CS 218 Design and Analysis of Algorithms

Nutan Limaye

Indian Institute of Technology, Bombay
nutan@cse.iitb.ac.in

Module 3: NP hardness and reductions

Are all problems efficiently solvable?

In Module 1 and 2 we saw many problems that are efficiently solvable.

Are all problems efficiently solvable?

In Module 1 and 2 we saw many problems that are efficiently solvable.

One might feel that indeed all problems ARE efficiently solvable.

Are all problems efficiently solvable?

In Module 1 and 2 we saw many problems that are efficiently solvable.

One might feel that indeed all problems ARE efficiently solvable.

Is this true?

Are all problems efficiently solvable?

In Module 1 and 2 we saw many problems that are efficiently solvable.

One might feel that indeed all problems ARE efficiently solvable.

Is this true?

The answer is

Are all problems efficiently solvable?

In Module 1 and 2 we saw many problems that are efficiently solvable.

One might feel that indeed all problems ARE efficiently solvable.

Is this true?

The answer is we do not know.

Are all problems efficiently solvable?

In Module 1 and 2 we saw many problems that are efficiently solvable.

One might feel that indeed all problems ARE efficiently solvable.

Is this true?

The answer is we do not know.

Are all problems efficiently solvable?

There are problems believed to be impossible to solve efficiently.

Are all problems efficiently solvable?

There are problems believed to be impossible to solve efficiently.

Unfortunately, many of them are extremely important problems.

Are all problems efficiently solvable?

There are problems believed to be impossible to solve efficiently.

Unfortunately, many of them are extremely important problems.

Which we would like to solve efficiently.

Are all problems efficiently solvable?

There are problems believed to be impossible to solve efficiently.

Unfortunately, many of them are extremely important problems.

Which we would like to solve efficiently.

In this module we will see some examples of such problems.

Are all problems efficiently solvable?

There are problems believed to be impossible to solve efficiently.

Unfortunately, many of them are extremely important problems.

Which we would like to solve efficiently.

In this module we will see some examples of such problems.

We will also build a theory around such problems.

Are all problems efficiently solvable?

There are problems believed to be impossible to solve efficiently.

Unfortunately, many of them are extremely important problems.

Which we would like to solve efficiently.

In this module we will see some examples of such problems.

We will also build a theory around such problems.

Scheduling problem

Jobs and processors.

Scheduling problem

Jobs and processors.

We have 2 processors. P_1, P_2.

Scheduling problem

Jobs and processors.

We have 2 processors. P_1, P_2.

We have jobs $J_1, \ldots, J_n$ with durations $d_1, \ldots, d_n$, respectively.

Scheduling problem

Jobs and processors.

We have 2 processors. P_1, P_2.

We have jobs $J_1, \ldots, J_n$ with durations $d_1, \ldots, d_n$, respectively. We assume that d_is are positive integers.

Scheduling problem

Jobs and processors.

We have 2 processors. P_1, P_2.

We have jobs $J_1, \ldots, J_n$ with durations $d_1, \ldots, d_n$, respectively. We assume that d_is are positive integers.

A schedule $\mathcal{S}$ is an ordering of these jobs on the two processors.

Scheduling problem

Jobs and processors.

We have 2 processors. P_1, P_2.

We have jobs $J_1, \ldots, J_n$ with durations $d_1, \ldots, d_n$, respectively. We assume that d_is are positive integers.

A schedule $\mathcal{S}$ is an ordering of these jobs on the two processors.

E.g. If we have jobs J_1, J_2, J_3, J_4. A schedule $\mathcal{S} = (\sigma_1, \sigma_2)$ could be $(\langle 1, 4 \rangle, \langle 3, 2 \rangle)$.

Scheduling problem

Jobs and processors.

We have 2 processors. P_1, P_2.

We have jobs $J_1, \ldots, J_n$ with durations $d_1, \ldots, d_n$, respectively. We assume that d_is are positive integers.

A schedule $\mathcal{S}$ is an ordering of these jobs on the two processors.

E.g. If we have jobs J_1, J_2, J_3, J_4. A schedule $\mathcal{S} = (\sigma_1, \sigma_2)$ could be $(\langle 1, 4\rangle, \langle 3, 2\rangle)$.

This says that schedule job 1 followed by job 4 on P_1 and schedule job 3 on followed by 2 on P_2.

Scheduling problem

Jobs and processors.

We have 2 processors. P_1, P_2.

We have jobs $J_1, \ldots, J_n$ with durations $d_1, \ldots, d_n$, respectively. We assume that d_is are positive integers.

A schedule $\mathcal{S}$ is an ordering of these jobs on the two processors.

E.g. If we have jobs J_1, J_2, J_3, J_4. A schedule $\mathcal{S} = (\sigma_1, \sigma_2)$ could be $(\langle 1, 4\rangle, \langle 3, 2\rangle)$.

This says that schedule job 1 followed by job 4 on P_1 and schedule job 3 on followed by 2 on P_2.

Scheduling problem

Jobs and processors.

We have 2 processors. P_1, P_2.

We have jobs $J_1, \ldots, J_n$ with durations $d_1, \ldots, d_n$, respectively. We assume that d_is are positive integers.

Scheduling problem

Jobs and processors.

We have 2 processors. P_1, P_2.

We have jobs $J_1, \ldots, J_n$ with durations $d_1, \ldots, d_n$, respectively. We assume that d_is are positive integers.

A schedule $\mathcal{S} = (\sigma_1, \sigma_2)$ is an ordering of these jobs on the two processors.

Let S_1 be the set of jobs scheduled on P_1.

Scheduling problem

Jobs and processors.

We have 2 processors. P_1, P_2.

We have jobs $J_1, \ldots, J_n$ with durations $d_1, \ldots, d_n$, respectively. We assume that d_is are positive integers.

A schedule $\mathcal{S} = (\sigma_1, \sigma_2)$ is an ordering of these jobs on the two processors.

Let S_1 be the set of jobs scheduled on P_1.

The total completion time of a schedule $\mathcal{S}$ is $\max\left\{\sum_{i \in S_1} d_i, \sum_{i \in [n] \setminus S_1} d_i\right\}$.

Scheduling problem

Given: jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively

Scheduling problem

Given: jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find: a schedule for these jobs on 2 processors that

Scheduling problem

Given: jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find: a schedule for these jobs on 2 processors that minimises the total completion time.

Scheduling problem

Given: jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find: a schedule for these jobs on 2 processors that minimises the total completion time.

Quite similar to the interval scheduling problem from Module 1.

Scheduling problem

Given:	jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find:	a schedule for these jobs on 2 processors that minimises the total completion time.

Quite similar to the interval scheduling problem from Module 1.

Some key differences.

Scheduling problem

Given: jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find: a schedule for these jobs on 2 processors that minimises the total completion time.

Quite similar to the interval scheduling problem from Module 1.

Some key differences.

No begin and end times, just durations.

Scheduling problem

Given: jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find: a schedule for these jobs on 2 processors that minimises the total completion time.

Quite similar to the interval scheduling problem from Module 1.

Some key differences.

No begin and end times, just durations.

Multiple processors instead of just one.

Scheduling problem

Given:	jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find:	a schedule for these jobs on 2 processors that minimises the total completion time.

Quite similar to the interval scheduling problem from Module 1.

Some key differences.

No begin and end times, just durations.

Multiple processors instead of just one.

All jobs must be eventually done.

Scheduling problem

Given: jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find: a schedule for these jobs on 2 processors that minimises the total completion time.

Quite similar to the interval scheduling problem from Module 1.

Some key differences.

No begin and end times, just durations.

Multiple processors instead of just one.

All jobs must be eventually done.

Will this problem be harder or easier to solve?

Scheduling problem

Given: jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find: a schedule for these jobs on 2 processors that minimises the total completion time.

Quite similar to the interval scheduling problem from Module 1.

Some key differences.

No begin and end times, just durations.

Multiple processors instead of just one.

All jobs must be eventually done.

Will this problem be harder or easier to solve?

Scheduling problem

Given: jobs $j_1, \dots, j_n$ with durations $d_1, \dots, d_n$ respectively
Find: a schedule for these jobs on 2 processors that minimises the total completion time.

A possible schedule. $\mathcal{S} = (\langle 1, 3, 5, \dots, n-1 \rangle, \langle 2, 4, 6, \dots, n \rangle)$.

Scheduling problem

Given: jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find: a schedule for these jobs on 2 processors that minimises the total completion time.

A possible schedule. $\mathcal{S} = (\langle 1, 3, 5, \ldots, n-1 \rangle, \langle 2, 4, 6, \ldots, n \rangle)$. Say n even.

Will this work?

Scheduling problem

Given: jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find: a schedule for these jobs on 2 processors that minimises the total completion time.

A possible schedule. $\mathcal{S} = (\langle 1, 3, 5, \ldots, n-1 \rangle, \langle 2, 4, 6, \ldots, n \rangle)$. Say n even.

Will this work?

Scheduling problem

Given: jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find: a schedule for these jobs on 2 processors that minimises the total completion time.

A greedy heuristic.

Scheduling problem

Given: jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find: a schedule for these jobs on 2 processors that minimises the total completion time.

A greedy heuristic.

Schedule a job on the processor which is currently least loaded.

Scheduling problem

Given:	jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find:	a schedule for these jobs on 2 processors that minimises the total completion time.

A greedy heuristic.

Schedule a job on the processor which is currently least loaded.

Will this work?

Scheduling problem

Given:	jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find:	a schedule for these jobs on 2 processors that minimises the total completion time.

A greedy heuristic.

Schedule a job on the processor which is currently least loaded.

Will this work?

Scheduling problem

Given:	jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find:	a schedule for these jobs on 2 processors that minimises the total completion time.

A modified greedy heuristic.

Scheduling problem

Given:	jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find:	a schedule for these jobs on 2 processors that minimises the total completion time.

A modified greedy heuristic.

Sort the jobs in ascending order of their duration.

Scheduling problem

Given:	jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find:	a schedule for these jobs on 2 processors that minimises the total completion time.

A modified greedy heuristic.

Sort the jobs in ascending order of their duration.

Use this order and do as before, i.e.

Scheduling problem

Given:	jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find:	a schedule for these jobs on 2 processors that minimises the total completion time.

A modified greedy heuristic.

Sort the jobs in ascending order of their duration.
Use this order and do as before, i.e.
Schedule a job on the processor which is currently least loaded.

Scheduling problem

Given: jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find: a schedule for these jobs on 2 processors that minimises the total completion time.

A modified greedy heuristic.

Sort the jobs in ascending order of their duration.
Use this order and do as before, i.e.
Schedule a job on the processor which is currently least loaded.

Will this work?

Scheduling problem

Given: jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively
Find: a schedule for these jobs on 2 processors that minimises the total completion time.

A modified greedy heuristic.

Sort the jobs in ascending order of their duration.
Use this order and do as before, i.e.
Schedule a job on the processor which is currently least loaded.

Will this work?