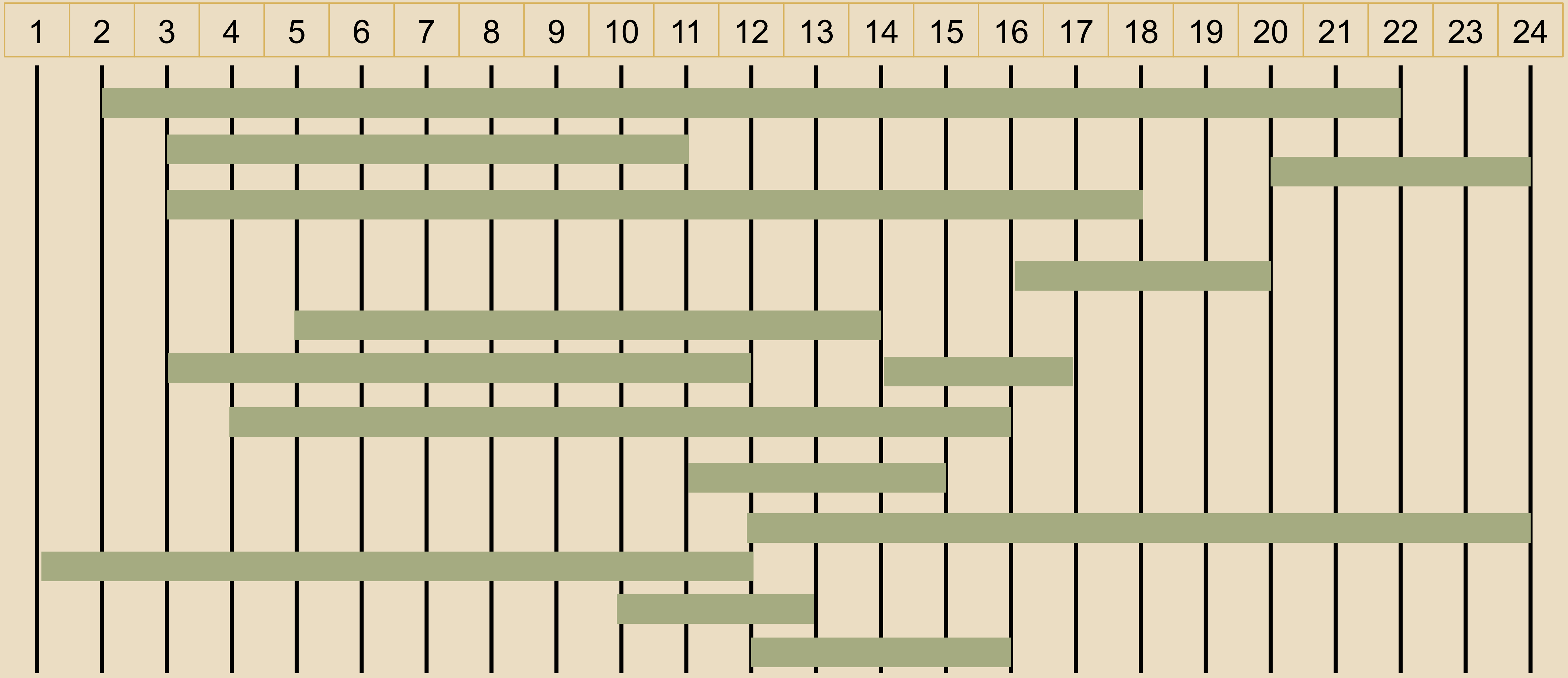
- Certainly you won't need more than n rooms.
- So how many ways can you assign n events to n rooms?

EVENT SCHEDULING WITH MULTIPLE ROOMS

Suppose you have a conference to plan with n events and you have an unlimited supply of rooms. How can you assign events to rooms in such a way as to minimize the number of rooms?

Ideas for a greedy algorithm?

EVENT SCHEDULING



EVENT SCHEDULING WITH MULTIPLE ROOMS

Suppose you have a conference to plan with n events and you have an unlimited supply of rooms. How can you assign events to rooms in such a way as to minimize the number of rooms?

- Greedy choice:
  - Number each room from 1 to n.
  - Sort the events by earliest start time.
  - Put the first event in room 1.
  - For events $2 \ldots n$, put each event in the smallest numbered room that is available.

TECHNIQUES TO PROVE OPTIMALITY

Some general methods to prove optimality:

- Modify-the-solution, aka Exchange: most general
- Greedy-stays-ahead: often the most intuitive
- Greedy-achieves-the-bound: also used in approximation, LP, network flow
- Unique-local-optimum: dangerously close to a common fallacy

Which one to use is up to you.

ACHIEVES-THE-BOUND

1. Logically determine a bound on the value of the solution that must be satisfied by any valid answer.

2. Then show that the greedy strategy achieves this bound and therefore is optimal.

ACHIEVES-THE-BOUND

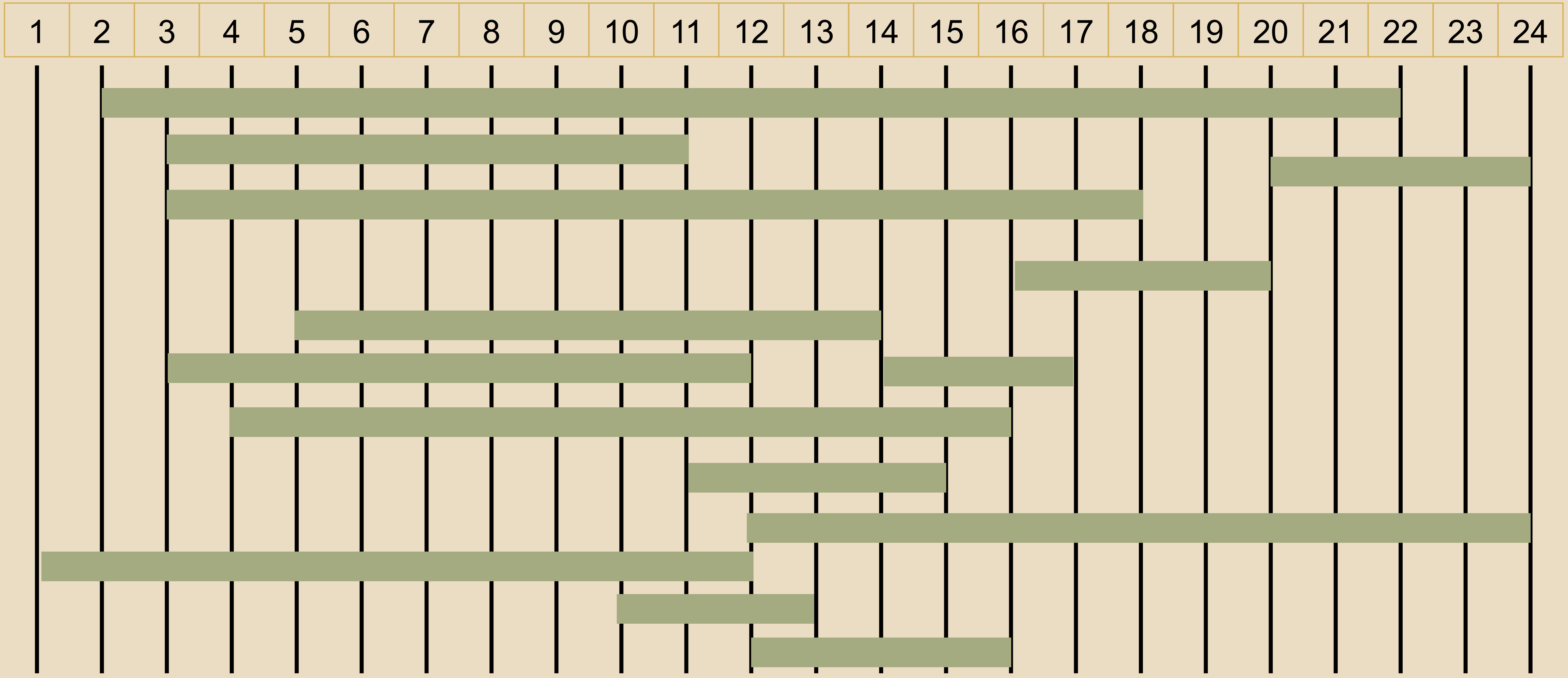
- Let t be any time during the conference.
- Let $B(t)$ be the set of events taking place at time t.

Bounding Lemma: Any valid schedule requires at least $|B(t)|$ rooms.

Proof:
There are $|B(t)|$ events taking place at time t.
They all need to be in different rooms.
So we need at least $|B(t)|$ rooms.

- Let $L = max(|B(t)|)$ over all t.
- Then L is a lower bound on the number of rooms needed.

EVENT SCHEDULING



ACHIEVES-THE-BOUND

Achieves-the-Bound Lemma: Let k be the number of rooms picked by the greedy algorithm. Then at some point t, $|B(t)| \geq k$. In other words there are at least k events happening at time t.

Proof:

Let t be the starting time of the first event to be scheduled in room k.

Then by the greedy choice, room k was the least number room available at that time.

This means at time t, there was an event happening in rooms room 1, room 2, ..., room $k - 1$. And plus an event happening in room k

Therefore $|B(t)| \geq k$.

CONCLUSION: GREEDY IS OPTIMAL

- Let GS be the greedy solution.
- Let OS be any other schedule.

- Let L = max |B(t)| over all t.

- By the Bounding lemma, Cost(OS) ≥ L.
- By the achieves-the-bound lemma, Cost(GS) = |B(t)| ≤ L for some t.
- Putting the two together, Cost(GS) ≤ Cost(OS).

ACHIEVES-THE-BOUND

The way it works:

- Argue that when the greedy solution **reaches its peak cost**, it reveals a **bound**.
- Then show this bound is also a lower bound on the cost of any other solution.
- So we are showing : Cost(GS) ≤ Bound ≤ Cost (OS)

This is a proof technique that does not work in all cases.