Enumeration and Heuristics

Dynamic Programming

- Consider min $\{cx : ax \geq \beta, x \in \{0,1\}^n\}$.
- Let $S = \max\{|a_i| : i = 1, \dots, n\}$.
- Define a directed graph $D = (V, A)$ with vertex set

$$V = \{0, \dots, n\} \times \{-nS, \dots, nS\}$$

- and arc set A defined by

$$\big((j, \delta), (i, \delta')\big) \in A \Leftrightarrow j = i - 1 \text{ and } \delta' - \delta \in \{0, a_i\}$$

- The length of $\big((i-1, \delta), (i, \delta)\big)$ is 0.
- The length of $\big((i-1, \delta), (i, \delta + a_i)\big)$ is c_i.
- Any directed path P in D from $(0,0)$ to (n, β') for some $\beta' \geq \beta$ yields a feasible solution x:
  - $x_i = 0$ if $\big((i-1, \delta), (i, \delta)\big) \in P$ for some δ;
  - $x_i = 1$ if $\big((i-1, \delta), (i, \delta + a_i)\big) \in P$ for some δ.
- The length of P is equal to cx.
- So we can solve the problem by finding a shortest path from $(0,0)$ to (n, β') for some $\beta' \geq \beta$.

Primal algorithms

- Improving search
  - begin at a feasible solution $\mathbf{x}^{(0)}$
  - advance along a sequence of feasible solutions $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$, ...with ever-improving objective value
  - move between feasible solutions via improving and feasible move directions $\Delta\mathbf{x}$:

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda \Delta\mathbf{x}$$

  - Guaranteed to find local optimal solutions under mild conditions

Improving search for discrete optimization

- Success of branch-and-bound for ILP depends largely on quality of LP relaxations
- But we also need "good" feasible solutions
- Some ILPs (and discrete optimization models in general) may be especially resistant to branch-and-bound-type techniques
- What can we do?
- Improving search can help us find good feasible solutions

Discrete neighborhoods and move sets

- Optimization model with discrete variables

  ⇒ Want the neighborhood of a current solution to be binary/integer

- We can define neighborhoods and control candidates for improving and feasible directions
- Example:

$$\begin{aligned} \max \quad & 20x_1 - 4x_2 + 14x_3 \\ \text{s.t.} \quad & 2x_1 + x_2 + 4x_3 \leq 5 \\ & x_1, x_2, x_3 \in \{0, 1\} \end{aligned}$$

- Suppose the current solution is $\mathbf{x}^{(t)} = (1, 1, 0)$
- Suppose the neighborhood consists of all feasible solutions that differ in at most one component.
- Example:

$$\begin{aligned} \max \quad & 20x_1 - 4x_2 + 14x_3 \\ \text{s.t.} \quad & 2x_1 + x_2 + 4x_3 \leq 5 \\ & x_1, x_2, x_3 \in \{0, 1\} \end{aligned}$$

- Neighborhood of $\mathbf{x}^{(t)} = (1, 1, 0)$:

$$\begin{aligned} (1,1,0) + (1,0,0) &= (2,1,0) & & \\ (1,1,0) + (-1,0,0) &= (0,1,0) & & \text{feasible} \\ (1,1,0) + (0,1,0) &= (1,2,0) & & \\ (1,1,0) + (0,-1,0) &= (1,0,0) & & \text{feasible and improving} \\ (1,1,0) + (0,0,1) &= (1,1,1) & & \\ (1,1,0) + (0,0,-1) &= (1,1,-1) & & \end{aligned}$$

Discrete improving search

0. **Initialization.**
   - Choose any starting feasible solution $\mathbf{x}^{(0)}$
   - Set solution index $t \leftarrow 0$
1. **Stopping.**
   - If no neighboring solution of $\mathbf{x}^{(t)}$is both improving and feasible, stop

   $\Rightarrow$ $\mathbf{x}^{(t)}$ is a local optimal solution
2. **Move.** Choose some improving feasible move as $\Delta\mathbf{x}^{(t+1)}$
3. **Step.** Update
$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \Delta\mathbf{x}^{(t+1)}$$
4. **Increment.** Increment $t \leftarrow t+1$, return to Step 1

The art of choosing a neighborhood

- The solution produced by local search depends on the neighborhood on the move set employed
- Larger neighborhoods generally result in superior local optimal solutions, but take longer to examine

Multistart search

- Different initial solutions lead to different local optimal solutions
- All globally optimal solutions are local optimal solutions
- Idea: start improving search from different initial solutions and take the best one

Escape from local optima: allow nonimproving moves

- Another idea: allow nonimproving feasible moves
- Rationale: we might be able to "escape" local optimal solutions and move to a better region
- Problem: if we don't "escape" far enough, we will just cycle back to the same local optimal solution
- Three popular methods:
  - Tabu search
  - Simulated annealing
  - Genetic algorithms

Tabu search

- **Tabu search** allows nonimproving moves and deals with cycling by temporarily forbidding moves that would return to a solution recently visited

- Makes certain solutions "tabu" ("taboo"?)

0. **Initialization.**
    - Choose any starting feasible solution $\mathbf{x}^{(0)}$
    - Choose iteration limit $t_{\max}$
    - Set incumbent solution $\hat{\mathbf{x}}$
    - Set solution index $t \leftarrow 0$
    - No moves are tabu
1. **Stopping.**
    - If no non-tabu move $\Delta\mathbf{x}$ in move set $\mathcal{M}$ leads to a feasible neighbor of $\mathbf{x}^{(t)}$, or if $t = t_{\max}$, stop.

    ⇒ Incumbent solution $\hat{\mathbf{x}}$ is approximate optimum
2. **Move.** Choose some non-tabu move $\Delta\mathbf{x} \in \mathcal{M}$ as $\Delta\mathbf{x}^{(t+1)}$
3. **Step.** Update
$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \Delta\mathbf{x}^{(t+1)}$$
4. **Incumbent solution.** If the objective function value of $\mathbf{x}^{(t+1)}$ is superior to that of the incumbent solution $\hat{\mathbf{x}}$, replace $\hat{\mathbf{x}} \leftarrow \mathbf{x}^{(t+1)}$
5. **Tabu list.**
    - Remove from the tabu list any moves that have been on it for a sufficient number of iterations
    - Add a collection of moves that includes any returning from $\mathbf{x}^{(t+1)}$ to $\mathbf{x}^{(t)}$
6. **Increment.** Increment $t \leftarrow t + 1$, return to Step 1

Simulated annealing

- **Simulated annealing** accepts nonimproving moves with probability

- Name comes from the annealing process of slowly cooling metals to improve strength

- Suppose we are maximizing

- If the move is improving ($\Delta\text{obj} > 0$), it is accepted

- If the move is nonimproving ($\Delta\text{obj} \leq 0$), it is accepted with probability

$$\text{probability of acceptance} = e^{\Delta\text{obj}/q}$$

where $q \geq 0$ is the **temperature** parameter

- The probability of accepting a nonimproving move declines the more it worsens the objective function

- The probability of accepting a nonimproving move declines as the temperature cools

- As the algorithm progresses, the temperature cools ($q \to 0$)

- This description is for maximization problems

0. **Initialization.**
   - Choose any starting feasible solution $\mathbf{x}^{(0)}$
   - Choose iteration limit $t_{\max}$
   - Set large initial temperature q
   - Set incumbent solution $\hat{\mathbf{x}}$
   - Set solution index $t \leftarrow 0$

1. **Stopping.**
   - If no move $\Delta\mathbf{x}$ in move set $\mathcal{M}$ leads to a feasible neighbor of $\mathbf{x}^{(t)}$, or if $t = t_{\max}$, stop.

   $\Rightarrow$ Incumbent solution $\hat{\mathbf{x}}$ is approximate optimum

2. **Provisional move.**
   - Randomly choose a feasible move $\Delta\mathbf{x} \in \mathcal{M}$ as $\Delta\mathbf{x}^{(t+1)}$
   - Compute
   $$\Delta\text{obj} = (\text{obj. val. at } \mathbf{x}^{(t)} + \Delta\mathbf{x}^{(t+1)}) - (\text{obj. val. at } \mathbf{x}^{(t)})$$

3. **Acceptance.** If $\Delta\mathbf{x}^{(t+1)}$ improves ($\Delta\text{obj} > 0$), or with probability $e^{\Delta\text{obj}/q}$ if $\Delta\text{obj} \leq 0$, accept $\Delta\mathbf{x}^{(t+1)}$ and update
   $$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \Delta\mathbf{x}^{(t+1)}$$

   Otherwise, return to Step 2

4. **Incumbent solution.** If the objective function value of $\mathbf{x}^{(t+1)}$ is superior to that of the incumbent solution $\hat{\mathbf{x}}$, replace $\hat{\mathbf{x}} \leftarrow \mathbf{x}^{(t+1)}$

2. **Temperature reduction.** If a sufficient number of iterations have passed since the last temperature change, reduce temperature q

3. **Increment.** Increment $t \leftarrow t + 1$, return to Step 1

Genetic algorithms

- **Genetic algorithms** evolve approximately optimal solutions by operations "combining" members of an improving **population** of individual solutions
- The best solution so far will always be part of the population
- Each **generation** will consist of a spectrum of solutions
  - Some will be feasible
  - Some will be nearly as good as the best
  - Some will be poor
- New solutions are created by "combining" pairs of individual solutions in the population
- Standard method for combining solutions: **crossover**
  - Take pair of "parent" solutions to produce "children" solutions
  - Break both parent vectors at same point and swap
  - Example:

$$\mathbf{x}^{(1)} = (1, 0, 1, 1, 0, \mid 0, 1, 0, 0)$$
$$\mathbf{x}^{(2)} = (0, 1, 1, 0, 1, \mid 1, 0, 0, 1)$$

$$\Rightarrow \mathbf{x}^{(3)} = (1, 0, 1, 1, 0, \mid 1, 0, 0, 1)$$
$$\Rightarrow \mathbf{x}^{(4)} = (0, 1, 1, 0, 1, \mid 0, 1, 0, 0)$$

- How to select pairs of solutions in current population to crossover?
- How to decide which new/old solutions will survive in the next population?
- How to maintain diversity in the population?
- **Elitist** strategy: form each new generation as a mix of
  - Elite solutions: best solutions from previous generation
  - Immigrant solutions: solutions taken arbitrarily from previous generation to promote diversity
  - Crossover solutions

0. **Initialization.**
   - Choose population size p
   - Choose initial feasible solutions $x^{(1)}, \ldots, x^{(p)}$

- Set generation limit $t_{\max}$
- Set population subdivisions p_e for elites, p_i for immigrants, and p_c for crossovers
- Set generation index $t \leftarrow 0$

1. **Stopping.** If $t = t_{\max}$, stop and report the best solution of the current population as an approximate optimum
2. **Elite.** Initialize the population of generation $t+1$ with copies of the p_e best solutions in the current generation
3. **Immigrants.** Arbitrarily choose p_i new immigrant feasible solutions and include them in the population of generation $t+1$
4. **Crossovers.**
   - Choose $p_c/2$ (disjoint) pairs of solutions from the generation t population
   - Execute crossover on each pair at an independently chosen random cut point
   - Put these crossovers into the population of generation $t+1$
5. **Increment.** Increment $t \leftarrow t+1$, return to Step 1