MVE165/MMG631
Linear and integer optimization with applications
Lecture 8a
Theory and algorithms for discrete optimization models

Ann-Brith Strömberg

2018–04–20

Methods for ILP: Overview (Ch. 14.1)

Enumeration

- Implicit enumeration: Branch–and–bound

Relaxations

- Decomposition methods: Solve simpler problems repeatedly
- Add valid inequalities to an LP $\Rightarrow$ "cutting plane methods"
- Lagrangian relaxation

Heuristic algorithms – optimum *not* guaranteed

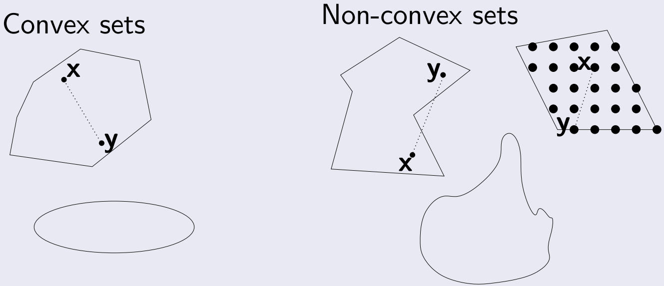
- "Simple" rules $\Rightarrow$ feasible solutions (usually fairly good but not optimal)
- Construction heuristics
- Local search heuristics

Convex sets

A set S is convex if, for any elements $\mathbf{x}, \mathbf{y} \in S$ it holds that

$$\alpha \mathbf{x} + (1 - \alpha)\mathbf{y} \in S \text{ for all } 0 \leq \alpha \leq 1$$

Examples:



- Linear optimization problems have convex feasible sets
- Integrality requirements $\Rightarrow$ nonconvex feasible set

Consider a minimization integer linear program (ILP)

$$
[\text{ILP}] \qquad z^* := \begin{array}{rrcll} \min & \mathbf{c}^\top \mathbf{x} & & & \\ \text{subject to} & \mathbf{A}\mathbf{x} & \leq & \mathbf{b} & \\ & \mathbf{x} & \geq & \mathbf{0} & \text{and integer} \end{array}
$$

- The feasible set $X = \{\mathbf{x} \in Z^n_+ \,|\, \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ is *non*-convex
- How can one prove that a solution $\mathbf{x}^* \in X$ is optimal?

- We cannot use strong duality/complementarity as for linear optimization (where X is polyhedral $\Rightarrow$ convexity)

- Bounds on the optimal value
  - Optimistic estimate $\underline{z} \leq z^*$ from a *relaxation* of ILP
  - Pessimistic estimate $\bar{z} \geq z^*$ from a *feasible solution* to ILP

Optimistic estimates of z^* from relaxations

- Either: Enlarge the set X by removing constraints $\Longrightarrow X^{\text{relax}} \supseteq X$
- Or: Replace $\mathbf{c}^\top \mathbf{x}$ by an underestimating function f, i.e., such that $f(\mathbf{x}) \leq \mathbf{c}^\top \mathbf{x}$ for all $\mathbf{x} \in X$
- Or: Do both of the above

$\Rightarrow$ solve a *relaxation* of (ILP)

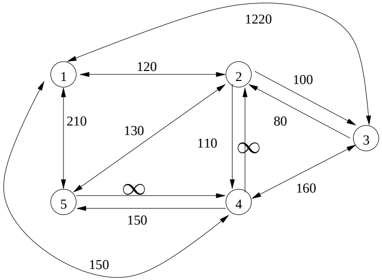
Example: enlarge X by relaxing the integrality requirements

- $X = \{\mathbf{x} \geq \mathbf{0} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b},\ \mathbf{x} \text{ integer }\}$
- $X^{\mathsf{LP}} = \{\mathbf{x} \geq \mathbf{0} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$

$$\Rightarrow \quad z^{\mathsf{LP}} := \min_{\mathbf{x} \in X^{\mathsf{LP}}} \mathbf{c}^\top \mathbf{x}$$

- It holds that $z^{\mathsf{LP}} \leq z^*$ since $X \subseteq X^{\mathsf{LP}}$

- Given n connected cities
- Distance on each connection
- Find the shortest tour that passes through all the cities



- $V = \{1, \ldots, n\}$: the set of cities
- d_{ij}: distance from city i to city j
- Binary variable $x_{ij} \iff$ connection from i to j

- Computationally hard to solve due to *combinatorial explosion*
- Several versions of the TSP: Euclidean, metric, symmetric ...

An ILP formulation of the TSP problem

$$
\begin{array}{llcll}
\min & \displaystyle\sum_{i\in V}\sum_{j\in V} d_{ij}x_{ij}, & & & \\
\text{s.t.} & \displaystyle\sum_{j\in V} x_{ij} & = & 1, & i\in V, \qquad (1)\\
 & \displaystyle\sum_{i\in V} x_{ij} & = & 1, & j\in V, \qquad (2)\\
 & \displaystyle\sum_{i\in U, j\in V\setminus U} x_{ij} & \geq & 1, & \forall U\subset V : 2\leq |U|\leq |V|-2, \qquad (3)\\
 & x_{ij} & \text{binary} & & i,j\in V \qquad (4)
\end{array}
$$

- Cf. the assignment problem $\boxed{\text{DRAW GRAPH * 2 !}}$
- Enter and leave each city exactly once ⇔ (1) and (2) $\boxed{\text{DRAW!}}$
- Constraints (3): *subtour elimination* $\boxed{\text{DRAW!}}$
- Alternative formulation of (3): $\boxed{\text{DRAW!}}$
  $\sum_{(i,j)\in U} x_{ij} \leq |U|-1, \quad \forall U\subset V : 2\leq |U|\leq |V|-2$

Relaxation principles that yield more tractable problems

Linear programming relaxation

Remove integrality requirements (*enlarge* X), but still an exponential number of constraints (3)

Combinatorial relaxation

E.g. remove subtour constraints (3) $\Rightarrow$ minimum-cost assignment (*enlarge* X)

DRAW!

Lagrangean relaxation $\Rightarrow$ Lagrange dual

Move "complicating" constraints to the objective function, with penalties for infeasible solutions; then find "optimal" penalties (*enlarge* X *and construct a function* f *such that* $f(\mathbf{x}) \leq \mathbf{c}^\top \mathbf{x}$, $\forall \mathbf{x} \in X$)

Tight bounds

- Suppose that $\bar{\mathbf{x}} \in X$ is a feasible solution to ILP (min-problem) and that $\underline{\mathbf{x}}$ solves a relaxation of ILP
- Then, it holds that

$$\underline{z} := \mathbf{c}^\top \underline{\mathbf{x}} \leq z^* \leq \mathbf{c}^\top \bar{\mathbf{x}} =: \bar{z}$$

- If $\bar{z} - \underline{z} \leq \varepsilon$ then the value of the solution candidate $\bar{\mathbf{x}}$ is at most ε from the optimal value z^*
- Efficient solution methods for ILP *combine relaxation and heuristic methods* to find tight bounds (small $\varepsilon \geq 0$)

$$\text{[ILP]} \qquad z^* = \min_{\mathbf{x} \in X} \; \mathbf{c}^\top \mathbf{x}, \qquad \text{where } X \subset Z^n$$

- Divide–&–Conquer: a general principle to partition and search the feasible space
- Branch–&–Bound: Divide–and–conquer for finding *optimal* solutions to optimization problems with integrality requirements
- Can be adapted to different types of models
- Can be combined with other (e.g. heuristic) algorithms
- Also called implicit enumeration and tree search
- *Idea:* Enumerate all feasible solutions by a successive partitioning of X into a family of subsets
- Enumeration organized in a tree using graph search; it is made implicit by utilizing approximations of z^* from relaxations of [ILP] for pruning branches from the tree

Branch–&–bound for ILP: Main concepts

Relaxation: a simplification of [ILP] in which some constraints are removed

- Purpose: to get simple (i.e., polynomially solvable) (*node*) *subproblems*, and optimistic approximations of z^*
- Examples: *remove integrality requirements*, remove or Lagrangean relax complicating (linear) constraints (e.g., sub-tour constraints)

Branching strategy: rules for partitioning a subset of X

- Purpose: exclude the solution to a relaxation if it is *not* feasible in [ILP] $\Longleftrightarrow$ a *partitioning* of the feasible set
- Examples: Branch on fractional values, subtours, etc

B&B: Main concepts (continued)

Tree search strategy: defines the order in which the nodes in the B&B tree are created and searched

- Purpose: quickly *find good feasible solutions* $\Longrightarrow$ limit the size of the tree
- Examples: depth-, breadth-, best-first.

Node cutting criteria: rules for deciding when a subset should not be further partitioned

- Purpose: *avoid searching parts* of the tree that *cannot* contain an optimal solution
- Cut off a node (i.e., prune a whole branch) if the corresponding *node subproblem* has
  - no feasible solution, or
  - an optimal solution which is feasible in [ILP], or
  - an optimal objective value that is worse (higher) than that of any known feasible solution

ILP: Example of a Branch–&–Bound solution

- Relax the integrality requirements $\Longrightarrow$ the node subproblem is a linear (continuous) optimization problem
- Branch over fractional variable values
- Here: the tree is searched in depth-first order
- Here: branches are pruned due to integrality/infeasibility