Course Presentation

- Communication media
  - Blackboard (for private communications)
  - Mail, Fora, Blog, Grades, Documents (photocopies),
  - Web-site http://www.imada.sdu.dk/~marco/DM87/
    Lecture plan, Syllabus, Links, Exam documents
- 30 hours of lectures + 10 hours practical exercises
- Schedule:
  1. Lectures:
     Mondays 12:00-13:45, Thursdays 8:15-10:00
     Weeks 5-10, 13-16
     Last lecture (preliminary date): Thursday, April 17
  2. Exam: June
Course Content

- Review of Optimization Methods:
  - Mathematical Programming,
  - Constraint Programming,
  - Heuristics
  - Problem Specific Algorithms (Dynamic Programming, Branch and Bound)
- Introduction to Scheduling, Terminology, Classification.
  - Single Machine Models
  - Parallel Machine Models
  - Flow Shops and Flexible Flow Shops
  - Job Shops, Open Shops
- Introduction to Timetabling, Terminology, Classification
  - Interval Scheduling, Reservations
  - Educational Timetabling
  - Workforce and Employee Timetabling
  - Transportation Timetabling
- Introduction to Vehicle Routing, Terminology, Classification
  - Capacitated Vehicle Routing
  - Vehicle Routing with Time Windows

Evaluation

Final Assessment (10 ECTS)
- Oral exam: 30 minutes + 5 minutes defense project
  - meant to assess the base knowledge
- Group project:
  - free choice of a case study among few proposed ones
  - Deliverables: program + report
  - meant to assess the ability to apply

Course Material

- Literature
  - Supplementary Articles: will be indicated during the course
- Slides
- Class exercises (participatory)

Useful Previous Knowledge for this Course

- Algorithms and data structures
- Programming A and B
- Networks and Integer Programming
- Heuristics for Optimization
- Software Methodologies and Engineering
Course Goals and Project Plan

How to Tackle Real-life Optimization Problems:
▶ Formulate (mathematically) the problem
▶ Model the problem and recognize possible similar problems
▶ Search in the literature (that is, on the Internet) for:
  ▶ complexity results (is the problem $NP$-hard?)
  ▶ solution algorithms for simplified problem
  ▶ solution algorithms for original problem
▶ Design solution algorithms
▶ Test experimentally with the goals of:
  ▶ configuring
  ▶ tuning parameters
  ▶ comparing
  ▶ studying the behavior (prediction of scaling and deviation from optimum)

The problem Solving Cycle

Outline

1. Course Introduction

2. Scheduling
   Problem Classification

Scheduling

▶ Manufacturing
  ▶ Project planning
  ▶ Single, parallel machine and job shop systems
  ▶ Flexible assembly systems
    Automated material handling (conveyor system)
  ▶ Lot sizing
  ▶ Supply chain planning
▶ Services
  ⇒ different algorithms
Problem Definition

Given: a set of jobs $J = \{J_1, \ldots, J_n\}$ that have to be processed by a set of machines $M = \{M_1, \ldots, M_m\}$

Find: a schedule, i.e., a mapping of jobs to machines and processing times subject to feasibility and optimization constraints.

Notation:
$n, j, k$ jobs
$m, i, h$ machines

Visualization

Scheduling are represented by Gantt charts
▶ machine-oriented

Data Associated to Jobs

▶ Processing time $p_{ij}$
▶ Release date $r_j$
▶ Due date $d_j$ (deadline)
▶ Weight $w_j$

▶ A job $J_j$ may also consist of a number $n_j$ of operations $O_{j1}, O_{j2}, \ldots, O_{jn_j}$ and data for each operation.
▶ Associated to each operation a set of machines $\mu_{jl} \subseteq M$

Data that depend on the schedule (dynamic)
▶ Starting times $S_{ij}$
▶ Completion time $C_{ij}, C_j$
Problem Classification

A scheduling problem is described by a triplet $\alpha | \beta | \gamma$.

- $\alpha$ machine environment (one or two entries)
- $\beta$ job characteristics (none or multiple entry)
- $\gamma$ objective to be minimized (one entry)


The $\alpha | \beta | \gamma$ Classification Scheme

**Machine Environment** $\alpha_1 \alpha_2 | \beta_1 \ldots \beta_{13} | \gamma$

- single machine/multi-machine ($\alpha_1 = \alpha_2 = 1$ or $\alpha_2 = m$)
- parallel machines: identical ($\alpha_1 = P$), uniform $p_j/v_i$ ($\alpha_1 = Q$), unrelated $p_j/v_{ij}$ ($\alpha_1 = R$)
- multi operations models: Flow Shop ($\alpha_1 = F$), Open Shop ($\alpha_1 = O$), Job Shop ($\alpha_1 = J$), Mixed (or Group) Shop ($\alpha_1 = X$)

**Job Characteristics** $\alpha_1 \alpha_2 | \beta_1 \ldots \beta_{13} | \gamma$

- $\beta_1 = prmp$ presence of preemption (resume or repeat)
- $\beta_2$ precedence constraints between jobs (with $\alpha = P, F$
  acyclic digraph $G = (V, A)$
  - $\beta_2 = prec$ if $G$ is arbitrary
  - $\beta_2 = \{chains, intree, outtree, tree, sp-graph\}$
- $\beta_3 = r_j$ presence of release dates
- $\beta_4 = p_j = p$ precessing times are equal
- $(\beta_5 = d_j$ presence of deadlines$)$
- $\beta_6 = \{s\text{-batch}, p\text{-batch}\}$ batching problem
- $\beta_7 = \{s_{jk}, s_{jik}\}$ sequence dependent setup times

**Job Characteristics (2)** $\alpha_1 \alpha_2 | \beta_1 \ldots \beta_{13} | \gamma$

- $\beta_8 = brkdwn$ machines breakdowns
- $\beta_9 = M_j$ machine eligibility restrictions (if $\alpha = Pm$)
- $\beta_{10} = prmu$ permutation flow shop
- $\beta_{11} = block$ presence of blocking in flow shop (limited buffer)
- $\beta_{12} = nwt$ no-wait in flow shop (limited buffer)
- $\beta_{13} = recrc$ Recirculation in job shop
The α | β | γ Classification Scheme

Objective (always \( f(C_j) \))

- Lateness: \( L_j = C_j - d_j \)
- Tardiness: \( T_j = \max\{C_j - d_j, 0\} = \max\{L_j, 0\} \)
- Earliness: \( E_j = \max\{d_j - C_j, 0\} \)
- Unit penalty: \( U_j = \begin{cases} 1 & \text{if } C_j > d_j \\ 0 & \text{otherwise} \end{cases} \)

Other Objectives

Non regular objectives

- Min \( \sum w_j E_j + \sum w''_j T_j \) (just in time)
- Min waiting times
- Min set up times/costs
- Min transportation costs

DM87 – Scheduling, Timetabling and Routing 22

Exercises

Example 1.1.3 (Scheduling Tasks in a Central Processing Unit [CPU])

One of the functions of a multitasking computer operating system is to schedule the time that the CPU devotes to the different programs that have to be executed. The exact processing times are usually not known in advance. However, the distribution of these random processing times may be known in advance, including their expected values and variances. In addition, each task usually has a certain priority level (the operating system typically allows operators and users to specify the priority level or weight of each task). In this case, the objective is to minimize the expected sum of the weighted completion times for all tasks.

To avoid the situation where relatively short tasks remain in the system for a long time waiting for much longer tasks with a higher priority, the operating system slices the tasks into little pieces. The operating system then rotates these slices on the CPU so that, in any given time interval, the CPU spends some amount of time on each task. This way, if by chance the processing time of one of the tasks is very short, the task will be able to leave the system relatively quickly.
Exercises

Example 1.1.2 (Gate Assignments at an Airport)
Consider an airline terminal at a major airport. There are dozens of gates and hundreds of airplanes arriving and departing each day. The gates are not all identical and neither are the planes. Some of the gates are at locations with a lot of space where large planes (widebodies) can be accommodated easily. Other gates are in locations where it is difficult to bring in the planes. Certain planes may actually have to be towed to their gates.
Planes arrive and depart according to a certain schedule. However, the schedule is subject to a significant amount of randomness that may be weather related or due to events at other airports. During the time that a plane occupies a gate, the arriving passengers have to be deplaned, the plane has to be serviced, and the departing passengers have to be boarded. The scheduled departure time can be viewed as a due date, and the airline’s performance is measured accordingly. However, if it is known in advance that the plane cannot land at the next airport because of anticipated congestion at the scheduled arrival time, then the plane does not take off (such a policy is followed to conserve fuel). If a plane is not allowed to take off, operating policies usually prescribe that passengers remain in the terminal rather than on the plane. If boarding is postponed, a plane may remain at a gate for an extended period of time, thus preventing other planes from using the gate.

Classes of Schedules

Nondelay schedule
A feasible schedule is called nondelay if no machine is kept idle while an operation is waiting for processing.
There are optimal schedules that are nondelay for most models with regular objective function.

Active schedule
A feasible schedule is called active if it is not possible to construct another schedule by changing the order of processing on the machines and having at least one operation finishing earlier and no operation finishing later.
There exists for $Jm||\gamma$ ($\gamma$ regular) an optimal schedule that is active.
nondelay $\Rightarrow$ active
active $\not\Rightarrow$ nondelay

Semi-active schedule
A feasible schedule is called semi-active if no operation can be completed earlier without changing the order of processing on any one of the machines.

Solutions

Distinction between
$\blacktriangleright$ sequence
$\blacktriangleright$ schedule
$\blacktriangleright$ scheduling policy

Feasible schedule
A schedule is feasible if no two time intervals overlap on the same machine, and if it meets a number of problem specific constraints.

Optimal schedule
A schedule is optimal if it minimizes the given objective.

Complexity Hierarchy
A problem $A$ is reducible to $B$ if a procedure for $B$ can be used also for $A$.

Ex: $1||\sum C_j \propto 1||\sum w_j C_j$

Complexity hierarchy describe relationships between different scheduling problems.

Interest in characterizing the borderline: polynomial vs NP-hard problems