DM204, 2011
SCHEDULING, TIMETABLING AND ROUTING

Lecture 2
Single Machine Problems
Polynomial Cases

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Course Overview

- ✔ Scheduling
  - ✔ Classification
  - ✔ Complexity issues
  - RCPSP
  - Single Machine
  - Parallel Machine and Flow Shop Models
  - Job Shop
  - Resource Constrained Project Scheduling Model

- Timetabling
  - Sport Timetabling
  - Reservations and Education
  - University Timetabling
  - Crew Scheduling
  - Public Transports

- Vechicle Routing
  - Capacited Models
  - Time Windows models
  - Rich Models

Outline

1. Resource Constrained Project Scheduling Model
2. Dispatching Rules
3. Single Machine Models

RCPSP

Resource Constrained Project Scheduling Model

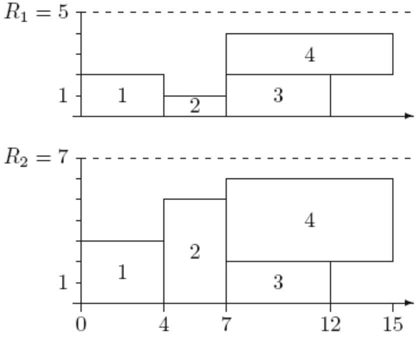
Given:

- activities (jobs) $j = 1, \ldots, n$
- renewable resources $i = 1, \ldots, m$
- amount of resources available R_i
- processing times p_j
- amount of resource used r_{ij}
- precedence constraints $j \rightarrow k$

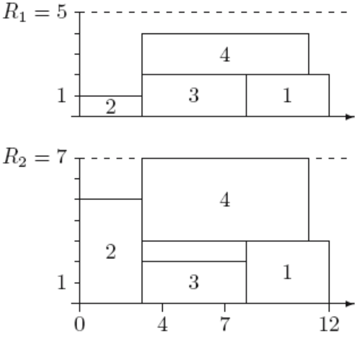
Further generalizations

- Time dependent resource profile $R_i(t)$
  given by (t_i^μ, R_i^μ) where $0 = t_i^1 < t_i^2 < \ldots < t_i^{m_i} = T$
  Disjunctive resource, if $R_k(t) = \{0, 1\}$; cumulative resource, otherwise
- Multiple modes for an activity j
  processing time and use of resource depends on its mode m: p_{jm}, r_{jkm}.

An Example



(a) A feasible schedule



(b) An optimal schedule

Modeling

Assignment 1

- A contractor has to complete n activities.
- The duration of activity j is p_j
- each activity requires a crew of size W_j.
- The activities are not subject to precedence constraints.
- The contractor has W workers at his disposal
- his objective is to complete all n activities in minimum time.

Assignment 2

- Exams in a college may have different duration.
- The exams have to be held in a gym with W seats.
- The enrollment in course j is W_j and
- all W_j students have to take the exam at the same time.
- The goal is to develop a timetable that schedules all n exams in minimum time.
- Consider both the cases in which each student has to attend a single exam as well as the situation in which a student can attend more than one exam.

Assignment 3

- In a basic high-school timetabling problem we are given m classes $c_1, \ldots, c_m$,
- h teachers $a_1, \ldots, a_h$ and
- T teaching periods $t_1, \ldots, t_T$.
- Furthermore, we have lectures $i = l_1, \ldots, l_n$.
- Associated with each lecture is a unique teacher and a unique class.
- A teacher a_j may be available only in certain teaching periods.
- The corresponding timetabling problem is to assign the lectures to the teaching periods such that
  - each class has at most one lecture in any time period
  - each teacher has at most one lecture in any time period,
  - each teacher has only to teach in time periods where he is available.

Assignment 4

- A set of jobs $J_1, \dots, J_g$ are to be processed by auditors $A_1, \dots, A_m$.
- Job J_l consists of n_l tasks ($l = 1, \dots, g$).
- There are precedence constraints $i_1 \to i_2$ between tasks i_1, i_2 of the same job.
- Each job J_l has a release time r_l, a due date d_l and a weight w_l.
- Each task must be processed by exactly one auditor. If task i is processed by auditor A_k, then its processing time is p_{ik}.
- Auditor A_k is available during disjoint time intervals $[s_k^\nu, l_k^\nu]$ ($\nu = 1, \dots, m$) with $l_k^\nu < s_k^\nu$ for $\nu = 1, \dots, m_k - 1$.
- Furthermore, the total working time of A_k is bounded from below by H_k^- and from above by H_k^+ with $H_k^- \leq H_k^+$ ($k = 1, \dots, m$).
- We have to find an assignment $\alpha(i)$ for each task $i = 1, \dots, n := \sum_{l=1}^{g} n_l$ to an auditor $A_{\alpha(i)}$ such that
  - each task is processed without preemption in a time window of the assigned auditor
  - the total workload of A_k is bounded by H_k^- and H_k^k for $k = 1, \dots, m$.
  - the precedence constraints are satisfied,
  - all tasks of J_l do not start before time r_l, and
  - the total weighted tardiness $\sum_{l=1}^{g} w_l T_l$ is minimized.

Outline

Dispatching rules

Distinguish static and dynamic rules.

- Service in random order (SIRO)
- Earliest release date first (ERD=FIFO)
  - tends to min variations in waiting time
- Earliest due date (EDD)
- Minimal slack first (MS)
  - $j^* = \arg\min_j \{\max(d_j - p_j - t, 0)\}$.
  - tends to min due date objectives (T,L)

- (Weighted) shortest processing time first (WSPT)
  - $j^* = \arg\max_j \{w_j / pj\}$.
  - tends to min $\sum w_j C_j$ and max work in progress and
- Longest processing time first (LPT)
  - balance work load over parallel machines
- Shortest setup time first (SST)
  - tends to min C_{max} and max throughput
- Least flexible job first (LFJ)
  - eligibility constraints

- Critical path (CP)
  - first job in the CP
  - tends to min C_{max}
- Largest number of successors (LNS)
- Shortest queue at the next operation (SQNO)
  - tends to min idleness of machines

Dispatching Rules in Scheduling

	RULE	DATA	OBJECTIVES
Rules Dependent on Release Dates and Due Dates	ERD	r_j	Variance in Throughput Times
	EDD	d_j	Maximum Lateness
	MS	d_j	Maximum Lateness
Rules Dependent on Processing Times	LPT	p_j	Load Balancing over Parallel Machines
	SPT	p_j	Sum of Completion Times, WIP
	WSPT	p_j, w_j	Weighted Sum of Completion Times, WIP
	CP	p_j, $prec$	Makespan
	LNS	p_j, $prec$	Makespan
Miscellaneous	SIRO	-	Ease of Implementation
	SST	s_{jk}	Makespan and Throughput
	LFJ	M_j	Makespan and Throughput
	SQNO	-	Machine Idleness

When dispatching rules are optimal?

	RULE	DATA	ENVIRONMENT
1	SIRO	—	—
2	ERD	r_j	$1 \mid r_j \mid \mathrm{Var}(\sum(C_j - r_j)/n)$
3	EDD	d_j	$1 \mid\mid L_{\max}$
4	MS	d_j	$1 \mid\mid L_{\max}$
5	SPT	p_j	$Pm \mid\mid \sum C_j;\ Fm \mid p_{ij} = p_j \mid \sum C_j$
6	WSPT	w_j, p_j	$Pm \mid\mid \sum w_j C_j$
7	LPT	p_j	$Pm \mid\mid C_{\max}$
8	SPT-LPT	p_j	$Fm \mid block, p_{ij} = p_j \mid C_{\max}$
9	CP	$p_j, prec$	$Pm \mid prec \mid C_{\max}$
10	LNS	$p_j, prec$	$Pm \mid prec \mid C_{\max}$
11	SST	s_{jk}	$1 \mid s_{jk} \mid C_{\max}$
12	LFJ	M_j	$Pm \mid M_j \mid C_{\max}$
13	LAPT	p_{ij}	$O2 \mid\mid C_{\max}$
14	SQ	—	$Pm \mid\mid \sum C_j$
15	SQNO	—	$Jm \mid\mid \gamma$

Composite dispatching rules

Why composite rules?

- Example: $1 \mid\mid \sum w_j T_j$:
  - WSPT, optimal if due dates are zero
  - EDD, optimal if due dates are loose
  - MS, tends to minimize T

➤ The efficacy of the rules depends on instance factors

Instance characterization

- Job attributes: {weight, processing time, due date, release date}
- Machine attributes: {speed, num. of jobs waiting, num. of jobs eligible}
- Possible instance factors:
  - $1 \mid \mid \sum w_j T_j$

$$\theta_1 = 1 - \frac{\bar{d}}{C_{max}} \qquad \text{(due date tightness)}$$

$$\theta_2 = \frac{d_{max} - d_{min}}{C_{max}} \qquad \text{(due date range)}$$

  - $1 \mid s_{jk} \mid \sum w_j T_j$

$$(\theta_1,\ \theta_2 \text{ with estimated } \hat{C}_{max} = \sum_{j=1}^{n} p_j + n\bar{s})$$

$$\theta_3 = \frac{\bar{s}}{\bar{p}} \qquad \text{(set up time severity)}$$

- $1 \,|\,|\, \sum w_j T_j$, dynamic apparent tardiness cost (ATC)

$$I_j(t) = \frac{w_j}{p_j} \exp\left(-\frac{\max(d_j - p_j - t, 0)}{K\bar{p}}\right)$$

- $1 \,|\, s_{jk} \,|\, \sum w_j T_j$, dynamic apparent tardiness cost with setups (ATCS)

$$I_j(t, l) = \frac{w_j}{p_j} \exp\left(-\frac{\max(d_j - p_j - t, 0)}{K_1\bar{p}}\right) \exp\left(\frac{-s_{jk}}{K_2\bar{s}}\right)$$

after job l has finished.

Outline

1. Resource Constrained Project Scheduling Model

2. Dispatching Rules

3. Single Machine Models

Outlook

$1 \mid\mid \sum w_j C_j$: weighted shortest processing time first is optimal

$1 \mid\mid \sum_j U_j$: Moore's algorithm

$1 \mid prec \mid L_{max}$: Lawler's algorithm, backward dynamic programming in $O(n^2)$ [Lawler, 1973]

$1 \mid\mid \sum h_j(C_j)$: dynamic programming in $O(2^n)$

$1 \mid\mid \sum w_j T_j$: local search and dynasearch

$1 \mid r_j, (prec) \mid L_{max}$: branch and bound

$1 \mid s_{jk} \mid C_{max}$: in the special case, Gilmore and Gomory algorithm optimal in $O(n^2)$

$1 \mid\mid \sum w_j T_j$: column generation approaches

Summary

Single Machine Models:

- C_{max} is sequence independent
- if $r_j = 0$ and h_j is monotone non decreasing in C_j then optimal schedule is nondelay and has no preemption.

$1 \mid \mid \sum w_j C_j$

[Total weighted completion time]

Theorem

The weighted shortest processing time first (WSPT) rule is optimal.

Extensions to $1 \mid prec \mid \sum w_j C_j$

- in the general case strongly NP-hard
- chain precedences:
  process first chain with highest ρ-factor up to, and included, job with highest ρ-factor.
- polytime algorithm also for tree and sp-graph precedences

Extensions to $1 \mid r_j, prmp \mid \sum w_j C_j$

- in the general case strongly NP-hard
- preemptive version of the WSPT if equal weights
- however, $1 \mid r_j \mid \sum w_j C_j$ is strongly NP-hard

$1 \mid \mid \sum_j U_j$

[Number of tardy jobs]

- [Moore, 1968] algorithm in $O(n \log n)$
  - Add jobs in increasing order of due dates
  - If inclusion of job j^* results in this job being completed late discard the scheduled job k^* with the longest processing time
- $1 \mid \mid \sum_j w_j U_j$ is a knapsack problem hence NP-hard

Dynamic programming

Procedure based on divide and conquer

Principle of optimality the completion of an optimal sequence of decisions must be optimal

- Break down the problem into stages at which the decisions take place
- Find a recurrence relation that takes us backward (forward) from one stage to the previous (next)
- Typical technique: labelling with dominance criteria

(In scheduling, backward procedure feasible only if the makespan is schedule independent, eg, single machine problems without setups, multiple machines problems with identical processing times.)

$1 \mid prec \mid h_{max}$

- $h_{max} = \max\{h_1(C_1), h_2(C_2), \ldots, h_n(C_n)\}$, h_j regular
- special case: $1 \mid prec \mid L_{max}$ [maximum lateness]
- solved by backward dynamic programming in $O(n^2)$ [Lawler, 1978]

J set of jobs already scheduled;
J^c set of jobs still to schedule;
$J' \subseteq J^c$ set of schedulable jobs

Step 1: Set $J = \emptyset$, $J^c = \{1, \ldots, n\}$ and J' the set of all jobs with no successor

Step 2: Select j^* such that $j^* = \arg\min_{j \in J'} \{h_j \left(\sum_{k \in J^c} p_k\right)\}$;
add j^* to J; remove j^* from J^c; update J'.

Step 3: If J^c is empty then stop, otherwise go to Step 2.

- For $1 \mid \mid L_{max}$ Earliest Due Date first
- $1|r_j|L_{max}$ is instead strongly NP-hard