IO543: Scheduling

Lecture 2

Brian Denton
Associate Professor, IOE

IOE Bld 2893
btdenton@umich.edu

Reminder

- Assignment # 1 due January 19

First Class Survey: Interests

Manufacturing systems
Supply Chain Management
Healthcare
Energy
Warehouse scheduling
Project scheduling
Parallel Machine Scheduling
Sports
Airlines
How to classify problems
Theory behind scheduling
Maintenance problems

Computer Science
Surgery Scheduling
Patient flow problems
Shipbuilding and Design
Military applications
Renewable Energy
Stochastic Scheduling
Using AMPL to model and solve scheduling problems
Transportation
Petroleum
Shift Scheduling

First Class Survey: Questions

Q. Will there be an in-class assignment every class? No

Q. Will this course require programming? Yes

Q. Will this course teach programming? No

Q. What programming languages do we need to know? Matlab and AMPL

Q. I have no Matlab or AMPL knowledge, can I take this course? Yes

Q. Is the course application oriented or theoretical? Yes

Q. Is the final exam cumulative? Can we bring a “cheat sheet”? Yes

Q. Are the assignments collaborative? No

Q. Can the homework assignments be handwritten? Yes

First Class Survey: Questions

Q. How much of IOE316 will we use? Markov Processes? Some

Q. How much linear programming will there be in the course? A lot

Q. Are the supplemental references on integer programming and dynamic programming available online? Yes

Linear Programming by Robert Vanderbei, available online through UM Library:

Introduction to Dynamic Programming by Dreyfus and Law (ctools)

Q. What is the most challenging part of this course? Depends on your background

Q. Are the midterm and final exam dates in the schedule firm? Yes

Today

Goals:

- Practice classifying scheduling problems
- Introduction to *Complexity*

Classification

Scheduling problems are described by a triplet:

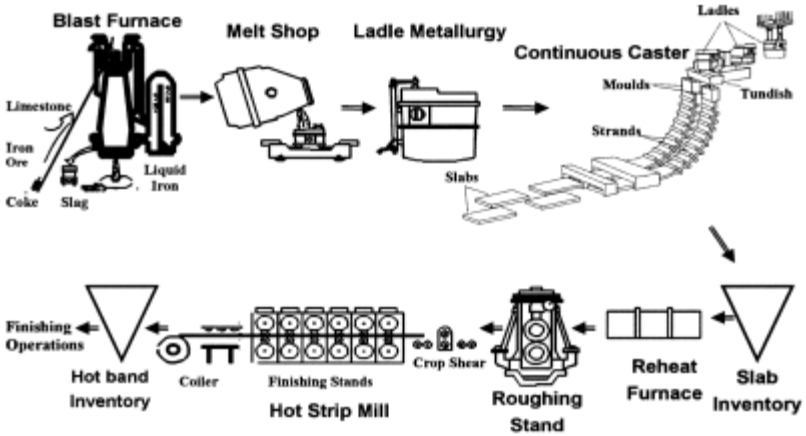
$$\alpha|\beta|\gamma$$

α defines the <u>machine environment</u>, β defines <u>processing characteristics and constraints</u>, and γ defines the <u>criteria</u>.

Source: Pinedo Sections 2.1, 2.2

Machine Environments

- Single Machine (1)



- Identical Parallel Machines (Pm)
- Parallel Machines with Different Speeds (Qm)
- Unrelated Machines in Parallel (Rm)

Machine Environments

- Flow Shop (Fm)
- Flexible Flow Shop (FFc)

- Job Shop (Jm)
- Flexible Job Shop (FJc)

Processing Characteristics

Possible entries for the β field are:

- Release dates (r_j)
- Preemptions (*prmp*)
- Precedence constraints (*prec*)
- Sequence dependent setups (s_{jk})
- Batch processing (*batch(b)*)
- Permutation (*prmu*)
- Blocking (*block*)
- No-wait(*nwt*)
- Recirculation (*rcrc*)

Criteria

Possible entries for the γ field are:

- Makespan (C_{max})
- Total Completion Time ($\sum C_j$)
- Total Weighted Completion Time ($\sum w_j C_j$)
- Maximum Lateness (L_{max})
- Total Tardiness ($\sum T_j$)
- Total Weighted Tardiness ($\sum w_j T_j$)
- Number of Tardy Jobs ($\sum U_j$)
- Weighted Number of Tardy Jobs ($\sum w_j U_j$)

Example 1

<u>Problem Description:</u> There are 5 machines in series. Each job must be processed on each machine and all jobs have to follow the same route. Jobs have release dates and due dates and the goal is to minimize average tardiness among jobs.

What is the triplet $\alpha|\beta|\gamma$ for this problem?

Example 2

<u>Problem Description:</u> There are 3 identical machines. Each job must be processed on one of the machines and any job can be processed on any machine. The time it takes to complete the last job must be minimized.

What is the triplet $\alpha|\beta|\gamma$ for this problem?

Example 3

<u>Problem Description:</u> There are *m* machines and each job has its own route to follow through the machines. In some cases jobs must visit the same machine a second time. The goal is to minimize the number of tardy jobs.

What is the triplet $\alpha|\beta|\gamma$ for this problem?

Complexity Primer

Introduction:

- Complexity theory provides a means for **classifying** scheduling problems on the basis of difficulty

Definitions:

- A **problem** is a generic description of a problem
- An **instance** of a problem is a specific case based on specified numerical data

Problem vs. an **Instance**

- An example of a problem is: Pm| | Cmax

- An example of an instance is: two identical machines are to have 6 jobs scheduled on either machine with processing times: 7,6,4,5,3,8.

Complexity Primer

The **size** of an instance of a problem is a measure of the length of the data string required to define it

What is the size of Pm| | Cmax?

The **size of a scheduling problem** is normally defined by the **number of jobs**

Complexity Primer

- The **efficiency** of an algorithm is often measured by the maximum (**worst-case**) number of computational steps to find an optimal solution

- For scheduling problems efficiency is characterized by the **limiting growth rate** in the number of jobs (n):

Example: # steps = $10 + 20n + 7n^5 \approx O(n^5)$

Examples of Time Complexity Functions

Time complexity function	Size n					
	10	20	30	40	50	60
n	.00001 second	.00002 second	.00003 second	.00004 second	.00005 second	.00006 second
n^2	.0001 second	.0004 second	.0009 second	.0016 second	.0025 second	.0036 second
n^3	.001 second	.008 second	.027 second	.064 second	.125 second	.216 second
n^5	.1 second	3.2 seconds	24.3 seconds	1.7 minutes	5.2 minutes	13.0 minutes
2^n	.001 second	1.0 second	17.9 minutes	12.7 days	35.7 years	366 centuries
3^n	.059 second	58 minutes	6.5 years	3855 centuries	2×10^8 centuries	1.3×10^{13} centuries

Exponential Time

*From Gary and Johnson, “Computers and Intractability,” 1979

Complexity Primer

Examples of **Easy** problems:

- Finding the largest number in a list is O(n)
- Sorting a sequence of n numbers is O(n log(n))
- The *assignment problem* is $O(n^3)$

Decision Problems

Complexity Theory is built around **Decision Problems**

<u>Decision Problem Example:</u> Given job processing times, p_j, is there a sequence with total completion time less than 12?

Jobs	1	2	3	4
p_j	6	4	2	1

<u>Optimization Problem Counterpart:</u> What is the sequence with minimum completion time?

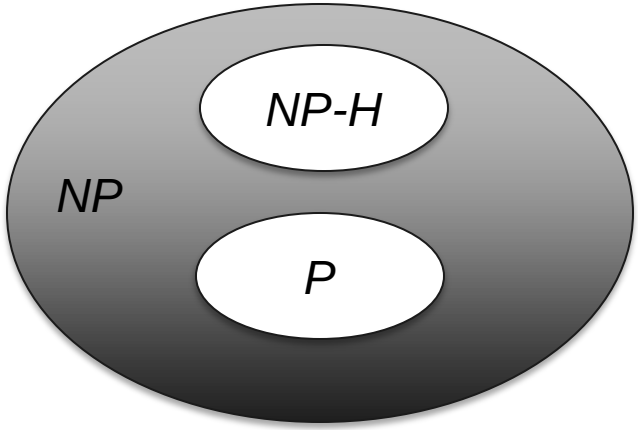
Complexity Primer

Classes of decision problems:

- Definition (**Class *NP***): NP contains all **decision problems** for which a given solution can be **verified** in a polynomial number of steps.

- Definition (**Class *P***): *P* contains all decision problems for which there is an **algorithm** that solves the problem in a **polynomial number of steps**.

Complexity Primer



- <u>Definition (NP-Hardness)*:*</u> A problem is ***NP-Hard*** if the entire class of *NP* problems polynomially reduces to it.

- <u>Question:</u> Does *P* = *NP*?

Get Rich!

Clay Mathematics Institute

Dedicated to increasing and disseminating mathematical knowledge

HOME | ABOUT CMI | PROGRAMS | NEWS & EVENTS | AWARDS | SCHOLARS | PUBLICATIONS

First Clay Mathematics Institute Millennium Prize Announced

Prize for Resolution of the Poincaré Conjecture Awarded to Dr. Grigoriy Perelman

March 18, 2010. The Clay Mathematics Institute (CMI) announces today that Dr. Grigoriy Perelman of St. Petersburg, Russia, is the recipient of the Millennium Prize for resolution of the Poincaré conjecture. The citation for the award reads:

The Clay Mathematics Institute hereby awards the Millennium Prize for resolution of the Poincaré conjecture to Grigoriy Perelman.

Birch and Swinnerton-Dyer Conjecture
Hodge Conjecture
Navier-Stokes Equations
P vs NP
Poincaré Conjecture
Riemann Hypothesis
Yang-Mills Theory

Rules
Millennium Meeting Videos

Learn more at: http://www.claymath.org/millennium/

Examples of NP-Hard Problems

PARTITION: Given positive integers $a_1, \ldots, a_t$ and

$$b = \left(\frac{1}{2}\right) \sum_{j=1}^{t} a_j$$

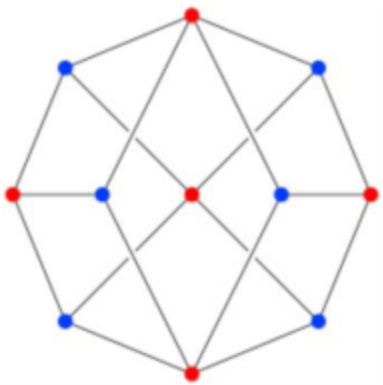
Are there two disjoint subsets S_1 and S_2 such that

$$\sum_{j \in S_i} a_j = b$$

for $i = 1,2$?

Examples of NP-Hard Problems

HAMILTONIAN CIRCUIT: For a graph G = (N,A) with node set N and arc set A, does there exist a circuit (tour) that visits all nodes in N exactly once and returns to the start node?



CLIQUE: For a graph G = (N,A) with node set N and arc set A, does there exist a clique of size c?

Examples of NP-Hard Problems

The very first problem classified as NP-Hard was the **Satisfiability** (SAT) problem:

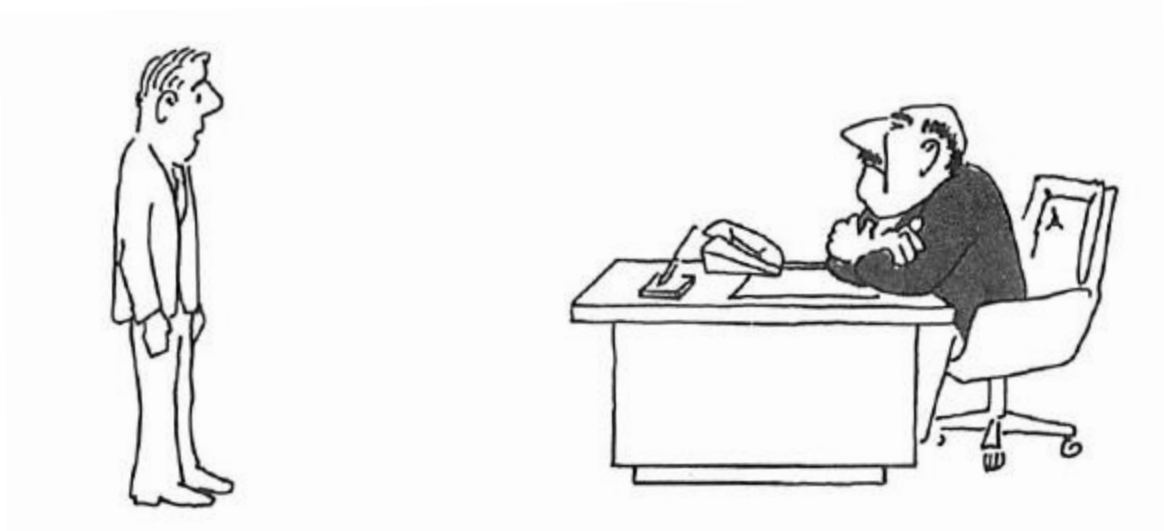
SATISFIABILITY: Given a set of variables and a collection of clauses defined over the variables, is there an assignment of value to the variables for which each one of the clauses is true?

Cooke's Theorem says that every problem in NP reduces to Satisfiability (see Garey and Johnson)

Examples of NP-Hard Problems

Why do we care about these examples of NP-Hard Problems?

So this this doesn't happen to you….



"I can't find an efficient algorithm, I guess I'm just too dumb."

*From Garey and Johnson, "Computers and Intractability," 1979

Examples of NP-Hard Problems

This would be better but unfortunately Complexity Theory doesn't help here….

"I can't find an efficient algorithm, because no such algorithm is possible!"

Examples of NP-Hard Problems

This is the best you can do for now…..

"I can't find an efficient algorithm, but neither can all these famous people."

Classifying Scheduling Problems

If you can show that an NP-Hard problem is a special case of a scheduling problem then the scheduling problem is NP-Hard

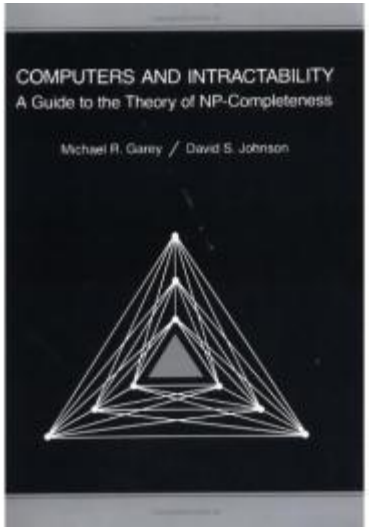
Question:

Why does PARTITION reduce to P2| | Cmax?

Learning More

Pinedo, Appendix D, provides a summary of complexity theory suitable for this course

If you are curious and want to learn more see the tutorial on ctools or the book Garey and Johnson, “Computers and Intractability”:



Single Machine Scheduling

Consider the following instance of $1|\ |\sum C_j$

Jobs	1	2	3	4
p_j	6	4	2	1

Q1. What is the optimal sequence for this <u>instance</u>?

Q2. What is the optimal sequence for this <u>problem</u>?

Q3. What is the <u>complexity</u> of this problem?

For Next Time

Answer Q3 for $1|\sum w_j C_j$

For next class read Sections 2.3 and 3.1, and 3.3 (we will come back to Section 3.2 next week)