17. Rounding and relaxation

- Decision problems
- Easy and hard examples
- Rounding
- Convex relaxation
- Convex hull
Decision problems

A decision problem is a yes/no question.

Examples

• Does the following sequence contain the pattern $A, A, G$?

• Is there a subset of these numbers that sums to zero?
  $\{-7, -3, -1, 5, 8\}$
NP decision problems

A decision problem is in the class $\text{NP}^1$ if instances where the answer is “yes” have efficiently verifiable proofs of the fact that the answer is indeed “yes”. Here, “efficient” means polynomial-time in the length of the instance.

Examples

- Does the following sequence contain the pattern $A, A, G$?

- Is there a subset of these numbers that sums to zero?
  \{−7, −3, −1, 5, 8\}

  The red elements prove that the answer is indeed “yes”.

(1) “Nondeterministic Polynomial time”
NP decision problems

- The easiest problems are \textbf{P}, they can be \textit{solved} efficiently.

  \[
  \]
  If the sequence has length \(n\), the subsequence can be found using \(n\) comparisons (efficient).

- The hardest problems are \textbf{NP-complete}. We don’t know any better way to solve these problems aside from checking every possibility...

  \[
  \Rightarrow \{-7, -3, -1, 5, 8\}. \text{ Need to check all subsets! If the sequence has length } n, \text{ there are } 2^n \text{ subsets (exponential).}
  \]
P = NP?

- It’s not actually known whether there is such a thing as “hard” problems in NP! It could be possible that P=NP (all NP problems are solvable in polynomial time).

- This is perhaps the most famous unsolved problem in theoretical computer science.

- Most people believe that P≠NP.
Examples in P

- **pattern-matching:** Given a string $x_1x_2\ldots x_n$, does it contain a substring $y_1y_2\ldots y_k$? (e.g. linear search)

- **sorting a list:** Given a set of numbers $\{x_1, \ldots, x_n\}$ sort it in ascending order. (e.g. bubble sort, quicksort)

- **linear equations:** Solve a system of $n$ linear equations in $n$ variables (e.g. Gaussian elimination)

- **linear programming:** Solve a linear program with $n$ variables and $n$ constraints. (e.g. ellipsoid method)
NP-complete problems

• Traveling salesman (TSP): Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?

• Boolean satisfiability (SAT): Given an expression using \( n \) boolean variables and the operators AND, OR, NOT, and parentheses, is there a choice of the variables that makes the expression true? Example:

\[
(x_1 \lor \neg x_2) \land (\neg x_1 \lor x_2 \lor x_3) \land \neg x_1
\]
NP-complete problems

- **k-coloring**: Given a graph, can we color the edges using \( k \) colors so that each edge connects two vertices of a different color? This is in \( \mathbf{P} \) for \( k = 2 \) only.

- **vertex cover**: Given a graph, can we select \( k \) of the vertices so that every vertex is at most one edge away from a selected vertex?
NP-complete problems

- **Integer (linear) programs**: Solving a linear program with integer constraints on the variables.

Every NP problem can be represented as an integer program!
Easy instances

- All problems in NP can be written as IPs.
- (including problems in P)
- So some IPs must be easy to solve...

\[
\begin{align*}
\text{maximize} & \quad a_1z_1 + \cdots + a_nz_n \\
\text{subject to:} & \quad z_1 + \cdots + z_n = 1 \\
& \quad z_i \in \{0, 1\}
\end{align*}
\]

- Same as max\{a_1, \ldots, a_n\}, which can efficiently be solved!
Bad news

Suppose you’d like to solve a SAT problem with \( n \) variables by brute force (checking all \( 2^n \) combinations), and you can check \( 10^9 \) combinations per second.

<table>
<thead>
<tr>
<th>( n )</th>
<th>time it will take to solve</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1 microsecond</td>
</tr>
<tr>
<td>30</td>
<td>1 second</td>
</tr>
<tr>
<td>50</td>
<td>13 days</td>
</tr>
<tr>
<td>70</td>
<td>374 centuries</td>
</tr>
<tr>
<td>100</td>
<td>( 2908 \times ) (current age of the universe)</td>
</tr>
</tbody>
</table>
Good news

Very large NP-complete problems are solved in practice!

- SAT problems with a million variables
- TSP with a million variables (1000 cities)

How is this possible?

- Instances occurring in practice have special structures that can be exploited.
- Efficient approximation algorithms sometimes exist. Example: get within $\varepsilon$ of optimal in polynomial time.
Rounding

Back to the standard IP formulation:

\[
\begin{align*}
\text{maximize} & \quad c^T x \\
\text{subject to:} & \quad Ax \leq b \\
\text{where} & \quad x \in \mathbb{Z}
\end{align*}
\]

Idea:

- Solve the problem for \( x \in \mathbb{R} \) instead (a regular LP).
- Round each \( x_i \) in the solution to the nearest integer.
- This usually \textbf{does not} work!
Rounding

- If LP solution is already integral, then it is also the exact solution to the original IP. (e.g. min cost flow problems)
- Rounding can lead to an infeasible point
- Rounding can produce a point far from the optimal point

true optimum (○), relaxed optimum (●), rounded (●)
Convex relaxation

\[
\text{minimize } \quad \min_{x \in S} f(x)
\]

Two ideas we will discuss:

1. *Function relaxation*: if \( f \) is troublesome, bound it with a function that is easier to work with, e.g. a convex function.

2. *Constraint relaxation*: If \( S \) is troublesome, find a bigger set that is easier to work with, e.g. a convex set.
Function relaxation

\[ f_{\text{opt}} = \min_{x \in S} f(x) \]

Suppose we can find \( g \) such that \( g(x) \leq f(x) \) for all \( x \). In other words, \( g \) is a lower bound on \( f \).
Function relaxation

- Solve $g_{\text{opt}} = \min_{x \in S} g(x)$ and let $\hat{x}$ be the corresponding $x$.
- We have the bounds: $g_{\text{opt}} = g(\hat{x}) \leq f_{\text{opt}} \leq f(\hat{x})$.
- If $f(\hat{x}) = g_{\text{opt}}$ then the bound is tight and $f_{\text{opt}} = f(\hat{x})$.

Pick a convex $g$ so that $g_{\text{opt}}$ and $\hat{x}$ are easy to find!
Suppose we can find some set $C$ such that $S \subseteq C$. In other words, $C$ is a superset of $S$. 
• Solve $h_{\text{opt}} = \min_{x \in C} f(x)$ and let $\tilde{x}$ be the optimal $x$.

• We have the bound: $h_{\text{opt}} = f(\tilde{x}) \leq f_{\text{opt}} \leq f(x)$ for $x \in S$.

• If $\tilde{x} \in S$ then the bound is tight and $f_{\text{opt}} = f(\tilde{x})$.

Pick a convex $C$ so that $h_{\text{opt}}$ and $\tilde{x}$ are easy to find!
Common relaxations

1. Boolean constraint:

   \[ x \in \{0, 1\} \quad \implies \quad 0 \leq x \leq 1 \]

   If \( x_{\text{opt}} \) is 0 or 1, relaxation is exact.

2. Convex equality:

   \[ f(x) = 0 \quad \implies \quad f(x) \leq 0 \]

   If \( f(x_{\text{opt}}) = 0 \), relaxation is exact.

3. A constraint you don’t like:

   \[ x \neq 3 \quad \implies \quad \text{just remove the constraint!} \]

   If \( x_{\text{opt}} \neq 3 \), relaxation is exact.
Convex hull

The **convex hull** of a set $S$, written $\text{conv}(S)$ is the smallest convex set that contains $S$.

Equivalent definitions:

- The set of all affine combinations of all points in $S$
- The intersection of all convex sets containing $S$