Midterm 2 is graded (and sorted). Distribution on the website soon.

\[
\text{Mean (stddev) } = 26.7 (6.2) \quad \left( \frac{2}{3} \text{ pt } > \text{ MT1} \right)
\]

\[
\text{MT1: } 26.2 (8.5)
\]

HW 10 out — due Tue.

HW 11 out next Mon/Tue. “due” one week later = last day of class.

<table>
<thead>
<tr>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>21</td>
<td>28</td>
<td>Today</td>
<td>28</td>
</tr>
<tr>
<td>26</td>
<td>HW 1D due</td>
<td>26</td>
<td>ICES</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Last day</td>
<td>24</td>
<td>Review</td>
<td>Final</td>
</tr>
<tr>
<td>Any ?s</td>
<td></td>
<td>25</td>
<td>7pm</td>
<td></td>
</tr>
</tbody>
</table>

Hard problems:

- NP-hard — no poly time algo unless miracle occurs \( (P = NP) \)
- \( ETH \) \( \Rightarrow \) no subexp. time algo
- \( SETH \) \( \Rightarrow \) no subquadratic-time algorithm, or worse

We have to solve these anyway!

- Exploit structure in input
- \( k \)-means Clustering
- Instead of optimal compute provably good solutions close to optimal
- Approximation algorithm
Scheduling $n$ jobs $T[1..n]$—running times
$m$ machines—identical

compute assignment $M[1..n]
run job $j$ on machine $M[j]$ minimize $\max \sum_{j: M[j]=i} T[j]$

NP-hard from 3PARTITION

1. Sort $[1..n]$ in decreasing order
2. For $j \leftarrow 1$ to $n$
   Assign job $j$ to machine finishing first so far

Guaranteed to compute makespan $\leq \frac{2}{3}$ OPT

Claim: Makespan $\leq 2$ OPT

Proof: Machine $i$ ends last
Job $j$ ends last $T[j] \leq OPT$

because algorithm $M-T[j]$ Total time for any machine
single

$OPT \geq \sum_{j=1}^{n} T[j]/m \geq \sum_{j=1}^{n} T[j]$ for all $j$

$M-T[j] \leq m \min_{i} \text{Total}[i] \leq \frac{1}{m} \sum_{i} \text{Total}[i] = OPT$

$\Rightarrow M \leq 2 \cdot OPT$
Makespan ≤ X + Y
X ≤ OPT
Y ≤ OPT

1. Sort T[1..n] in decreasing order
2. For j = 1 to n
   Assign job j to machine finishing first so far
   Guaranteed to compute makespan ≤ \frac{4}{3} OPT

Claim: Makespan ≤ \frac{3}{2} OPT

Proof: Machine i finishes last
Job j finishes last

- If only one job assigned to machine i

  T[j] ≤ T[1]...T[m]

  First m+1 jobs:

  T[m] + T[m+1] ≤ OPT

  T[j] ≤ T[m+1] ≤ \frac{3}{2} OPT

  M - T[j] ≤ OPT

  M ≤ \frac{3}{2} OPT
Vertex Cover

\[ \text{min } \# \text{vertices cover every edge} \]

Do reductions preserve approximation? NO!

Max Indset is NP-hard to approx
within a factor of \( O(n^{1-\varepsilon}) \) for all \( \varepsilon \).

Sort nodes by decreas degree

Mark max deg node \( v \)

discard \( v \) and neighbors

recurse

\[ H_n \text{-approx} \]