Lecture 3: Reductions from 3-partition

* Two NP-complete problems useful for reducing to arithmetic (summing) problems:

(2-) Partition: given integers \( A = \{a_1, a_2, \ldots, a_n\} \), partition \( A \) into two sets \( A = A_1 \cup A_2 \) of equal sum: \( \frac{\sum A}{2} = \sum A_1 = \sum A_2 = t \) [Karp 1972]

3-Partition: given integers \( A = \{a_1, a_2, \ldots, a_n\} \), partition \( A \) into \( n/3 \) sets \( A_i \) of equal sum, \( \frac{\sum A}{(n/3)} = \sum A_i = t \)

- can assume each \( a_i \in (t/4, t/2) \)
- each set \( A_i \) contains exactly 3 items
- can make each \( a_i \) close to \( t/3 \); add huge number \( (n^{100} \cdot \max A) \) to each \( a_i \)

3-Partition along with 3-SAT might be the most important problems to prove NP-hardness.
*Two types of NP-hardness for number problems:

- **Weakly NP-hard** = NP-hard
  - Allow numbers to have value exponential in \( n \)
  - Encoding length = \( \log(2^{nc}) = nc \) still polynomial

- **Strongly NP-hard** = NP-hard even when restricted to numbers with value polynomial in \( n \)
  (i.e., even if numbers encoded in unary)

*Corresponding algorithmic notions:

- **Pseudopolynomial** = polynomial in \( n \) & largest number
- **Weakly polynomial** = polynomial in \( n \) & \( \log(\text{largest number}) \)
- **Strongly polynomial** = polynomial in \( n \) \( \Rightarrow \# \) numbers

Weak NP-hardness precludes polynomial algorithm (assuming \( P \neq NP \)) but leaves possible pseudopolynomial

Assuming \( P \neq NP \):

- \( \Rightarrow \) weakly NP-hard
- \( \Rightarrow \) strongly NP-hard
difficulty

- **strongly polynomial**
- **weakly polynomial**
- **pseudo polynomial**
Multiprocessor scheduling: [Garey & Johnson - SICOMP 1975]
- given n jobs with processing times \( a_1, a_2, \ldots, a_n \)
- given p processors (each sequential & identical)
- assign jobs to processors to minimize maximum completion time (makespan)

\( a_i \) as is

Reduction from Partition: \( p = 2 \) \( \Rightarrow \) weakly NP-hard
Reduction from 3-Partition: \( p = \frac{n}{3} \) \( \Rightarrow \) strongly NP-hard

(This was Garey & Johnson's motivation for introducing 3-partition in 1975.)

Claim: jobs finishable in makespan \( \epsilon \)
\( \Leftrightarrow \) (3-Partition instance has a solution)

Claim: jobs finishable in makespan \( \epsilon \)

\( \Leftrightarrow \) (3-Partition instance has a solution)
Rectangle packing:
- given $n$ rectangles & target rectangle
- can you pack former into latter?
  - rotate & translate to fit without overlap

- special case: exact packing — no gaps
  \[ \text{hardness result is stronger theorem} \]

Reduction from Partition:
\[
A = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} \\
B = \begin{bmatrix} 2\varepsilon \ll 1 \\
t = \frac{2\varepsilon}{\sqrt{a_i}} \end{bmatrix}
\]

Reduction from 3-Partition:
\[
B = \begin{bmatrix} \frac{n}{3}\varepsilon \ll 1 \\
t = \frac{2\varepsilon}{a_i / (\sqrt[3]{n})} \end{bmatrix}
\]

Scaling trick to make all dimensions integral:
\[
A = \begin{bmatrix} \frac{n}{3} a_i \rightarrow 1 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{n}{3} \end{bmatrix}
\]

[for 3-partition]

Here, just adding $\frac{n}{3}$ to each $a_i$ suffices:
\[
A = \begin{bmatrix} \frac{n}{3} a_i + a_i \rightarrow 1 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{n}{3} \end{bmatrix}
\]

[Demaine & Demaine - G&C 2007]
Edge-matching puzzles: [Demaine & Demaine - G&C 2007]
- given unit square tiles, each side labeled with a "color"
- given target rectangle
- goal: put tiles in target such that tiles sharing an edge have matching colors

No numbers ⇒ can't use Partition!

Reduction from 3-Partition: (like rect. packing)
- $a_i$ gadget: $\leftrightarrow a_i \leftarrow$ effectively unary encoding!
- if $i$ colors go together, forced to make this
- but some could go on boundary...
- frame gadget: ("infrastructure")

- unique colors forced on boundary
- ⇒ frame construction forced
- target shape: $(\frac{\sqrt{3}}{3} + 2) \times (t + 2)$
- ⇒ $a_i$ construction forced (no boundary left)
- ⇒ effectively rectangle packing
Signed edge-matching puzzles: (lizards etc.)
- Colors come in matching pairs: 
  a & A, b & B, etc.
- Color does not match itself - only its mate

Reduction from unsigned edge-matching puzzles:

- Interior colors (x, y, z, w) are unique pairs
  \( \Rightarrow \) must assemble \( 2 \times 2 \)

(assuming frame to prevent boundary use)
  \( \Rightarrow \) acts like unsigned tile
Jigsaw puzzles:  
- no guiding picture  
- ambiguous mates (fitting ≠ correct)

Reduction from signed edge-matching puzzles:

\[ \begin{array}{c}
\begin{array}{c}
\text{b shape} \\
\text{lower case} \\
\text{tab}
\end{array}
\end{array} \Rightarrow
\begin{array}{c}
\begin{array}{c}
\text{a shape} \\
\text{upper case}
\end{array}
\end{array} \]

Note that pocket & tab shapes are different for different colors.

For rectangular boundary:

- even square

Polyomino packing:  
- given polyominoes = edge-to-edge joinings (like Tetris)
- given target rectangle
- goal: exact pack former into latter

- rectangle packing is a special case \( \Rightarrow \) done
- but piece areas are \( \geq n \)
- what if areas are polylog?
- \[\text{OPEN}: \text{logarithmic area}\]

Reduction from jigsaw puzzles:

\[ \begin{array}{c}
\begin{array}{c}
\text{binary encoding of color}
\end{array}
\end{array} \Rightarrow
\begin{array}{c}
\begin{array}{c}
\text{\( \Theta(n) \)}
\end{array}
\end{array} \]

- can get equal areas
Closing the loop (in terms of cycle of reductions) [Demaine & Demaine - G&G 2007]
reduction from polyomino packing to unsigned edge-matching puzzles

\[
\begin{array}{c}
\text{unique pairs}
\end{array}
\Rightarrow
\begin{array}{c}
\text{use frame, but with } \\
\text{with } y = y
\end{array}
\]

So: all 4 puzzle types are NP-complete & constant-factor equivalent: can convert one to the other with O(1) factor blowup

3-partition \rightarrow unsigned edge matching
\rightarrow polyomino packing
\rightarrow signed edge matching
\rightarrow jigsaw

Demaine & Demaine G&G 2007

http://erikdemaine.org/Papers/Jigsaw_GC
Packing squares into a square: strongly NP-complete

[Leung, Tam, Wong, Young, Chin - JPDC 1990]

- **motivation:** scheduling square jobs on grid supercomputer

- infrastructure to build rectangular space

- squares of dimension $a_i + B \ll$ huge $\Rightarrow \approx B$

- pack into rectangle of height $\approx 3B$:

- total slop $\leq \max(A) \cdot (3B + t)$
  
  $< B^2 < $ one square

  if $B > 4 \max(A)$

  $\Rightarrow$ "doesn't help"