Scheduling

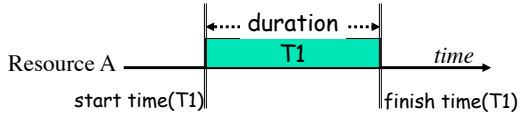
- Scheduling is the "problem of allocating scarce resources to activities over time." [Baker 1974]
- Typically, planning is deciding *what* to do, and scheduling is deciding *when* to do it.
- Generally, scheduling is determining how to accomplish some set of tasks while:
  - obeying task constraints (e.g., due dates, task interactions) [feasibility criteria]
  - minimizing resource usage (e.g., time, personnel) [optimization criteria]

Scheduling Problem

- Given:
  - set of resources/machines $M := \{M_1,...M_m\}$
  - set of tasks/jobs with constraints, temporal descriptions and resource requirements $J := \{J_1,...J_n\}$
  - an objective function
- find a solution (mapping of tasks to times on resources) that
  - satisfies constraints
  - minimizes objective function

Task Characteristics

- Non-preemptive:
  - once started, must be allowed to finish.



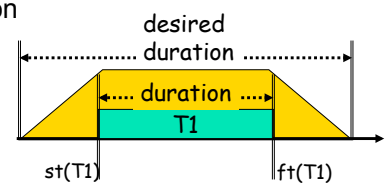
- Preemptive:
  - can be interrupted and re-started later.

Task Characteristics (cont.)

- Single-Stage:
  - each job consists of a single operation
- Multi-Stage:
  - Each job may consist of many operations performed in a prescribed order on different machines

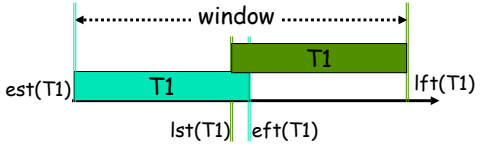
Task Characteristics (cont.)

- Processing time:
  - fixed duration
  - flexible: penalty function for less than desired duration



Task Characteristics (cont.)

- Time windows:
  - must be scheduled within some interval



  - release dates: beginning of window
  - deadlines: end of window
  - slack: LST - EST

Task Characteristics (cont.)

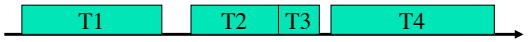
- Precedence Constraints:
  - Task A before Task B
  - start(B)-finish(A)>=0
- Resource Constraints:
  - must be on a particular resource or particular type of resource
- Alternatives
  - resources
  - times

Task Characteristics (cont.)

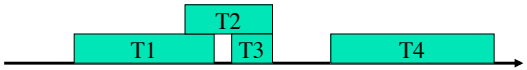
- Job Weights:
  - Relative importance of different tasks
- Set-up Times
  - task dependent interaction effects
  - depending on what preceded task in schedule, it may take extra time to prepare resource to execute the task
  - e.g., making different colored plastics in a vat, extruding different viscosity materials, re-orienting antenna to receive signal

Resource Characteristics: Capacity

- Unit capacity: one task at a time



- Multi-capacity:
  - *n* tasks at a time
  - certain sum of task size requirements at a time



Resource Characteristics: Duplication

- Parallel machines: additional machines available
  - Identical: processing time of a job is independent of which machine
  - Uniform: machines have different speeds, but changes times for all jobs uniformly
  - Unrelated: machines produce different job related processing times

Resource Characteristics: Capacity Types

- consumable:
  - monotonically decreasing capacity
- renewable:
  - variable (up and down) capacity
- reusable:
  - fixed, unchangable capacity

Optimization Metrics

- Resource based
  - minimize number of resources used/cost of resources allocated
  - maximize resource utilization: percentage of time that resources are not idle
  - minimum peak resource usage
- Task based
  - cumulative value of tasks scheduled
  - cumulative value of time slots for tasks
  - maximum weighted # of tasks

Optimization Metrics (cont.)

- Time based
  - Makespan: duration between start of first task and completion of last task (aka Maximal Completion Time)
  - Total Weighted Flow Time: sum of weighted completion times
  - Maximum Tardiness: largest delay from desired finish time to actual end time for any task
  - Total Weighted Tardiness: Sum of the delays between desired and actual finish times
  - Total Weighted Number of Late/Discarded Tasks

Feasibility

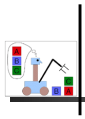
- Overconstrained:
  - Sometimes not all constraints can be honored (e.g., *soft* constraints)
  - decide which constraints to *relax* then decide when to do the tasks
- Oversubscribed:
  - Sometimes not all tasks can be accommodated given fixed resources.
  - decide *which* tasks to discard then decide when to do the tasks that remain

Example: Classroom Scheduling (Aka Timetabling)

- Tasks:
  - 20 Recitations @ 50 minutes = ~20 hours
    - CS150A, CS150B, CS160A, CS160B, CS160C, CS160D, CS160E, CS160F, CS160G, CS160H, CS160I, CS160J, CS253A, CS253B, CS253C, CS253D, CS270A, CS270B, CS270C, CS270D
  - 11 Labs @100 minutes = ~22 hours
    - CS161A, CS161B, CS161C, CS161D, CS161E, CS161F, CS161G, CS200A, CS200B, CS200C, CS200D
- Resources: 2 Rooms, ~8 GTAs + ~3 UTAs
- Constraints?

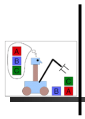
Timetabling

"A timetabling problem is a problem with four parameters: T , a finite set of times; R, a finite set of resources; M, a finite set of meetings; and C, a finite set of constraints. The problem is to assign times and resources to the meetings so as to satisfy the constraints as far as possible. " [E.K.Burke, J.H.Kingston and D.deWerra. "Applications to timetabling" In: J. Gross and J. Yellen (eds.) *The Handbook of Graph Theory*, Chapman Hall/CRC Press, 2004, 445-474.]



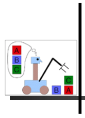
Types of Timetabling

- Nurse/hospital scheduling
- Sports (e.g., tournaments, leagues)
- Transportation (e.g., airport flights, crews)
- Educational
  - school (weekly class schedules in K-12)
  - course (lectures in university)
  - examination (finals)



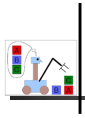
Examination timetabling characteristics

- 1 exam per course
- Different possible constraints:
  - At most 2 exams per day for each student
  - Students should not have consecutive exams
  - Number of possible slots (periods) may vary
  - Allow multiple exams in the same room or not



Exam Timetabling Definition

- Assign a set of exams $E=e_1,e_2,\ldots e_e$ into a limited number of ordered timeslots $T=t_1,t_2,\ldots t_t$ and rooms of certain capacity in each timeslot $C=c_1,c_2,\ldots c_t$, subject to certain constraints.



Hard constraints

- Exams being taken by the same students cannot be assigned at the same time. x_i is timeslot for exam i, D_{ij} is the number of students in both exams i and j:

$$x_i \in T, x_i \neq x_j \forall i,j \in E, i \neq j, D_{ij} > 0$$

- Number of students taking an exam cannot exceed the capacity of the rooms. s_i is the number of students in exam i:

$$\sum_{i \in E} s_i \leq C_t, x_i = t, t \in T$$

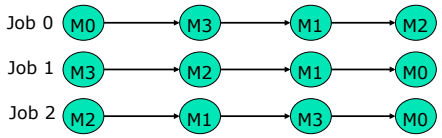
Soft Constraints

- At most 1 or 2 exams per day for each student
- Spread exams as evenly as possible
- Allow some groups of exams to take place at the same time, on the same day or at one location
- Number of possible slots (periods) may vary
- Allow multiple exams in the same room or not
- Schedule all exams or largest exams as early as possible
- Conflicting exams on the same day to be located nearby
- Students should not have consecutive exams

$$\sum_{i \in E} \sum_{j \in E, j \neq i} \text{if } (x_j = x_i + 1), D_{ij}$$

Job Shop Scheduling

- n jobs on m machines
- each job must be processed on each machine exactly once for a fixed duration in a pre-defined order (job routing order).
- unit capacity, non-preemptive, no time windows or setups
- minimize makespan (time from start of earliest job to finish of latest job)



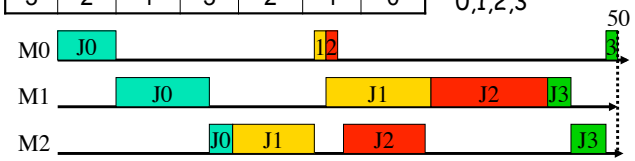
JSP (cont.)

- Solution:
  - is processing order for all jobs on each machine.
  - specifies est (earliest start time) for each job/ machine s.t. precedence and resource constraints are satisfied.
- (n!)^m possible solutions of which a subset are feasible.

Example JSP

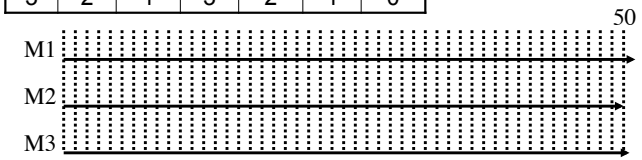
	Oi,1		Oi,2		Oi,3	
Job	Dur	Rout	Dur	Rout	Dur	Rout
0	5	0	8	1	2	2
1	7	2	3	0	9	1
2	1	0	7	2	10	1
3	2	1	3	2	1	0

Solution:
0,1,2,3



Example JSP

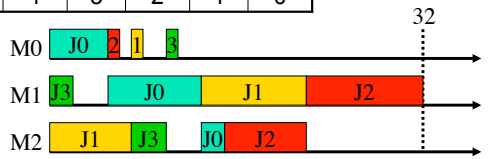
	Oi,1		Oi,2		Oi,3	
Job	Dur	Rout	Dur	Rout	Dur	Rout
0	5	0	8	1	2	2
1	7	2	3	0	9	1
2	1	0	7	2	10	1
3	2	1	3	2	1	0



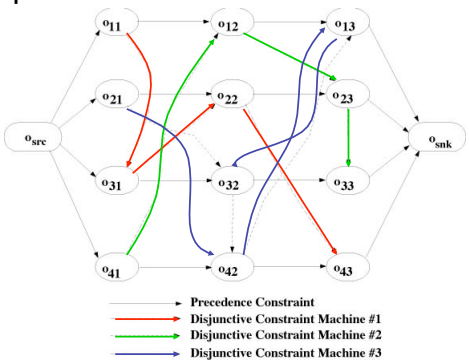
Example JSP

	Oi,1		Oi,2		Oi,3	
Job	Dur	Rout	Dur	Rout	Dur	Rout
0	5	0	8	1	2	2
1	7	2	3	0	9	1
2	1	0	7	2	10	1
3	2	1	3	2	1	0

Solution:
0,1,2,3
3,0,1,2
1,3,0,2



Example JSP (cont.)



JSP Taxonomy of Schedules

Feasible Solution: a job processing order on each machine without any cyclic dependencies

Feasible Schedule: specifies a start time s.t. all constraints are satisfied

Inadmissible: operations start later than their est

Semi-Active: some operations are scheduled at their est

- Global left shift: move an operation earlier into machine idle time

Active: no global left shifts are possible (difficult to determine)

Non-Delay: no machine is idle if an operation is available

Optimal: minimizes the makespan

Algorithms for Scheduling

- Local Search
- Heuristic Constructive (e.g., LDS, HBSS)
- Constraint Programming (e.g.,dynamic programming, edge-finding)
- Genetic Algorithms
- Hybrid

[Note: makespan computation tends to be most expensive aspect]

An Aside: Critical Operations, Paths and Blocks

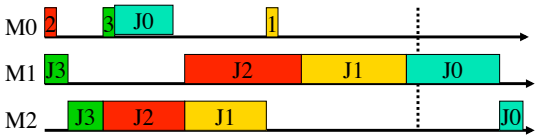
Critical Operation: operations in which est=lst. If start time is delayed, makespan will increase.

Critical Path: contiguous sequence of critical operations starting at t=0 and ending at makespan with no machine idle time.

Critical Blocks: subsequences of critical paths with all operations on the same machine.

Critical Operations, Paths and Blocks Example

Greedy Solution: 3,2,1,0



Critical operations:
$J01, J12, J11, J21, J22, J31, J32$

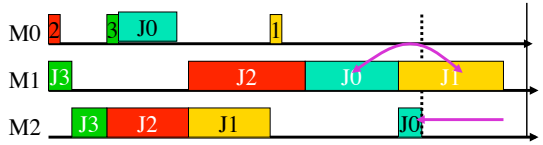
Critical path:
$J31, J32, J22, J21, J11, J01, J02$

Critical blocks:
$\{J32, J22\}, \{J21, J11, J01\}$

J01 is J0 on M1

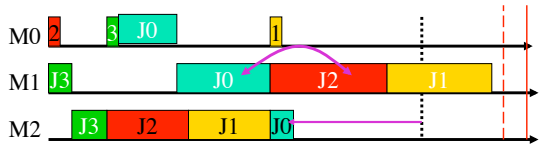
Critical Blocks Based Move Operators

- N1:
  - invert the order of a pair of adjacent operations on the same critical block
  - focuses on only 1) feasible neighbors and 2) those operations that can change the makespan
  - fully connected search space: there exists a path from any feasible solution to a global optimal

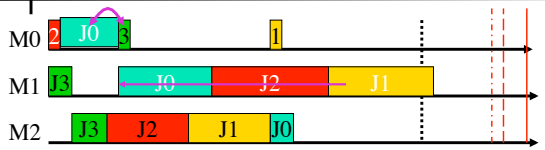


Critical Blocks Based Move Operators (cont.)

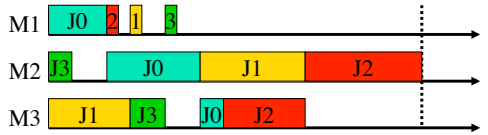
- N5, like N1 except
  - swap only adjacent operations that include either first or last operations in the critical block
  - do not swap either the first two operations in the first critical block or the last two operations in the last critical block
  - not completely connected



Effect of Moves (cont.)

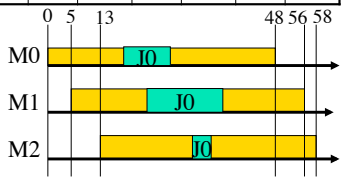


Recall the optimal…



Computing EST, LFT…

	$O_{i,1}$		$O_{i,2}$		$O_{i,3}$	
Job	Dur	Rout	Dur	Rout	Dur	Rout
0	5	0	8	1	2	2
1	7	2	3	0	9	1
2	1	0	7	2	10	1
3	2	1	3	2	1	0



EST

$EST_{0,0}$= 0

$EST_{0,1}=EFT_{0,0}=EST_{0,0}+D_{0,0}=0+5=5$

$EST_{0,2}=EFT_{0,1}=EST_{0,1}+D_{0,1}=5+8=13$

LFT

Worst Case LFT $\Sigma D_{i,j} = 58$

$LFT_{0,2}=LFTWC=58$

$LFT_{0,1}=LST_{0,2}=LFT_{0,2}-D_{0,2}=58-2=56$

$LFT_{0,0}=LST_{0,1}=LFT_{0,1}-D_{0,1}=56-8=48$

Slack Based Heuristics

- order operations in increasing (LST – EST)
- greedy version:
  - until all operations scheduled:
    - schedule open operation that has predecessors scheduled and minimizes slack at its EST
    - update ESTs and LSTs
- minslack: impose ordering on pairs of unsequenced tasks with minimal maximal slack:
  - (max[lft(A)-est(B), lft(B)-est(A)] – d(A) – d(B))

Constraint Programming

Edge Finding:

- determining whether an operation can, cannot, or must execute before (or after) other operations on a machine.
- Rules: (O is set of operations on machine, A is a particular operation, D is duration)

$$\forall O, \forall A \notin O, LFT_{O \cup \{A\}} - EST_O < \sum_{i \in O} D_i + D_A \Rightarrow A << O$$

$$\forall O, \forall A \notin O, LFT_O - EST_{O \cup \{A\}} < \sum_{i \in O} D_i + D_A \Rightarrow A >> O$$

$$A << O \Rightarrow end(A) \leq \min_{i \in O}(LFT_i - D_i)$$

$$A >> O \Rightarrow start(A) \geq \max_{i \in O}(EST_i + D_i)$$

A<<O when its LFT is before O's
A>>O when its EST is after O's
A<<O when its end is before O's LST
A>>O when its start is after O's EFT

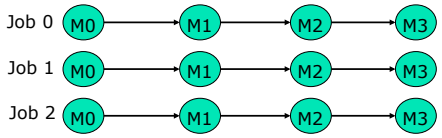
Edge-Finding (cont.)

1. Compute a schedule (*S*) by iteratively applying priority rule:
   - whenever a resource is free and an operation is available, schedule the operation with the smallest LFT
2. Given *S*, for each operation *A*,
   1. compute the set of operations (P) not finished at EST(A).
   2. For each P (in decreasing order of LFT(P)), update ordering and start/end times based on rules.

Benchmark Application: Manufacturing Scheduling

Flow Shop (JSP):

- n jobs on m machines
- each job must be processed on each machine exactly once for a fixed duration in the *same* pre-defined order
- unit capacity, non-preemptive, no time windows or setups
- minimize makespan
- Permutation: infinite buffer between machines



Permutation Operators

- Solution to FSP is permutation of jobs
- Search operators for permutations:
  - Transpose: swap positions of 2 adjacent jobs

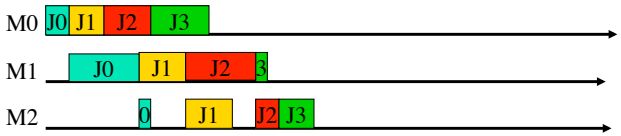
  - Exchange: swap positions of 2 different jobs

  - Insertion or Shift: remove job from one position and move to another

FSP Example

Job	J_0	J_1	J_2	J_3
p_{j0}	2	3	4	5
p_{j1}	6	4	6	1
p_{j2}	1	4	2	3

Solution:
0,1,2,3



JSP Difficulty

- Analysis:
  - Optimization is NP-hard
  - No polynomial time algorithm can guarantee a solution within 20% of optimal makespan.
- Observations:
  - Given fixed *n* and *m*, workflow JSPs are typically more difficult than random JSPs.
  - Given fixed *n* and *m*, flowshop JSPs are typically more difficult than workflow JSPs.
  - Given fixed *m* and *wf*,square JSPs are typically more difficult than rectangular JSPs (more jobs than machines).
  - Given fixed *n*,*m* and *wf*, relative problem difficulty tends to be algorithm independent.

Local Search for JSP

- state of the art [Nowicki & Smutnicki 2003]
  - solutions represented as disjunctive graph with edges only between adjacent operations
  - JSP specific move operators (N5 for N&S03)
  - initiate search from random semi-active or greedy solutions
  - tabu search: checks for cycles in recent solutions (tabu tenure=8) and re-starts if cycles or if no progress over some time
  - path relinking to generate new solutions for re-starts