Interval Scheduling

Arash Rafiey

Interval Scheduling Problem

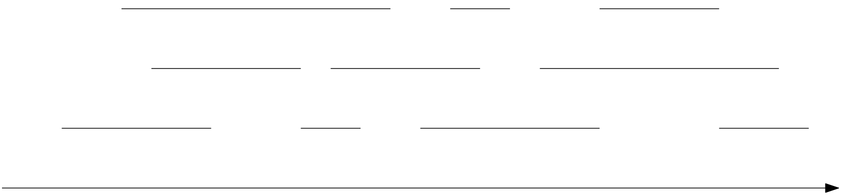
Problem Statement: We have a resource and many people request to use the resource for periods of time.

Conditions:

- the resource can be used by at most one person at a time.
- we can accept only *compatible* requests (overlap-free).

Goal: maximize the number of *compatible* requests accepted.

Key Idea: greedy approach.



Greedy Algorithm for Interval Scheduling Problem

(1) Initially let R be the set of all requests, and let A be empty;

(2) While R is not empty:

    (i) Choose a request $i \in R$ that has the smallest finishing time

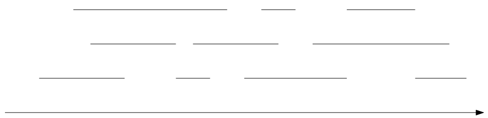
    (ii) Add request i to A;

    (iii) Delete all requests from R that are not compatible with request i

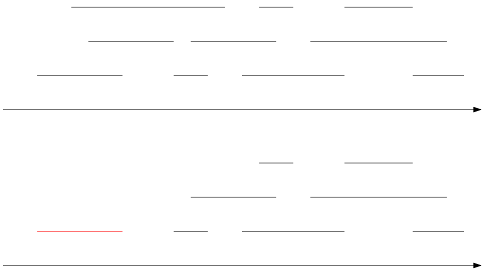
(3) EndWhile

(4) Return the set A as the set of accepted requests

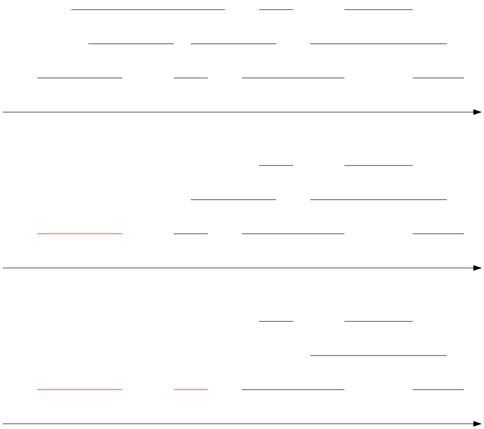
Example



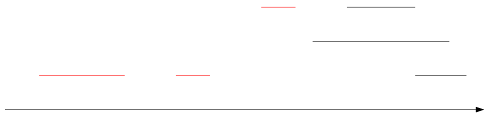
Example



Example



Example(Con.)



Example(Con.)

Example(Con.)

Let OPT be the optimal solution (optimal scheduling). Let S be the scheduling produced by the greedy algorithm.

Let OPT be the optimal solution (optimal scheduling). Let S be the scheduling produced by the greedy algorithm.

Let I be the first interval from left to right that is in S but not in OPT.

Let OPT be the optimal solution (optimal scheduling). Let S be the scheduling produced by the greedy algorithm.

Let I be the first interval from left to right that is in S but not in OPT.

Suppose J is the interval in OPT that has been considered instead of I. We know that the right end of J is not before the right end of I.

Let OPT be the optimal solution (optimal scheduling). Let S be the scheduling produced by the greedy algorithm.

Let I be the first interval from left to right that is in S but not in OPT.

Suppose J is the interval in OPT that has been considered instead of I. We know that the right end of J is not before the right end of I.

In other words, the finish time of J is not smaller than the finish time of I.

Let OPT be the optimal solution (optimal scheduling). Let S be the scheduling produced by the greedy algorithm.

Let I be the first interval from left to right that is in S but not in OPT.

Suppose J is the interval in OPT that has been considered instead of I. We know that the right end of J is not before the right end of I.

In other words, the finish time of J is not smaller than the finish time of I.

Now obtain $OPT1$ by removing J from OPT and adding I. It is clear that no interval after J in OPT has overlap with I. Therefore $OPT1$ is a valid scheduling and have the same number of intervals as OPT. Therefore OPT_1 is optimal.

Let OPT be the optimal solution (optimal scheduling). Let S be the scheduling produced by the greedy algorithm.

Let I be the first interval from left to right that is in S but not in OPT.

Suppose J is the interval in OPT that has been considered instead of I. We know that the right end of J is not before the right end of I.

In other words, the finish time of J is not smaller than the finish time of I.

Now obtain $OPT1$ by removing J from OPT and adding I. It is clear that no interval after J in OPT has overlap with I. Therefore $OPT1$ is a valid scheduling and have the same number of intervals as OPT. Therefore OPT_1 is optimal.

By keep repeading the above procedure we eventually get OPT_r that is the same as S. Therefore S is optimal.

How many resources do we need to finish all the requests ? (minimum number of compatible sets of all the requests (in each set there is no overlap))

Scheduling to Minimize Lateness

Problem Statement: We have a resource and many people request to use the resource for periods of time.

Conditions:

- the resource can be used by at most one person at a time.
- we can accept only *compatible* requests (overlap-free).
- the request i has deadline d_i, and it requires time interval of length t_i.
- *lateness* of each request i is $f(i) - d_i$ ($f(i)$ finish time of i)

Notation: we assign each request an interval $[s(i), f(i)]$ where $t_i = f(i) - s(i)$.

Goal: minimize the *maximum lateness*.

Scheduling to Minimize Lateness

Key Idea: Earliest Deadline First.

Key Idea: Earliest Deadline First.

Algorithm

(i) Order the jobs in order of their deadlines $d_1 \leq d_2 \leq \cdots \leq d_n$;

(ii) Initially $f = s$;

(iii) for jobs $i = 1, \cdots, n$:
Do job i in the time interval $[s(i), f(i)]$ where $s(i) = f$ and $f(i) = s(i) + t_i$;
Let $f = f + t_i$;

(iv) return intervals $[s(i), f(i)]$ for $i = 1, \cdots, n$;

Scheduling to Minimize Lateness

Why this works ?

Why this works ?

(*)Let assume there is an optimal solution with no idle time.

Scheduling to Minimize Lateness

Why this works ?

(*)Let assume there is an optimal solution with no idle time.

Definition

A schedule A' has an inversion if a job i with deadline d_i is scheduled before another job j with earlier deadline $d_j < d_i$.

Scheduling to Minimize Lateness

Why this works ?

(*)Let assume there is an optimal solution with no idle time.

Definition

A schedule A' has an inversion if a job i with deadline d_i is scheduled before another job j with earlier deadline $d_j < d_i$.

Lemma

All schedules with no inversions and no idle time have the same maximum lateness.

Why this works ?

(*)Let assume there is an optimal solution with no idle time.

Definition

A schedule A' has an inversion if a job i with deadline d_i is scheduled before another job j with earlier deadline $d_j < d_i$.

Lemma

All schedules with no inversions and no idle time have the same maximum lateness.

Theorem

There is an optimal solution that has no inversions and no idle time.

Suppose at time t the optimal processes J_i first and then J_j with $d_j < d_i$. We show that we obtain another solution (as good as optimal) in which J_j is processed before J_i.

Suppose at time t the optimal processes J_i first and then J_j with $d_j < d_i$. We show that we obtain another solution (as good as optimal) in which J_j is processed before J_i.

Let L_i be the lateness of J_i and L_j be the lateness of J_j. By definition $L_i = t + t_i - d_i$ and $L_j = t + t_i + t_j - d_j$

Since $d_j < d_i$ we have $L_i < L_j$.

Suppose at time t the optimal processes J_i first and then J_j with $d_j < d_i$. We show that we obtain another solution (as good as optimal) in which J_j is processed before J_i.

Let L_i be the lateness of J_i and L_j be the lateness of J_j. By definition $L_i = t + t_i - d_i$ and $L_j = t + t_i + t_j - d_j$

Since $d_j < d_i$ we have $L_i < L_j$.

Now consider a solution S obtained from optimal by exchanging J_i and J_j at time t (all the other jobs remained unchanged).

Suppose at time t the optimal processes J_i first and then J_j with $d_j < d_i$. We show that we obtain another solution (as good as optimal) in which J_j is processed before J_i.

Let L_i be the lateness of J_i and L_j be the lateness of J_j. By definition $L_i = t + t_i - d_i$ and $L_j = t + t_i + t_j - d_j$

Since $d_j < d_i$ we have $L_i < L_j$.

Now consider a solution S obtained from optimal by exchanging J_i and J_j at time t (all the other jobs remained unchanged).

Let L'_i be the lateness of J_i and L'_j be the lateness of J_j in S. By definition $L'_i = t + t_j + t_i - d_i$ and $L'_j = t + t_j - d_j$

$L'_j \leq L_j$ because $t + t_j - d_j < t + t_i + t_j - d_j$.

Suppose at time t the optimal processes J_i first and then J_j with $d_j < d_i$. We show that we obtain another solution (as good as optimal) in which J_j is processed before J_i.

Let L_i be the lateness of J_i and L_j be the lateness of J_j. By definition $L_i = t + t_i - d_i$ and $L_j = t + t_i + t_j - d_j$

Since $d_j < d_i$ we have $L_i < L_j$.

Now consider a solution S obtained from optimal by exchanging J_i and J_j at time t (all the other jobs remained unchanged).

Let L'_i be the lateness of J_i and L'_j be the lateness of J_j in S. By definition $L'_i = t + t_j + t_i - d_i$ and $L'_j = t + t_j - d_j$

$L'_j \leq L_j$ because $t + t_j - d_j < t + t_i + t_j - d_j$.

Case 1. $L'_i < L'_j$ **:** Then we can consider S instead of optimal (the maximum lateness does not happen at time t).

Suppose at time t the optimal processes J_i first and then J_j with $d_j < d_i$. We show that we obtain another solution (as good as optimal) in which J_j is processed before J_i.

Let L_i be the lateness of J_i and L_j be the lateness of J_j. By definition $L_i = t + t_i - d_i$ and $L_j = t + t_i + t_j - d_j$

Since $d_j < d_i$ we have $L_i < L_j$.

Now consider a solution S obtained from optimal by exchanging J_i and J_j at time t (all the other jobs remained unchanged).

Let L'_i be the lateness of J_i and L'_j be the lateness of J_j in S. By definition $L'_i = t + t_j + t_i - d_i$ and $L'_j = t + t_j - d_j$

$L'_j \leq L_j$ because $t + t_j - d_j < t + t_i + t_j - d_j$.

Case 1. $L'_i < L'_j$ **:** Then we can consider S instead of optimal (the maximum lateness does not happen at time t).

Case 2 $L'_j < L'_i$ **:**
Now $L'_i < L_j$ because $t + t_j + t_i - d_i < t + t_i + t_j - d_j$.

Therefore we consider S instead of optimal (S is also good).

Scheduling Jobs with Deadlines and Profits

Problem Statement: We have a resource and many people request to use the resource for periods of time.

Conditions:

- the resource can be used by at most one person at a time.
- we can accept only *compatible* requests (overlap-free).
- the request i has deadline d_i, and it has to be done before its deadline.
- each job has a profit g_i.
- each job has a unit length.

Goal: find a feasible schedule with *maximum profit* .

Scheduling to Minimize Lateness

Key Idea: Find the best place for the job with maximum profit.

Scheduling to Minimize Lateness

Key Idea: Find the best place for the job with maximum profit.

Algorithm

(i) Sort the jobs so that: $g_1 \geq g_2 \geq \cdots \geq g_n$;

(ii) for jobs $i = 1, \cdots, n$:
schedule job i in the <u>latest</u> possible free slot meeting its deadline ;
if there is no such slot, do not schedule i;

(iv) return intervals $[s(i), f(i)]$ for $i = 1, \cdots, n$;

Minimizing the Average Completion Time

Problem Definition:

1. We are given a set of n jobs $J_1, J_2, \ldots, J_n$
2. Each job J_i, $1 \leq i \leq n$ has a processing time p_i
3. Each job J_i, $1 \leq i \leq n$ has weight $w_i \geq 0$

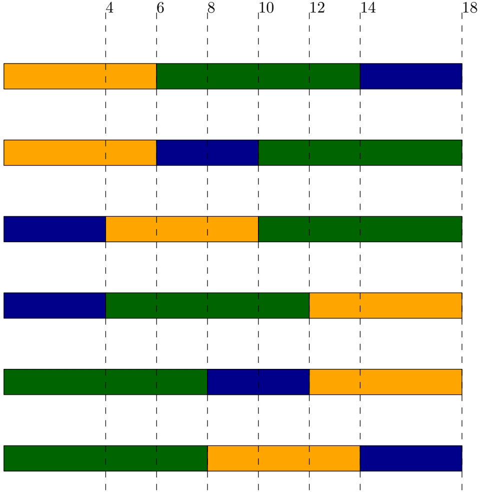
Goal :

Scheduling of the jobs to minimize :

$$\sum_{i=1}^{i=n} w_i C_i$$

Here C_i is the completion time of job J_i.

$W_{orange} = 2 \qquad W_{blue} = 2 \qquad W_{green} = 6$



$Sum = 2 \times 6 + 6 \times 14 + 2 \times 18 = 132$

$Sum = 2 \times 6 + 2 \times 10 + 6 \times 18 = 140$

$Sum = 2 \times 4 + 2 \times 10 + 6 \times 18 = 136$

$Sum = 2 \times 4 + 6 \times 12 + 2 \times 18 = 116$

$Sum = 6 \times 8 + 2 \times 12 + 2 \times 18 = 108$

$Sum = 6 \times 8 + 2 \times 14 + 2 \times 18 = 112$

Order the jobs based on $\frac{p_i}{w_i}$.

Suppose $\frac{p_1}{w_1} \leq \frac{p_2}{w_2} \leq \ldots \frac{p_{n-1}}{w_{n-1}} \leq \frac{p_n}{w_n}$

Then the ordering of the jobs is $J_1, J_2, \ldots, J_n$.

Proof : Let OPT be the optimal solution in which J_j is right before J_i and $\frac{p_j}{w_j} > \frac{p_i}{w_i}$. Otherwise, our ordering is optimal.

Let X be the value average completion time of OPT.
Let OPT1 be the schedule in which we switch J_i and J_j (J_i comes right before J_j in OPT1).
We compare X and $X1$ (which is the average time of OPT1).
Suppose job J_j starts at time t in OPT. Thus, we have

$X = U + (t + p_j)w_j + (t + p_j + p_i)w_i + V$ and
$X1 = U + (t + p_i)w_i + (t + p_i + p_j)w_j + V$.
Here $t + p_j$ is the C_j (finish time of J_j), and $C_i = t + p_i + p_j$ is the finish time of J_i in OPT. Similarly $(t + p_i)$ and $(t + p_i + p_j)$ are the finish time of J_i and J_j in OPT1.

Now $X - X1 = p_j w_i - p_i w_j > 0$, because $\frac{p_j}{w_j} > \frac{p_i}{w_i}$.
Therefore, OPT1 has a smaller average time which is a contradiction to OPT being optimal.