Branch & Bound Algorithms

- Standard technique: very simple, but basis of all commercial codes (enhanced by other techniques + tricks)

- Idea: divide-and-conquer approach to explore the set of feasible integer solutions using bounds on opt. cost to avoid full exploration

\[
\begin{align*}
\text{max } C^T x \\
\text{s.t. } x \in F \subseteq F: \text{ feasible region}
\end{align*}
\]

Branching:
If we cannot solve original problem directly = optimize over subproblems
Partition \( F \) into subsets \( F_1, \ldots, F_k \),
then \( \text{max } C^T x = \max \left\{ \text{max } C^T x \right\} \).

At each \( i \) might be still hard to solve \( F_i \)\( \text{ recursed.} \)

... in extreme case: full enumeration!
(avoid by bounding)

Bounding:
- Let \( z \) be some lower bound on the optimal cost: \( z \leq \text{max } C^T x \).
- Note: any feas. solution for a subproblem \( i \) gives a lower bound on the opt. cost = update \( z \) if \( z_i \geq z \).
- For each subproblem \( F_i \):
  - either solve it optimally and update \( z \) if \( z_i \geq z \) (might be difficult)
  - or compute upper bound \( b(F_i) \geq \text{max } C^T x \).
    * if \( b(F_i) < z \) then ignore subproblem \( F_i \)
    * otherwise branch or insert more into computing opt. \( z \)
Very generic framework. Questions (flexibility in):

1) How to break into subproblems
2) Which subproblem to choose?
3) How to obtain upper bounds?

1) How to break into subproblems? No several ways, problem dependent:
   - Example A: general ILP
   - Example B: TSP in directed graphs

2) Which subproblems to choose?
   - Some typical ways: DFS, BFS, Best-First-Search (best bound), and various others or combinations

3) How to obtain upper bounds?
   - E.g.: • LP relaxation
   - • Other relaxations (Lagrangean relax, next)
   - • Utilize cutting planes

Ex: General ILP

- Bounding: LP relaxation
- If solution \( x \) is integral, 1 no further branching, update \( x \)
- Otherwise, pick some non-integral component \( x_i \)
- and create two subproblems by adding one of the two constraints:
  \[ x_i \leq \lfloor x_i^* \rfloor \quad \text{or} \quad x_i \geq \lceil x_i^* \rceil \]

\[
\begin{align*}
F_i & \quad x^* \in Z \\
F_{i1} & \quad \text{if } x_i \geq \lfloor x_i^* \rfloor \\
F_{i2} & \quad \text{if } x_i \geq \lceil x_i^* \rceil \\
F_{i1} & \quad = F_i \cup \{ x_j \mid x_j \leq \lfloor x_j^* \rfloor \} \\
F_{i2} & \quad = F_i \cup \{ x_j \mid x_j \geq \lceil x_j^* \rceil \}
\end{align*}
\]
Ex. TSP in directed graphs

\[
\begin{align*}
\min & \sum_{ij} C_{ij} x_{ij} \\
\text{s.t.} & \sum_j x_{ij} = 1 & i = 1, \ldots, n \\
& \sum_i x_{ij} = 1 & j = 1, \ldots, n \\
& \sum_{ij} x_{ij} \leq |S| - 1 & S \subseteq \{1, \ldots, n\} \\
& x_{ij} \in \{0, 1\}
\end{align*}
\]

\[\rightarrow \text{Relaxation without subtour elimination constraint.} \]
\[= \text{assignment problem (exercise lecture)} \]
\[= \text{can be solved efficiently to optimality.} \]

Branch & Bound for TSP:

- **Bounding strategy**: Solve assignment problem relaxation \[\Rightarrow x^*\]
- **Branching**: If \[x^*\] satisfies is a tour \[= 1\] optimal for problem
  - Otherwise, branch as follows:
  - Choose shortest cycle (w.r.t. \# edges) \[e_1, \ldots, e_k\]
  - and create subproblems by setting one \[x_{e_i} = 0\] for \[i = 1, \ldots, k\].

Comment on finding feasible solutions for TSP (heuristics)

- **Nearest neighbor (Greedy)**
  - Start at any node; visit the nearest node not yet visited (repeat); return to start node
  - Fast, easy to implement; empirically quite good
    \[1.26 \text{times opt cost}\] on TSPLIB 1997. (Not a guarantee)
- **Tour improvement: 2-OPT (k-OPT, local search)**
  - Given a tour, consider any two non-adjacent edges; remove them and reconnect (unique way) to new tour.
  - If cost improved, then keep; otherwise, discard and continue.