Introduction to Mathematical Programming
(Mixed Integer Linear Programming)
Algorithmics, 186.814, VU 4.0

Daniel Obszelka

Algorithms and Complexity Group
Institute of Logic and Computation
TU Wien

WS 2019/20

Topics of this part

- Model-based Algorithms
- Linear Programming
  - Introduction
  - Geometry
  - Algorithms
  - Duality Theory
- (Mixed) Integer Linear Programming
  - Formulations
  - Relaxations/Bounds
  - Branch-and-Bound

(Mixed) Integer Linear Programming

Motivation

- model and solve real-world optimization problems
- generally applicable for a wide variety of problems
- (mostly) no specialized algorithms necessary
- here we concentrate on applying the tool of integer linear programming
  (detailed proofs of underlying theorems $\Rightarrow$ literature)

Literature

Main resource:

- Laurence A. Wolsey, *Integer Programming*, 1998, Wiley

Further literature:

- Alexander Schrijver, *Theory of Linear and Integer Programming*, 1998, Wiley
- George L. Nemhauser, Laurence A. Wolsey, *Integer and Combinatorial Optimization*, 1999, Wiley

(Mixed) Integer Linear Programming

Formulations

Real-World Problems

- scheduling (trains, airline crews, timetables, ...)
- planning (production, electricity generation, ...)
- telecommunications (network design, routing, ...)
- transportation (dial-a-ride, vehicle routing, ...)
- cutting and packing
- ...

Linear Program (LP)

$$\begin{aligned} \max \quad & \boldsymbol{c}\boldsymbol{x} \\ & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \geq \mathbf{0}. \end{aligned}$$

$\boldsymbol{c}$ ... n-dimensional row vector
$\boldsymbol{x}$ ... n-dimensional column vector
$\boldsymbol{A}$ ... $m \times n$ matrix
$\boldsymbol{b}$ ... m-dimensional column vector

Mixed Integer (Linear) Program (MI(L)P)

$$
\begin{aligned}
\max \quad & \boldsymbol{c_1 x_1} + \boldsymbol{c_2 x_2} \\
& \boldsymbol{A_1 x_1} + \boldsymbol{A_2 x_2} \leq \boldsymbol{b} \\
& \boldsymbol{x_1} \geq \boldsymbol{0} \\
& \boldsymbol{x_2} \geq \boldsymbol{0} \text{ and integer.}
\end{aligned}
$$

$\boldsymbol{c_1}$ ... n-dimensional row vector
$\boldsymbol{x_1}$ ... n-dimensional column vector
$\boldsymbol{c_2}$ ... p-dimensional row vector
$\boldsymbol{x_2}$ ... p-dimensional column vector
$\boldsymbol{A_1}$ ... $m \times n$ matrix
$\boldsymbol{A_2}$ ... $m \times p$ matrix
$\boldsymbol{b}$ ... m-dimensional column vector

Integer Program (IP)

$$\begin{aligned} \max \quad & \boldsymbol{cx} \\ & \boldsymbol{Ax} \leq \boldsymbol{b} \\ & \boldsymbol{x} \geq \mathbf{0} \text{ and integer.} \end{aligned}$$

$\boldsymbol{c}$ … n-dimensional row vector
$\boldsymbol{x}$ … n-dimensional column vector
$\boldsymbol{A}$ … $m \times n$ matrix
$\boldsymbol{b}$ … m-dimensional column vector

Binary Integer Program (BIP)

$$
\begin{aligned}
\max \quad & \boldsymbol{c}\boldsymbol{x} \\
& \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\
& \boldsymbol{x} \in \{0,1\}^n.
\end{aligned}
$$

$\boldsymbol{c}$ ... n-dimensional row vector
$\boldsymbol{x}$ ... n-dimensional column vector
$\boldsymbol{A}$ ... $m \times n$ matrix
$\boldsymbol{b}$ ... m-dimensional column vector

Combinatorial Optimization Problem (COP)

Given:

- a finite set $N = \{1, \ldots, n\}$
- weights $c_j, \forall j \in N$
- a set $\mathcal{F}$ of feasible subsets of N

The problem of finding a minimum weight feasible subset

$$\arg\min_{S \in \mathcal{F}} \left\{ \sum_{j \in S} c_j \right\}$$

is a *combinatorial optimization problem*.

Many COPs can be formulated as (B)IPs.

How to solve IPs?

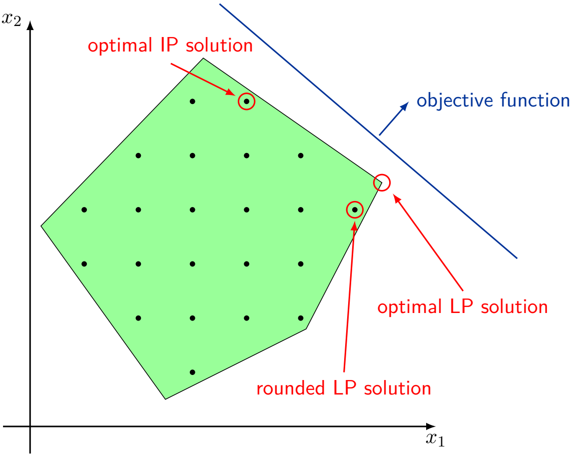


Figure: Rounding optimal LP solution? $\Rightarrow$ Does not work!

Formulating (B)IPs

1. define necessary (and maybe additional) variables
2. define a set of constraints using the variables:
   feasible points $\Leftrightarrow$ feasible solutions of the problem
3. define objective function using the variables

Example

Variables for a COP can be defined as the n-dimensional *incidence vector* $\boldsymbol{x^S} \in \{0,1\}^n$ of subset $S \subseteq N$:

- $x_j^S = 1$ if $j \in S$
- $x_j^S = 0$ if $j \notin S$

The Assignment Problem

Definition

Given:

- n jobs, n persons
- cost c_{ij} for assigning person i to job j
  (suitability of persons for jobs)

The goal is to find an assignment with minimum cost.

The Assignment Problem - BIP

1. Variables: $x_{ij} \in \{0, 1\},\ \forall i, j \in \{1, \ldots, n\}$
   $x_{ij} = 1$ if person i is assigned to job j, $x_{ij} = 0$ otherwise
2. Constraints:
   - Each person i is assigned exactly one job:

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad \forall i \in \{1, \ldots, n\}$$

   - Each job j is done by exactly one person:

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad \forall j \in \{1, \ldots, n\}$$

3. Objective function: minimize the total assignment costs

$$\min \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}.$$

The 0-1 Knapsack Problem

Definition

Given:

- n possible projects
- cost a_j for project j
- estimated profit c_j for selecting project j
- available budget B

The goal is to find a subset of projects that maximizes the total profit while not exceeding the budget.

The 0-1 Knapsack Problem - BIP

1. Variables: $x_j \in \{0, 1\},\ \forall j \in \{1, \ldots, n\}$,
   $x_j = 1$ if project j is selected, $x_j = 0$ otherwise
2. Constraint: the budget must not be exceeded

$$\sum_{j=1}^{n} a_j x_j \leq B$$

3. Objective function: maximize the total profit

$$\max \quad \sum_{j=1}^{n} c_j x_j$$

The Set Covering Problem

Definition

Given:

- n potential locations for fire stations
- cost c_j for building a fire station at location j
- m regions that have to be serviced in case of fire
- sets S_j of regions that can be serviced within 5 minutes by a fire station at location j

The goal is to find a set of locations for fire stations with minimal total building costs. Each region has to be reachable by at least one fire station within 5 minutes.

The Set Covering Problem - BIP

1. Variables: $x_j \in \{0, 1\},\ \forall j \in \{1, \ldots, n\}$
   $x_j = 1$ if fire station is built at location j, $x_j = 0$ otherwise
2. Constraint: each region has to be reachable within 5 minutes by at least one fire station

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1, \qquad \forall i \in \{1, \ldots, m\}$$

   ($a_{ij} = 1$ if $i \in S_j$ and $a_{ij} = 0$ if $i \notin S_j$)
3. Objective function: minimize the total building costs

$$\min \quad \sum_{j=1}^{n} c_j x_j$$

The Combinatorial Explosion

We can solve combinatorial optimization problems by simple enumeration

$\Rightarrow$ number of possible (feasible and infeasible) solutions:

- assignment problem: $n!$ (permutations)
- knapsack problem: 2^n (power set)
- set covering problem: 2^n (power set)

$\Rightarrow$ we need a more intelligent algorithm!

MIPs - Modeling of Fixed Costs

Non-linear fixed charge cost function:

$$h(x) = \begin{cases} 0 & \text{if} \quad x = 0, \\ f + px & \text{if} \quad 0 < x \leq C,\ f > 0,\ p > 0. \end{cases}$$

Question: How can we model such functions?

1. Additional variable $y \in \{0, 1\}$:
   - $y = 0$ if $x = 0$
   - $y = 1$ if $x > 0$
2. Additional constraint: $x \leq Cy$
3. Objective function: $\min \quad fy + px$

Note: $x = 0, y = 1$ is feasible in model but infeasible for $h(x)$.
Nevertheless, this case cannot occur in an optimal solution.

Uncapacitated Facility Location (UFL)

Definition

Given:

- set $N = \{1, \ldots, n\}$ of potential depots
- fixed cost f_j for opening depot j
- set $M = \{1, \ldots, m\}$ of clients
- transportation cost c_{ij} for delivering the total demand of client i from depot j

The problem is to decide which depots to open and which depots serve each client. The sum of fixed depot opening and transportation costs should be minimized.

Uncapacitated Facility Location - MIP

1. Variables:
   - $x_{ij} \geq 0,\ \forall i \in M, j \in N$: fraction of demand of client i satisfied by depot j
   - $y_j \in \{0, 1\},\ \forall j \in N$: $y_j = 1$ if depot j is opened, else $y_j = 0$
2. Constraints:
   - The demand of client i has to be satisfied:

$$\sum_{j \in N} x_{ij} = 1, \qquad \forall i \in M$$

   - Linking x and y variables (m is an upper bound):

$$\sum_{i \in M} x_{ij} \leq m y_j, \qquad \forall j \in N$$

3. Objective function: minimize the sum of all costs

$$\min \quad \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} + \sum_{j \in N} f_j y_j$$

Uncapacitated Lot Sizing (ULS)

Definition

Given:

- fixed cost f_t for producing in period t
- unit production cost p_t in period t
- unit storage cost h_t in period t
- demand d_t in period t

The goal is to find a production plan for a single product for the next n time periods. The sum of fixed, production, and storage costs should be minimized.

Uncapacitated Lot Sizing - MIP

1. **(Obvious) variables:**
   - $x_t \geq 0,\ \forall t \in \{1, \ldots, n\}$: amount produced in period t
   - $s_t \geq 0,\ \forall t \in \{0, \ldots, n\}$: stock at the end of period t $(s_0 = s_n = 0)$
   - $y_t \in \{0, 1\},\ \forall t \in \{1, \ldots, n\}$: $y_t = 1$ if production occurs in period t, else $y_t = 0$
2. **Objective function:** minimize the sum of production, storage, and fixed costs

$$\min \quad \sum_{t=1}^{n} p_t x_t + \sum_{t=1}^{n} h_t s_t + \sum_{t=1}^{n} f_t y_t$$

Uncapacitated Lot Sizing - MIP

3. **Constraints:**
   - Flow conservation in each period t:

$$s_{t-1} + x_t - d_t = s_t, \qquad \forall t \in \{1, \ldots, n\}$$

   Note: $s_t = \sum_{i=1}^{t} x_i - \sum_{i=1}^{t} d_i$
   - Linking x and y variables:

$$x_t \leq M_t y_t, \qquad \forall t \in \{1, \ldots, n\}$$

   Question: How large must M_t be? Is there an upper bound on x_t?

$$x_t \leq \left(\sum_{i=t}^{n} d_i \right) y_t, \qquad \forall t \in \{1, \ldots, n\}$$

Discrete Alternatives or Disjunctions

How can we model disjunctions? (e.g., in scheduling problems)

- $x \in \mathbb{R}^n,\ \mathbf{0} \leq \boldsymbol{x} \leq \boldsymbol{u}$
- Either $\boldsymbol{a^1 x} \leq b_1$ or $\boldsymbol{a^2 x} \leq b_2$

For instance that way:

1. Binary variables $y_1, y_2 \in \{0, 1\}$
2. Constraints:

$$\begin{aligned} \boldsymbol{a^i x} - b_i &\leq M(1 - y_i), & i \in \{1, 2\} \\ y_1 + y_2 &= 1 \end{aligned}$$

$M \geq \max_{i \in \{1,2\}} \{\boldsymbol{a^i x} - b_i \mid \mathbf{0} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$

Formulation

Definition

A subset of $\mathbb{R}^n$ described by a finite set of linear constraints $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ is a *polyhedron*.

Definition

A polyhedron $P \subseteq \mathbb{R}^{n+p}$ is a *formulation* for a set $X \subseteq \mathbb{Z}^n \times \mathbb{R}^p$ if and only if $X = P \cap (\mathbb{Z}^n \times \mathbb{R}^p)$.

Are there more equivalent formulations for set X?

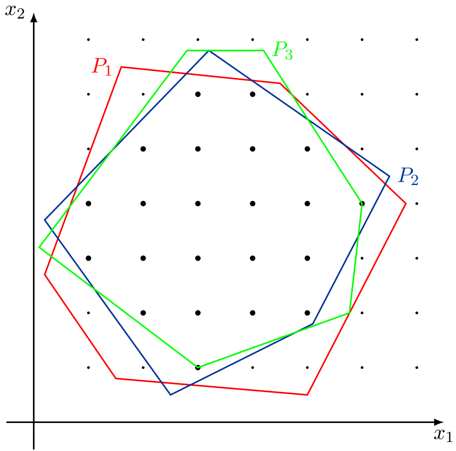


Figure: Dots denote integer values. Big dots denote set X of feasible solutions.

Equivalent Formulation for UFL

- Variables stay the same:
  - $x_{ij} \geq 0, \ \forall i \in M, j \in N$: fraction of demand of client i satisfied by depot j
  - $y_j \in \{0, 1\}, \ \forall j \in N$: $y_j = 1$ if depot j is opened, else $y_j = 0$
- Constraints

$$\sum_{i \in M} x_{ij} \leq m y_j, \qquad \forall j \in N$$

are replaced by

$$x_{ij} \leq y_j, \qquad \forall i \in M, j \in N$$

Question: Is this equivalent formulation better or worse?

Extended Formulation for ULS

1. Variables:
   - $w_{it} \geq 0,\ \forall i, t \in \{1, \ldots, n\}, i \leq t$: amount produced in period i to satisfy demand in period t
   - $y_t \in \{0, 1\},\ \forall t \in \{1, \ldots, n\}$: $y_t = 1$ if production occurs in period t, else $y_t = 0$
2. Constraints:
   - Demand satisfaction in period t:

$$\sum_{i=1}^{t} w_{it} = d_t, \qquad \forall t \in \{1, \ldots, n\}$$

   - Variable upper bounds by linking w and y variables:

$$w_{it} \leq d_t y_i, \qquad \forall i, t \in \{1, \ldots, n\},\ i \leq t$$

Extended Formulation for ULS

3. Objective function is modified for simplicity (no storage costs)

$$\min \quad \sum_{t=1}^{n} p_t x_t + \sum_{t=1}^{n} f_t y_t$$

$\Rightarrow$ variables x_t are expressed by w_{it}:

$$x_t = \sum_{i=t}^{n} w_{ti}$$

Comparing Formulations

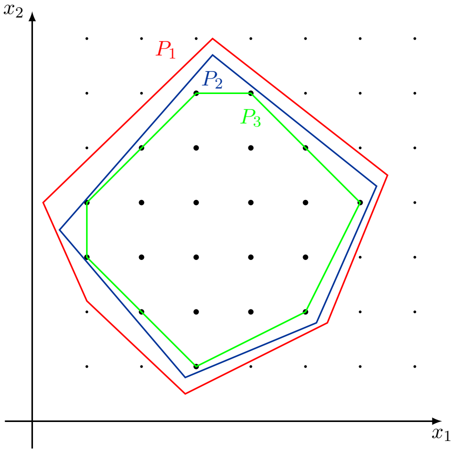


Figure: All formulations are feasible for set X but which is the "best"?

Ideal Formulation

Definition

A formulation P for a set $X \subseteq \mathbb{R}^n$ is *ideal* iff $P = \text{conv}(X)$.

Consequence:

IP: $\{\max \boldsymbol{cx} \mid \boldsymbol{x} \in X\}$ can be replaced by
LP: $\{\max \boldsymbol{cx} \mid \boldsymbol{x} \in \text{conv}(X)\}$
$\Rightarrow$ optimal solution is extreme point of $\text{conv}(X)$
$\Rightarrow$ IP can be solved in polynomial time.

Question: Does the last statement have a hook?

Problem:

In general $\text{conv}(X)$ has no simple characterization and exponentially many constraints.

Good Formulations

$X \subseteq \mathrm{conv}(X) \subseteq P$, for all formulations P, suggests:

Definition

Given a set $X \subseteq \mathbb{R}^n$ and formulations P_1 and P_2 for X,
P_1 is a *better formulation* than P_2 if $P_1 \subset P_2$.

Four different cases:

1. $P_1 = P_2 : \boldsymbol{x} \in P_1 \Leftrightarrow \boldsymbol{x} \in P_2$
2. $P_1 \subset P_2 : (\boldsymbol{x} \in P_1 \Rightarrow \boldsymbol{x} \in P_2) \wedge (\exists \boldsymbol{x} \in P_2 : \boldsymbol{x} \notin P_1)$
3. $P_1 \supset P_2 : (\boldsymbol{x} \in P_2 \Rightarrow \boldsymbol{x} \in P_1) \wedge (\exists \boldsymbol{x} \in P_1 : \boldsymbol{x} \notin P_2)$
4. $P_1 \neq P_2 : (\exists \boldsymbol{x} \in P_1 : \boldsymbol{x} \notin P_2) \wedge (\exists \boldsymbol{x} \in P_2 : \boldsymbol{x} \notin P_1)$

Comparing Formulations for UFL

Let P_1 be the formulation with constraints

$$\sum_{i \in M} x_{ij} \leq m y_j, \qquad \forall j \in N, \tag{1}$$

and P_2 the one with

$$x_{ij} \leq y_j, \qquad \forall i \in M, j \in N. \tag{2}$$

If $(\boldsymbol{x}, \boldsymbol{y}) \in P_2$ then by summing constraints (2) over $i \in M$ we see that $(\boldsymbol{x}, \boldsymbol{y}) \in P_1 \Rightarrow P_2 \subseteq P_1$.

Suppose, $m = n$ and each depot serves exactly one client:
$x_{ij} = 1$ if $i = j$, and $x_{ij} = 0$ otherwise, $y_j = 1/m$, lies in $P_1 \setminus P_2$.
$\Rightarrow P_2 \subset P_1$
$\Rightarrow P_2$ is better than P_1.

Remark: y_j need not be integers here because P_1 and P_2 are polyhedra.

Projections

The two formulations for ULS use different sets of variables. How can we compare them?

Let $P \subseteq \mathbb{R}^n$ be a formulation for integer set $X \subseteq \mathbb{Z}^n$

$$\min\{\boldsymbol{c}'\boldsymbol{x} \mid \boldsymbol{x} \in P \cap \mathbb{Z}^n\}$$

and a second extended formulation $Q \subseteq \mathbb{R}^n \times \mathbb{R}^p$

$$\min\{\boldsymbol{c}'\boldsymbol{x} \mid (\boldsymbol{x}, \boldsymbol{w}) \in Q \cap (\mathbb{Z}^n \times \mathbb{R}^p)\}.$$

Definition

The *projection* of polyhedron $Q \subseteq \mathbb{R}^n \times \mathbb{R}^p$ onto the subspace $P \subseteq \mathbb{R}^n$, denoted $\operatorname{proj}_{\boldsymbol{x}} Q$, is defined as:

$$\operatorname{proj}_{\boldsymbol{x}} Q = \{\boldsymbol{x} \in \mathbb{R}^n \mid (\boldsymbol{x}, \boldsymbol{w}) \in Q \text{ for some } \boldsymbol{w} \in \mathbb{R}^p\}.$$

Comparing Formulations for ULS

Let P_1 be the formulation with constraints

$$\begin{aligned} s_{t-1} + x_t - d_t = s_t, &\qquad \forall t \in \{1, \ldots, n\}, \\ x_t \le M_t y_t, &\qquad \forall t \in \{1, \ldots, n\}, \end{aligned}$$

and $P_2 = \operatorname{proj}_{\boldsymbol{x},\boldsymbol{s},\boldsymbol{y}} Q_2$, where Q_2 is defined by

$$\begin{aligned} \sum_{i=1}^{t} w_{it} = d_t, &\qquad \forall t \in \{1, \ldots, n\}, \\ w_{it} \le d_t y_i, &\qquad \forall i, t \in \{1, \ldots, n\},\ i \le t, \\ x_t = \sum_{i=t}^{n} w_{ti}, &\qquad \forall t \in \{1, \ldots, n\}. \end{aligned}$$

Comparing Formulations for ULS

- It can be shown that P_2 is an ideal formulation.
  $\Rightarrow$ ULS can be solved in polynomial time by solving the LP of P_2.
- From above, $P_2 \subseteq P_1$. To show $P_2 \subset P_1$:
  E.g., the point $x_t = d_t,\ s_t = 0,\ y_t = d_t/M,\ \forall t \in \{1, \ldots, n\}$ lies in $P_1 \setminus P_2$.

$\Rightarrow P_2$ is better than P_1 (and even ideal).

The Traveling Salesman Problem (TSP)

Definition

Given:

- set $N = \{1, \ldots, n\}$ of cities
- cost c_{ij} for traveling from city i to city j

The goal is to find a tour with minimum traveling cost. Each city has to be visited exactly once and the salesman has to return to his starting city at the end.

The Traveling Salesman Problem - BIP

1. Variables: $x_{ij} \in \{0, 1\}, \ \forall i, j \in N, \ i \neq j$

   $$x_{ij} = \begin{cases} 1 & \text{if the salesman travels directly from city } i \text{ to } j, \\ 0 & \text{otherwise.} \end{cases}$$

2. Constraints:
   - The salesman leaves each city i exactly once:

   $$\sum_{j \in N, j \neq i} x_{ij} = 1, \qquad \forall i \in N$$

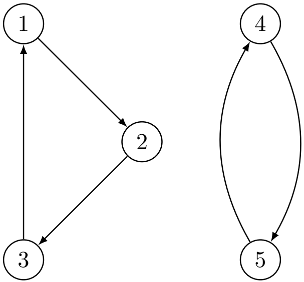
   - The salesman arrives at each city j exactly once:

   $$\sum_{i \in N, i \neq j} x_{ij} = 1, \qquad \forall j \in N$$

3. Objective function: minimize the total traveling costs

   $$\min \quad \sum_{i \in N} \sum_{j \in N, j \neq i} c_{ij} x_{ij}$$

The Traveling Salesman Problem - BIP



- Problem: subtours are allowed!
- Additional constraints to eliminate these subtours are required!

Sequential Formulation

- introduced by Miller, Tucker and Zemlin (1960)
- additional (continuous) variables u_i, are used to indicate the order in which the cities are visited
- additional constraints:

$$\begin{aligned} & u_i + x_{ij} \leq u_j + M \cdot (1 - x_{ij}), && \forall i, j \in N \setminus \{1\},\ i \neq j, \\ & 1 \leq u_i \leq n - 1, && \forall i \in N \setminus \{1\} \end{aligned}$$

- M has to inactivate a constraint if $x_{ij} = 0 \Rightarrow M = n - 2$

Single-Commodity Flow Formulation

- introduced by Gavish and Graves (1978)
- additional (continuous) variables f_{ij} represent the amount of "flow" on arc (i, j)
- additional constraints: node 1 sends out $n - 1$ "goods" and any other node consumes exactly one

$$
\begin{aligned}
&\sum_{j,j \neq 1} f_{1j} = n - 1 && \\
&\sum_{i,i \neq j} f_{ij} - \sum_{k,k \neq j} f_{jk} = 1, && \forall j \in N \setminus \{1\}, \\
&0 \leq f_{ij} \leq (n-1) \cdot x_{ij}, && \forall i, j \in N,\ i \neq j
\end{aligned}
$$

Multi-Commodity Flow Formulation

- introduced by Wong (1980)
- additional (continuous) variables f_{ij}^k represent the amount of "flow" on arc (i, j) for commodity k
- additional constraints: node 1 sends out $n - 1$ different commodities, each one dedicated to a specific node

$$
\begin{aligned}
&\sum_{j, j \neq 1} f_{1j}^k - \sum_{j, j \neq 1} f_{j1}^k = 1, && \forall k \in N \setminus \{1\}, \\
&\sum_{i, i \neq k} f_{ik}^k = 1, && \forall k \in N \setminus \{1\}, \\
&\sum_{i, i \neq j} f_{ij}^k - \sum_{i, i \neq j} f_{ji}^k = 0, && \forall j, k \in N \setminus \{1\},\ j \neq k, \\
&0 \leq f_{ij}^k \leq x_{ij}, && \forall i, j \in N,\ i \neq j,\ \forall k \in N \setminus \{1\}
\end{aligned}
$$

TSP Formulations

Let P_{MTZ}, P_{SCF}, and P_{MCF} be the polyhedra of the Miller-Tucker-Zemlin formulation, the single-commodity flow formulation, and the multi-commodity flow formulation, respectively. Then:

$$P_{\mathrm{MCF}} \subset P_{\mathrm{SCF}} \subset P_{\mathrm{MTZ}}.$$

So MCF is the strongest formulation. Unfortunately, the size of MCF is significantly larger than that of the other two formulations:

Formulation	#variables	#constraints
MTZ	$n(n-1)+(n-1)$	n^2+1
SCF	$n(n-1)+n(n-1)$	n^2+2n
MCF	$n(n-1)+n(n-1)^2$	n^3-n^2+2n

(Mixed) Integer Linear Programming

Relaxations and Bounds

Optimality Condition

Given IP or COP $\max\{c(\boldsymbol{x}) \mid \boldsymbol{x} \in X \subseteq \mathbb{Z}^n\}$

Definition

A solution $\boldsymbol{x}^*$ with $z^* = c(\boldsymbol{x}^*)$ is optimal if there is a lower bound $\underline{z} \leq z^*$ and an upper bound $\overline{z} \geq z^*$, such that $\underline{z} = z^* = \overline{z}$.

An algorithm tries to find an increasing sequence of lower bounds and a decreasing sequence of upper bounds

$$\underline{z_1} < \underline{z_2} < \cdots < \underline{z_l} \leq z^* \leq \overline{z_u} < \cdots < \overline{z_2} < \overline{z_1}$$

and can stop when

$$\overline{z_u} - \underline{z_l} \leq \epsilon \ (\geq 0).$$

But how can we derive lower and upper bounds?

Primal Bounds

Definition

Lower bounds in maximization problems and upper bounds in minimization problems are called primal bounds.

Every feasible solution $\boldsymbol{x} \in X$ is a primal (in our case lower) bound $\underline{z} = c(\boldsymbol{x}) \leq z^*$.
$\Rightarrow$ Use (meta-)heuristics to derive feasible solutions and (hopefully) good primal bounds.

(see lecture *Heuristic Optimization Techniques* in winter term)

Dual Bounds

Definition

Upper bounds in maximization problems and lower bounds in minimization problems are called dual bounds.

In most cases dual bounds are obtained by relaxations:
a difficult (maximization) IP is replaced by a simpler optimization problem with an optimal value at least as large as z^*, e.g., by removing some constraints.

Relaxations

Definition

A problem RP $z^R = \max\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in Y \subseteq \mathbb{R}^n\}$ is a relaxation of IP $z^* = \max\{c(\boldsymbol{x}) \mid \boldsymbol{x} \in X \subseteq \mathbb{R}^n\}$ if:

1. $X \subseteq Y$, and
2. $f(\boldsymbol{x}) \geq c(\boldsymbol{x}),\ \forall \boldsymbol{x} \in X$.

Theorem

If RP is a relaxation of IP, then $z^R \geq z^*$.

Proof.

If $\boldsymbol{x}^*$ is an optimal solution of IP, $\boldsymbol{x}^* \in X \subseteq Y$, and $z^* = c(\boldsymbol{x}^*) \leq f(\boldsymbol{x}^*)$. As $\boldsymbol{x}^* \in Y, f(\boldsymbol{x}^*)$ is a lower bound on z^R, and so $z^* \leq f(\boldsymbol{x}^*) \leq z^R$. $\square$

Linear Programming Relaxations

Definition

Given an IP $\max\{\boldsymbol{cx} \mid \boldsymbol{x} \in P \cap \mathbb{Z}^n\}$ with formulation P, the LP relaxation is the LP $z^{LP} = \max\{\boldsymbol{cx} \mid \boldsymbol{x} \in P\}$.

Since $P \cap \mathbb{Z}^n \subseteq P$ and the objective function is not changed, the LP relaxation is a relaxation.

Better formulations give $tighter$ (dual) bounds:

Theorem

Let P_1 and P_2 be two formulations for IP $\max\{\boldsymbol{cx} \mid \boldsymbol{x} \in X \subseteq \mathbb{Z}^n\}$, and P_1 is better than P_2, i.e. $P_1 \subset P_2$.
If $z_i^{LP} = \max\{\boldsymbol{cx} \mid \boldsymbol{x} \in P_i\},\ i \in \{1, 2\}$, then
$z_1^{LP} \leq z_2^{LP},\ \forall \boldsymbol{c} \in \mathbb{R}^n$.

Optimality by Relaxations

In some cases relaxations can prove optimality:

Theorem

1. *If relaxation RP is infeasible, the original IP is infeasible.*
2. *Let $\boldsymbol{x}^*$ be an optimal solution to RP. If $\boldsymbol{x}^* \in X$ and $f(\boldsymbol{x}^*) = c(\boldsymbol{x}^*)$, then $\boldsymbol{x}^*$ is optimal for IP.*

Proof.

1. As RP is infeasible, $Y = \emptyset$. Since $X \subseteq Y$ holds for a relaxation, $X = \emptyset$.
2. As $\boldsymbol{x}^* \in X$, $z^* \geq c(\boldsymbol{x}^*) = f(\boldsymbol{x}^*) = z^R$. Since $z^* \leq z^R$, we get $c(\boldsymbol{x}^*) = z^* = z^R$.

$\square$

Combinatorial Relaxations

Definition

If the relaxation is a COP, we speak of a combinatorial relaxation.

Preferably, these COPs are easy problems which can be solved efficiently, e.g.:
When considering the TSP on a digraph $D = (V, A)$, feasible tours T correspond to assignments containing no subtours $\Rightarrow$

$$z^{TSP} = \min_{T \subseteq A} \left\{ \sum_{(i,j) \in T} c_{ij} \mid T \text{ is a tour} \right\} \geq$$

$$z^{ASS} = \min_{T \subseteq A} \left\{ \sum_{(i,j) \in T} c_{ij} \mid T \text{ is an assignment} \right\}$$

Combinatorial Relaxation for Symmetric TSP

Given an undirected graph $G = (V, E)$ with edge costs $c_e, \ \forall e \in E$, we search for an undirected tour with minimum costs.

- Every tour consists of two edges incident to node 1, and a path through nodes $\{2, \ldots, n\}$.
- A path is a special case of a tree.

Definition

A 1-tree is a subgraph containing two edges adjacent to node 1, and the edges of a tree on nodes $\{2, \ldots, n\}$.

Every tour is a 1-tree $\Rightarrow$

$$z^{STSP} = \min_{T \subseteq E} \left\{ \sum_{e \in T} c_e \mid T \text{ is a tour} \right\} \geq$$

$$z^{\textit{1-tree}} = \min_{T \subseteq E} \left\{ \sum_{e \in T} c_e \mid T \text{ is a 1-tree} \right\}$$

Combinatorial Relaxation for the Knapsack Problem

Given the set

$$X = \{\boldsymbol{x} \in \mathbb{Z}_+^n : \sum_{j=1}^{n} a_j x_j \leq B\}$$

a relaxation of set X is

$$X' = \{\boldsymbol{x} \in \mathbb{Z}_+^n : \sum_{j=1}^{n} \lfloor a_j \rfloor x_j \leq \lfloor B \rfloor\}.$$

Question: Why is this relaxation easier to solve?

There are efficient pseudo-polynomial dynamic programming algorithms for instances with integer weights.

Duality

- LP duality provides a standard way to obtain dual bounds.
- Note: Any dual feasible solution provides a dual bound, but a relaxation must be solved to optimality to get a dual bound.

Theorem

Given the IP $z^* = \max\{\boldsymbol{cx} \mid \boldsymbol{Ax} \leq \boldsymbol{b}, \boldsymbol{x} \in \mathbb{Z}^n_+\}$ and the associated dual problem DIP $w^* = \min\{\boldsymbol{ub} \mid \boldsymbol{uA} \geq \boldsymbol{c}, \boldsymbol{u} \in \mathbb{Z}^m_+\}$. Let LP and DLP be the according linear programming relaxations, respectively. Then, IP and DP form a weak dual pair. Strong duality does not hold in general.

Proof.

Since LP and DLP are relaxations, $z^* \leq z^{LP}$ and $w^{DLP} \leq w^*$. Because of strong LP duality, $z^{LP} = w^{DLP}$ holds. Thus, $z^* \leq z^{LP} = w^{DLP} \leq w^*$. □

Optimality by Duality

Similar to relaxations, dual problems can sometimes prove optimality:

Theorem

Let IP $\max\{c(\boldsymbol{x}) \mid \boldsymbol{x} \in X\}$ *and DIP* $\min\{\omega(\boldsymbol{u}) \mid \boldsymbol{u} \in U\}$ *be a weak dual pair.*

1. *If DIP is unbounded, IP is infeasible.*
2. *If* $\boldsymbol{x}^* \in X$ *and* $\boldsymbol{u}^* \in U$ *satisfy* $c(\boldsymbol{x}^*) = \omega(\boldsymbol{u}^*)$, *then* $\boldsymbol{x}^*$ *is optimal for IP and* $\boldsymbol{u}^*$ *is optimal for DIP.*

(Mixed) Integer Linear Programming

Branch-and-Bound

Divide and Conquer

Given a problem $z = \max\{\boldsymbol{cx} \mid \boldsymbol{x} \in S\}$.
Use divide and conquer approach:

1. break the problem into smaller and easier problems
2. solve all smaller problems
3. put information together to solve original problem

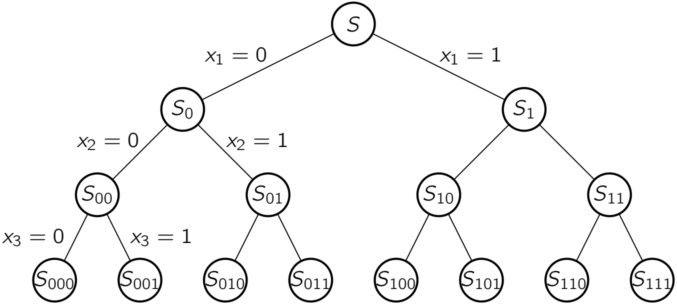
Theorem

Let $S = S_1 \cup \cdots \cup S_K$ be a decomposition of S into smaller sets, and let $z^k = \max\{\boldsymbol{cx} \mid \boldsymbol{x} \in S_k\}$, $\forall k \in \{1, \ldots, K\}$. Then $z = \max_k z^k$.

A divide and conquer approach can be represented by an enumeration tree.

Binary Branching

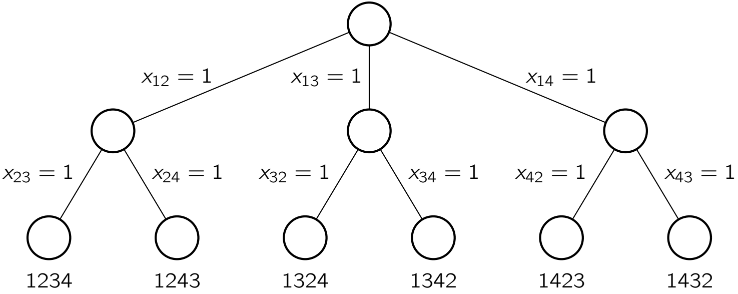
E.g. $S \subseteq \{0,1\}^3$:



A leaf $S_{i_1 i_2 i_3}$ is non-empty if and only if $\boldsymbol{x} = (i_1, i_2, i_3) \in S$. The leaves directly correspond to points in $\{0,1\}^3$.

Multiway Branching

E.g., asymmetric TSP with $n = 4$:



Leaves correspond to the $(n-1)!$ *feasible* tours. Here we used problem-specific branching.

Implicit Enumeration

Complete enumeration is computationally impossible for $|\boldsymbol{x}| > 20$ for many problems. $\Rightarrow$

- How can we benefit from bound information to prune complete subtrees without explicitly enumerating them?
- How can we put together bound information?

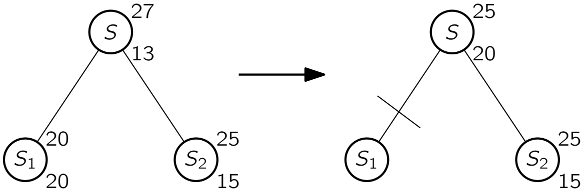
Theorem

Let $S = S_1 \cup \cdots \cup S_K$ be a decomposition of S into smaller sets, and let $z^k = \max\{\boldsymbol{cx} \mid \boldsymbol{x} \in S_k\},\ \forall k \in \{1, \ldots, K\},\ \overline{z}^k$ be an upper bound and $\underline{z}^k$ be a lower bound on z^k. Then $\overline{z} = \max_k \overline{z}^k$ is an upper bound and $\underline{z} = \max_k \underline{z}^k$ a lower bound on z.

Note: For minimization problems $\overline{z} = \min_k \overline{z}^k$ and $\underline{z} = \min_k \underline{z}^k$.

Prune by Optimality

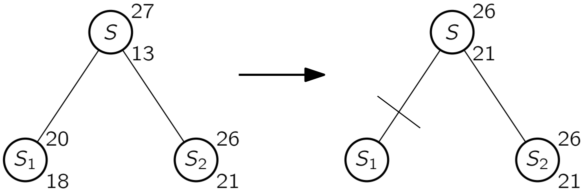
Given a maximization problem and a decomposition of S into two sets $S_1,\ S_2$, and lower and upper bounds on these problems:



- $\overline{z} = \max_k \overline{z}^k = \max\{20, 25\} = 25$ and
  $\underline{z} = \max_k \underline{z}^k = \max\{20, 15\} = 20$
- $\overline{z}^1 = \underline{z}^1 = z^1 \Rightarrow$ problem z^1 is solved.

In general, if $z^t = \max\{\boldsymbol{cx} \mid \boldsymbol{x} \in S_t\}$ is solved, branch S_t can be pruned.

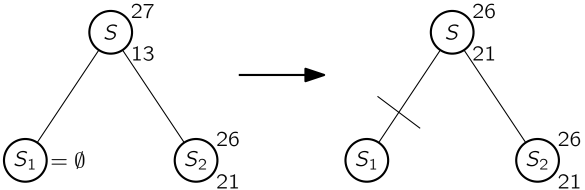
Prune by Bound



- $\overline{z} = \max_k \overline{z}^k = \max\{20, 26\} = 26$ and
  $\underline{z} = \max_k \underline{z}^k = \max\{18, 21\} = 21$
- $\underline{z} = 21$ and $\overline{z}^1 = 20 \Rightarrow$ no optimal solution can lie in S_1.

In general, if $\overline{z}^t \leq \underline{z}$, branch S_t can be pruned.

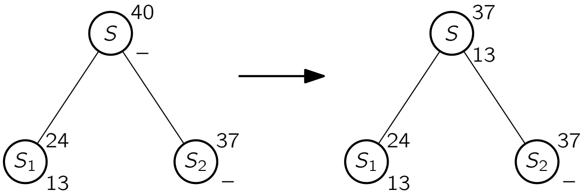
Prune by Infeasibility



- $\overline{z} = \max_k \overline{z}^k = \max\{26\} = 26$ and
  $\underline{z} = \max_k \underline{z}^k = \max\{21\} = 21$
- $S_1 = \emptyset \Rightarrow z^1$ is infeasible.

In general, if $S_t = \emptyset$, branch S_t can be pruned.

No Pruning Possible



- $\overline{z} = \max_k \overline{z}^k = \max\{24, 37\} = 37$ and
  $\underline{z} = \max_k \underline{z}^k = \max\{13\} = 13$

Here, no pruning is possible $\Rightarrow$ both subproblems have to be examined further.

Open Questions

The implementation of an implicit enumeration algorithm seems quite easy, but:

- What relaxation or dual problem should be used to provide dual bounds?
  Should we use a weak bound which can be calculated quickly or a stronger bound which takes more time to obtain?
- How should we separate a set S into smaller sets?
  Should we use binary or multiway branching? Should the separation rule be fixed or obtained dynamically depending on bounds and solutions?
- In what order should we examine the open subproblems?
  FIFO, LIFO, or something else?

LP-based Branch-and-Bound

- Most common way to solve (M)IPs
- LP relaxation used as dual (upper) bound
- Binary branching by splitting set S on fractional variables: if x_j is integer and $\overline{x}_j = x_j^{LP} \notin \mathbb{Z}$ then

$$S_1 = S \cap \{x \mid x_j \leq \lfloor \overline{x}_j \rfloor\},$$
$$S_2 = S \cap \{x \mid x_j \geq \lceil \overline{x}_j \rceil\},$$

  $\Rightarrow S_1 \cup S_2 = S,\ S_1 \cap S_2 = \emptyset,\ \boldsymbol{x}^{LP} \notin LP(S_1),\ \boldsymbol{x}^{LP} \notin LP(S_2)$
  $\Rightarrow \max\{\overline{z}^1, \overline{z}^2\} \leq \overline{z} \Rightarrow$ the upper bound monotonically decreases.

- Updating the incumbent solution (best primal bound): if $\boldsymbol{x}^{t,LP} \in S$ (feasible for original problem)
  $\Rightarrow \underline{z} = \max\{\underline{z}, z^t\} \Rightarrow$ prune branch S_t by optimality

Algorithm 1: LP-based Branch-and-Bound

problem list $L : \max\{\boldsymbol{cx} \mid \boldsymbol{x} \in S\}$
$\underline{z} = -\infty$, incumbent $\boldsymbol{x}^* = NULL$
while $L \neq \emptyset$ **do**
- choose set S_i and remove it from L
- solve $\overline{z}^i = LP(S_i)$
- $\boldsymbol{x}^{i,LP}$ = optimal LP solution
- **if** $S_i = \emptyset$ **then** prune S_i by infeasibility
- **else if** $\overline{z}^i \leq \underline{z}$ **then** prune S_i by bound
- **else if** $\boldsymbol{x}^{i,LP} \in S$ **then** /* $\overline{z}^i = \underline{z}^i = z^i$ */
  - **if** $z^i \geq \underline{z}$ **then**
    - update primal bound $\underline{z} = z^i$
    - update incumbent $\boldsymbol{x}^* = \boldsymbol{x}^{i,LP}$
  - prune S_i by optimality
- **else** $L = L \cup \{S_{i,1}, S_{i,2}\}$

How to Choose a Branching Variable

Typically, there is a set C of integer variables that are fractional in the current LP solution. Which one should be chosen to branch on?

- Random variable x_i
- Most fractional variable x_i:

$$i = \arg\max_{j \in C} \min\{f_j, 1 - f_j\} \quad \text{with } f_j = x_j^{LP} - \lfloor x_j^{LP} \rfloor$$

- Strong branching: spend more time on the decision
  1. try each $x_j \in C$ as branching variable, branch up (U) and down (D) and resolve all LPs $\Rightarrow z_j^U, z_j^D, \ \forall x_j \in C$
  2. choose actual branching variable x_i with largest decrease of dual bound

$$i = \arg\min_{j \in C} \max\{z_j^U, z_j^D\}$$

How to Choose a Subproblem

- Random choice
- Depth-First-Search strategy at first goes down the enumeration tree to quickly find primal bounds $\Rightarrow$
  - mostly, pruning is only possible with a strong primal bound
  - the LP relaxation is easier to resolve by just adding one simple constraint (beneficial when using simplex algorithms)
- Best-Node-First strategy chooses problem i with best dual (largest upper) bound $\overline{z}^i = \max_t \overline{z}^t \Rightarrow$
  - the total number of examined problems is minimized
  - a problem with $\overline{z}^t < z$ is never separated (which would be pruned later)

In practice, a combination of the last two strategies is used.

Additional Features in Practice

Commercial branch-and-bound systems (e.g., IBM ILOG CPLEX, Gurobi) often include:

- a preprocessor to simplify the formulations by reducing the number of constraints and variables
- different simplex variants and interior point methods chosen automatically or by hand
- various branching and problem selection options
- user-defined priorities on integer variables to control branching
- various primal heuristics based on LP solutions to obtain good primal bounds
- user-defined cutoff values

If All Else Fails

If the problem is too hard, such that

- no feasible solution has been found, or
- the gap between upper and lower bound is very large, or
- the system runs out of memory due to a very large list of not yet examined problems,

then

- find better primal bounds by external (meta-)heuristics, or
- find better dual bounds by improving (tightening) the formulation for the problem, or use other dual bounds, e.g., Lagrangian relaxation.

(Mixed) Integer Linear Programming

Outlook

Advanced Topics

- dealing with exponentially many constraints
- dealing with exponentially many variables
- decomposition approaches
- special problem classes
- (strong) valid inequalities
- ...

$\Rightarrow$ VU Mathematical Programming