UNIVERSITAT POMPEU FABRA

A Polynomial Algorithm for a Special Case of the One-Machine Scheduling Problem with Time-Lags

Helena Ramalhinho Lourenço

Universitat Pompeu Fabra Barcelona

Barcelona, Spain
helena.ramalhinho@upf.edu
http://www.econ.upf.edu/~ramalhin/

1

Outline of the Presentation

- Introduction
- The one machine scheduling problem with time-lags
- The early-late algorithm
- Computational results
- Conclusions
- Directions of future work.

2

Introduction

- We consider a one-machine scheduling problem minimizing the maximum lateness.
- Each job is associated with
  - a release date,
  - a processing time
  - a delivery time.
- There are precedence constraints between some pairs of jobs as well as a time interval, the finish-start time-lags.

3

Introduction

- In presence of these constraints, the problem is *NP*-hard even if preemption is allowed.
- We consider a special case of the one-machine preemption scheduling problem with time-lags in chain form.
- And we propose a polynomial algorithm to solve it.
- One of the applicability is to obtain lower bounds for NP-hard one-machine and job-shop scheduling problems.

4

The one machine scheduling problem

- The one-machine scheduling problem:
  - a set of jobs have to be scheduled on one machine.
  - each job has a release date, a processing time, and a delivery time.
  - Each job cannot be processed before its release time.
  - At most one job can be processed at a time, all jobs can be simultaneously delivered.
  - Preemption is allowed.

5

The one machine scheduling problem

- There also can exists precedence constrains, <, between the jobs.
  - In presence of precedence constraints, a job cannot start processing before the previous one has been finished.
  - the completion time of the first job and the starting time of the second job there must exists a time interval known as finish-start time-lags.
  - These time lags have a chain form:

$$J_i < J_k < \ldots < J_l$$

6

The one machine scheduling problem

- Notation
  - n jobs $J=\{J_1, J_2, \ldots, J_n\}$
  - r_j : release date
  - d_j : due date
  - q_j : delivery time
  - C_j : completion time
  - Lateness
    $$L_j = C_j - d_j$$
- The objective is to minimize the maximum lateness (L_{max})

7

The Early-Late Algorithm

- Previous…
  - The Longest Tail Rule (LTRTL) schedules the jobs sequentially choosing at each step the job with the longest delivery time among those not scheduled yet.
    - * Can easily be adapted to allow preemption (pLTRTL).
    - * Obtains a feasible solution for the problem.
  - The Horn's algorithm, Horn (1974) is the similar to the LTRTL but considers preemption and no time-lags.

8

The Early-Late Algorithm

- The early-late algorithm finds the optimal solution of the problem.
- It is particular case of the enumeration method proposed by Lourenço (1993) to the general one-machine scheduling problem with time lags (in a general form)
- It considers only polynomial number of tree nodes of the previous enumeration method.

9

The Early-Late Algorithm

- The basic idea of the algorithm is to obtain a lower bound and an upper bound at each node i of the tree.
- The lower bound is obtained by applying the Horn's rule to a modified instance (ignoring the time lags).
- The upper bound is obtained by applying the pLTRTL (considering the time lags).

10

The Early-Late Algorithm

- If the upper bound is equal to the lower bound the algorithm stops because that optimal solution was found.
  - Node i, instance I_i

$$L_{\max}(\sigma_i, I_i) = LB(I_i)$$

- Otherwise some job is scheduled late or early…

11

The Early-Late Algorithm

$$L_{\max}(\sigma_i, I_i) = \min_{j \in K} \bar{r}_j + \sum_{j \in K} p_j + q_c > LB(I_i) \geq$$

$$\min_{j \in K} r_j + \sum_{j \in K} p_j + \min_{j \in K} q_j = L(K)$$

- One chain job is scheduled late if:

$$\min_{j \in K} \bar{r}_j > \min_{j \in K} r_j$$

- One chain job is schedule early if:

$$q_c > \min_{j \in K} q_j$$

12

UNIVERSITAT POMPEU FABRA

The Early-Late Algorithm

- Then the instance I_i is modified by applying some dominance rules.
- The only modifications needed are the release dates and delivery times (priorities) of the chain jobs.
- When a modification is made, it means that we are changing from one node to another in the search tree of the enumerative method.
- The algorithm runs in polynomial time.

13

Computational Results

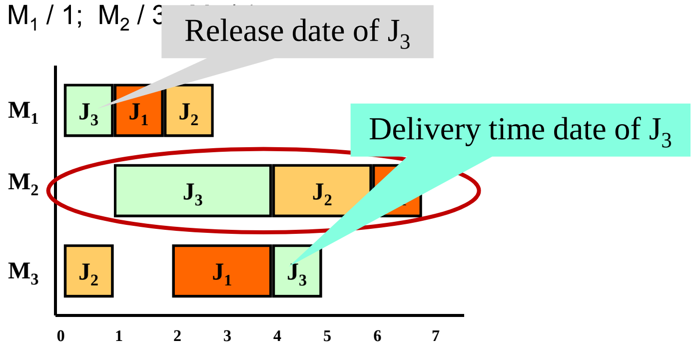
- We consider 10 instances of the one-machine scheduling problem obtained by relaxations of the job-shop scheduling problem.
- The results obtained are lower bounds for the job-shop scheduling problem.

14

UNIVERSITAT POMPEU FABRA

The Job-Shop Scheduling

- Example:
  - * J_1: M_1 / 1; M_3 / 2; M_2 / 1;
  - * J_2: M_3 / 1; M_1 / 1; M_2 / 2;
  - * J_3: M_1 / 1; M_2 / 3



15

Computational Results

- Horn's Rule. Problem: No time lags, pmtn.
- Early-Late algorithm. Problem: Chain time-lags, pmtn.
- Enumerative method. Problem: All time lags, pmtn.
- Carlier's algorithm. Problem: No time lags, no pmtn.
- Enumerative method. Problem: Chain time-lags, no pmtn.
- Enumerative method. Problem: All time lags, no pmtn.

16

Computational Results

	Polynomial		Enumerative methods			
Examples	1	2	3	4	5	6
(LA19)	798	807	813	807	807	832
(MT10)	911	911	911	911	911	911
(MT10)	917	917	917	917	917	917
(MT10)	836	836	836	836	836	836
(MT10)	884	884	884	892	892	892
(ABZ5)	1101	1116	1116	1108	1116	1116
(LA19)	735	752	752	747	755	755
(ABZ5)	1147	1157	1157	1147	1157	1157
(MT10)	884	884	884	892	892	892
(MT10)	918	918	918	918	918	918

Conclusions

- In this work, we considered a special case of the one-machine scheduling with time-lags in a chain form and allowing preemption.
- The main contribution of this work is the presentation of a polynomial algorithm, the early-late algorithm.
- This algorithm can be used to obtain lower-bounds within branch-and-bound methods to other complex scheduling problems, as the job-shop problem.

Future Research

- Consider precedence constraints in a bipartite graph form and study how to solve such problems.
- Apply the early-late algorithm within a Matheuristic method to solve complex sheduling problems.
  - Apply search heuristics combined with exact methods.

19