CSL 356: Analysis and Design of Algorithms

Ragesh Jaiswal

CSE, IIT Delhi

Greedy 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.

Greedy Approximation: Examples

- Let S be a set containing n elements. A set of subsets $\{S_1, \dots, S_m\}$ of S is called a covering set if each element in S is present in at least one of the subsets $S_1, \dots, S_m$.
- Problem (Set Cover): Given a set S containing n elements and m subsets $S_1, \dots, S_m$ of S. Find a covering set of S of minimum cardinality.
- Example:
  - $S = \{a, b, c, d, e, f\}$
  - $S_1 = \{a, b\}, S_2 = \{a, c\}, S_3 = \{a, c\}, S_4 = \{d, e, f\}, S_5 = \{e, f\}$
  - $\{S_1, S_2, S_3, S_4\}$ is a covering set.
  - $\{S_1, S_2, S_4\}$ is a covering set of minimum size.

Greedy Approximation: Examples

- Problem (Set Cover): Given a set S containing n elements and m subsets $S_1, \dots, S_m$ 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.

Greedy Approximation: Examples

- Problem (Set Cover): Given a set S containing n elements and m subsets $S_1, \dots, S_m$ of S. Find a covering set of S of minimum cardinality.
- Greedy strategy: Give preference to the subset that covers most number of elements.

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

Greedy Approximation: Examples

- Problem (Set Cover): Given a set S containing n elements and m subsets $S_1, \ldots, S_m$ of S. Find a covering set of S of minimum cardinality.
- Greedy strategy: Give preference to the subset that covers most number of elements.

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

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

Greedy Approximation: Examples

- <u>Claim</u>: Let k be the optimal cardinality of the covering set. Then the greedy algorithm outputs a covering set with cardinality at most $k * \ln(n)$.
- <u>Proof</u>: Let N_t be the number of uncovered elements after t iterations of the loop.
  - <u>Claim</u>: $N_t \leq (1 - 1/k) * N_{t-1}$

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

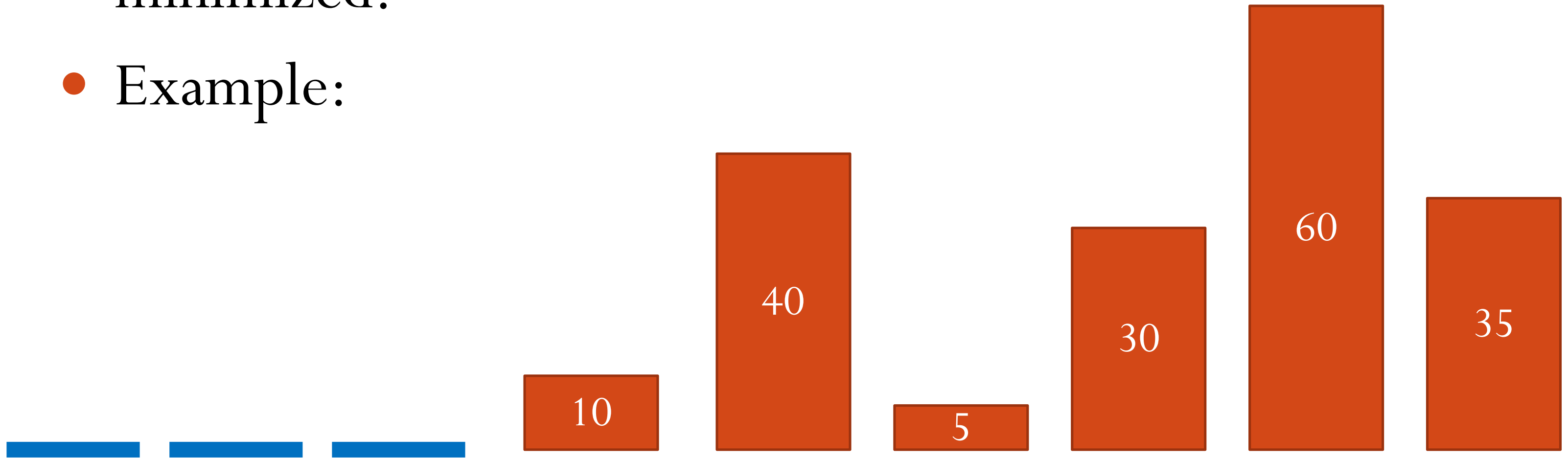
Greedy Approximation: Examples

- Claim: Let k be the optimal cardinality of the covering set. Then the greedy algorithm outputs a covering set with cardinality at most $k * \ln(n)$.
- Proof: Let N_t be the number of uncovered elements after t iterations of the loop.
  - Claim: $N_t \leq (1 - 1/k) * N_{t-1}$
  - Claim: $N_{(k \cdot \ln(n))} < 1$
    - Using the fact that $(1 - x) \leq e^{-x}$ and the equality holds only for $x = 0$.

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

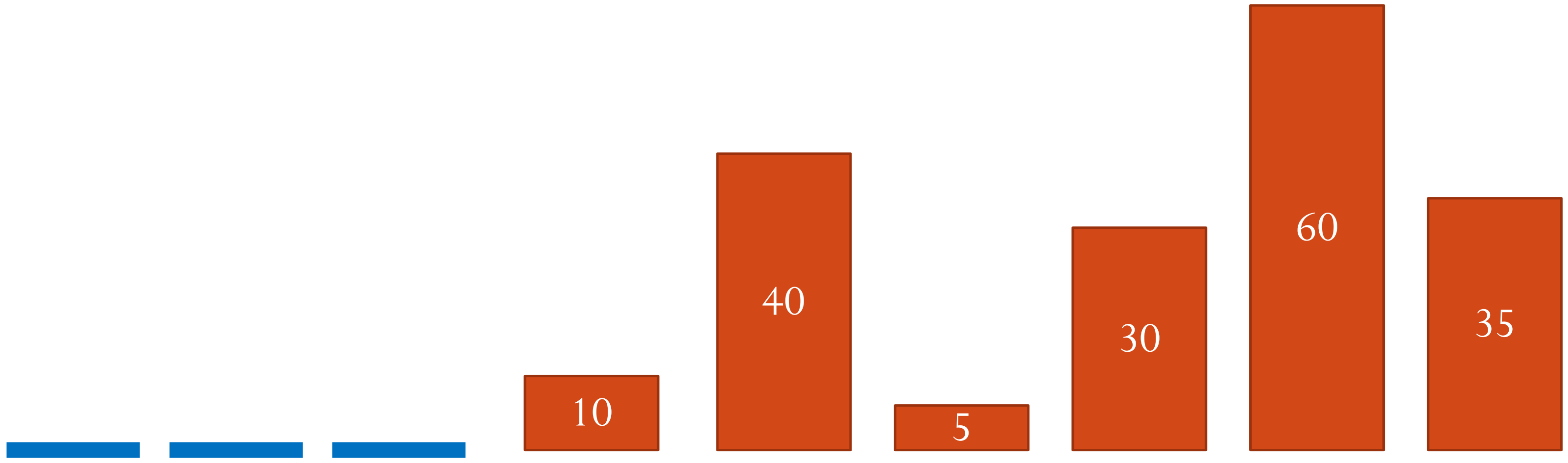
Greedy Approximation: Examples

- 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 required by any machine to perform this job. Assign these n jobs on m machines such that the maximum finishing time is minimized.
- Example:



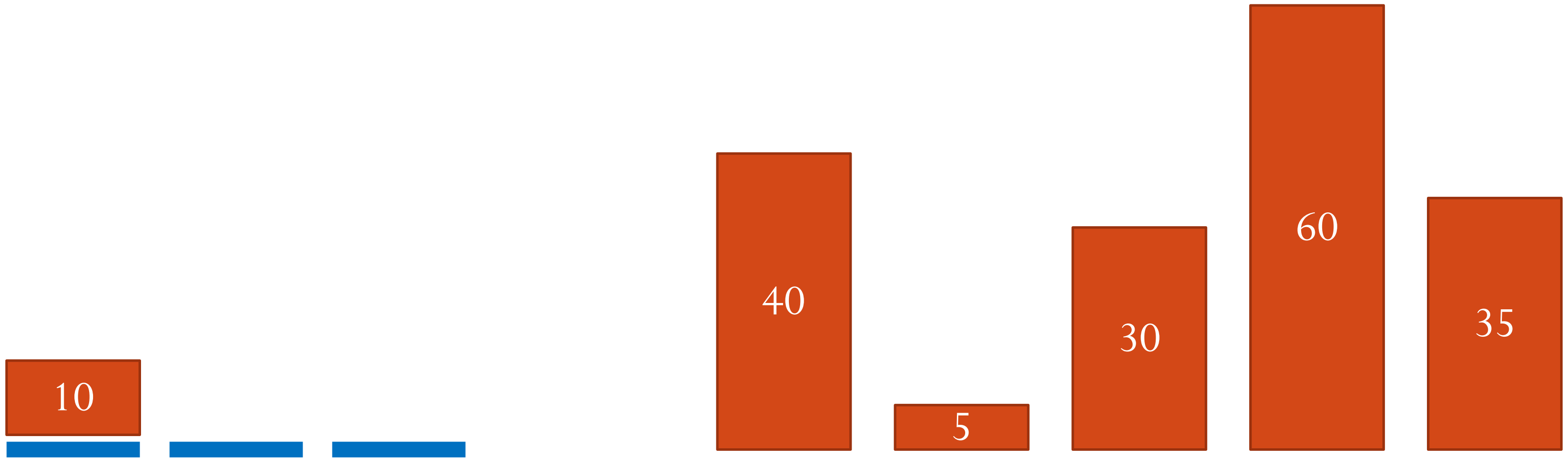
Greedy Approximation: Examples

- 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 required by any machine to perform this job. Assign these n jobs on m machines such that the maximum finishing time is minimized.
- Greedy Strategy: Assign the next job to a machine with least load



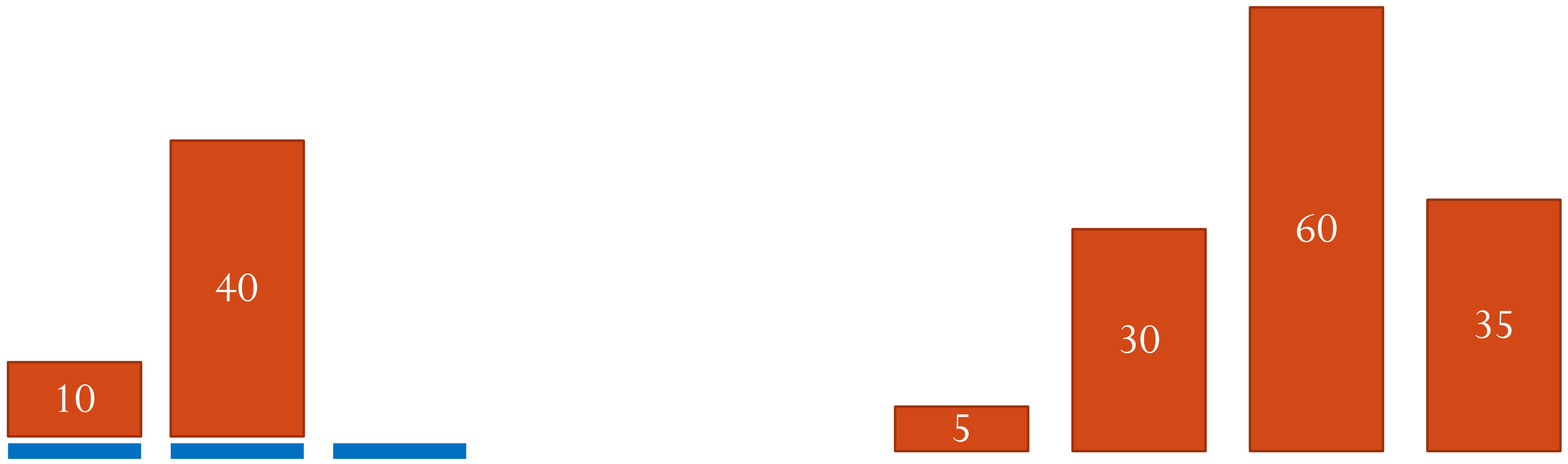
Greedy Approximation: Examples

- 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 required by any machine to perform this job. Assign these n jobs on m machines such that the maximum finishing time is minimized.
- Greedy Strategy: Assign the next job to a machine with least load



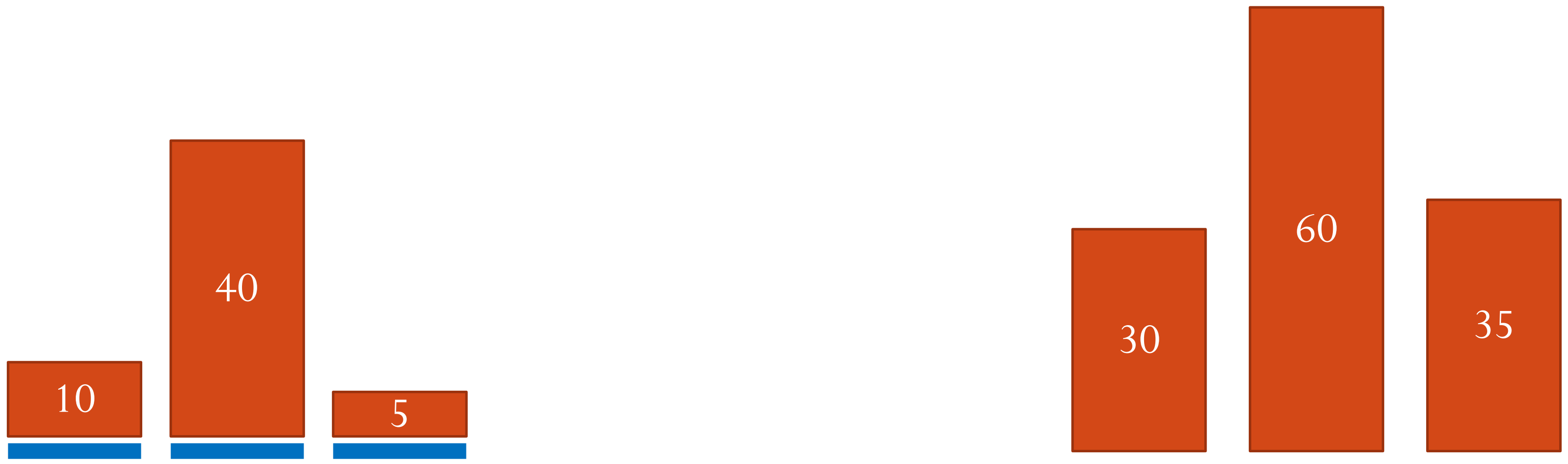
Greedy Approximation: Examples

- 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 required by any machine to perform this job. Assign these n jobs on m machines such that the maximum finishing time is minimized.
- Greedy Strategy: Assign the next job to a machine with least load



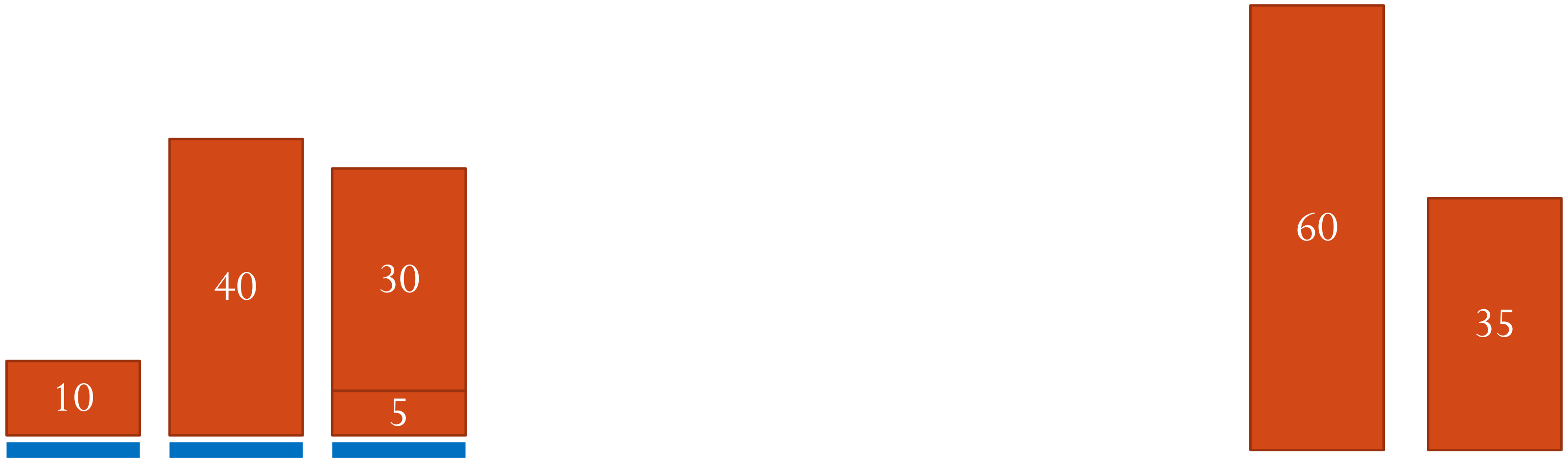
Greedy Approximation: Examples

- 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 required by any machine to perform this job. Assign these n jobs on m machines such that the maximum finishing time is minimized.
- Greedy Strategy: Assign the next job to a machine with least load



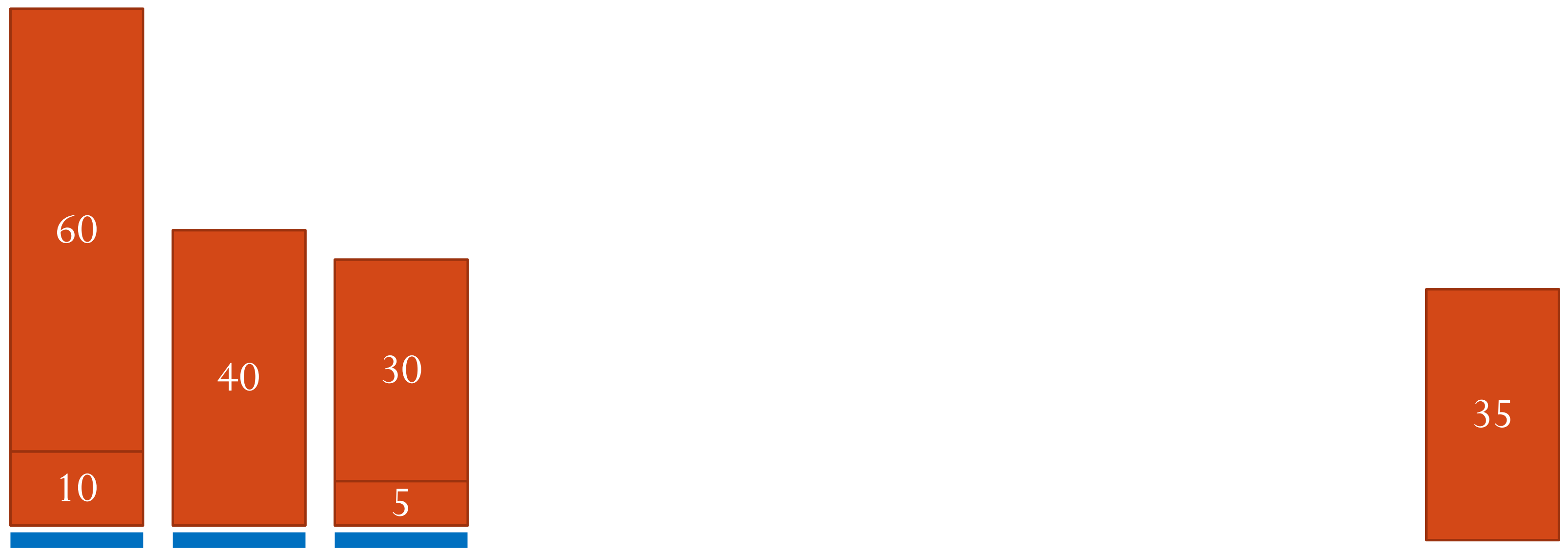
Greedy Approximation: Examples

- 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 required by any machine to perform this job. Assign these n jobs on m machines such that the maximum finishing time is minimized.
- Greedy Strategy: Assign the next job to a machine with least load



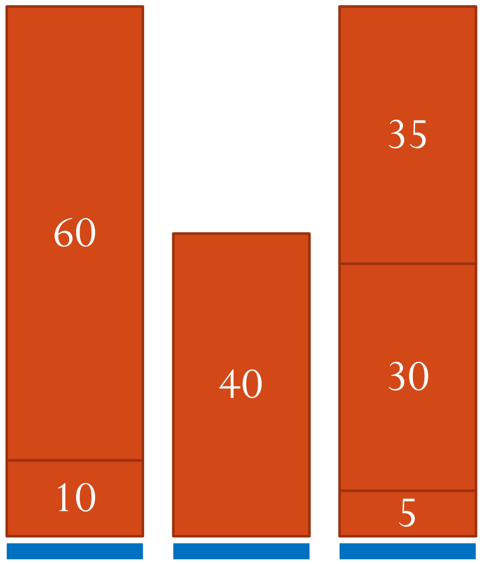
Greedy Approximation: Examples

- 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 required by any machine to perform this job. Assign these n jobs on m machines such that the maximum finishing time is minimized.
- Greedy Strategy: Assign the next job to a machine with least load



Greedy Approximation: Examples

- 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 required by any machine to perform this job. Assign these n jobs on m machines such that the maximum finishing time is minimized.
- Greedy Strategy: Assign the next job to a machine with least load



- Is this the optimal Solution?

Greedy Approximation: Examples

```
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: $G \leq 2 * OPT$
    - Claim: $OPT \geq \frac{d(1) + \ldots + d(n)}{m}$
    - Claim: 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: $s \leq \frac{(d(1) + \ldots + d(n))}{m}$

Greedy Approximation: Examples

- Let OPT be the optimal value.
- Let G denote the maximum finishing time of a machine as per the greedy assignment.
- Claim: $G \leq 2 * OPT$
- Proof:
  - Claim 1: $OPT \geq \frac{d(1)+ \ldots + d(n)}{m}$
  - Claim 2: For any job t, $OPT \geq d(t)$
  - Let the jth 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 3: $s \leq \frac{d(1)+ \ldots + d(n)}{m}$
  - So, $G \leq s + d(i)$

Greedy Approximation: Examples

- Let OPT be the optimal value.
- Let G denote the maximum finishing time of a machine as per the greedy assignment.
- Claim: $G \leq 2 * OPT$
- Proof:
  - Claim 1: $OPT \geq \frac{(d(1)+ \ldots + d(n))}{m}$
  - Claim 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 3: $s \leq \frac{d(1)+ \ldots + d(n)}{m}$
  - So, $G \leq s + d(i)$
  - This implies $G \leq \frac{d(1)+ \ldots + d(n)}{m} + d(i)$ (from Claim 3)

Greedy Approximation: Examples

- Let OPT be the optimal value.
- Let G denote the maximum finishing time of a machine as per the greedy assignment.
- Claim: $G \leq 2 * OPT$
- Proof:
  - Claim 1: $OPT \geq \frac{d(1) + \dots + d(n)}{m}$
  - Claim 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 3: $s \leq \frac{d(1) + \dots + d(n)}{m}$
  - So, $G \leq s + d(i)$
  - This implies $G \leq \frac{d(1) + \dots + d(n)}{m} + d(i)$ (from Claim 3)
  - This implies $G \leq OPT + d(i)$ (from Claim 1)

Greedy Approximation: Examples

- Let OPT be the optimal value.
- Let G denote the maximum finishing time of a machine as per the greedy assignment.
- Claim: $G \leq 2 * OPT$
- Proof:
  - Claim 1: $OPT \geq \frac{d(1) + \ldots + d(n)}{m}$
  - Claim 2: For any job t, $OPT \geq d(t)$
  - Let the jth 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 3: $s \leq \frac{d(1) + \ldots + d(n)}{m}$
  - So, $G \leq s + d(i)$
  - This implies $G \leq \frac{d(1) + \ldots + d(n)}{m} + d(i)$ (from Claim 3)
  - This implies $G \leq OPT + d(i)$ (from Claim 1)
  - This implies $G \leq OPT + OPT$ (from Claim 2)

End

Problems to think about:

1. Consider the following algorithm for minimum makespan problem:
    - Sort the jobs in decreasing order of duration. Let L be the sorted list of jobs.
    - While all jobs are not assigned
        - Assign the next job in L to a machine with least load

    Let G be the maximum finishing time as per greedy algorithm above and let OPT be the maximum finishing time as per the optimal schedule. Then $G \leq (4/3) * OPT$