Single machine models

Observation:

- for non-preemptive problems and regular objectives, a sequence in which the jobs are processed is sufficient to describe a solution

Dispatching (priority) rules

- static rules - not time dependent
  e.g. shortest processing time first, earliest due (in absence of release dates) date first
- dynamic rules - time dependent
  e.g. largest *available* job (i.e. $r_j \leq t$); t current time
- for some problems dispatching rules lead to optimal solutions

Single machine models: $1||\sum w_j C_j$

Given:

- n jobs with processing times $p_1, \ldots, p_n$ and weights $w_1, \ldots, w_n$

Consider case: $w_1 = \ldots = w_n (= 1)$:

Single machine models: $1||\sum w_j C_j$

Given:

- n jobs with processing times $p_1, \ldots, p_n$ and weights $w_1, \ldots, w_n$

Consider special case: $w_1 = \ldots = w_n (= 1)$:

- SPT-rule: shortest processing time first
- Theorem: SPT is optimal for $1||\sum C_j$
  Proof: by an exchange argument (on board)
- Complexity: $O(n \log(n))$

Single machine models: $1||\sum w_jC_j$

General case

- WSPT-rule: weighted shortest processing time first, i.e. sort jobs by increasing p_j/w_j-values
- Theorem: WSPT is optimal for $1||\sum w_jC_j$
  Proof: by an exchange argument (exercise)
- Complexity: $O(n\log(n))$

Further results:

- $1|tree|\sum w_jC_j$ can be solved by in polynomial time ($O(n\log(n))$) (see [Brucker 2004])
- $1|prec|\sum C_j$ is NP-hard in the strong sense (see [Brucker 2004])

Single machine models: $1|prec|f_{max}$

Given:

- n jobs with processing times $p_1, \ldots, p_n$
- precedence constraints between the jobs
- non-decreasing functions $f_1, \ldots, f_n$
- objective criterion $f_{max} = \max\{f_1(C_1), \ldots, f_n(C_n)\}$

Observation:

- completion time of last job $= \sum p_j$

Single machine models: $1|prec|f_{max}$

Given:

- n jobs with processing times $p_1, \ldots, p_n$
- precedence constraints between the jobs
- regular functions $f_1, \ldots, f_n$
- objective criterion $f_{max} = \max\{f_1(C_1), \ldots, f_n(C_n)\}$

Observation:

- completion time of last job $= \sum p_j$

Method

- plan backwards from $\sum p_j$ to 0
- from all available jobs (jobs from which all successors have already been scheduled), schedule job which is 'cheapest' on that position

Single machine models: $1|prec|f_{max}$

S set of already scheduled jobs (initial: $S = \emptyset$)

J set of all jobs, which successors have been scheduled (initial: all jobs without successors)

t time where next job will be completed (initial: $t = \sum p_j$)

Algorithm $1|prec|f_{max}$ (Lawler's Algorithm)

REPEAT

select $j \in J$ such that $f_j(t) = \min_{k \in J} f_k(t)$;
schedule j such that it completes at t;
add j to S, delete j from J and update J;
$t := t - p_j$;

UNTIL $J = \emptyset$.

Single machine models: $1|prec|f_{max}$

- Theorem: Algorithm $1|prec|f_{max}$ is optimal for $1|prec|f_{max}$
  Proof: on the board
- Complexity: $O(n^2)$

Problem $1||L_{max}$:

- Earliest due date first (EDD) is optimal for $1||L_{max}$ (Jackson's EDD rule)
- Proof: special case of Lawler's algorithm

Problem $1|r_j|C_{max}$:

Problem $1||L_{max}$:

- Earliest due date first (EDD) is optimal for $1||L_{max}$ (Jackson's EDD rule)
- Proof: special case of Lawler's algorithm

Problem $1|r_j|C_{max}$:

- $1|r_j|C_{max} \propto 1||L_{max}$
  - define $d_j := K - r_j$, with constant $K > \max r_j$
  - reversing the optimal schedule of this $1||L_{max}$-problem gives an optimal schedule for the $1|r_j|C_{max}$-problem

Problem $1|prec|L_{max}$:

- if $d_j < d_k$ whenever $j \to k$, any EDD schedule respects the precedence constraints, i.e. in this case EDD is optimal
- defining $d_j := \min\{d_j, d_k - p_k\}$ if $j \to k$ does not increase L_{max} in any feasible schedule

Problem $1|prec|L_{max}$:

- if $d_j < d_k$ whenever $j \to k$, any EDD schedule respects the precedence constraints, i.e. in this case EDD is optimal
- defining $d_j := \min\{d_j, d_k - p_k\}$ if $j \to k$ does not increase L_{max} in any feasible schedule

Algorithm $1|prec|L_{max}$

1. make due dates consistent: set $d_j = \min\{d_j, \min_{k|j\to k} d_k - p_k\}$
2. apply EDD rule with modified due dates

Remarks on Algorithm $1|prec|L_{max}$

- leads to an optimal solution
- Step 1 can be realized in $O(n^2)$
- problem $1|prec|L_{max}$ can be solved without knowledge of the processing times, whereas Lawler's Algorithm (which also solves this problem) in general needs this knowledge (Exercise),
- Problem $1|r_j, prec|C_{max} \propto 1|prec|L_{max}$

Problem $1|r_j|L_{max}$:

- problem $1|r_j|L_{max}$ is NP-hard
- Proof: by reduction from 3-PARTITION (on the board)

Problem $1|pmtn, r_j|L_{max}$:

- preemptive EDD-rule: at each point in time, schedule an available job (job, which release date has passed) with earliest due date.
- preemptive EDD-rule leads to at most k preemptions ($k =$ number of distinct release dates)

Problem $1|pmtn, r_j|L_{max}$:

- preemptive EDD-rule: at each point in time, schedule an available job (job, which release date has passed) with earliest due date.
- preemptive EDD-rule leads to at most k preemptions ($k =$ number of distinct release dates)
- preemptive EDD solves problem $1|pmtn, r_j|L_{max}$
- Proof (on the board) uses following results:
  - $L_{max} \geq r(S) + p(S) - d(S)$ for any $S \subset \{1, \ldots, n\}$, where $r(S) = \min_{j \in S} r_j,\ p(S) = \sum_{j \in S} p_j,\ d(S) = \max_{j \in S} d_j$
  - preemptive EDD leads to a schedule with $L_{max} = \max_{S \subset \{1,\ldots,n\}} r(S) + p(S) - d(S)$

Remarks on preemptive EDD-rule for $1|pmtn, r_j|L_{max}$:

- can be implemented in $O(n \log(n))$
- is an 'on-line' algorithm
- after modification of release and due-dates, preemptive EDD solves also $1|prec, pmtn, r_j|L_{max}$

Approximation algorithms for problem $1|r_j|L_{max}$:

- a polynomial algorithm A is called an α-approximation for problem P if for every instance I of P algorithm A yields an objective value $f_A(I)$ which is bounded by a factor α of the optimal value $f^*(I)$; i.e. $f_A(I) \leq \alpha f^*(I)$

Approximation algorithms for problem $1|r_j|L_{max}$:

- a polynomial algorithm A is called an α-approximation for problem P if for every instance I of P algorithm A yields an objective value $f_A(I)$ which is bounded by a factor α of the optimal value $f^*(I)$; i.e. $f_A(I) \leq \alpha f^*(I)$
- for the objective L_{max}, α-approximation does not make sense since L_{max} may get negative
- for the objective T_{max}, an α-approximation with a constant α implies $\mathcal{P} = \mathcal{NP}$ (if $T_{max} = 0$ an α-approximation is optimal)

The head-body-tail problem $(1|r_j, d_j < 0|L_{max})$

- n jobs
- job j: release date r_j (head), processing time p_j (body), delivery time q_j (tail)
- starting time $S_j \geq r_j$;
- completion time $C_j = S_j + p_j$
- delivered at $C_j + q_j$
- goal: minimize $\max_{j=1}^{n} C_j + q_j$

The head-body-tail problem ($1|r_j, d_j < 0|L_{max}$), (cont.)

- define $d_j = -q_j$, i.e. the due dates get negative!
- result: $\min_{j=1}^{n} C_j + q_j = \min_{j=1}^{n} C_j - d_j = \min_{j=1}^{n} L_j = L_{max}$
- head-body-tail problem equivalent with $1|r_j|L_{max}$-problem with negative due dates
  Notation: $1|r_j, d_j < 0|L_{max}$
- an instance of the head-body-tail problem defined by n triples (r_j, p_j, q_j) is equivalent to an inverse instance defined by n triples (q_j, p_j, r_j)
- for the head-body-tail problem considering approximation algorithms makes sense

The head-body-tail problem $(1|r_j, d_j < 0|L_{max})$, (cont.)

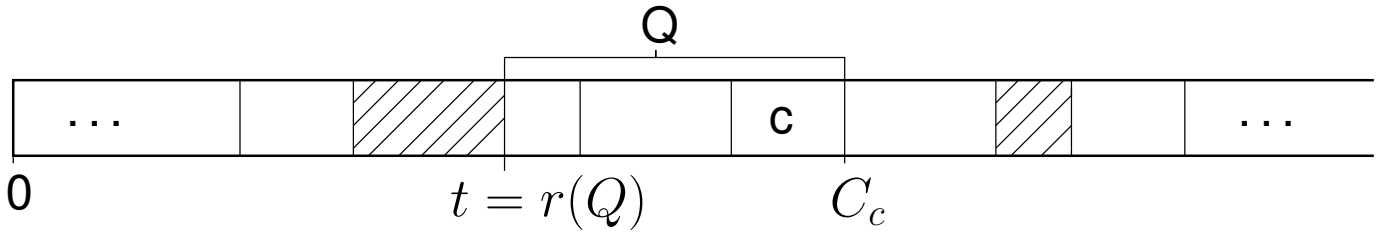
- $L_{max} \geq r(S) + p(S) + q(S)$ for any $S \subset \{1, \ldots, n\}$, where

$$r(S) = \min_{j \in S} r_j, \; p(S) = \sum_{j \in S} p_j, \; q(S) = \min_{j \in S} q_j$$

(follows from $L_{max} \geq r(S) + p(S) - d(S)$ - slide 5)

Approximation ratio for EDD for problem $1|r_j, d_j < 0|L_{max}$

- structure of a schedule S



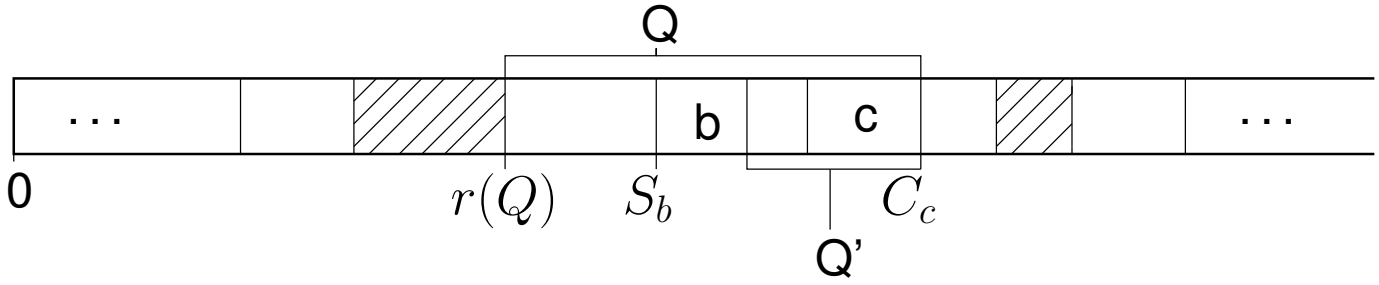
- critical job c of a schedule: job with $L_c = \max L_j$
- critical sequence Q: jobs processed in the interval $[t, C_c]$, where t is the earliest time that the machine is not idle in $[t, C_c]$
- if $q_c = \min_{j \in Q} q_j$ the schedule is optimal since then

$$L_{max}(S) = L_c = C_c - d_c = r(Q) + p(Q) + q(Q) \leq L^*_{max}$$

- Notation: L^*_{max} denotes the optimal value

Approximation ratio for EDD for problem $1|r_j, d_j < 0|L_{max}$

- structure of a schedule



- interference job b: last scheduled job from Q with $q_b < q_c$
- Lemma: For the objective value $L_{max}(EDD)$ of an EDD schedule we have: (Proofs on the board)
  1. $L_{max}(EDD) - L^*_{max} < q_c$
  2. $L_{max}(EDD) - L^*_{max} < p_b$
- Theorem: EDD is 2-approximation algorithm for $1|r_j, d_j < 0|L_{max}$