# Operations Research as Empirical Modeling

John Hooker
Carnegie Mellon University

June 2009

# OR as Modeling

- Current practice: research in OR is primarily about **algorithms**.
  - **Computational performance** is very important in research literature.

# OR as Modeling

- Current practice: research in OR is primarily about **algorithms**.
  - **Computational performance** is very important oin research literature.
- **Thesis:** OR should be primarily about **empirical modeling**.
  - Algorithmic work should be **auxiliary**.

## Modeling as Science?

- Conventional wisdom: Modeling is an **art**, not a science
  - Unlike algorithms.

# Modeling as Science?

- Conventional wisdom: Modeling is an **art**, not a science
  - Unlike algorithms.
- This is based on confusion.
  - **Creation** of models is an art.
  - Creation of **algorithms** is also an art!

# Modeling as Science?

- Conventional wisdom: Modeling is an **art**, not a science
  - Unlike algorithms.
- This is based on confusion.
  - **Creation** of models is an art.
  - Creation of **algorithms** is also an art!
- **Most of the sciences** build empirical models.
  - Quantum mechanics, thermodynamics, genetics, etc.
  - OR should do likewise.
  - OR models are **prescriptive**, as opposed to descriptive.

# Taking the Modeling Task Seriously

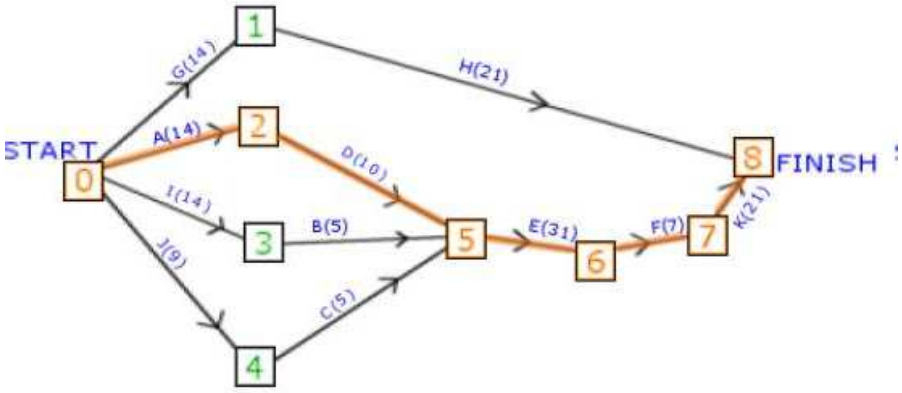
- An **empirical model** is a succinct and rigorous description of a natural phenomonon…
  - …from which one can deduce **empirically testable** consequences.
  - Usually written in a formal language (e.g., mathematics).
  - Should be **explanatory**.

# Taking the Modeling Task Seriously

- An **empirical model** is a succinct and rigorous description of a natural phenomonon…
  - …from which one can deduce **empirically testable** consequences.
  - Usually written in a formal language (e.g., mathematics).
  - Should be **explanatory**.
- **OR models** are rigorous and make testable predictions (e.g., optimal solutions).
  - But too often they are **not explanatory**.
  - For example, long list of inequality constraints.

# Empirical Modeling in OR

- One **can find** empirical modeling in OR.
- For example, **network flow** models.
  - They **elucidate** the problem.
  - Easy to deduce consequences and test the model.
- In particular, **critical path** models.
  - Very simple explanation of why projects get **behind schedule**.



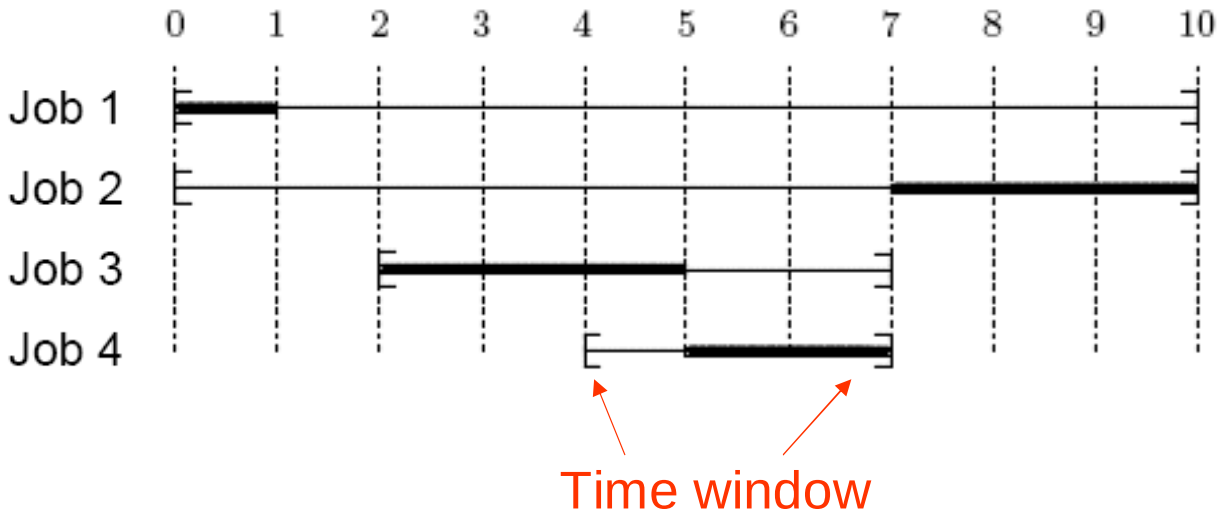
# Edge Finding for Disjunctive Scheduling

Schedule jobs so that they do not overlap in time.

| Job | Release time | Dead-line | Processing time |
|---|---|---|---|
| $j$ | $r_j$ | $d_j$ | $p_j$ |
| 1 | 0 | 10 | 1 |
| 2 | 0 | 10 | 3 |
| 3 | 2 | 7 | 3 |
| 4 | 4 | 7 | 2 |

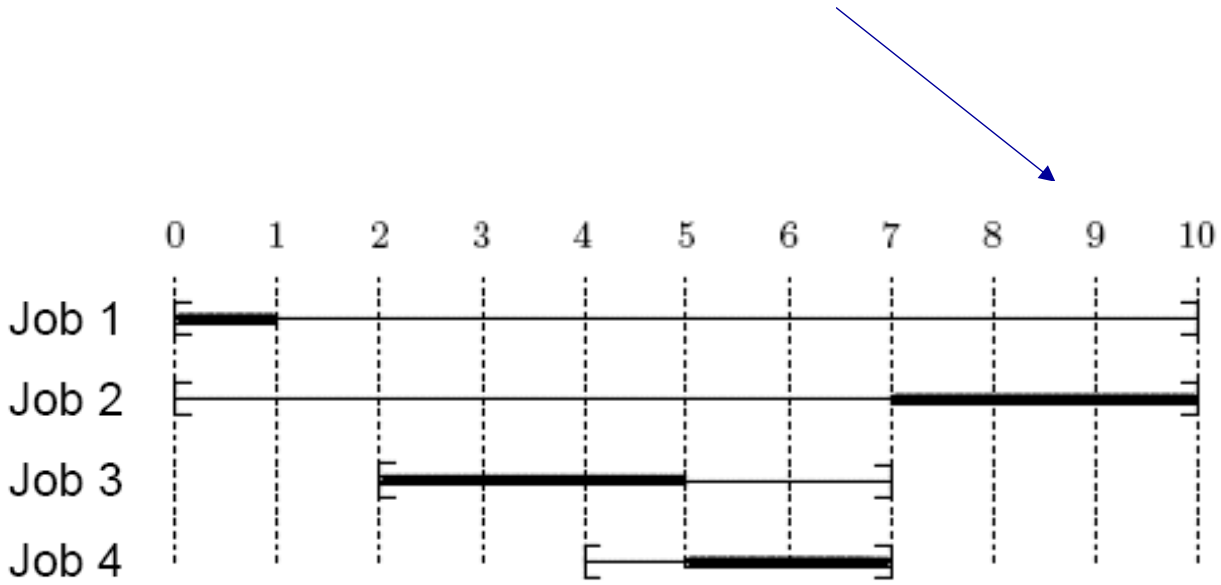
# Edge finding for disjunctive scheduling

A feasible (min makespan) solution:



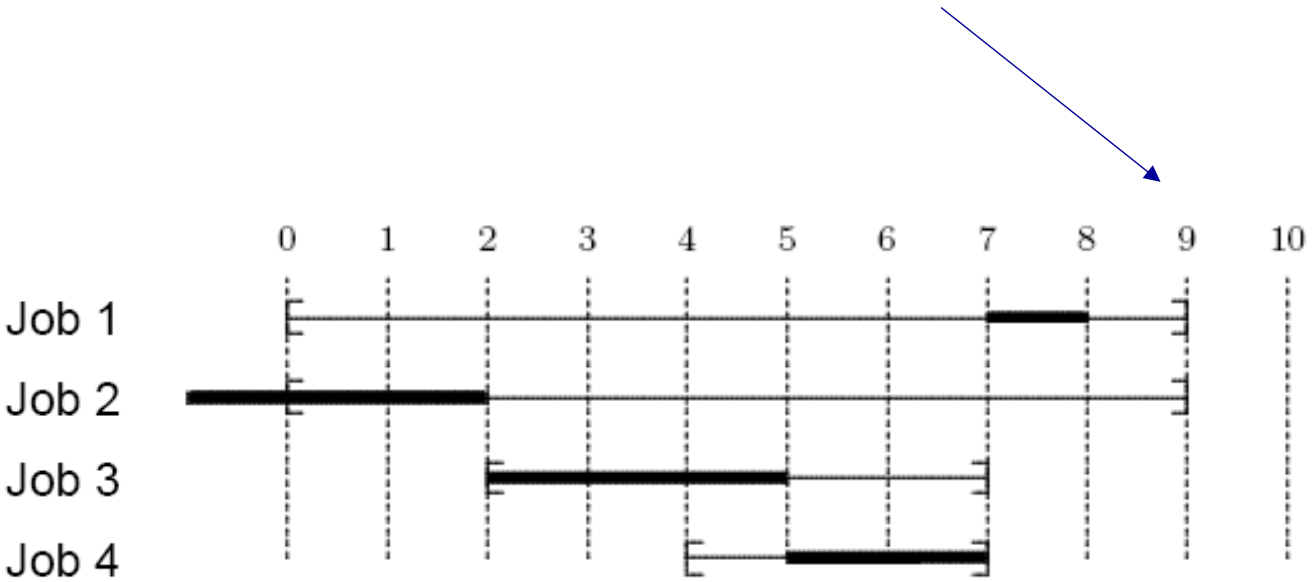
# Edge finding for disjunctive scheduling

But let's reduce 2 of the deadlines to 9:



# Edge finding for disjunctive scheduling

But let's reduce 2 of the deadlines to 9:



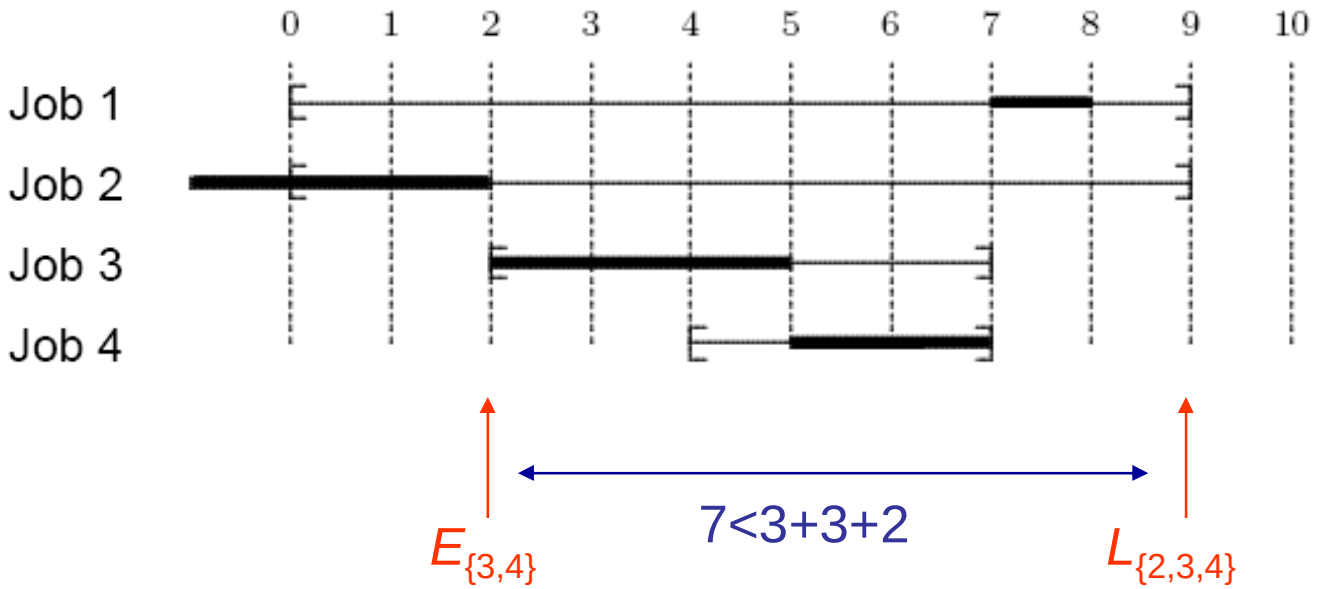

We will use edge finding to prove that there is no feasible schedule.

# Edge finding for disjunctive scheduling

We can deduce that job 2 must precede jobs 3 and 4: $2 \ll \{3,4\}$

Because if job 2 is not first, there is not enough time for all 3 jobs within the time windows:

$$L_{\{2,3,4\}} - E_{\{3,4\}} < p_{\{2,3,4\}}$$



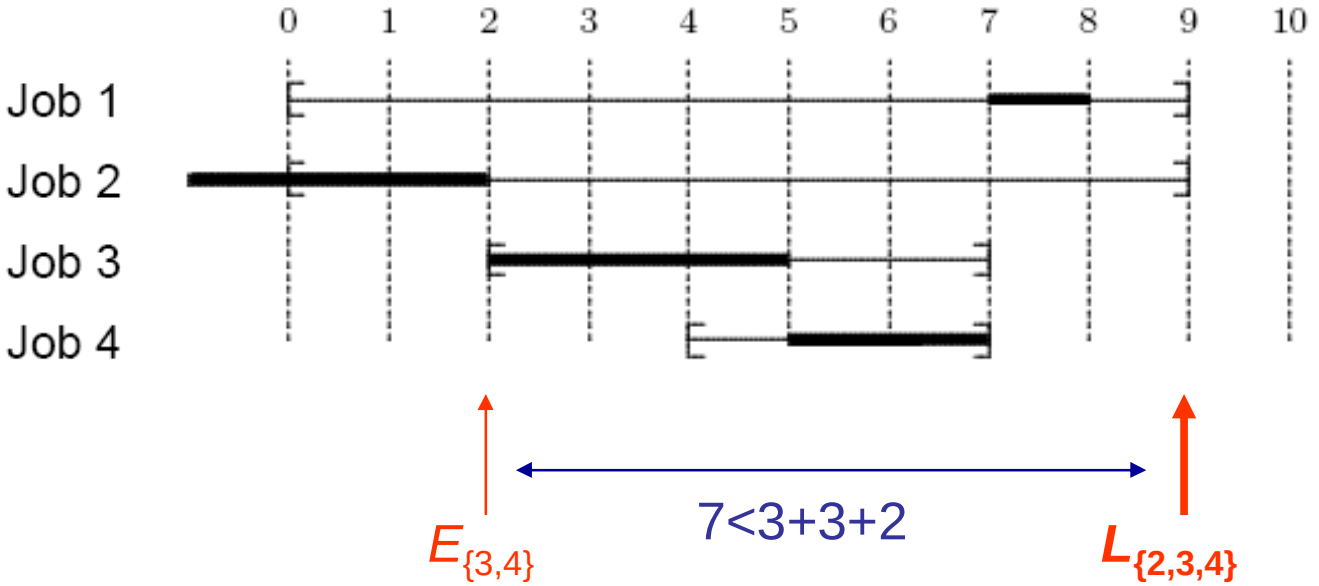
# Edge finding for disjunctive scheduling

We can deduce that job 2 must precede jobs 3 and 4: $2 \ll \{3,4\}$

Because if job 2 is not first, there is not enough time for all 3 jobs within the time windows:

$$L_{\{2,3,4\}} - E_{\{3,4\}} < p_{\{2,3,4\}}$$

Latest deadline



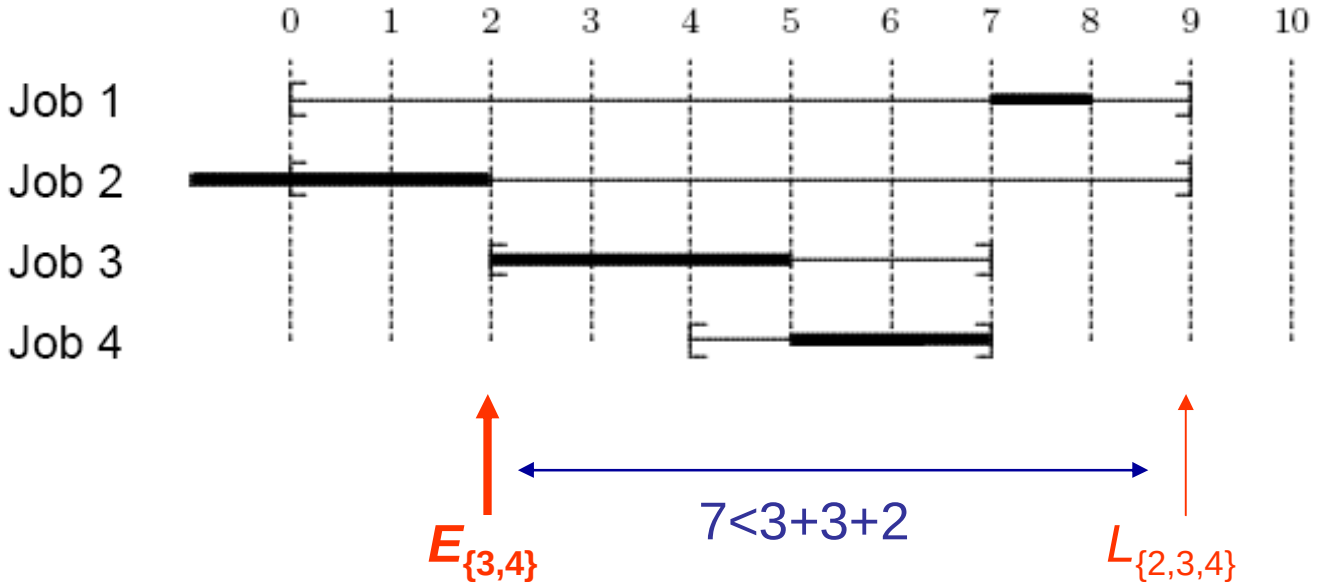
# Edge finding for disjunctive scheduling

We can deduce that job 2 must precede jobs 3 and 4: $2 \ll \{3,4\}$

Because if job 2 is not first, there is not enough time for all 3 jobs within the time windows:

$$L_{\{2,3,4\}} - E_{\{3,4\}} < p_{\{2,3,4\}}$$

Earliest release time



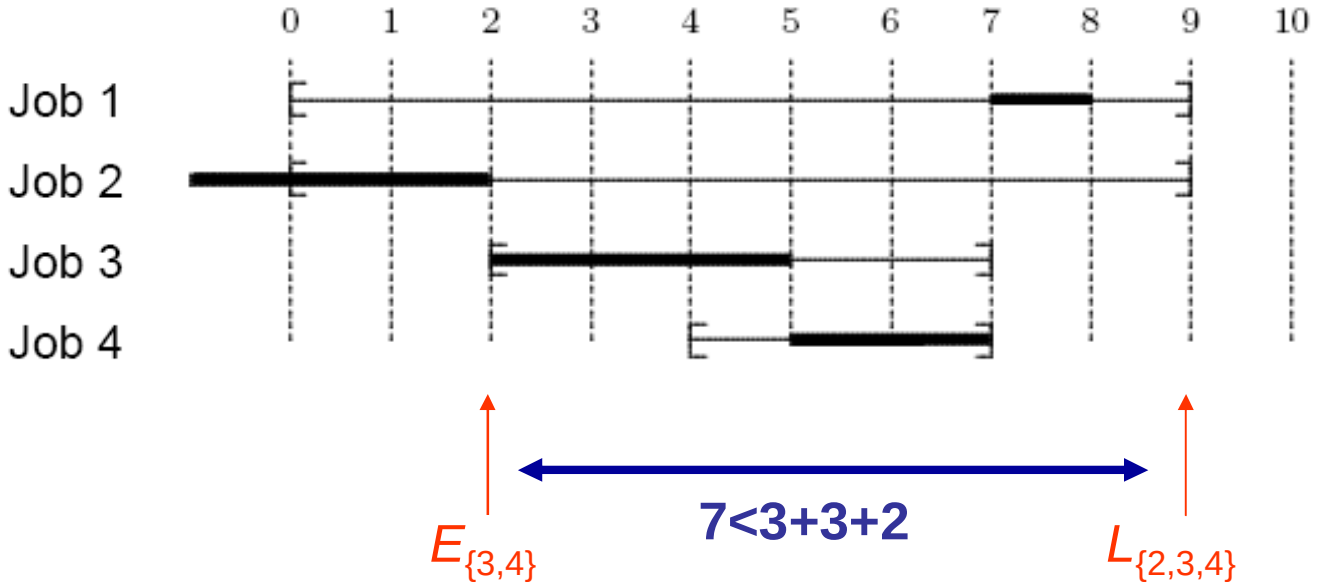
# Edge finding for disjunctive scheduling

We can deduce that job 2 must precede jobs 3 and 4: $2 \ll \{3,4\}$

Because if job 2 is not first, there is not enough time for all 3 jobs within the time windows:

$$L_{\{2,3,4\}} - E_{\{3,4\}} < p_{\{2,3,4\}}$$

Total processing time

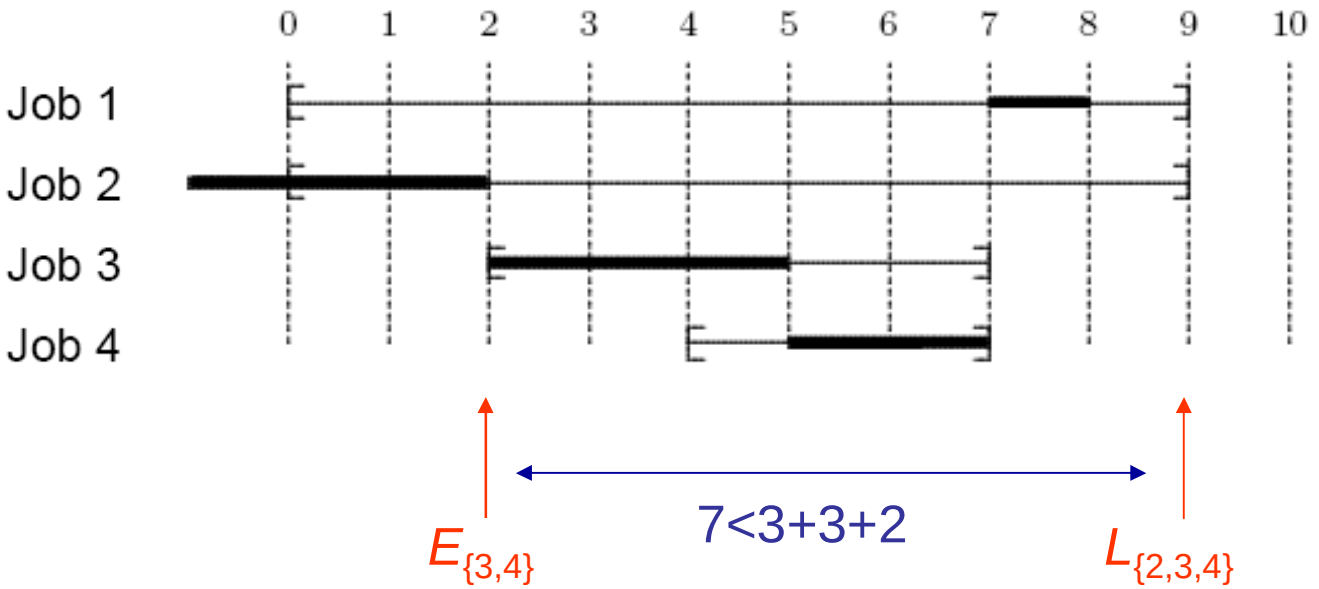


# Edge finding for disjunctive scheduling

We can deduce that job 2 must precede jobs 3 and 4: $2 \ll \{3,4\}$

So we can tighten deadline of job 2 to minimum of

$$L_{\{3\}} - p_{\{3\}} = 4 \qquad L_{\{4\}} - p_{\{4\}} = 5 \qquad L_{\{3,4\}} - p_{\{3,4\}} = 2$$



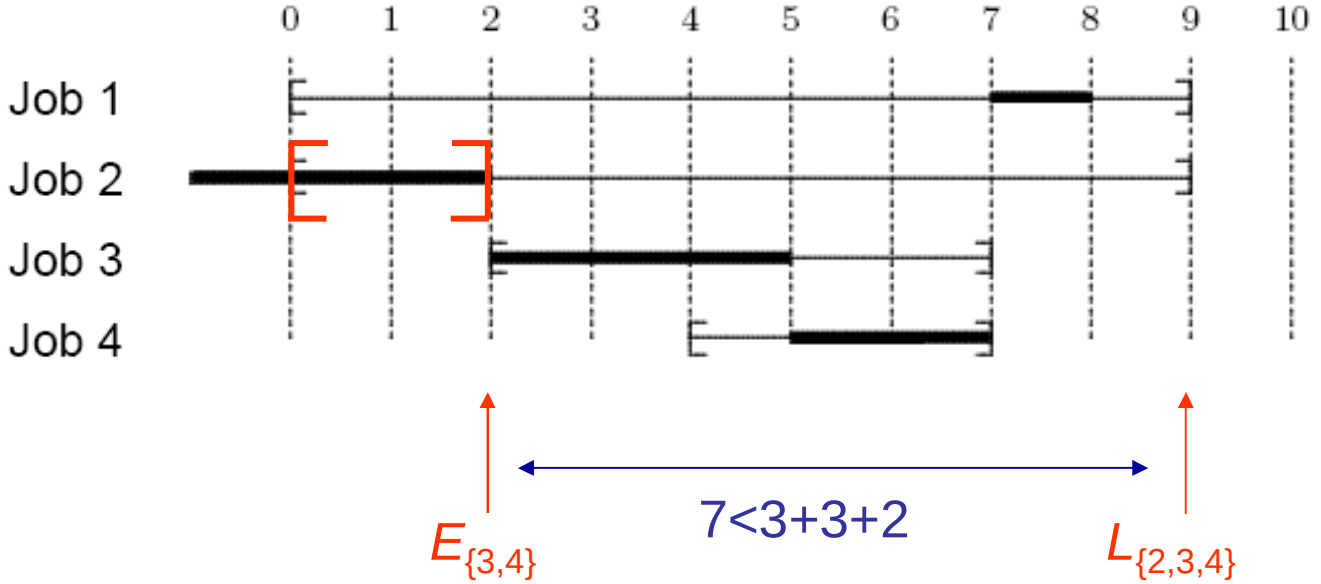
# Edge finding for disjunctive scheduling

We can deduce that job 2 must precede jobs 3 and 4: $2 \ll \{3,4\}$

So we can tighten deadline of job 2 to minimum of

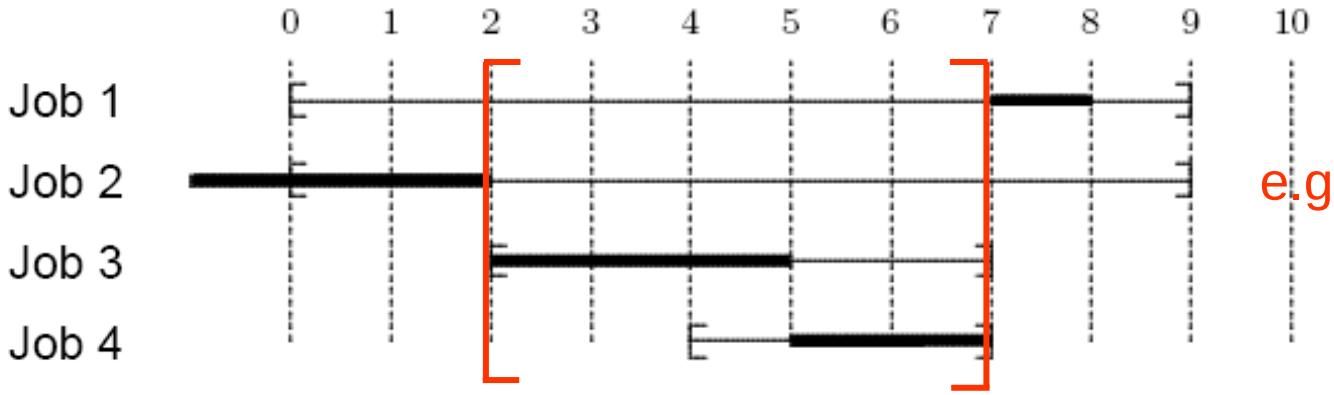
$$L_{\{3\}} - p_{\{3\}} = 4 \qquad L_{\{4\}} - p_{\{4\}} = 5 \qquad L_{\{3,4\}} - p_{\{3,4\}} = 2$$

Since time window of job 2 is now too narrow, there is no feasible schedule.



# Edge finding for disjunctive scheduling

**Key result:** We only have to consider sets $J$ whose time windows lie within some interval.



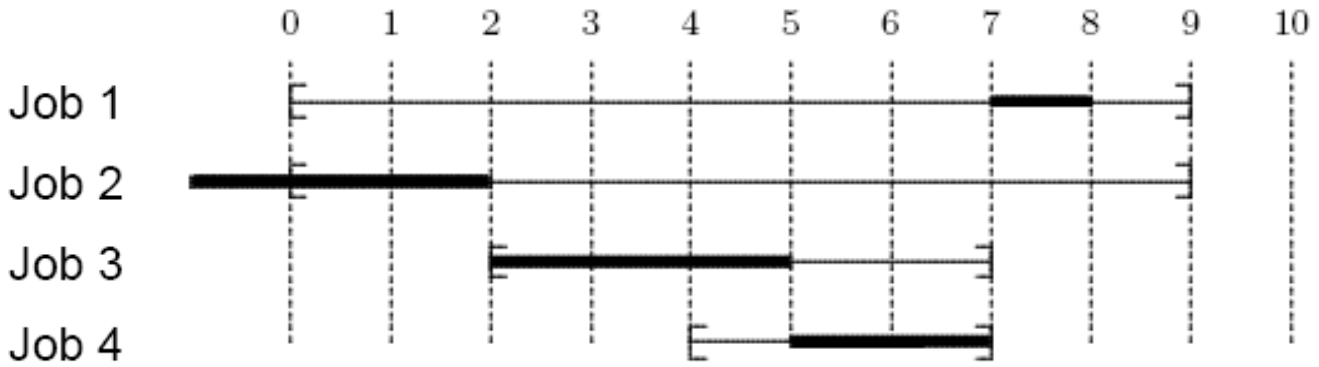

e.g., $J = \{3,4\}$

There are a polynomial number of intervals defined by release times and deadlines.

Updating of time windows can also be done in polytime.

# Edge finding for disjunctive scheduling

This not only allows us to identify some infeasible schedules, but provides some **understanding** of why there may be no feasible schedule.




…even though the theory is not complete.

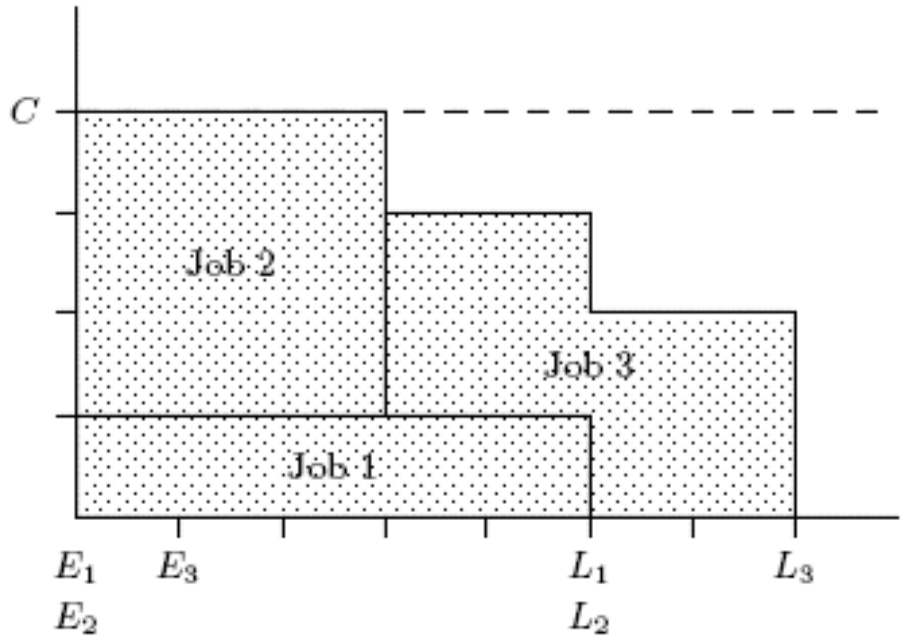
# Edge Finding for Cumulative Scheduling

Schedule jobs in parallel so that total rate of resource consumption is within the limit $C$.

Resource consumption rate

| $j$ | $p_j$ | $c_j$ | $E_j$ | $L_j$ |
|---|---|---|---|---|
| 1 | 5 | 1 | 0 | 5 |
| 2 | 3 | 3 | 0 | 5 |
| 3 | 4 | 2 | 1 | 7 |

A feasible solution:

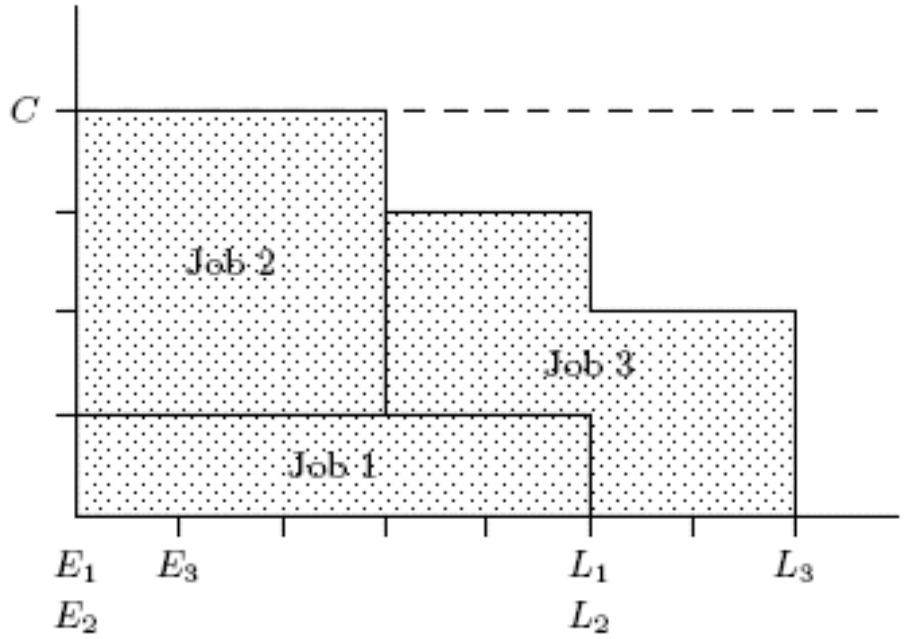


# Edge finding for cumulative scheduling

We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

Because the total **energy** required exceeds the area between the earliest release time and the later deadline of jobs 1,2:

$$e_3 + e_{\{1,2\}} > C \cdot \left( L_{\{1,2\}} - E_{\{1,2,3\}} \right)$$



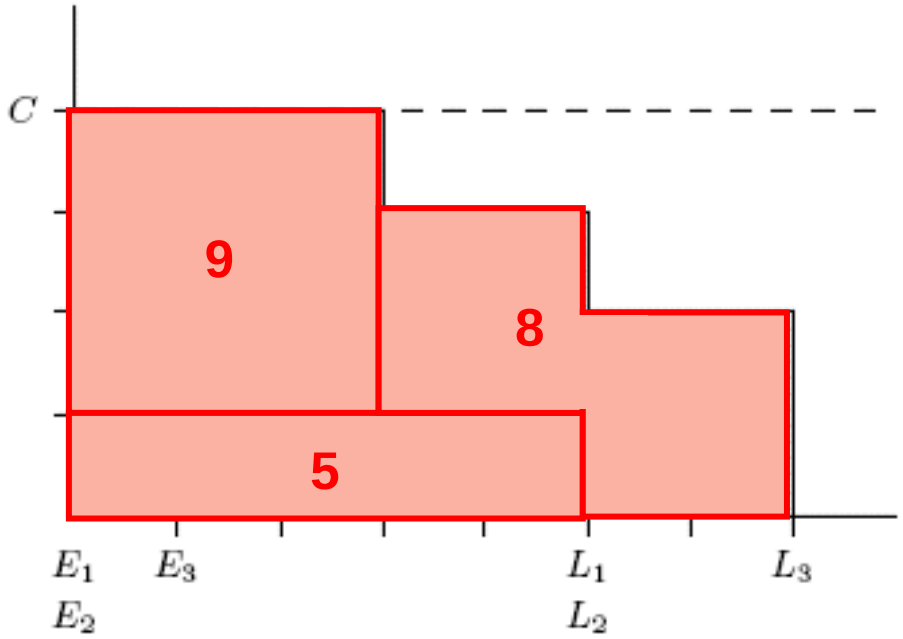
# Edge finding for cumulative scheduling

We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

Because the total **energy** required exceeds the area between the earliest release time and the later deadline of jobs 1,2:

$$e_3 + e_{\{1,2\}} > C \cdot \left( L_{\{1,2\}} - E_{\{1,2,3\}} \right)$$

Total energy required = 22



# Edge finding for cumulative scheduling

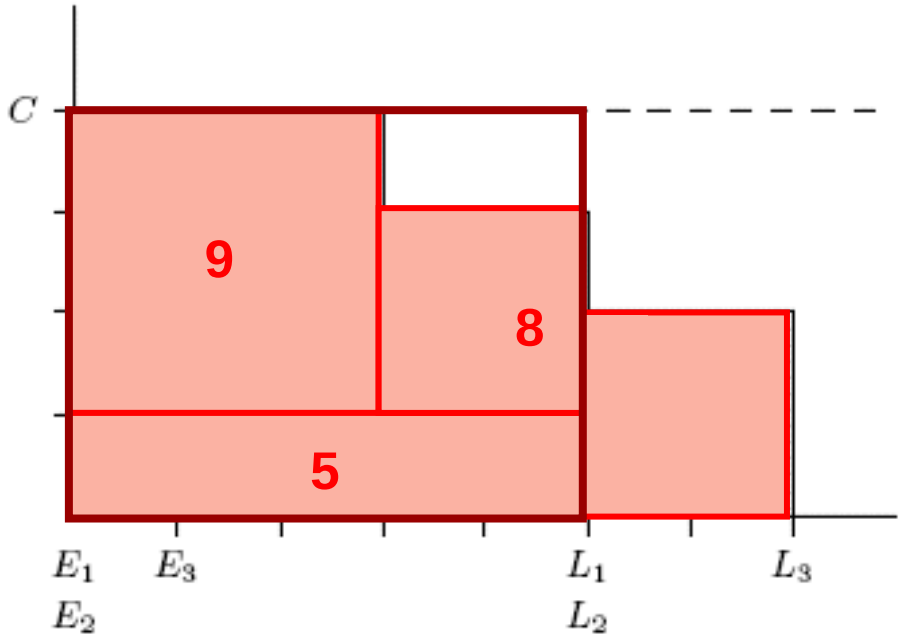
We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

Because the total **energy** required exceeds the area between the earliest release time and the later deadline of jobs 1,2:

$$e_3 + e_{\{1,2\}} > C \cdot \left( L_{\{1,2\}} - E_{\{1,2,3\}} \right)$$

Total energy required = 22

Area available = 20



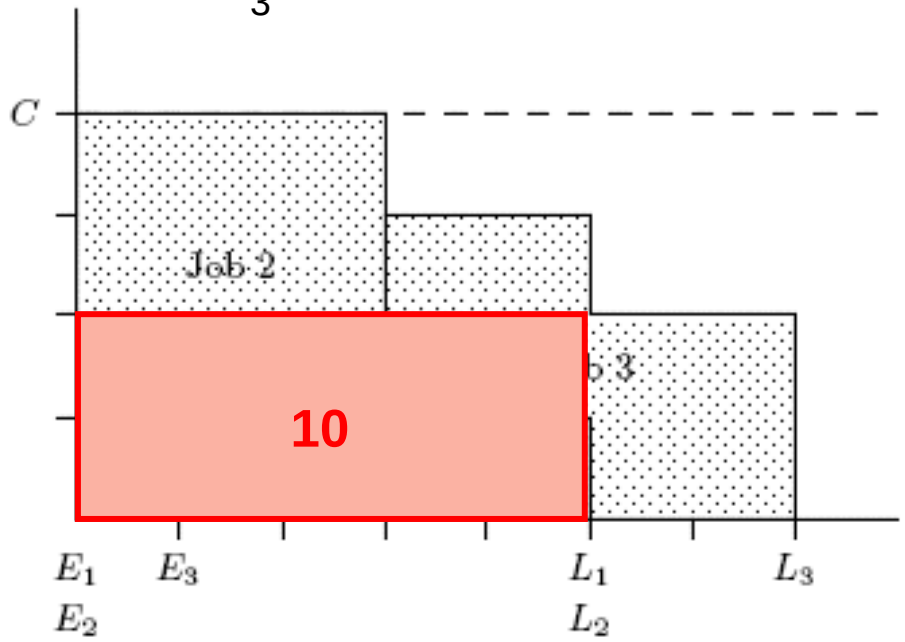
# Edge finding for cumulative scheduling

We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

We can update the release time of job 3 to

$$E_{\{1,2\}} + \frac{e_J - (C - c_3)(L_{\{1,2\}} - E_{\{1,2\}})}{c_3}$$

Energy available for jobs 1,2 if space is left for job 3 to start anytime = 10



# Edge finding for cumulative scheduling

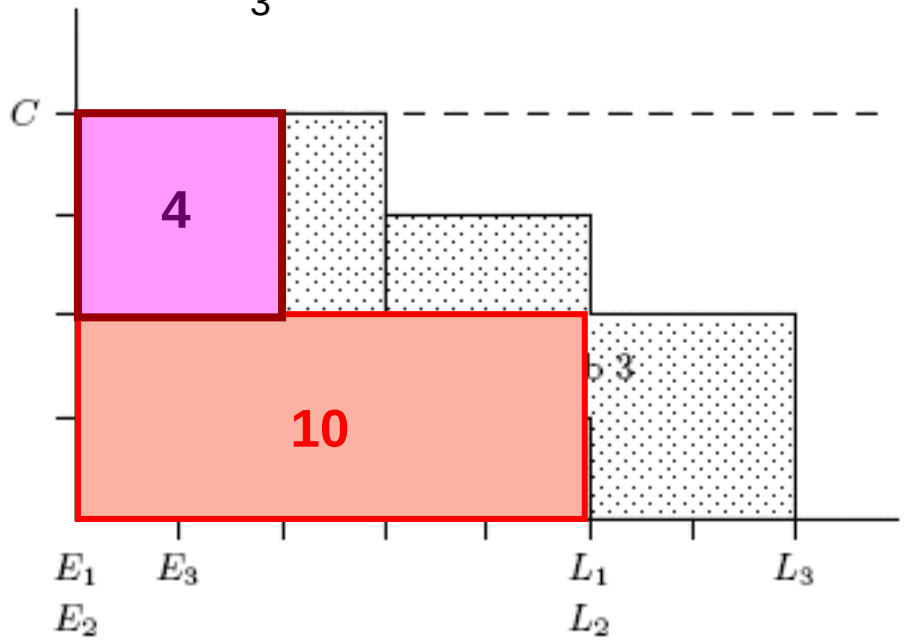
We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

We can update the release time of job 3 to

$$E_{\{1,2\}} + \frac{e_J - (C - c_3)(L_{\{1,2\}} - E_{\{1,2\}})}{c_3}$$

Energy available for jobs 1,2 if space is left for job 3 to start anytime = 10

Excess energy required by jobs 1,2 = 4



# Edge finding for cumulative scheduling

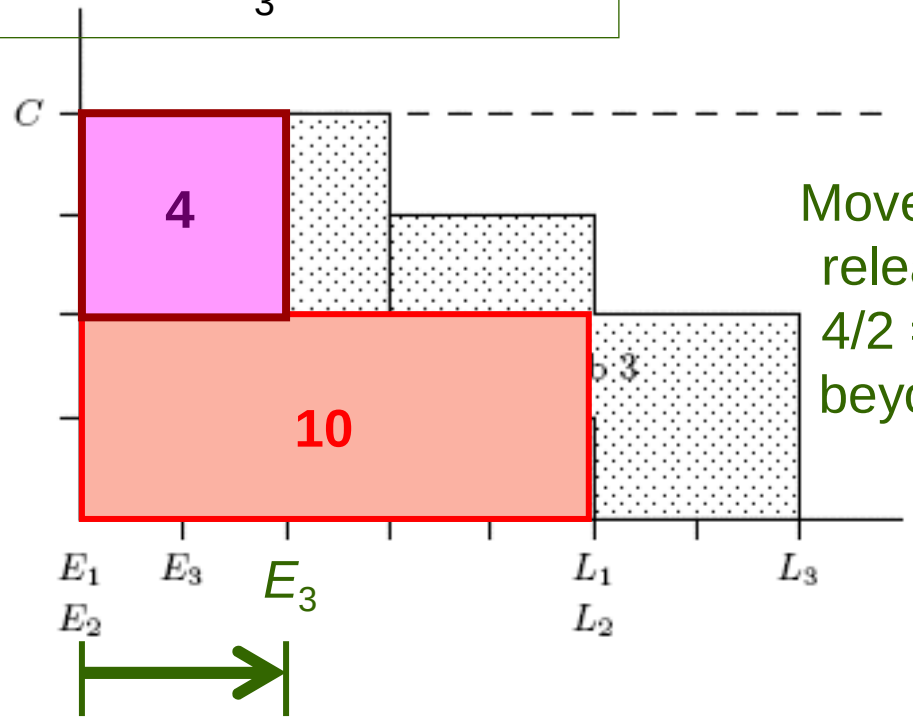
We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

We can update the release time of job 3 to

$$E_{\{1,2\}} + \frac{e_J - (C - c_3)(L_{\{1,2\}} - E_{\{1,2\}})}{c_3}$$

Energy available for jobs 1,2 if space is left for job 3 to start anytime = 10

Excess energy required by jobs 1,2 = 4




Move up job 3 release time 4/2 = 2 units beyond $E_{\{1,2\}}$

# Edge finding for cumulative scheduling

- There is an $O(n^2)$ algorithm that finds all applications of the edge finding rules.

# Edge finding for cumulative scheduling

- There is an $O(n^2)$ algorithm that finds all applications of the edge finding rules.
- The concept of **energy** of provides a (partial) **explanation** of why there may be no feasible cumulative schedule.

# Edge finding for cumulative scheduling

- There is an $O(n^2)$ algorithm that finds all applications of the edge finding rules.
- The concept of **energy** of provides a (partial) **explanation** of why there may be no feasible cumulative schedule.
- Other propagation rules for cumulative scheduling use the concept of energy.
    - Extended edge finding.
    - Timetabling.
    - Not first/not last rules.
    - Energetic reasoning.

# Subject Matter of OR Models

- Chemistry studies molecules, biology studies living things.
  - What does OR study?

## Subject Matter of OR Models

- Chemistry studies molecules, biology studies living things.
  - What does OR study?
- OR studies the **formal structure of human activities**.
  - Scheduling, production planning, logistics, queuing, product design, location decision, etc., etc.

# OR and Mathematics

- OR is **not mathematics**.
  - Mathematics operates at a higher level of abstraction but obtains fewer results.
  - OR **uses** mathematics.
  - Just as study of heat transfer uses differential equations.

# OR and Mathematics

- OR is **not mathematics**.
  - Mathematics operates at a higher level of abstraction but obtains fewer results.
  - OR **uses** mathematics.
  - Just as study of heat transfer uses differential equations.
- OR is **not applied mathematics**.
  - It does not simply apply mathematical theory.
  - It requires **theory formation** of its own.
  - It studies scheduling, planning, logistics, and design, not directed graphs, polyhedra, and other mathematical objects.

# OR and Mathematical Programming

- Let's grant that **mathematical programming** is mathematics.
  - Linear programming studies how polyhedra relate to linear functionals.
- Then mathematical programming is **not** a branch of OR.
  - It is a **tool** for OR – one that may be developed by OR people.
  - As when Newton developed fluxions (differential calculus) for mechanics.

# OR and Mathematical Programming

- Linear programming as developed by Dantzig/von Neumann is **OR as well as mathematics**.
  - Mathematics, yes – linear algebra, Farkas lemma, etc.
  - But it also explains economic/operational phenomena (reduced costs, basic solutions), 2-person games, etc.

## Role of Computation

- Why the preoccupation with computational issues?
  - Perhaps partly because it is a convenient gatekeeper.
  - It is the way we compute consequences of a model.
  - Granted, it is particularly crucial in OR, as in fluid mechanics.
- Yet computation should be the **servant** and not the master.

# OR as Empirical Science

- We should **reject** models that are **disconfirmed** in practice.
  - We rarely speak of a model as being “false.”
  - This is because we have forgot what modeling is all about.
  - An OR model can be false (fails to described reality) as an ecological model of the Indian Ocean can be false.

# OR as Empirical Science

- We should **reject** models that are **disconfirmed** in practice.
  - We rarely speak of a model as being “false.”
  - This is because we have forgot what modeling is all about.
  - An OR model can be false (fails to described reality) as an ecological model of the Indian Ocean can be false.
- A scheduling model is disconfirmed if schedules that it says are feasible cannot be implemented in practice.
  - Perhaps because setup times are sequence-dependent, which calls for a different model.
  - The aim is to **explain what is going on** at a structural level when people schedule activities.

# OR as Empirical Science

- Empirical ≠ nontheoretical.
  - Consider quantum electrodynamics.

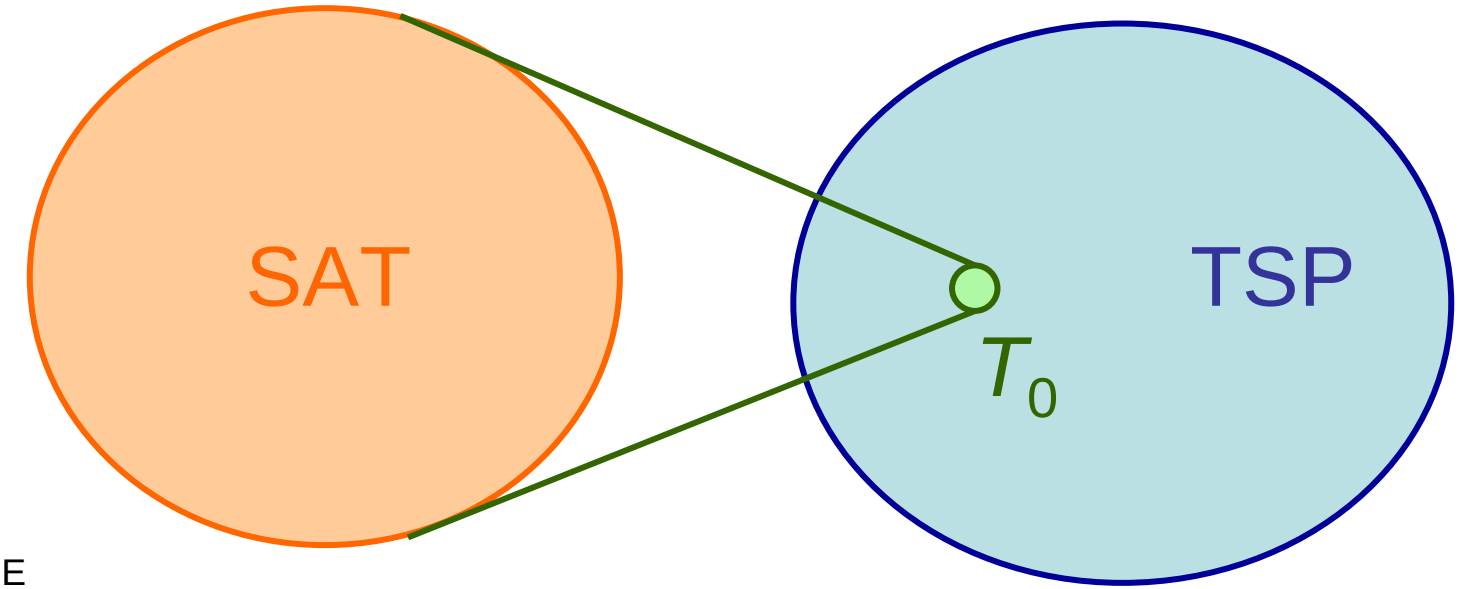
# OR as Empirical Science

- Empirical ≠ nontheoretical.
    - Consider quantum electrodynamics.
- An OR theory (model) is **intellectually interesting** to the extent that is explains our experience.
    - This is true even for the theory of computation, such as **NP-completeness theory**.
    - It is interesting only to the extent that it is **empirical**.

# NP-Completeness Theory

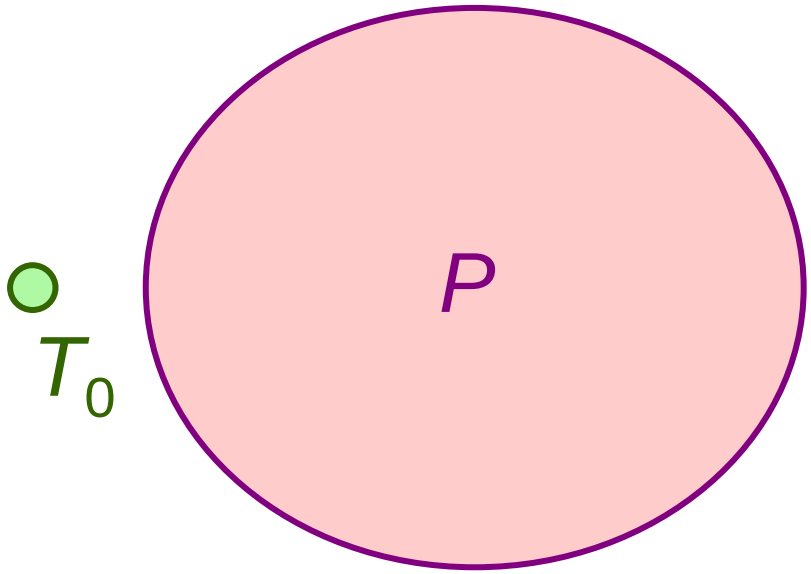
• TSP is judged to be **NP-complete** because there is a polynomial reduction of SAT to a very special class ($T_0$) of TSP instances.

  • On what grounds do we say that TSP instances are **characteristically** hard?



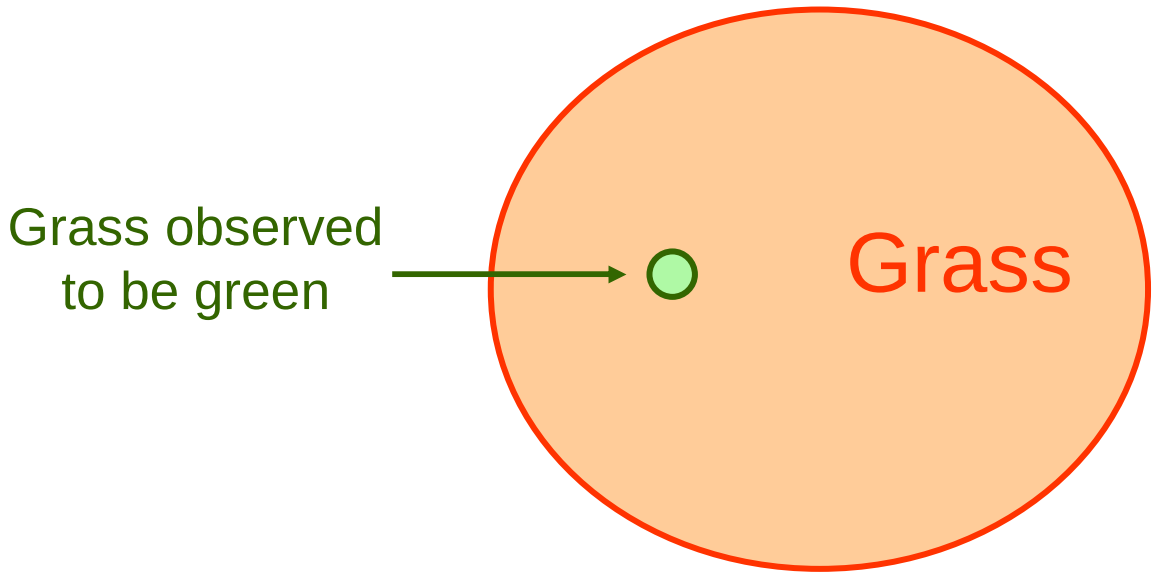
# NP-Completeness Theory

- We could just as well as that problems in $T_0 \cup P$ are characteristically hard!
  - $P$ = set of all easy (polynomially soluble) problems.



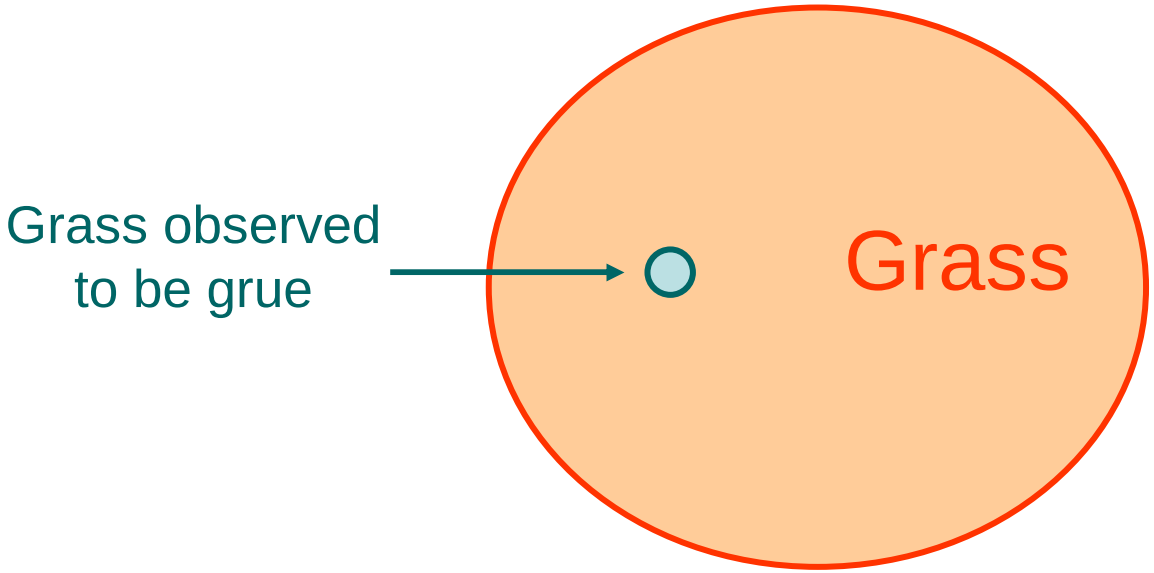
# NP-Completeness Theory

- We must show that TSP is a **natural kind**.
- **Green** is a natural kind.
  - Observing some green grass is evidence that grass is characteristically green.



# NP-Completeness Theory

- We must show that TSP is a **natural kind**.
- **Grue** (= green if date is prior to 2015, blue otherwise) is not a natural kind.
  - Observing some grue grass does not show that grass is characteristically grue.



## NP-Completeness Theory

- We know **empirically** that green is a natural kind.
  - In fact, physical theories help explain **why** it is a natural kind.

## NP-Completeness Theory

- We know **empirically** that green is a natural kind.
  - In fact, physical theories help explain **why** it is a natural kind.
- We know **empirically** that TSP is a natural kind (if we know it at all).
  - In computational experience, TSP instances outside $T_0$ tend to get hard as they scale up.
  - However, NP-completeness theory doesn't explain **why** TSP is a natural kind.
  - We need a more adequate empirical theory of tractability.

# Prescriptive Modeling

- Scientific modeling is normally **descriptive**.
  - Variables refer to natural phenomena.
- OR modeling is **prescriptive**.
  - Variables can refer to human **decisions**.

# Prescriptive Modeling

- Scientific modeling is normally **descriptive**.
  - Variables refer to phenomena.
- OR modeling is **prescriptive**.
  - Variables can refer to human **decisions**.
- This may have led OR away from explanatory modeling and postoptimality analysis.
  - Focus on recommendations for the client.
  - But to make responsible decisions, the client must understand the problem, too.

## Computational Technique

- OR should be defined by the **phenomenon it studies**…
  - rather than its computational techniques.

## Computational Technique

- OR should be defined by the **phenomenon it studies**…
  - rather than its computational techniques.
- **Practical** reasons not to define OR by its techniques:
  - Resistance to new techniques (cf. control theory).
  - No one understands what the field is about.

## Computational Technique

- OR should be defined by the **phenomenon it studies**…
  - rather than its computational techniques.
- **Practical** reasons not to define OR by its techniques:
  - Resistance to new techniques (cf. control theory).
  - No one understands what the field is about.
- **Fundamental** reason not to define OR by its techniques:
  - A field is energized by a **mystery**, not the techniques it uses to probe the mystery.

## OR's Task

- OR does not address perennial questions.
  - …as do physics, biology, cosmology.

# OR's Task

- OR does not address perennial questions.
  - …as do physics, biology, cosmology.
- But OR has the urgent task of understanding **complex human activities**.
  - Agriculture, manufacturing, transport, waste disposal, security, health care, information, financial services, justice.
  - They are increasingly unstable and unjust.
  - We must take charge of these systems and make them work.
  - OR must provide **understanding** as well as tools.

## Implications for the Field

- Publication.
  - Main criterion should be explanatory value rather than speed of the algorithm.
  - Publish enlightening analysis even when there is no algorithm.

## Implications for the Field

- Publication.
  - Main criterion should be explanatory value rather than speed of the algorithm.
  - Publish enlightening analysis even when there is no algorithm.
- Ancillary role of computation.
  - …although structural analysis can be a key to fast computation.
  - Mathematical proof, etc., continues to be important.

# Implications for the Field

- Benchmark problems.
  - Distinguish modeling from problem statement.
  - Get rid of problem sets like MIPLIB.
  - Replace the MIP models with problem statements.

# Implications for the Field

- Benchmark problems.
  - Distinguish modeling from problem statement.
  - Get rid of problem sets like MIPLIB.
  - Replace the MIP models with problem statements.
- Modeling.
  - Create perspicuous models.
  - Get rid of unreadable LP, MIP models.
  - Software tools should reflect centrality of modeling.
  - Get rid of formal modeling language as user interface.

# Implications for the Field

- Solving
  - Support **interaction of modeling and solution** to exploit problem structure.
  - Emphasize **postoptimality** and **what-if** analysis.
  - Use integrated solvers/modeling systems.
  - Get rid of individual IP solvers, LP solvers, CP solvers, GO solvers, etc. etc.
  - Domain-specific solvers are OK.

# Implications for the Field

- Solving
  - Support **interaction of modeling and solution** to exploit problem structure.
  - Emphasize **postoptimality** and **what-if** analysis.
  - Use integrated solvers/modeling systems.
  - Get rid of individual IP solvers, LP solvers, CP solvers, GO solvers, etc. etc.
  - Domain-specific solvers are OK.
- Research direction.
  - Deeper understanding of wider variety of operational problems.