DM204 – Spring 2013

Scheduling, Timetabling and Routing

Lecture 4

Single Machine Problems
MILP approaches
Advanced Methods for MILP

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Course Overview

- ✔ Scheduling
  - ✔ Classification
  - ✔ Complexity issues
  - ✔ 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

Complexity resume

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

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

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

[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

```
1  LIST := {S};
2  U:=value of some heuristic solution;
3  current_best := heuristic solution;
4  while LIST ≠ ∅
5     Choose a branching node k from LIST;
6     Remove k from LIST;
7     Generate children child(i), i = 1, ..., n_k, and calculate corresponding lower
            bounds LB_i;
8     for i:=1 to n_k
9        if LB_i < U then
10          if child(i) consists of a feasible solution then
11             U:=LB_i;
12             current_best:=solution corresponding to child(i)
13          else add child(i) to LIST
14 // else prune
```

Pruning: (i) by bound (ii) by optimality (iii) by infeasibility

Branch and Bound

Branch and bound *vs* backtracking

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

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

≠ branch and bound is used only for optimization problems.

Branch and bound *vs* A*

= In A* the admissible heuristic mimics bounding

≠ In A* there is no branching. It is a search algorithm.

≠ 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
- 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
  (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

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

$1||\sum w_j T_j$

- **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, introduce costs
  $\sum_{t=1}^{T-p_j} c_{jt} = h_j(t+p_j)\ \forall j = 1..n; t = 1..(T-p_j)$

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

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

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

[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.

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

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

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

Outline

IP Models
Sequencing variables

$1|prec|\sum w_j C_j$

Sequencing (linear ordering) variables

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

$$\text{s.t. } x_{kj} + x_{jl} + x_{lk} \geq 1 \qquad j,k,l = 1,\ldots,n; j \neq k, k \neq l$$

$$x_{kj} + x_{jk} = 1 \qquad \forall j,k = 1,\ldots,n, j \neq k$$

$$x_{jk} \in \{0,1\} \qquad j,k = 1,\ldots,n$$

$$x_{jj} = 0 \qquad \forall j = 1,\ldots,n$$

IP Models
Completion time

$1|prec|C_{max}$

Completion time variables $\in \mathbb{R}$ and job precedences $\in \mathbb{B}$ for disjunctive constraints

$$\min \sum_{j=1}^{n} w_j z_j$$

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

IP Models
Time indexed variables

$1||\sum h_j(C_j)$

Time indexed variables

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

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

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

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

+ The LR of this formulation gives better bounds than the two preceding
+ Flexible with respect to objective function
– Pseudo-polynomial number of variables

Advanced Methods for MILP

Moved to next lecture

Back to our Time-Indexed Formulation

$$\begin{aligned} \max\ & c^T x \\ \text{s.t. } & Ax \le b \\ & Dx \le d \\ & x \in \mathbb{Z}^n_+ \end{aligned} \qquad \begin{aligned} \max\ & c^T x \\ \text{s.t. } & Ax \le b \\ & x \in P \end{aligned} \qquad \text{(IP)}$$

polytope $P = \{x \in \mathbb{Z}^n : Dx \le d\}$

Assuming that P is bounded and has a finite number of points $\{x^s\}, s \in Q$ it can be represented by its extreme points $x^1, \ldots, x^k$:

$$x^s = \sum_{k=1}^{K} \lambda_k x^k, \text{ with } \sum_{k=1}^{K} \lambda_k = 1, \lambda_k \ge 0$$

substituting in (IP) leads to DW master problem:

$$\begin{aligned} \max\ & \sum_k (cx^k)\lambda_k \\ \text{s.t. } & \sum_k (Ax^k)\lambda_k \le b \\ & \sum_{k=1}^{K} \lambda_k = 1 \\ & \lambda_k \ge 0 \end{aligned} \qquad \text{(MP)}$$

Dantzig-Wolfe decomposition

Reformulation:

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

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

$$x_{jt} \in X \qquad \text{for each } j = 1, \ldots, n;\ t = 1, \ldots, T - p_j + 1$$

where $X = \left\{ x \in \{0,1\} : \sum_{j=1}^{n} \sum_{s=t-p_j+1}^{t} x_{js} \leq 1, \text{ for each } t = 1, \ldots, T \right\}$

$x^l, l = 1, \ldots, L$ extreme points of X.

$$X = \left\{ \begin{array}{c} x \in \{0,1\} : x = \sum_{l=1}^{L} \lambda_l x^l \\ \sum_{l=1}^{L} \lambda_l = 1, \\ \lambda_l \in \{0,1\} \end{array} \right\}$$

matrix of X is interval matrix

extreme points are integral

they are pseudo-schedules

Dantzig-Wolfe decomposition

Substituting X in original model getting master problem

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

$$\pi \quad \text{s.t.} \quad \sum_{l=1}^{L} \left(\sum_{t=1}^{T-p_j+1} x_{jt}^l \right) \lambda_l = 1, \quad \text{for all } j = 1, \ldots, n \Longleftarrow \sum_{l=1}^{L} n_j^l \lambda_l = 1$$

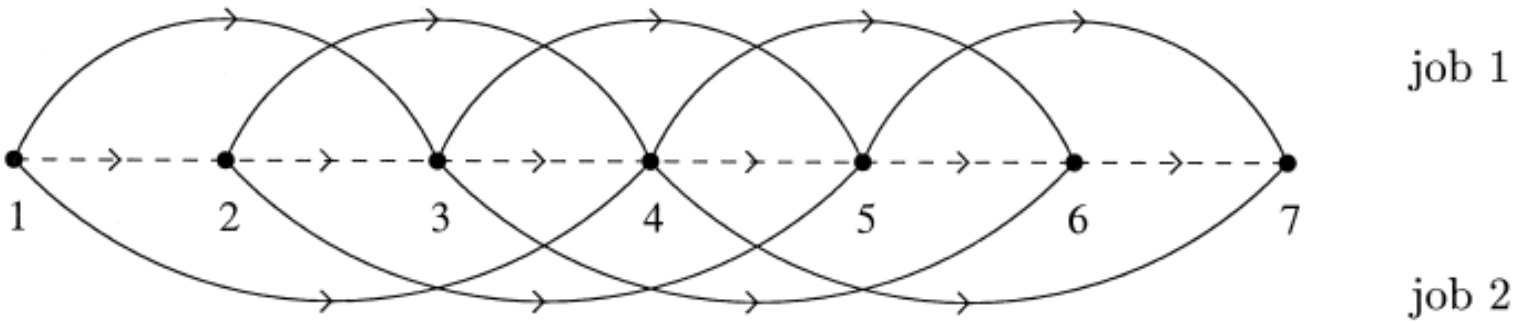
$$\alpha \qquad \sum_{l=1}^{L} \lambda_l = 1,$$

$$\lambda_l \in \{0,1\} \Longleftarrow \lambda_l \geq 0 \text{ LP-relaxation}$$

- n_j^l number of times job j appears in pseudo-schedule l
- solve LP-relaxation by column generation on pseudo-schedules x^l
- reduced cost of λ_k is $\bar{c}_k = \sum_{j=1}^{n} \sum_{t=1}^{T-p_j+1} (c_{jt} - \pi_j) x_{jt}^k - \alpha$

Pricing problem

- Subproblem solved by finding shortest path in a network N with
  - $1, 2, \ldots, T+1$ nodes corresponding to time periods
  - process arcs, for all j, t, $t \rightarrow t + p_j$ and cost $c_{jt} - \pi_j$
  - idle time arcs, for all t, $t \rightarrow t + 1$ and cost 0



- a path in this network corresponds to a pseudo-schedule in which a job may be started more than once or not processed.
- since network is directed and acyclic, shortest path found in $O(nT)$

Further Readings

- the lower bound on the master problem produced by the LP-relaxation of the restricted master problem can be tighten by inequalities

  J. van den Akker, C. Hurkens and M. Savelsbergh. Time-Indexed Formulations for Machine Scheduling Problems: Column Generation. INFORMS Journal On Computing, 2000, 12(2) , 111-124

- *A. Pessoa, E. Uchoa, M.P. de AragÃ£o and R. Rodrigues. Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. 2010, 2, 259-290*

  proposes another time index formulation that dominates this one. They can solve consistently instances up to 100 jobs.

References

Fisher M.L. (2004). **The Lagrangian relaxation method for solving integer programming problems**. *Management Science*, 50(12), pp. 1861–1871. This article originally appeared in Management Science, January 1981, Volume 27, Number 1, pp. 1-18, published by The Institute of Management Sciences.

Pan Y. and Shi L. (2007). **On the equivalence of the max-min transportation lower bound and the time-indexed lower bound for single-machine scheduling problems**. *Mathematical Programming*, 110(3), pp. 543–559.