DM204, 2010
SCHEDULING, TIMETABLING AND ROUTING

Lecture 18

Single Machine Models, Branch and Bound
Parallel Machines, PERT

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Organization

- Who is taking the oral exam?
- Watch out the schedule.
- Exercise sessions from now mainly about help for the project.
- Resume and Outlook
- Watch out the list of questions

Course Overview

- ✔ Problem Introduction
  - ✔ Scheduling classification
  - ✔ Scheduling complexity
  - ✔ RCPSP
- ✔ General Methods
  - ✔ Integer Programming
  - ✔ Constraint Programming
  - ✔ Heuristics
  - ✔ Dynamic Programming
  - Branch and Bound

- Scheduling
  - Single Machine
  - Parallel Machine and Flow Shop Models
  - Job Shop
  - Resource Constrained Project Scheduling Model
- Timetabling
  - Reservations and Education
  - University Timetabling
  - Crew Scheduling
  - Public Transports
- Vechicle Routing
  - Capacited Models
  - Time Windows models
  - Rich Models

Outline

Outlook

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

$1 \,|\,|\, \sum_j U_j$: Moore's algorithm

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

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

$1 \,|\,|\, \sum w_j T_j$: local search and dynasearch

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

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

$1 \,|\,|\, \sum w_j T_j$: column generation approaches

Multicriteria

Outline

$1 \mid r_j \mid L_{max}$

[Maximum lateness with release dates]

- Strongly NP-hard (reduction from 3-partition)
- might have optimal schedule which is not non-delay

$1 \mid r_j \mid L_{max}$

[Maximum lateness with release dates]

- Strongly NP-hard (reduction from 3-partition)
- might have optimal schedule which is not non-delay
- Branch and bound algorithm (valid also for $1 \mid r_j, prec \mid L_{max}$)
  - **Branching**:
    schedule from the beginning (level k, $n!/(k-1)!$ nodes)
    elimination criterion: do not consider job j_k if:

    $$r_j > \min_{l \in J} \{\max(t, r_l) + p_l\} \qquad J \text{ jobs to schedule, } t \text{ current time}$$

  - **Lower bounding**: relaxation to preemptive case for which EDD is optimal

Branch and Bound

S root of the branching tree

```
LIST := {S};
U:=value of some heuristic solution;
current_best := heuristic solution;
while LIST ≠ ∅
   Choose a branching node k from LIST;
   Remove k from LIST;
   Generate children child(i), i = 1, ..., n_k, and calculate corresponding lower
         bounds LB_i;
   for i:=1 to n_k
      if LB_i < U then
         if child(i) consists of a single solution then
            U:=LB_i;
            current_best:=solution corresponding to child(i)
         else add child(i) to LIST
```

Branch and Bound

Branch and bound *vs* backtracking

$=$ a state space tree is used to solve a problem.

$\neq$ branch and bound does not limit us to any particular way of traversing the tree (backtracking is depth-first)

$\neq$ branch and bound is used only for optimization problems.

Branch and bound *vs* A^*

$=$ In A^* the admissible heuristic mimics bounding

$\neq$ In A^* there is no branching. It is a search algorithm.

$\neq$ A^* is best first

Branch and Bound

[Jens Clausen (1999). Branch and Bound Algorithms
- Principles and Examples.]

- Eager Strategy:
  1. select a node
  2. branch
  3. for each subproblem compute bounds and compare with incumbent solution
  4. discard or store nodes together with their bounds

  (Bounds are calculated as soon as nodes are available)

- Lazy Strategy:
  1. select a node
  2. compute bound
  3. branch
  4. store the new nodes together with the bound of the father node

(often used when selection criterion for next node is max depth)

Components

1. Initial feasible solution (heuristic) – might be crucial!
2. Bounding function
3. Strategy for selecting
4. Branching
5. Fathoming (dominance test)

Bounding

$$\min_{s \in P} g(s) \leq \left\{ \begin{array}{l} \min_{s \in P} f(s) \\ \min_{s \in S} g(s) \end{array} \right\} \leq \min_{s \in S} f(s)$$

P: candidate solutions; $S \subseteq P$ feasible solutions

- relaxation: $\min_{s \in P} f(s)$
- solve (to optimality) in P but with g

Bounding

$$\min_{s \in P} g(s) \leq \left\{ \begin{array}{c} \min_{s \in P} f(s) \\ \min_{s \in S} g(s) \end{array} \right\} \leq \min_{s \in S} f(s)$$

P: candidate solutions; $S \subseteq P$ feasible solutions

- relaxation: $\min_{s \in P} f(s)$
- solve (to optimality) in P but with g
- Lagrangian relaxation combines the two

Bounding

$$\min_{s \in P} g(s) \leq \left\{ \begin{array}{c} \min_{s \in P} f(s) \\ \min_{s \in S} g(s) \end{array} \right\} \leq \min_{s \in S} f(s)$$

P: candidate solutions; $S \subseteq P$ feasible solutions

- relaxation: $\min_{s \in P} f(s)$
- solve (to optimality) in P but with g
- Lagrangian relaxation combines the two
- should be polytime and strong (trade off)

Strategy for selecting next subproblem

- best first
  (combined with eager strategy but also with lazy)
- breadth first
  (memory problems)
- depth first
  works on recursive updates (hence good for memory)
  but might compute a large part of the tree which is far from optimal

Strategy for selecting next subproblem

- best first
  (combined with eager strategy but also with lazy)
- breadth first
  (memory problems)
- depth first
  works on recursive updates (hence good for memory)
  but might compute a large part of the tree which is far from optimal
  (enhanced by alternating search in lowest and largest bounds combined with branching on the node with the largest difference in bound between the children)
  (it seems to perform best)

Branching

- dichotomic
- polytomic

Branching

- dichotomic
- polytomic

Overall guidelines

- finding good initial solutions is important
- if initial solution is close to optimum then the selection strategy makes little difference
- Parallel B&B: distributed control or a combination are better than centralized control
- parallelization might be used also to compute bounds if few nodes alive
- parallelization with static work load distribution is appealing with large search trees

$1 \mid \mid \sum w_j T_j$

- **Branching:**
  - work backward in time
  - elimination criterion:
    if $p_j \leq p_k$ and $d_j \leq d_k$ and $w_j \geq w_k$ then there is an optimal schedule with j before k

$1 \mid \mid \sum w_j T_j$

- **Branching:**
  - work backward in time
  - elimination criterion:
    if $p_j \leq p_k$ and $d_j \leq d_k$ and $w_j \geq w_k$ then there is an optimal schedule with j before k

- **Lower Bounding:**
  relaxation to preemptive case
  transportation problem

$$\min \sum_{j=1}^{n} \sum_{t=1}^{C_{max}} c_{jt} x_{jt}$$

$$\text{s.t.} \sum_{t=1}^{C_{max}} x_{jt} = p_j, \qquad \forall j = 1, \ldots, n$$

$$\sum_{j=1}^{n} x_{jt} \leq 1, \qquad \forall t = 1, \ldots, C_{max}$$

$$x_{jt} \geq 0 \qquad \forall j = 1, \ldots, n; \;\; t = 1, \ldots, C_{max}$$

[Pan and Shi, 2007]'s lower bounding through time indexed
Stronger but computationally more expensive

$$\min \sum_{j=1}^{n} \sum_{t=1}^{T-1} c_{jt} y_{jt}$$

s.t.

$$\sum_{t=1}^{T-p_j} c_{jt} \leq h_j(t + p_j)$$

$$\sum_{t=1}^{T-p_j} y_{jt} = 1, \qquad \forall j = 1, \ldots, n$$

$$\sum_{j=1}^{n} \sum_{s=t-p_j+1}^{t} y_{jt} \leq 1, \qquad \forall t = 1, \ldots, C_{max}$$

$$y_{jt} \geq 0 \qquad \forall j = 1, \ldots, n; \; t = 1, \ldots, C_{max}$$

Complexity resume

Single machine, single criterion problems $1 \,|\,|\, \gamma$:

C_{max}	$\mathcal{P}$
T_{max}	$\mathcal{P}$
L_{max}	$\mathcal{P}$
h_{max}	$\mathcal{P}$
$\sum C_j$	$\mathcal{P}$
$\sum w_j C_j$	$\mathcal{P}$
$\sum U$	$\mathcal{P}$
$\sum w_j U_j$	weakly $\mathcal{NP}$-hard
$\sum T$	weakly $\mathcal{NP}$-hard
$\sum w_j T_j$	strongly $\mathcal{NP}$-hard
$\sum h_j(C_j)$	strongly $\mathcal{NP}$-hard

Outline

Scheduling

$1|prec|\sum w_j C_j$

Sequencing (linear ordering) variables

$$\min \sum_{j=1}^{n} \sum_{k=1}^{n} w_j p_k x_{kj} + \sum_{j=1}^{n} w_j p_j$$

$$\begin{aligned}
\text{s.t. } & x_{kj} + x_{lk} + x_{jl} \geq 1 \qquad j,k,l = 1,\ldots,n j \neq k, k \neq l \\
& x_{kj} + x_{jk} = 1 \qquad \forall j,k = 1,\ldots,n, j \neq k \\
& x_{jk} \in \{0,1\} \qquad j,k = 1,\ldots,n \\
& x_{jj} = 0 \qquad \forall j = 1,\ldots,n
\end{aligned}$$

Scheduling

$1|prec|C_{max}$

Completion time variables

$$
\begin{aligned}
& \min \sum_{j=1}^{n} w_j z_j \\
\text{s.t. } & z_k - z_j \geq p_k \quad \text{for } j \to k \in A \\
& z_j \geq p_j, \quad \text{for } j = 1, \ldots, n \\
& z_k - z_j \geq p_k \quad \text{or} \quad z_j - z_k \geq p_j, \text{ for } (i,j) \in I \\
& z_j \in \mathbf{R}, \quad j = 1, \ldots, n
\end{aligned}
$$

Scheduling

$1||\sum h_j(C_j)$

Time indexed variables

$$\min \sum_{j=1}^{n} \sum_{t=1}^{T-p_j+1} h_j(t+p_j)x_{jt}$$

$$\text{s.t.} \quad \sum_{t=1}^{T-p_j+1} x_{jt} = 1, \quad \text{for all } j = 1,\ldots,n$$

$$\sum_{j=1}^{n} \sum_{s=\max 0, t-p_j+1}^{t} x_{js} \leq 1, \quad \text{for each } t = 1,\ldots,T$$

$$x_{jt} \in \{0,1\}, \quad \text{for each } j = 1,\ldots,n;\ t = 1,\ldots,T$$

Scheduling

$1||\sum h_j(C_j)$

Time indexed variables

$$\min \sum_{j=1}^{n} \sum_{t=1}^{T-p_j+1} h_j(t+p_j)x_{jt}$$

$$\text{s.t.} \quad \sum_{t=1}^{T-p_j+1} x_{jt} = 1, \quad \text{for all } j=1,\ldots,n$$

$$\sum_{j=1}^{n} \sum_{s=\max 0, t-p_j+1}^{t} x_{js} \leq 1, \quad \text{for each } t=1,\ldots,T$$

$$x_{jt} \in \{0,1\}, \quad \text{for each } j=1,\ldots,n;\ t=1,\ldots,T$$

+ This formulation gives better bounds than the two preceding

− pseudo-polynomial number of variables

Summary

$1 \mid \mid \sum w_j C_j$: 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

Multiobjective: Multicriteria Optimization

Multiobjective Scheduling

Multiobjective scheduling

Resolution process and decision maker intervention:

- a priori methods (definition of weights, importance)
  - goal programming
  - weighted sum
  - ...
- interactive methods
- a posteriori methods
  - Pareto optimality
  - ...

Lexicographic Order

$$\alpha \mid \beta \mid \gamma_1(opt)\gamma_2$$

Find all solutions optimal for γ_1 and among them those optimal for γ_2
Examples:

- $1||\sum C_j(opt), L_{\max}$
  solved by breaking ties: SPT/EDD
- $1||L_{\max}(opt), \sum C_j$
  impose $L_{\max}$ as hard constraint on due dates

Pareto Optimality

S set of solutions $\Longrightarrow \vec{z} \in \mathbf{R}^k$ image in the objective space

partial order structure defined in $\mathbf{R}^k$:
$\forall x, y \in \mathbf{R}^k : x \leq y \Leftrightarrow x_i \leq y_i \forall i = 1, \ldots, k$

Definition (Weak Pareto optimium)

$x \in S$ is a weak Pareto optimium (weakly efficient solution)
$\Leftrightarrow \nexists y \in S \mid \forall i = 1, \ldots, k,\ z_i(y) < z_i(x)$

Definition (Strong Pareto optimium)

$x \in S$ is a weak Pareto optimium (weakly efficient solution)
$\Leftrightarrow \nexists y \in S \mid \forall i = 1, \ldots, k,\ z_i(y) \leq z_i(x)$

Outline

1. Single Machine Models
   - Branch and Bound
   - Mathematical Programming Models

2. Parallel Machine Models
   - CPM/PERT

$Pm \,|\,|\, C_{max}$
(without preemption)

$P\infty \,|\, prec \,|\, C_{max}$ CPM

$Pm \mid \mid C_{max}$ (without preemption)

$P\infty \mid prec \mid C_{max}$ CPM

$Pm \mid \mid C_{max}$ LPT heuristic, approximation ratio: $\frac{4}{3} - \frac{1}{3m}$

$Pm \,|\,|\, C_{max}$ (without preemption)

$P\infty \,|\, prec \,|\, C_{max}$ CPM

$Pm \,|\,|\, C_{max}$ LPT heuristic, approximation ratio: $\frac{4}{3} - \frac{1}{3m}$

$Pm \,|\, prec \,|\, C_{max}$ strongly NP-hard, LNS heuristic (non optimal)

$Pm \mid \mid C_{max}$
(without preemption)

$P\infty \mid prec \mid C_{max}$ CPM

$Pm \mid \mid C_{max}$ LPT heuristic, approximation ratio: $\frac{4}{3} - \frac{1}{3m}$

$Pm \mid prec \mid C_{max}$ strongly NP-hard, LNS heuristic (non optimal)

$Pm \mid p_j = 1, M_j \mid C_{max}$ LFJ-LFM (optimal if M_j are nested)

Outline

1. Single Machine Models
   - Branch and Bound
   - Mathematical Programming Models

2. Parallel Machine Models
   - CPM/PERT

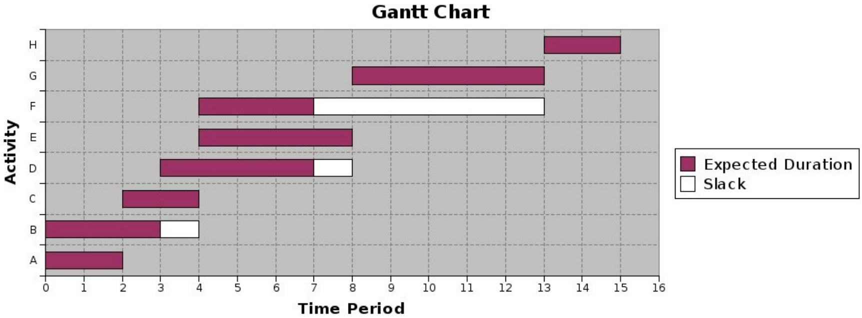
Project Planning

Milwaukee General Hospital Project			
Activity	**Description**	**Immediate Predecessor**	***Duration***
A	Build internal components	–	2
B	Modify roof and floor	–	3
C	Construct collection stack	A	2
D	Pour concrete and install frame	A,B	4
E	Build high-temperature burner	C	4
F	Install pollution control system	C	3
G	Install air pollution device	D,E	5
H	Inspect and test	F,G	2

Project Planning

Milwaukee General Hospital Project								
Activity	Description	Immediate Predecessor	*Duration*	EST	EFT	LST	LFT	Slack
A	Build internal components	–	2	0	2	0	2	0
B	Modify roof and floor	–	3	0	3	1	4	1
C	Construct collection stack	A	2	2	4	2	4	0
D	Pour concrete and install frame	A,B	4	3	7	6	10	3
E	Build high-temperature burner	C	4	4	8	6	10	2
F	Install pollution control system	C	3	4	7	10	13	6
G	Install air pollution device	D,E	5	8	13	8	13	0
H	Inspect and test	F,G	2	13	15	13	15	0
		Expected project duration			***15***			

Project Planning



Project Planning

Milwaukee General Hospital Projec			Expecte d						Time Estimates			Activity Varianc
Activity	Description	Immediate Predecessor	*(a+4m+b)/(*	EST	EFT	LST	LFT	Slack	*a*	*m*	*b*	*((b-a)/6)^2*
A	Build internal components	–	2	0	2	0	2	0	1	2	3	0.1111
B	Modify roof and floor	–	3	0	3	1	4	1	2	3	4	0.1111
C	Construct collection stack	A	2	2	4	2	4	0	1	2	3	0.1111
D	'our concrete and install frame	A,B	4	3	7	4	8	1	2	4	6	0.4444
E	3uild high-temperature burne	C	4	4	8	4	8	0	1	4	7	1.0000
F	nstall pollution control systerr	C	3	4	7	10	13	6	1	2	9	1.7778
G	Install air pollution device	D,E	5	8	13	8	13	0	3	4	11	1.7778
H	Inspect and test	F,G	2	13	15	13	15	0	1	2	3	0.1111
		Expected project duration			***15***	*Variance of project duration*						***3.1111***