COL351: Analysis and Design of Algorithms

Ragesh Jaiswal, CSE, IITD

Greedy (Approximation) Algorithms

Greedy (Approximation) Algorithms

- For some problems, even though the greedy strategy does not give an optimal solution but it might give a solution that is provably close to the optimal solution.

- Covering set: Let S be a set containing n elements. A set of subsets $\{S_1, ..., S_m\}$ of S is called a covering set if each element in S is present in at least one of the subsets $S_1, ..., S_m$.

Problem

Set Cover: Given a set S containing n elements and m subsets $S_1, ..., S_m$ of S. Find a covering set of S of minimum cardinality.

- Example
  - $S = \{a, b, c, d, e, f\}$
  - $S_1 = \{a, b\}$, $S_2 = \{a, c\}$, $S_3 = \{b, c\}$, $S_4 = \{d, e, f\}$, $S_5 = \{e, f\}$
  - $\{S_1, S_2, S_3, S_4\}$ is a covering set.
  - $\{S_1, S_2, S_4\}$ is a covering set of minimum cardinality.

Problem

Set Cover: Given a set S containing n elements and m subsets $S_1, ..., S_m$ of S. Find a covering set of S of minimum cardinality.

- Application: There are n villages and the government is trying to figure out which villages to open schools at so that it has to open minimum number of schools. The constraint is that no children should have to walk more than 3 miles to get to a school.

Problem

Set Cover: Given a set S containing n elements and m subsets $S_1, ..., S_m$ of S. Find a covering set of S of minimum cardinality.

- Greedy strategy: Give preference to the subsets that covers the most number of (remaining) elements.

Algorithm

`GreedySetCover`$(S, S_1, ..., S_m)$

- $T \leftarrow \{\}$; $R \leftarrow S$
- While R is not empty:
  - Pick a subset S_i that covers the maximum number of elements in R
  - $T \leftarrow T \cup \{S_i\}$; $R \leftarrow R - S_i$

Problem

Set Cover: Given a set S containing n elements and m subsets $S_1, ..., S_m$ of S. Find a covering set of S of minimum cardinality.

- Greedy strategy: Give preference to the subsets that covers the most number of (remaining) elements.

Algorithm

`GreedySetCover`$(S, S_1, ..., S_m)$
- $T \leftarrow \{\}$; $R \leftarrow S$
- While R is not empty:
  - Pick a subset S_i that covers the maximum number of elements in R
  - $T \leftarrow T \cup \{S_i\}$; $R \leftarrow R - S_i$

- Counterexample: $S = \{a, b, c, d, e, f, g, h\}, S_1 = \{a, b, c, d, e\}, S_2 = \{a, b, c, f\}, S_3 = \{d, e, g, h\}$.

Algorithm

`GreedySetCover`$(S, S_1, ..., S_m)$

- $T \leftarrow \{\}$; $R \leftarrow S$
- While R is not empty:
  - Pick a subset S_i that covers the maximum number of elements in R
  - $T \leftarrow T \cup \{S_i\}$; $R \leftarrow R - S_i$

- Claim 1: Let k be the cardinality of any optimal covering set. Then the greedy algorithm outputs a covering set with cardinality at most $k \cdot \ln n$.

Algorithm

`GreedySetCover`$(S, S_1, ..., S_m)$

- $T \leftarrow \{\}$; $R \leftarrow S$
- While R is not empty:
  - Pick a subset S_i that covers the maximum number of elements in R
  - $T \leftarrow T \cup \{S_i\}$; $R \leftarrow R - S_i$

- Claim 1: Let k be the cardinality of any optimal covering set. Then the greedy algorithm outputs a covering set with cardinality at most $k \cdot \ln n$.

Proof of Claim 1

- Let N_t be the number of uncovered elements after t iterations of the loop.
- Claim 1.1: $N_t \leq (1 - 1/k) \cdot N_{t-1}$.

Algorithm

`GreedySetCover`$(S, S_1, ..., S_m)$

- $T \leftarrow \{\}$; $R \leftarrow S$
- While R is not empty:
  - Pick a subset S_i that covers the maximum number of elements in R
  - $T \leftarrow T \cup \{S_i\}$; $R \leftarrow R - S_i$

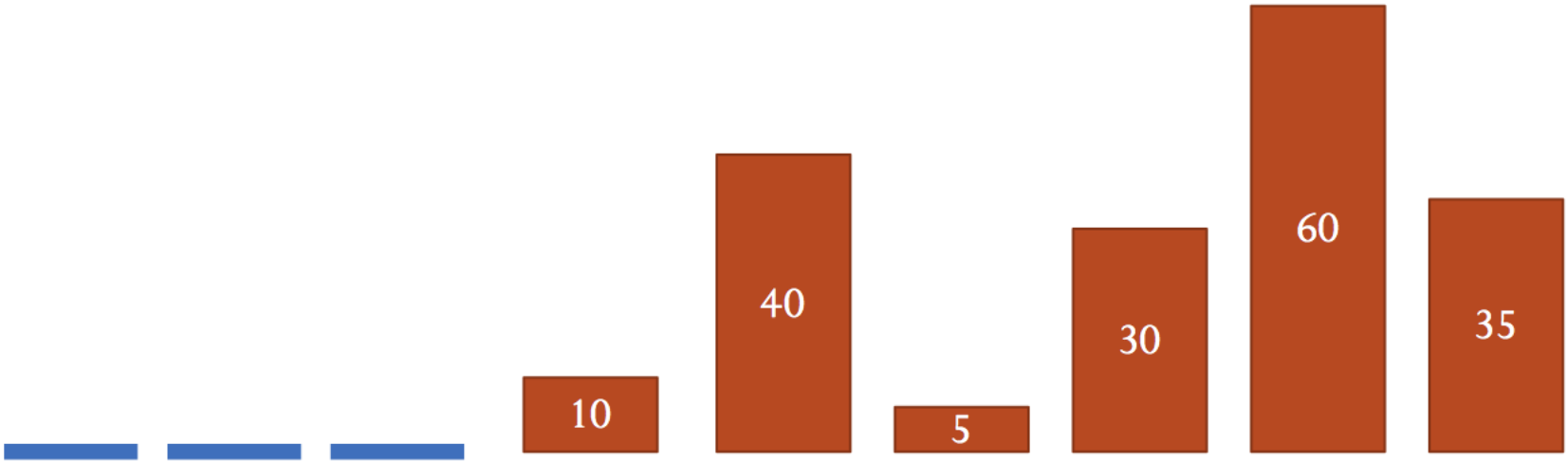
- Claim 1: Let k be the cardinality of any optimal covering set. Then the greedy algorithm outputs a covering set with cardinality at most $k \cdot \ln n$.

Proof of Claim 1

- Let N_t be the number of uncovered elements after t iterations of the loop.
- Claim 1.1: $N_t \leq (1 - 1/k) \cdot N_{t-1}$.
- Claim 1.2: $N_{k \cdot \ln n} < 1$.
  - Use the fact that $(1 - x) \leq e^{-x}$ and the equality holds only for $x = 0$.

Problem

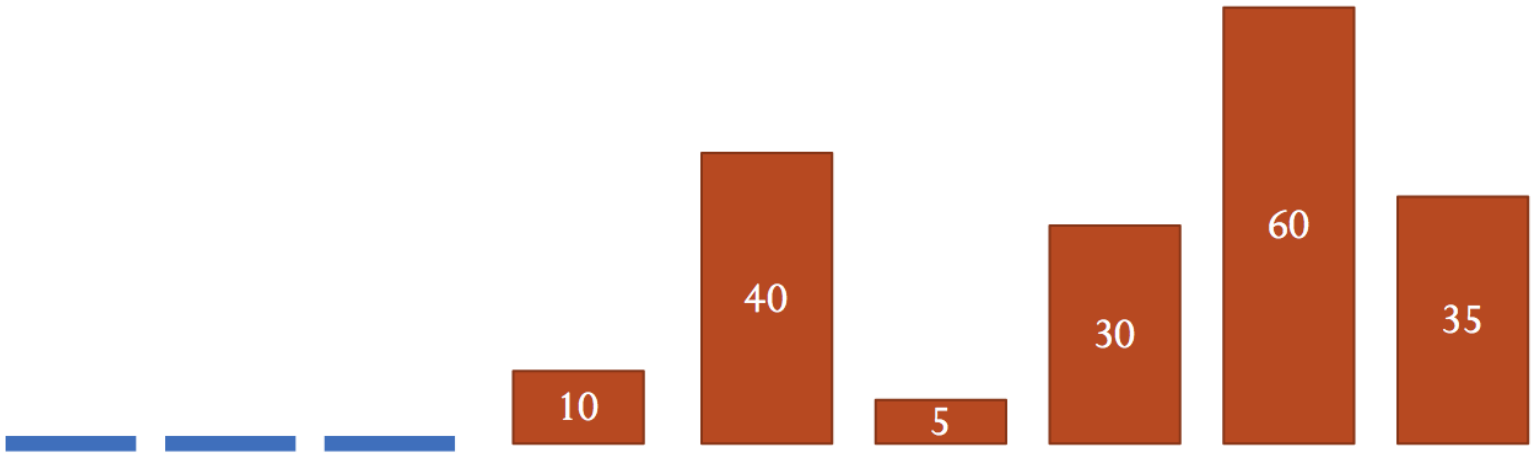
Minimum Makespan: You have m identical machines and n jobs. For each job i, you are given the duration of this job $d(i)$ that denotes the time that is required by any machine to perform this job. Assign these n jobs on m machine such that the maximum finishing time is minimized.



Problem

Minimum Makespan: You have m identical machines and n jobs. For each job i, you are given the duration of this job $d(i)$ that denotes the time that is required by any machine to perform this job. Assign these n jobs on m machine such that the maximum finishing time is minimized.

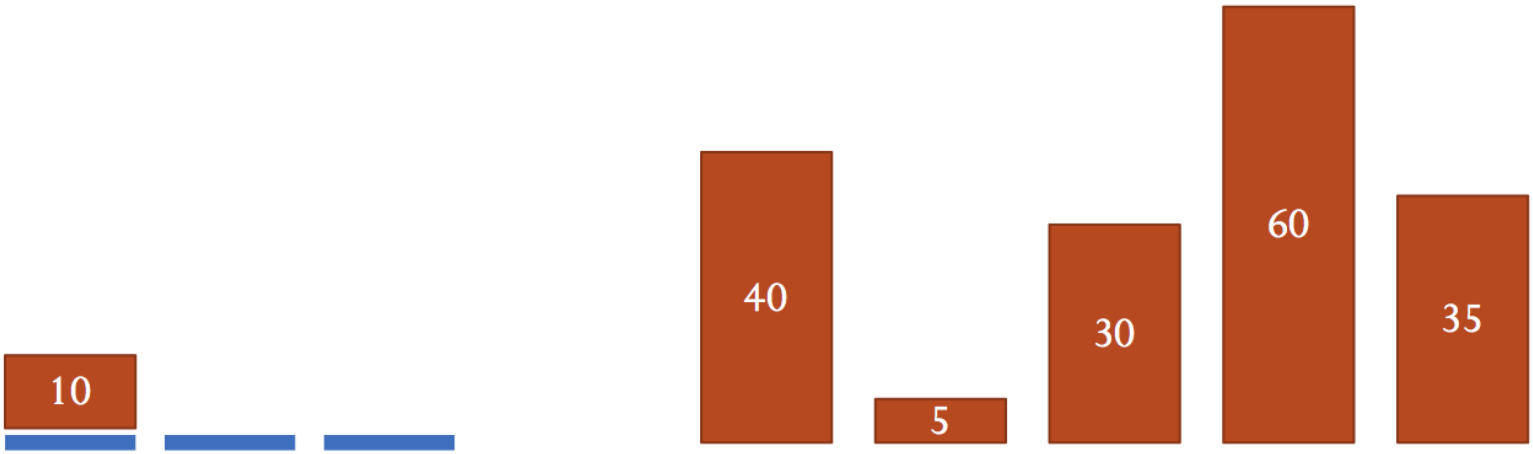
- Greedy strategy: Assign the next job to a machine with least load.



Problem

Minimum Makespan: You have m identical machines and n jobs. For each job i, you are given the duration of this job $d(i)$ that denotes the time that is required by any machine to perform this job. Assign these n jobs on m machine such that the maximum finishing time is minimized.

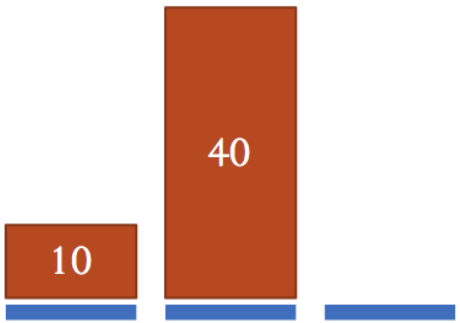
- Greedy strategy: Assign the next job to a machine with least load.

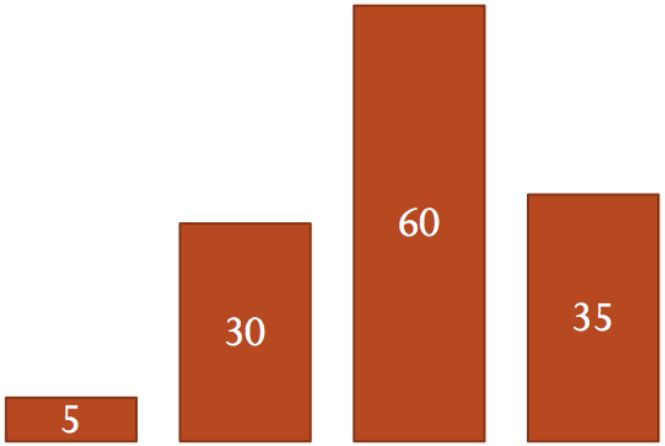


Problem

Minimum Makespan: You have m identical machines and n jobs. For each job i, you are given the duration of this job $d(i)$ that denotes the time that is required by any machine to perform this job. Assign these n jobs on m machine such that the maximum finishing time is minimized.

- Greedy strategy: Assign the next job to a machine with least load.

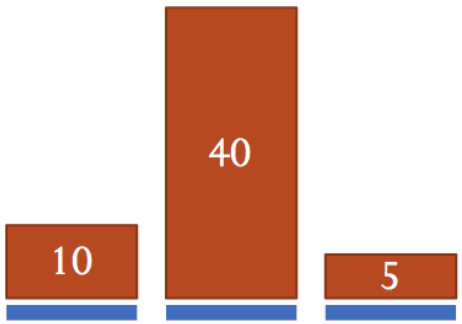


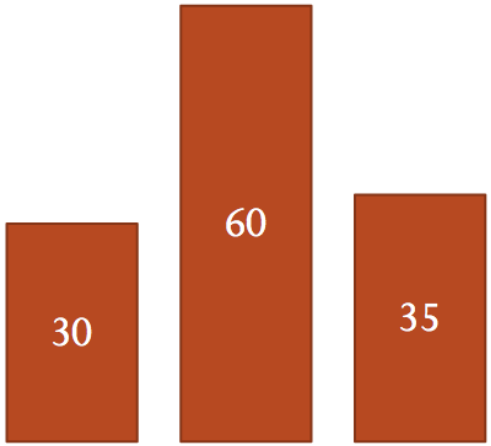


Problem

Minimum Makespan: You have m identical machines and n jobs. For each job i, you are given the duration of this job $d(i)$ that denotes the time that is required by any machine to perform this job. Assign these n jobs on m machine such that the maximum finishing time is minimized.

- Greedy strategy: Assign the next job to a machine with least load.

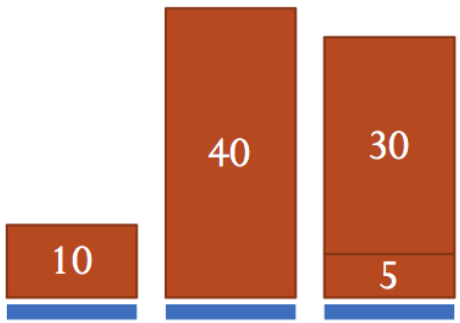


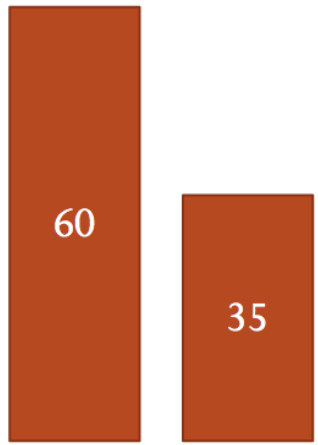


Problem

Minimum Makespan: You have m identical machines and n jobs. For each job i, you are given the duration of this job $d(i)$ that denotes the time that is required by any machine to perform this job. Assign these n jobs on m machine such that the maximum finishing time is minimized.

- Greedy strategy: Assign the next job to a machine with least load.

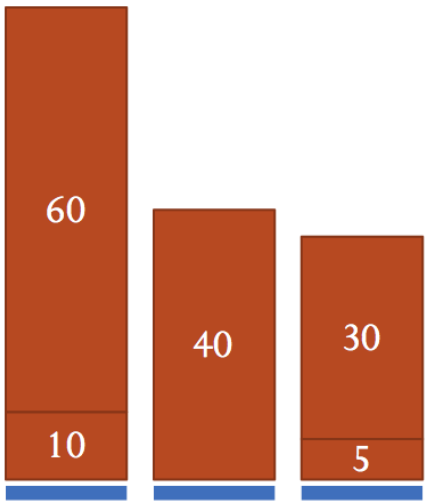




Problem

Minimum Makespan: You have m identical machines and n jobs. For each job i, you are given the duration of this job $d(i)$ that denotes the time that is required by any machine to perform this job. Assign these n jobs on m machine such that the maximum finishing time is minimized.

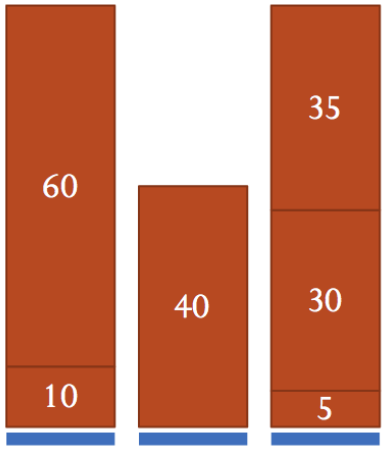
- Greedy strategy: Assign the next job to a machine with least load.



Problem

Minimum Makespan: You have m identical machines and n jobs. For each job i, you are given the duration of this job $d(i)$ that denotes the time that is required by any machine to perform this job. Assign these n jobs on m machine such that the maximum finishing time is minimized.

- Greedy strategy: Assign the next job to a machine with least load.

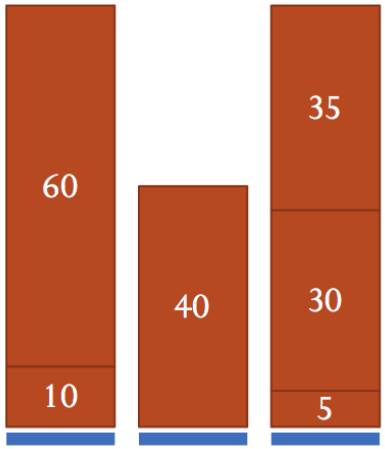


- Is this solution optimal?

Problem

Minimum Makespan: You have m identical machines and n jobs. For each job i, you are given the duration of this job $d(i)$ that denotes the time that is required by any machine to perform this job. Assign these n jobs on m machine such that the maximum finishing time is minimized.

- Greedy strategy: Assign the next job to a machine with least load.



- Is this solution optimal? No

Algorithm

`GreedyMakespan`

- While all jobs are not assigned
  - Assign the next job to a machine with least load

- Let OPT be the optimal value.
- Let G denote the maximum finishing time of a machine as per the greedy assignment.
- Claim 1: $G \leq 2 \cdot OPT$.

Algorithm

`GreedyMakespan`

- While all jobs are not assigned
  - Assign the next job to a machine with least load

- Let OPT be the optimal value.
- Let G denote the maximum finishing time of a machine as per the greedy assignment.
- Claim 1: $G \leq 2 \cdot OPT$.

Proof of Claim 1

- Claim 1.1: $OPT \geq \frac{d(1)+d(2)+...+d(n)}{m}$

Algorithm

`GreedyMakespan`

- While all jobs are not assigned
  - Assign the next job to a machine with least load

- Let OPT be the optimal value.
- Let G denote the maximum finishing time of a machine as per the greedy assignment.
- Claim 1: $G \leq 2 \cdot OPT$.

Proof of Claim 1

- Claim 1.1: $OPT \geq \frac{d(1)+d(2)+...+d(n)}{m}$
- Claim 1.2: For any job t, $OPT \geq d(t)$.

Algorithm

`GreedyMakespan`

- While all jobs are not assigned
  - Assign the next job to a machine with least load

- Let OPT be the optimal value.
- Let G denote the maximum finishing time of a machine as per the greedy assignment.
- Claim 1: $G \leq 2 \cdot OPT$.

Proof of Claim 1

- Claim 1.1: $OPT \geq \frac{d(1)+d(2)+\ldots+d(n)}{m}$
- Claim 1.2: For any job t, $OPT \geq d(t)$.
- Let the j^{th} machine finish last. Let i be the last job assigned to machine j. Let s be the start time of job i on machine j.
- Claim 1.3: $s \leq \frac{d(1)+d(2)+\ldots+d(n)}{m}$

Algorithm

`GreedyMakespan`

- While all jobs are not assigned
  - Assign the next job to a machine with least load

- Let OPT be the optimal value.
- Let G denote the maximum finishing time of a machine as per the greedy assignment.
- Claim 1: $G \leq 2 \cdot OPT$.

Proof of Claim 1

- Claim 1.1: $OPT \geq \frac{d(1)+d(2)+...+d(n)}{m}$
- Claim 1.2: For any job t, $OPT \geq d(t)$.
- Let the j^{th} machine finish last. Let i be the last job assigned to machine j. Let s be the start time of job i on machine j.
- Claim 1.3: $s \leq \frac{d(1)+d(2)+...+d(n)}{m}$
- So, $G \leq s + d(i)$
- This implies that $G \leq \frac{d(1)+...+d(n)}{m} + d(i)$ (using claim 1.3)

Algorithm

`GreedyMakespan`

- While all jobs are not assigned
  - Assign the next job to a machine with least load

- Let OPT be the optimal value.
- Let G denote the maximum finishing time of a machine as per the greedy assignment.
- Claim 1: $G \leq 2 \cdot OPT$.

Proof of Claim 1

- Claim 1.1: $OPT \geq \frac{d(1)+d(2)+...+d(n)}{m}$
- Claim 1.2: For any job t, $OPT \geq d(t)$.
- Let the j^{th} machine finish last. Let i be the last job assigned to machine j. Let s be the start time of job i on machine j.
- Claim 1.3: $s \leq \frac{d(1)+d(2)+...+d(n)}{m}$
- So, $G \leq s + d(i)$
- This implies that $G \leq \frac{d(1)+...+d(n)}{m} + d(i)$ (using claim 1.3)
- This implies that $G \leq OPT + d(i)$ (using claim 1.1)

Algorithm

`GreedyMakespan`

- While all jobs are not assigned
  - Assign the next job to a machine with least load

- Let OPT be the optimal value.
- Let G denote the maximum finishing time of a machine as per the greedy assignment.
- Claim 1: $G \leq 2 \cdot OPT$.

Proof of Claim 1

- Claim 1.1: $OPT \geq \frac{d(1)+d(2)+...+d(n)}{m}$
- Claim 1.2: For any job t, $OPT \geq d(t)$.
- Let the j^{th} machine finish last. Let i be the last job assigned to machine j. Let s be the start time of job i on machine j.
- Claim 1.3: $s \leq \frac{d(1)+d(2)+...+d(n)}{m}$
- So, $G \leq s + d(i)$
- This implies that $G \leq \frac{d(1)+...+d(n)}{m} + d(i)$ (using claim 1.3)
- This implies that $G \leq OPT + d(i)$ (using claim 1.1)
- This implies that $G \leq OPT + OPT$ (using claim 1.2)

End