CIS 410/510 (Spring 2020): Multi-Agent Systems and Game Theory

Lecture 2: Linear Programming

Thanh H. Nguyen
University of Oregon
Some slides are by Milind Tambe and Haifeng Xu
Announcement

- Programming assignment 1
  - Will be posted today (April 01)
  - Deadline: April 13, 2020
Example: Manufacturing

- A factory makes tables and chairs
- Profit of a table: $4
- Profit of a chair: $2
- Resources to make a table: 1 unit of metal and 3 units of wood
- Resources to make a chair: 2 units of metal and 1 unit of wood
- Total available resources: 6 units of metal and 9 units of wood

**Questions:** how many tables and chairs should the factory make to maximize profit?
Example: Manufacturing

- Number of tables: $x_1$
- Number of chairs: $x_2$
- #units of metal used: $x_1 + 2x_2$
- #units of wood used: $3x_1 + x_2$
- Total profit: $4x_1 + 2x_2$

$$\max_{x_1, x_2} 4x_1 + 2x_2$$
$$\text{s.t. } x_1 + 2x_2 \leq 6$$
$$3x_1 + x_2 \leq 9$$
$$x_1, x_2 \geq 0$$
Geometric Interpretation

\[
\begin{align*}
3x_1 + x_2 &= 9 \\
4x_1 + 2x_2 &= 6 \\
x_1 + 2x_2 &= 6 \\
x_1 + x_2 &= 13.2
\end{align*}
\]

\[
\begin{align*}
\text{max } 4x_1 + 2x_2 \\
\text{s.t. } x_1 + 2x_2 &\leq 6 \\
3x_1 + x_2 &\leq 9 \\
x_1, x_2 &\geq 0
\end{align*}
\]
Linear Programming

- Mathematical program, refers to an optimization problem
  - A set of input variables
  - Optimization objective expressed as function of input variables
    - Maximize or minimize
  - Set of constraints on input variables, as mathematical functions
General Linear Program

- Variables: $x_1, x_2, \ldots, x_n$
- Maximize (or minimize) $c_1x_1 + c_2x_2 + \ldots + c_nx_n$ (objective function)
- Subject to constraints:
  
  $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \sim b_i \quad i=1\ldots m$ (structural constraints)
  
  $x_j \geq 0 \quad j=1\ldots n$ (non-negativity constraints)

- $\sim$ refers to either $=$, $\leq$, or $\geq$.
- Some non-negativity constraints may be dropped
- Why is this a LINEAR program?
- Linear program can be solved in polynomial time!!!
Applications of Linear Programming

- Finding optimal strategy in zero-sum two player games
- Finding solution in Markov decision processes
- Min-cut / max-flow network problems
- Applications: transportation optimization, economic portfolio optimization, robotic control, scheduling generation, etc.
Mixed Integer Linear Programming (MILP)

- Some variables are constrained to be integer
- Example: Manufacturing
  - Number of tables and chairs are integer

When it is difficult to express something as an integer program, try using binary variables.
Practice 1: Max-Flow

- Flow network:
  - A directed graph
  - Each edge has a capacity and receives a flow

- Flow constraints
  - Capacity:
    - Flow does not exceed capacity
  - Flow conservation:
    - Flow leaving a node $v = \text{flow entering } v$

- Example: transport network

Thanh H. Nguyen
Practice 1: Max-Flow

- Given a directed graph $G=(V,E)$ with capacities $u_e$ on edges $e$, a source node $s$, a sink node $t$, find a maximum flow from $s$ to $t$ respecting the capacities.
Practice 1: Max-Flow

- Flow variables
  - $x_{s,1}, x_{s,2}, x_{1,3}, x_{1,4}, x_{2,4}, x_{3,t}, x_{4,t}$

- Flow constraints
  - Capacity:
    - Flow does not exceed capacity
  - Flow conservation:
    - Flow leaving a node $v = $ flow entering $v$

- Network flow: $x_{s,1} + x_{s,2}$
Practice 1: Max-Flow

Maximize network flow

Flow conservation

Capacity

\[
\begin{align*}
\text{maximize} & \quad x_{s,1} + x_{s,2} \\
\text{subject to} & \quad x_{s,1} = x_{1,2} + x_{1,4} \\
& \quad x_{s,2} = x_{2,4} \\
& \quad x_{1,3} = x_{3,t} \\
& \quad x_{1,4} + x_{2,4} = x_{4,t} \\
& \quad x_{s,1} \leq 4, x_{s,2} \leq 5, x_{1,3} \leq 2 \\
& \quad x_{1,4} \leq 1, x_{2,4} \leq 4 \\
& \quad x_{3,t} \leq 3, x_{4,t} \leq 2 \\
& \quad x_{s,1}, x_{s,2}, x_{1,3}, x_{1,4}, x_{2,4}, x_{3,t}, x_{4,t} \geq 0
\end{align*}
\]
Practice 2: Minimum Vertex Cover

- Given a undirected graph $G=(V,E)$ with a cost $c_v$ for every vertex $v$, find a vertex cover $S$ minimizes the total cost of nodes in $S$.

- Vertex cover: a set of vertices $S$ such that all edges in $E$ have at least one endpoint in $S$.

- Example: surveillance system
Practice 2: Minimum Vertex Cover

- Binary variables: \( x_v = 1 \) if \( v \) belongs to the cover set \( S \) and \( x_v = 0 \) otherwise

\[
\min_x \sum_v c_v x_v \quad \text{Total cost}
\]

\[
\text{s.t.} \quad x_u + x_v \geq 1, \ \forall (u, v) \in E
\]

\[
x_v \in \{0, 1\}, \ \forall v \in V
\]

At least one end point
Practice 3: Zero-sum Games

- **Zero-sum games**: defender’s utility is opposite to the attacker’s utility.

- **Example:**
  - Two targets
  - The defender can only protect one target at a time.
  - The attacker attacks one of the targets.
  - A defense strategy is a probability distribution over the two targets, i.e., the probabilities the defender protects each target.

<table>
<thead>
<tr>
<th></th>
<th>Target 1</th>
<th>Target 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Defender</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Target 1</strong></td>
<td>0, 0</td>
<td>-700, 700</td>
</tr>
<tr>
<td><strong>Target 2</strong></td>
<td>-100, 100</td>
<td>200, -200</td>
</tr>
</tbody>
</table>

Thinh H. Nguyen
Practice 3: Zero-sum Games

- Maximin: a strategy that maximizes my payoff, assuming my opponent knows my strategy and always minimizes my payoff.

- Minimax: a strategy that minimizes my opponent’s payoff, assuming he knows my strategy and always maximizes his own payoff.

- In zero-sum games, Maximin, and Minimax strategies are the same.

\[
\begin{array}{|c|c|}
\hline
\text{Target 1} & \text{Target 2} \\
\hline
0, 0 & -700, 700 \\
\hline
-100, 100 & 200, -200 \\
\hline
\end{array}
\]
Practice 3: Zero-sum Games

- Defender’s strategy
  - Protect target 1: $x_1$
  - Protect target 2: $x_2$

- If attacker attacks target 1
  - $EU_{\text{def}}(T_1 \text{ is attacked}) = 0x_1 - 100x_2$

- If attacker attacks target 2
  - $EU_{\text{def}}(T_2 \text{ is attacked}) = -700x_1 + 200x_2$

- Maximin:
  - Assumption: the attacker attacks target with the lowest expected utility for the defender

\[
\begin{array}{c|cc}
\text{Attacker} & \text{Target 1} & \text{Target 2} \\
\hline
\text{Defender} & 0, 0 & -700, 700 \\
\hline
\text{Target 1} & 0, 0 & -700, 700 \\
\text{Target 2} & -100, 100 & 200, -200 \\
\end{array}
\]
Practice 3: Zero-sum Games

- Linear program
  - Variable u: defender’s expected utility for playing \((x_1, x_2)\)

\[
\begin{align*}
\max_{x_1, x_2, u} & \quad u \\
s.t. & \quad u \leq 0x_1 - 100x_2 \\
& \quad u \leq -700x_1 + 200x_2 \\
& \quad x_1 + x_2 = 1 \\
& \quad x_1, x_2 \geq 0
\end{align*}
\]

\(\text{EU}_{\text{def}}(T_1 \text{ is attacked})\)

\(\text{EU}_{\text{def}}(T_2 \text{ is attacked})\)

<table>
<thead>
<tr>
<th></th>
<th>Target 1</th>
<th>Target 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacker</td>
<td>0, 0</td>
<td>-700, 700</td>
</tr>
<tr>
<td>Target 1</td>
<td>-100, 100</td>
<td>200, -200</td>
</tr>
</tbody>
</table>
IBM ILOG CPLEX Optimization Studio

- An optimization software package

- We will import Cplex as a module with Python API.
Example

- Optimization problem:

\[
\begin{align*}
\text{max} & \quad x_1 + 2x_2 + 3x_3 \\
\text{subject to:} & \\
-x_1 + x_2 + x_3 & \leq 20 \\
x_1 - 3x_2 + x_3 & \leq 30 \\
x_1, x_2, x_3 & \geq 0 \\
x_1 & \leq 40
\end{align*}
\]

```python
import sys
import cplex
from cplex.exceptions import CplexError

# data common to all populateby functions
my_obj = [1.0, 2.0, 3.0]
my_ub = [40.0, cplex.infinity, cplex.infinity]
my_colnames = ['x1', 'x2', 'x3']
my_rhs = [20.0, 30.0]
my_rownames = ['c1', 'c2']
my_sense = 'LL'
```
Example

\[
\begin{align*}
\text{max} \quad & x_1 + 2x_2 + 3x_3 \\
\text{subject to:} \quad & -x_1 + x_2 + x_3 \leq 20 \\
& x_1 - 3x_2 + x_3 \leq 30 \\
& x_1, x_2, x_3 \geq 0 \\
& x_1 \leq 40
\end{align*}
\]

- Note that:

```python
import cplex

import cplex
from cplex.exceptions import CplexError

# data common to all populateby functions
my_obj = [1.0, 2.0, 3.0]
my_ub = [40.0, cplex.infinity, cplex.infinity]
my_colnames = ['x1', 'x2', 'x3']
my_rhs = [20.0, 30.0]
my_rownames = ['c1', 'c2']
my_sense = 'LL'
```

```python
def populatebycolumn(prob):
    prob.objective.set_sense(prob.objective.sense.maximize)
    prob.linear_constraints.add(rhs=my_rhs, senses=my_sense,
                                names=my_rownames)
    c = [[[0, 1], [-1.0, 1.0]],
         [['c1', 1], [1.0, -3.0]],
         [[0, 'c2'], [1.0, 1.0]]]
    prob.variables.add(obj=my_obj, ub=my_ub, names=my_colnames,
                        columns=c)
```
Example

\[
\begin{align*}
\text{max} \quad & x_1 + 2x_2 + 3x_3 \\
\text{subject to:} \quad & -x_1 + x_2 + x_3 \leq 20 \\
& x_1 - 3x_2 + x_3 \leq 30 \\
& x_1, x_2, x_3 \geq 0 \\
& x_1 \leq 40
\end{align*}
\]

```python
import sys
import cplex
from cplex.exceptions import CplexError

# data common to all populateby functions
my_obj = [1.0, 2.0, 3.0]
my_ub = [40.0, cplex.infinity, cplex.infinity]
my_colnames = ["x1", "x2", "x3"]
my_rhs = [20.0, 30.0]
my_rownames = ["c1", "c2"]
my_sense = "LL"

def populatebyrow(prob):
    prob.objective.set_sense(prob.objective.sense.maximize)

    # since lower bounds are all 0.0 (the default), lb is omitted
    prob.variables.add(obj=my_obj, ub=my_ub, names=my_colnames)

    rows = [[[0, "x2", "x3"], [-1.0, 1.0, 1.0]],
            [["x1", 1, 2], [1.0, -3.0, 1.0]]]

    prob.linear_constraints.add(lin_expr=rows, senses=my_sense,
                                rhs=my_rhs, names=my_rownames)
```

Thanh H. Nguyen 3/31/20
```python
def lpex1(pop_method):
    try:
        my_prob = cplex.Cplex()
        if pop_method == "r":
            populatebyrow(my_prob)
        elif pop_method == "c":
            populatebycolumn(my_prob)
        else:
            raise ValueError('pop_method must be one of "r", or "c"')

    my_prob.solve()
    except CplexError as exc:
        raise

    numcols = my_prob.variables.get_num()

    print()
    print("Solution value = ", my_prob.solution.get_objective_value())

    x = my_prob.solution.get_values()
    for j in range(numcols):
        print("Column %d: Value = %10f " % (j, x[j]))

    my_prob.write("lpex1.lp")
```
Programming Assignment 1