Game-theoretic
Foundations of Multi-agent Systems

Lecture 4: Computing Solution Concepts of Normal-form Games

Seyed Majid Zahedi

UNIVERSITY OF
WATERLOO

Outline

1. Brief Overview of (Mixed Integer) Linear Programming
2. Dominated Strategies
3. Minmax and Maxmin Strategies
4. Nash Equilibrium
5. Correlated NE

Example: Reproduction of Two Paintings

Example: Reproduction of Two Paintings

- Painting 1 sells for \$30

Example: Reproduction of Two Paintings

- Painting 1 sells for \$30
- Painting 2 sells for \$20

Example: Reproduction of Two Paintings

- Painting 1 sells for \$30
- Painting 2 sells for \$20
- We have 16 units of blue, 8 green, 5 red

Example: Reproduction of Two Paintings

- Painting 1 sells for \$30
- Painting 2 sells for \$20
- We have 16 units of blue, 8 green, 5 red
- Painting 1 requires 4 blue, 1 green, 1 red

Example: Reproduction of Two Paintings

- Painting 1 sells for \$30
- Painting 2 sells for \$20
- We have 16 units of blue, 8 green, 5 red
- Painting 1 requires 4 blue, 1 green, 1 red
- Painting 2 requires 2 blue, 2 green, 1 red

Example: Reproduction of Two Paintings


- Painting 1 sells for \$30
- Painting 2 sells for \$20
- We have 16 units of blue, 8 green, 5 red
- Painting 1 requires 4 blue, 1 green, 1 red
- Painting 2 requires 2 blue, 2 green, 1 red

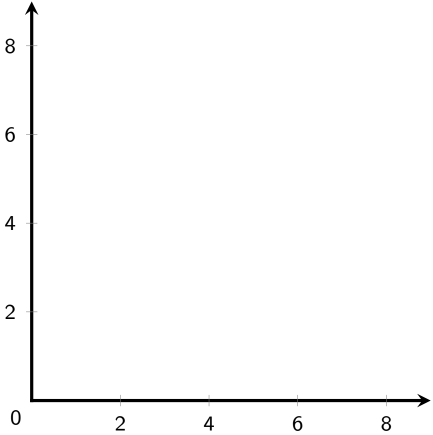
$$
\begin{aligned}
\text{max.} \quad & 3x + 2y \\
\text{s.t.} \quad & 4x + 2y \leq 16 \\
& x + 2y \leq 8 \\
& x + y \leq 5 \\
& x \geq 0 \\
& y \geq 0
\end{aligned}
$$

Solving Linear Program Graphically

$$
\begin{aligned}
\text{max.} \quad & 3x + 2y \\
\text{s.t.} \quad & 4x + 2y \le 16 \\
& x + 2y \le 8 \\
& x + y \le 5 \\
& x \ge 0 \\
& y \ge 0
\end{aligned}
$$

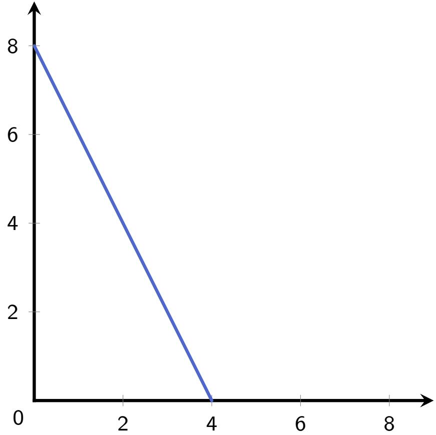
Solving Linear Program Graphically

$$
\begin{aligned}
\text{max.} \quad & 3x + 2y \\
\text{s.t.} \quad & 4x + 2y \leq 16 \\
& x + 2y \leq 8 \\
& x + y \leq 5 \\
& x \geq 0 \\
& y \geq 0
\end{aligned}
$$



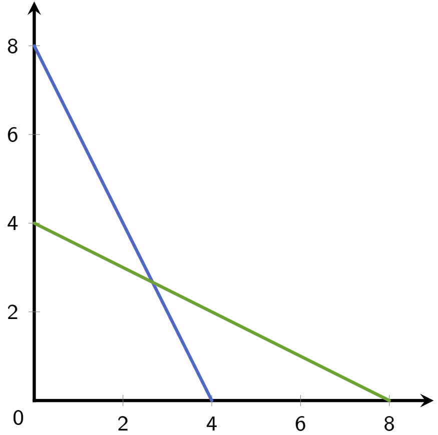
Solving Linear Program Graphically

$$
\begin{aligned}
\text{max.} \quad & 3x + 2y \\
\text{s.t.} \quad & 4x + 2y \leq 16 \\
& x + 2y \leq 8 \\
& x + y \leq 5 \\
& x \geq 0 \\
& y \geq 0
\end{aligned}
$$



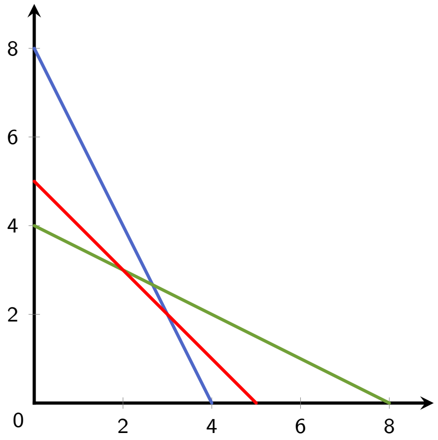
Solving Linear Program Graphically

$$
\begin{aligned}
\text{max.} \quad & 3x + 2y \\
\text{s.t.} \quad & 4x + 2y \leq 16 \\
& x + 2y \leq 8 \\
& x + y \leq 5 \\
& x \geq 0 \\
& y \geq 0
\end{aligned}
$$



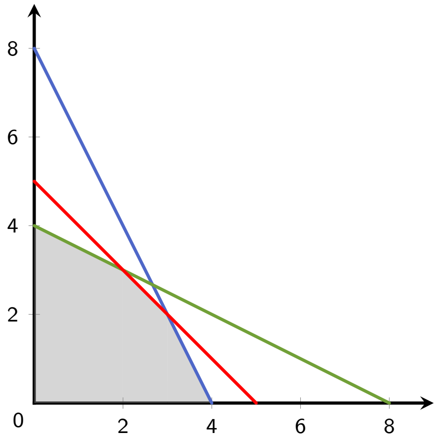
Solving Linear Program Graphically

$$\begin{aligned} \text{max.} \quad & 3x + 2y \\ \text{s.t.} \quad & 4x + 2y \leq 16 \\ & x + 2y \leq 8 \\ & x + y \leq 5 \\ & x \geq 0 \\ & y \geq 0 \end{aligned}$$



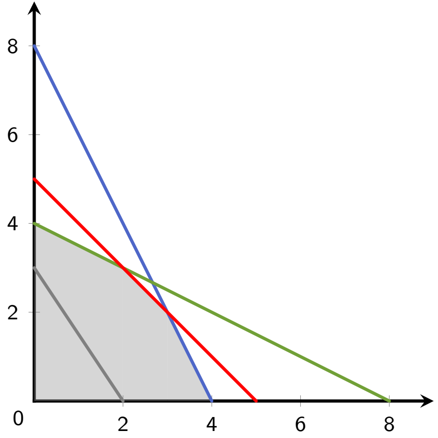
Solving Linear Program Graphically

$$
\begin{aligned}
\text{max.} \quad & 3x + 2y \\
\text{s.t.} \quad & 4x + 2y \leq 16 \\
& x + 2y \leq 8 \\
& x + y \leq 5 \\
& x \geq 0 \\
& y \geq 0
\end{aligned}
$$



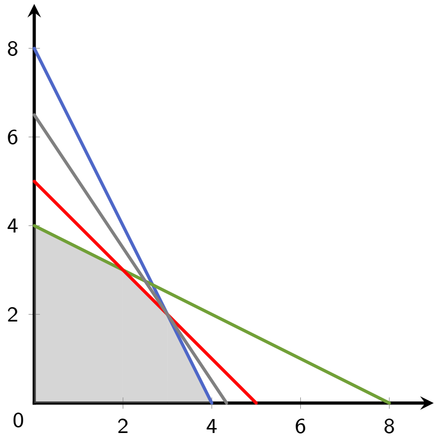
Solving Linear Program Graphically

$$\begin{aligned} \text{max.} \quad & 3x + 2y \\ \text{s.t.} \quad & 4x + 2y \le 16 \\ & x + 2y \le 8 \\ & x + y \le 5 \\ & x \ge 0 \\ & y \ge 0 \end{aligned}$$



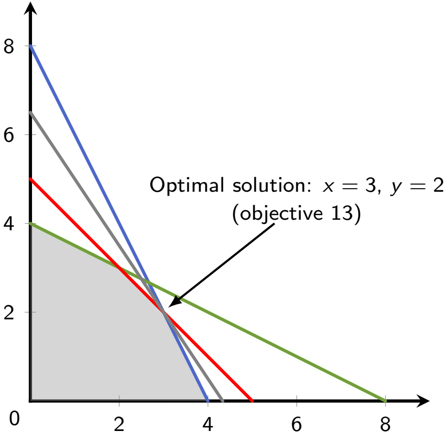
Solving Linear Program Graphically

$$
\begin{aligned}
\text{max.} \quad & 3x + 2y \\
\text{s.t.} \quad & 4x + 2y \le 16 \\
& x + 2y \le 8 \\
& x + y \le 5 \\
& x \ge 0 \\
& y \ge 0
\end{aligned}
$$



Solving Linear Program Graphically

$$
\begin{array}{ll}
\text{max.} & 3x + 2y \\
\text{s.t.} & 4x + 2y \leq 16 \\
 & x + 2y \leq 8 \\
 & x + y \leq 5 \\
 & x \geq 0 \\
 & y \geq 0
\end{array}
$$



Modified LP

$$\begin{aligned} \text{max.} \quad & 3x + 2y \\ \text{s.t.} \quad & 4x + 2y \leq 16 \\ & x + 2y \leq 8 \\ & x + y \leq 5 \\ & x \geq 0 \\ & y \geq 0 \end{aligned}$$

Modified LP

$$
\begin{aligned}
\text{max.}\quad & 3x + 2y \\
\text{s.t.}\quad & 4x + 2y \leq 15 \\
& x + 2y \leq 8 \\
& x + y \leq 5 \\
& x \geq 0 \\
& y \geq 0
\end{aligned}
$$

Modified LP

$$
\begin{aligned}
\text{max.} \quad & 3x + 2y \\
\text{s.t.} \quad & 4x + 2y \leq 15 \\
& x + 2y \leq 8 \\
& x + y \leq 5 \\
& x \geq 0 \\
& y \geq 0
\end{aligned}
$$

Modified LP

$$
\begin{aligned}
\text{max.} \quad & 3x + 2y \\
\text{s.t.} \quad & 4x + 2y \leq 15 \\
& x + 2y \leq 8 \\
& x + y \leq 5 \\
& x \geq 0 \\
& y \geq 0
\end{aligned}
$$

- Optimal solution: x = 2.5, y = 2.5

Modified LP

$$
\begin{aligned}
\text{max.} \quad & 3x + 2y \\
\text{s.t.} \quad & 4x + 2y \leq 15 \\
& x + 2y \leq 8 \\
& x + y \leq 5 \\
& x \geq 0 \\
& y \geq 0
\end{aligned}
$$

- Optimal solution: $x = 2.5$, $y = 2.5$
- Objective $= 7.5 + 5 = 12.5$

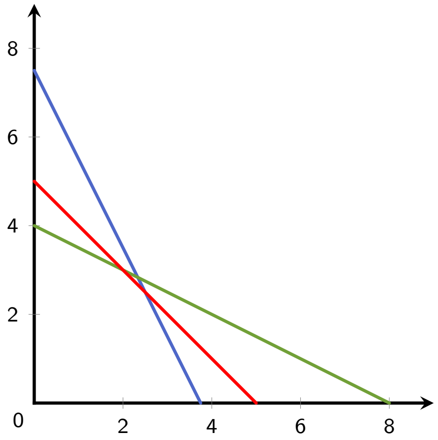
Modified LP

$$
\begin{aligned}
\text{max.} \quad & 3x + 2y \\
\text{s.t.} \quad & 4x + 2y \leq 15 \\
& x + 2y \leq 8 \\
& x + y \leq 5 \\
& x \geq 0 \\
& y \geq 0
\end{aligned}
$$

- Optimal solution: x = 2.5, y = 2.5
- Objective = 7.5 + 5 = 12.5
- Can we sell half paintings?

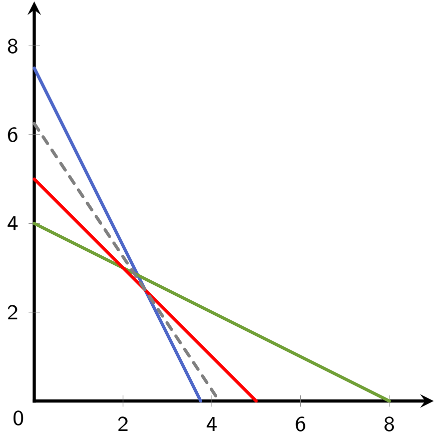
Integer Linear Program

$$\begin{aligned} \text{max.} \quad & 3x + 2y \\ \text{s.t.} \quad & 4x + 2y \leq 15 \\ & x + 2y \leq 8 \\ & x + y \leq 5 \\ & x \in \mathbb{N}_0 \\ & y \in \mathbb{N}_0 \end{aligned}$$



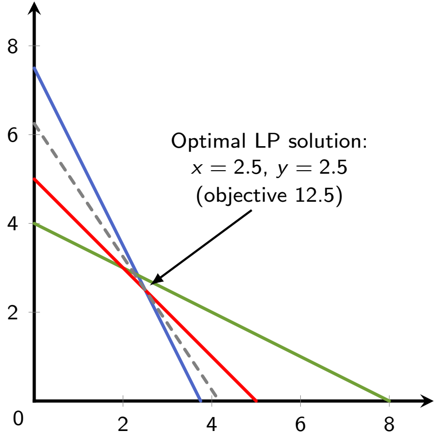
Integer Linear Program

$$
\begin{aligned}
\text{max.} \quad & 3x + 2y \\
\text{s.t.} \quad & 4x + 2y \leq 15 \\
& x + 2y \leq 8 \\
& x + y \leq 5 \\
& x \in \mathbb{N}_0 \\
& y \in \mathbb{N}_0
\end{aligned}
$$



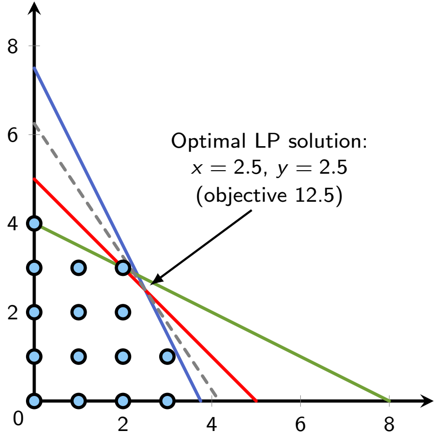
Integer Linear Program

$$\begin{aligned} \text{max.} \quad & 3x + 2y \\ \text{s.t.} \quad & 4x + 2y \leq 15 \\ & x + 2y \leq 8 \\ & x + y \leq 5 \\ & x \in \mathbb{N}_0 \\ & y \in \mathbb{N}_0 \end{aligned}$$



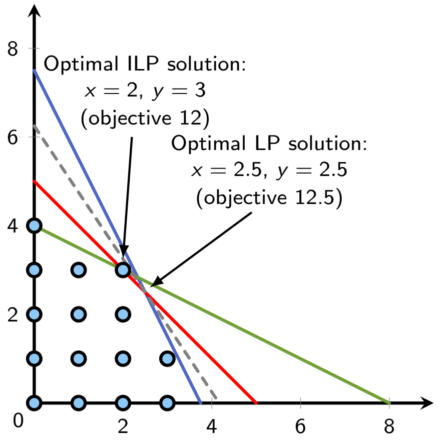
Integer Linear Program

$$
\begin{aligned}
\text{max.} \quad & 3x + 2y \\
\text{s.t.} \quad & 4x + 2y \leq 15 \\
& x + 2y \leq 8 \\
& x + y \leq 5 \\
& x \in \mathbb{N}_0 \\
& y \in \mathbb{N}_0
\end{aligned}
$$



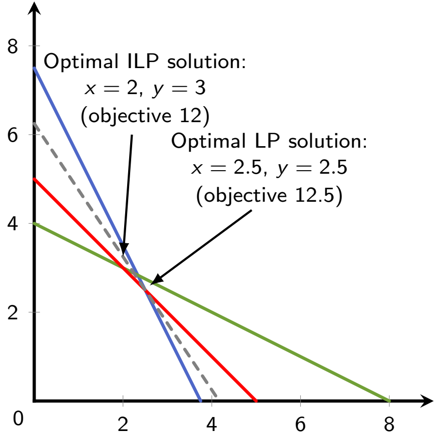
Integer Linear Program

$$
\begin{aligned}
\text{max.} \quad & 3x + 2y \\
\text{s.t.} \quad & {\color{blue} 4x + 2y \le 15} \\
& {\color{olive} x + 2y \le 8} \\
& {\color{red} x + y \le 5} \\
& x \in \mathbb{N}_0 \\
& y \in \mathbb{N}_0
\end{aligned}
$$



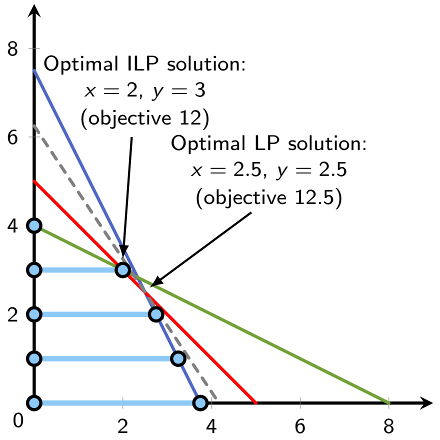
Mixed Integer Linear Program

$$\begin{aligned} \text{max.} \quad & 3x + 2y \\ \text{s.t.} \quad & 4x + 2y \le 15 \\ & x + 2y \le 8 \\ & x + y \le 5 \\ & x \ge 0 \\ & y \in \mathbb{N}_0 \end{aligned}$$



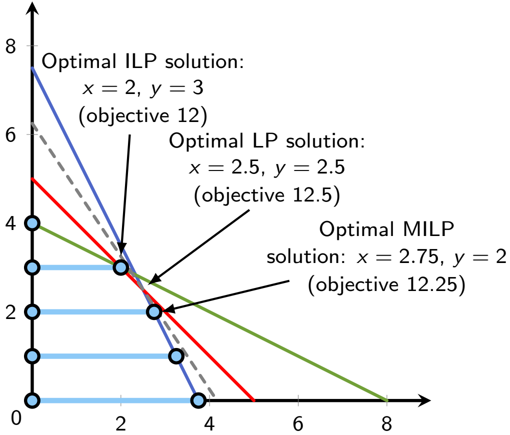
Mixed Integer Linear Program

$$\begin{aligned} \text{max.} \quad & 3x + 2y \\ \text{s.t.} \quad & 4x + 2y \leq 15 \\ & x + 2y \leq 8 \\ & x + y \leq 5 \\ & x \geq 0 \\ & y \in \mathbb{N}_0 \end{aligned}$$



Mixed Integer Linear Program

$$
\begin{aligned}
\text{max.} \quad & 3x + 2y \\
\text{s.t.} \quad & {\color{blue} 4x + 2y \le 15} \\
& {\color{olive} x + 2y \le 8} \\
& {\color{red} x + y \le 5} \\
& x \ge 0 \\
& y \in \mathbb{N}_0
\end{aligned}
$$



Solving Mixed Linear/Integer Programs

- Linear programs can be solved efficiently

Solving Mixed Linear/Integer Programs

- Linear programs can be solved efficiently
  - Simplex, ellipsoid, interior point methods, etc.

Solving Mixed Linear/Integer Programs

- Linear programs can be solved efficiently
  - Simplex, ellipsoid, interior point methods, etc.

- (Mixed) integer programs are NP-hard to solve

Solving Mixed Linear/Integer Programs

- Linear programs can be solved efficiently
  - Simplex, ellipsoid, interior point methods, etc.

- (Mixed) integer programs are NP-hard to solve
  - Many standard NP-complete problems can be modeled as MILP

Solving Mixed Linear/Integer Programs

- Linear programs can be solved efficiently
  - Simplex, ellipsoid, interior point methods, etc.
- (Mixed) integer programs are NP-hard to solve
  - Many standard NP-complete problems can be modeled as MILP
  - Search type algorithms such as branch and bound

Solving Mixed Linear/Integer Programs

- Linear programs can be solved efficiently
  - Simplex, ellipsoid, interior point methods, etc.

- (Mixed) integer programs are NP-hard to solve
  - Many standard NP-complete problems can be modeled as MILP
  - Search type algorithms such as branch and bound

- Standard packages for solving these

Solving Mixed Linear/Integer Programs

- Linear programs can be solved efficiently
  - Simplex, ellipsoid, interior point methods, etc.
- (Mixed) integer programs are NP-hard to solve
  - Many standard NP-complete problems can be modeled as MILP
  - Search type algorithms such as branch and bound
- Standard packages for solving these
  - Gurobi, MOSEK, GNU Linear Programming Kit, CPLEX, CVXOPT, etc.

Solving Mixed Linear/Integer Programs

- Linear programs can be solved efficiently
  - Simplex, ellipsoid, interior point methods, etc.
- (Mixed) integer programs are NP-hard to solve
  - Many standard NP-complete problems can be modeled as MILP
  - Search type algorithms such as branch and bound
- Standard packages for solving these
  - Gurobi, MOSEK, GNU Linear Programming Kit, CPLEX, CVXOPT, etc.
- LP relaxation of (M)ILP: remove integrality constraints

Solving Mixed Linear/Integer Programs

- Linear programs can be solved efficiently
  - Simplex, ellipsoid, interior point methods, etc.

- (Mixed) integer programs are NP-hard to solve
  - Many standard NP-complete problems can be modeled as MILP
  - Search type algorithms such as branch and bound

- Standard packages for solving these
  - Gurobi, MOSEK, GNU Linear Programming Kit, CPLEX, CVXOPT, etc.

- LP relaxation of (M)ILP: remove integrality constraints
  - Gives upper bound on MILP ($\sim$ admissible heuristic)

Exercise I: Knapsack-type Problem

- We arrive in room full of precious objects
- Can carry only 30kg out of the room
- Can carry only 20 liters out of the room
- Want to maximize our total value
- Unit of object A: 16kg, 3 liters, sells for \$11 (3 units available)
- Unit of object B: 4kg, 4 liters, sells for \$4 (4 units available)
- Unit of object C: 6kg, 3 liters, sells for \$9 (1 unit available)
- What should we take?

Exercise II: Cellphones (Set Cover)

- We want to have a working phone in every continent (besides Antarctica)
- But we want to have as few phones as possible
- Phone A works in NA, SA, Af
- Phone B works in E, Af, As
- Phone C works in NA, Au, E
- Phone D works in SA, As, E
- Phone E works in Af, As, Au
- Phone F works in NA, E

Exercise III: Hot-dog Stands

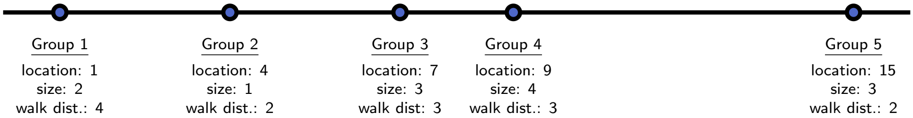
- We have two hot-dog stands to be placed in somewhere along beach
- We know where groups of people who like hot dogs are
- We also know how far each group is willing to walk
- Where do we put our stands to maximize # hot dogs sold? (price is fixed)

Exercise III: Hot-dog Stands

- We have two hot-dog stands to be placed in somewhere along beach
- We know where groups of people who like hot dogs are
- We also know how far each group is willing to walk
- Where do we put our stands to maximize # hot dogs sold? (price is fixed)

Exercise III: Hot-dog Stands

- We have two hot-dog stands to be placed in somewhere along beach
- We know where groups of people who like hot dogs are
- We also know how far each group is willing to walk
- Where do we put our stands to maximize # hot dogs sold? (price is fixed)



Outline

1. Brief Overview of (Mixed Integer) Linear Programming
2. Dominated Strategies
3. Minmax and Maxmin Strategies
4. Nash Equilibrium
5. Correlated NE

Recall: Strict Dominance

a_i strictly dominates s_i if $u_i(a_i, s_{-i}) > u_i(s_i, s_{-i})\ \forall s_{-i} \in S_{-i}$

Dominance by Pure Strategy

Algorithm 1: Determine whether s_i is strictly dominated by any pure strategy

```
for all a_i ∈ A_i where a_i ≠ s_i do
    dom ← true;
    forall a_{-i} ∈ A_{-i} do
        if u_i(s_i, a_{-i}) ≥ u_i(a_i, a_{-i}) then
            dom ← false;
            break;
    if dom = true then
        return true;
return false;
```

Dominance by Pure Strategy: Discussion

- Complexity of the Algorithm is $O(|A|)$, linear in the size of normal-form game

Dominance by Pure Strategy: Discussion

- Complexity of the Algorithm is $O(|A|)$, linear in the size of normal-form game
- Recall: a_i strictly dominates s_i if $u_i(a_i, s_{-i}) > u_i(s_i, s_{-i})\ \forall s_{-i} \in S_{-i}$
- This definition refers to mixed-strategy profile of other agents

Dominance by Pure Strategy: Discussion

- Complexity of the Algorithm is $O(|A|)$, linear in the size of normal-form game
- Recall: a_i strictly dominates s_i if $u_i(a_i, s_{-i}) > u_i(s_i, s_{-i})\ \forall s_{-i} \in S_{-i}$
- This definition refers to mixed-strategy profile of other agents
- In Alg. (1), we do not check every mixed-strategy profile of others, why?

Dominance by Pure Strategy: Discussion

- Complexity of the Algorithm is $O(|A|)$, linear in the size of normal-form game
- Recall: a_i strictly dominates s_i if $u_i(a_i, s_{-i}) > u_i(s_i, s_{-i})\ \forall s_{-i} \in S_{-i}$
- This definition refers to mixed-strategy profile of other agents
- In Alg. (1), we do not check every mixed-strategy profile of others, why?
  - Suppose a_i strictly dominates s_i for all a_{-i}

Dominance by Pure Strategy: Discussion

- Complexity of the Algorithm is $O(|A|)$, linear in the size of normal-form game
- Recall: a_i strictly dominates s_i if $u_i(a_i, s_{-i}) > u_i(s_i, s_{-i})\ \forall s_{-i} \in S_{-i}$
- This definition refers to mixed-strategy profile of other agents
- In Alg. (1), we do not check every mixed-strategy profile of others, why?
  - Suppose a_i strictly dominates s_i for all a_{-i}
  - Then, there is no s_{-i} for which $u_i(a_i, s_{-i}) \geq u_i(s_i, s_{-i})$

Dominance by Pure Strategy: Discussion

- Complexity of the Algorithm is $O(|A|)$, linear in the size of normal-form game
- Recall: a_i strictly dominates s_i if $u_i(a_i, s_{-i}) > u_i(s_i, s_{-i})\ \forall s_{-i} \in S_{-i}$
- This definition refers to mixed-strategy profile of other agents
- In Alg. (1), we do not check every mixed-strategy profile of others, why?
  - Suppose a_i strictly dominates s_i for all a_{-i}
  - Then, there is no s_{-i} for which $u_i(a_i, s_{-i}) \geq u_i(s_i, s_{-i})$
  - This holds because of the linearity of expectation

Weak Dominance by Mixed Strategy

- Checking if strategy s_i is weakly dominated by any mixed strategy

Weak Dominance by Mixed Strategy

- Checking if strategy s_i is weakly dominated by any mixed strategy

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{a_i \in A_i} p_{a_i} u_i(a_i, a_{-i}) \geq u_i(s_i, a_{-i}) && \forall a_{-i} \in A_{-i} \\
& \sum_{a_i \in A_i} p_{a_i} = 1 \\
& p_{a_i} \geq 0, && \forall a_i \in A_i
\end{aligned}
$$

Weak Dominance by Mixed Strategy

- Checking if strategy s_i is weakly dominated by any mixed strategy

$$
\begin{aligned}
\text{max.} \quad & \sum_{a_{-i} \in A_{-i}} \left[\left(\sum_{a_i \in A_i} p_{a_i} u_i(a_i, a_{-i}) \right) - u_i(s_i, a_{-i}) \right] \\
\text{s.t.} \quad & \sum_{a_i \in A_i} p_{a_i} u_i(a_i, a_{-i}) \geq u_i(s_i, a_{-i}) && \forall a_{-i} \in A_{-i} \\
& \sum_{a_i \in A_i} p_{a_i} = 1 \\
& p_{a_i} \geq 0, && \forall a_i \in A_i
\end{aligned}
$$

Weak Dominance by Mixed Strategy

- Checking if strategy s_i is weakly dominated by any mixed strategy

$$
\begin{aligned}
\text{max.} \quad & \sum_{a_{-i} \in A_{-i}} \left[\left(\sum_{a_i \in A_i} p_{a_i} u_i(a_i, a_{-i}) \right) - u_i(s_i, a_{-i}) \right] & \\
\text{s.t.} \quad & \sum_{a_i \in A_i} p_{a_i} u_i(a_i, a_{-i}) \geq u_i(s_i, a_{-i}) & \forall a_{-i} \in A_{-i} \\
& \sum_{a_i \in A_i} p_{a_i} = 1 & \\
& p_{a_i} \geq 0, & \forall a_i \in A_i
\end{aligned}
$$

- If optimal solution is strictly positive, then s_i is weakly dominated by $\{p_{a_i}\}$

Strict Dominance by Mixed Strategies

- Checking if strategy s_i is strictly dominated by any mixed strategy

Strict Dominance by Mixed Strategies

- Checking if strategy s_i is strictly dominated by any mixed strategy

$$\text{s.t.} \qquad \sum_{a_i \in A_i} p_{a_i} u_i(a_i, a_{-i}) \geq u_i(s_i, a_{-i}) + \epsilon \quad \forall a_{-i} \in A_{-i}$$

Strict Dominance by Mixed Strategies

- Checking if strategy s_i is strictly dominated by any mixed strategy

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{a_i \in A_i} p_{a_i} u_i(a_i, a_{-i}) \geq u_i(s_i, a_{-i}) + \epsilon \quad && \forall a_{-i} \in A_{-i} \\
& \sum_{a_i \in A_i} p_{a_i} = 1 \\
& p_{a_i} \geq 0, && \forall a_i \in A_i
\end{aligned}
$$

Strict Dominance by Mixed Strategies

- Checking if strategy s_i is strictly dominated by any mixed strategy

$$
\begin{aligned}
\text{max.} \quad & \epsilon \\
\text{s.t.} \quad & \sum_{a_i \in A_i} p_{a_i} u_i(a_i, a_{-i}) \geq u_i(s_i, a_{-i}) + \epsilon \quad && \forall a_{-i} \in A_{-i} \\
& \sum_{a_i \in A_i} p_{a_i} = 1 \\
& p_{a_i} \geq 0, && \forall a_i \in A_i
\end{aligned}
$$

Strict Dominance by Mixed Strategies

- Checking if strategy s_i is strictly dominated by any mixed strategy

$$
\begin{aligned}
\text{max.} \quad & \epsilon \\
\text{s.t.} \quad & \sum_{a_i \in A_i} p_{a_i} u_i(a_i, a_{-i}) \geq u_i(s_i, a_{-i}) + \epsilon \quad \forall a_{-i} \in A_{-i} \\
& \sum_{a_i \in A_i} p_{a_i} = 1 \\
& p_{a_i} \geq 0, \qquad \forall a_i \in A_i
\end{aligned}
$$

- If optimal solution is strictly positive, then s_i is strictly dominated by $\{p_{a_i}\}$

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

$0, 1$	$1, 0$
$1, 0$	$1, 0$
$1, 0$	$0, 1$

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

$0, 1$	$1, 0$
$1, 0$	$1, 0$
$1, 0$	$0, 1$

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

~~0, 1~~	~~1, 0~~
1, 0	1, 0
1, 0	0, 1

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

$\cancel{0, 1}$	$\cancel{1, 0}$
$1, 0$	$1, 0$
$1, 0$	$0, 1$

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

$\cancel{0,1}$	$\cancel{1,0}$
$\cancel{1,0}$	$1,0$
$\cancel{1,0}$	$0,1$

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

~~$0, 1$~~	~~$1, 0$~~
~~$1, 0$~~	$1, 0$
~~$1, 0$~~	$0, 1$

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

~~$0, 1$~~	~~$1, 0$~~
~~$1, 0$~~	$1, 0$
~~$1, 0$~~	~~$0, 1$~~

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

~~0, 1~~	~~1, 0~~
~~1, 0~~	1, 0
~~1, 0~~	~~0, 1~~

0, 1	1, 0
1, 0	1, 0
1, 0	0, 1

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

~~$0, 1$~~	~~$1, 0$~~
~~$1, 0$~~	$1, 0$
~~$1, 0$~~	~~$0, 1$~~

$0, 1$	$1, 0$
$1, 0$	$1, 0$
$1, 0$	$0, 1$

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

~~$0, 1$~~	~~$1, 0$~~
~~$1, 0$~~	$1, 0$
~~$1, 0$~~	~~$0, 1$~~

$0, 1$	$1, 0$
$1, 0$	$1, 0$
~~$1, 0$~~	~~$0, 1$~~

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

$\cancel{0, 1}$	$\cancel{1, 0}$
$\cancel{1, 0}$	$1, 0$
$\cancel{1, 0}$	$\cancel{0, 1}$

$0, 1$	$1, 0$
$1, 0$	$1, 0$
$\cancel{1, 0}$	$\cancel{0, 1}$

Path Dependency of Iterated Dominance

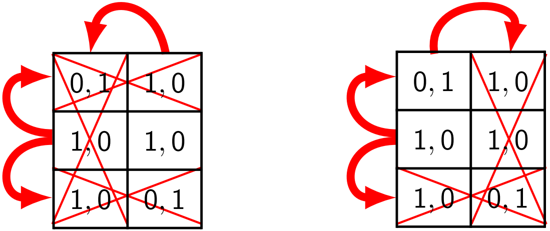
- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

~~0, 1~~	~~1, 0~~
~~1, 0~~	1, 0
~~1, 0~~	~~0, 1~~

0, 1	~~1, 0~~
1, 0	~~1, 0~~
~~1, 0~~	~~0, 1~~

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)



Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

~~$0,1$~~	~~$1,0$~~
~~$1,0$~~	$1,0$
~~$1,0$~~	~~$0,1$~~

~~$0,1$~~	~~$1,0$~~
$1,0$	~~$1,0$~~
~~$1,0$~~	~~$0,1$~~

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

$\cancel{0,1}$	$\cancel{1,0}$
$\cancel{1,0}$	$1,0$
$\cancel{1,0}$	$\cancel{0,1}$

$\cancel{0,1}$	$\cancel{1,0}$
$1,0$	$\cancel{1,0}$
$\cancel{1,0}$	$\cancel{0,1}$

$0,1$	$1,0$
$1,0$	$1,0$
$1,0$	$0,1$

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

~~0, 1~~	~~1, 0~~
~~1, 0~~	1, 0
~~1, 0~~	~~0, 1~~

~~0, 1~~	~~1, 0~~
1, 0	~~1, 0~~
~~1, 0~~	~~0, 1~~

0, 1	1, 0
1, 0	1, 0
1, 0	0, 1

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)

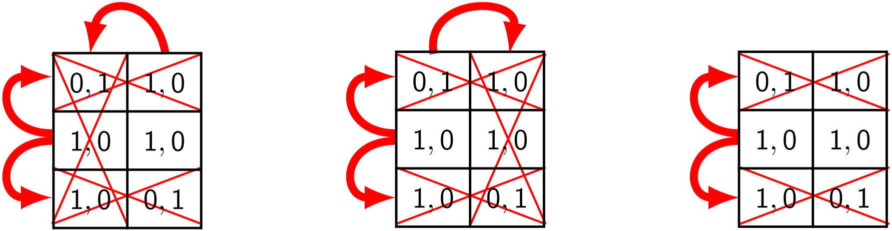
~~0, 1~~	~~1, 0~~
~~1, 0~~	1, 0
~~1, 0~~	~~0, 1~~

~~0, 1~~	~~1, 0~~
1, 0	~~1, 0~~
~~1, 0~~	~~0, 1~~

~~0, 1~~	~~1, 0~~
1, 0	1, 0
~~1, 0~~	~~0, 1~~

Path Dependency of Iterated Dominance

- Iterated weak dominance is path-dependent:
  - Sequence of eliminations may determine which solution we get (if any)



- Iterated strict dominance is path-independent:
  - Elimination process will always terminate at the same point

Computational Questions for Iterated Dominance

- Is there some elimination path under which s_i is eliminated?

Computational Questions for Iterated Dominance

- Is there some elimination path under which s_i is eliminated?
- Is there maximally reduced game where each agent has exactly 1 action?

Computational Questions for Iterated Dominance

- Is there some elimination path under which s_i is eliminated?

- Is there maximally reduced game where each agent has exactly 1 action?

- For strict dominance, both can be solved in polynomial time
  - Due to path-independence
  - Check if any strategy is dominated, remove it, repeat
  - With or without dominance by mixed strategies

Computational Questions for Iterated Dominance

- Is there some elimination path under which s_i is eliminated?
- Is there maximally reduced game where each agent has exactly 1 action?
- For strict dominance, both can be solved in polynomial time
  - Due to path-independence
  - Check if any strategy is dominated, remove it, repeat
  - With or without dominance by mixed strategies
- For weak dominance, both questions are NP-hard[1]
  - Even when all utilities are 0 or 1
  - With or without dominance by mixed strategies

[1] [Conitzer, Sandholm 05] and weaker version proved by [Gilboa, Kalai, Zemel 93]

Outline

1. Brief Overview of (Mixed Integer) Linear Programming
2. Dominated Strategies
3. Minmax and Maxmin Strategies
4. Nash Equilibrium
5. Correlated NE

Recall: Minmax and Maxmin

- Maxmin strategy for agent i (maxmin value for agent i)

$$\underset{s_i}{\operatorname{argmax}} \min_{s_{-i}} u_i(s_i, s_{-i})$$

Recall: Minmax and Maxmin

- Maxmin strategy for agent i (maxmin value for agent i)

$$\arg\max_{s_i} \min_{s_{-i}} u_i(s_i, s_{-i})$$

- Minmax strategy against agent i (minmax value for agent i)

$$\arg\min_{s_{-i}} \max_{s_i} u_i(s_i, s_{-i})$$

Maxmin Strategy and Value

- Finding maxmin strategy of agent i

Maxmin Strategy and Value

- Finding maxmin strategy of agent i

$$\text{s.t.} \qquad \sum_{a_i \in A_i} p_{a_i} u_i(a_i, a_{-i}) \geq U_i, \quad \forall a_{-i} \in A_{-i}$$

- Given p_{a_i}, first constraint ensures that U_i is less than any achievable expected utility for any pure strategies of opponents

Maxmin Strategy and Value

- Finding maxmin strategy of agent i

$$
\begin{aligned}
\text{s.t.} \qquad & \sum_{a_i \in A_i} p_{a_i} u_i(a_i, a_{-i}) \geq U_i, \quad && \forall a_{-i} \in A_{-i} \\
& \sum_{a_i \in A_i} p_{a_i} = 1 \\
& p_{a_i} \geq 0, && \forall a_i \in A_i
\end{aligned}
$$

- Given p_{a_i}, first constraint ensures that U_i is less than any achievable expected utility for any pure strategies of opponents

Maxmin Strategy and Value

- Finding maxmin strategy of agent i

$$\begin{aligned} \text{max.} \quad & U_i \\ \text{s.t.} \quad & \sum_{a_i \in A_i} p_{a_i} u_i(a_i, a_{-i}) \geq U_i, \quad \forall a_{-i} \in A_{-i} \\ & \sum_{a_i \in A_i} p_{a_i} = 1 \\ & p_{a_i} \geq 0, \qquad \forall a_i \in A_i \end{aligned}$$

- Given p_{a_i}, first constraint ensures that U_i is less than any achievable expected utility for any pure strategies of opponents
- Objective of this LP, U_i, is maxmin value of agent i

Outline

1. Brief Overview of (Mixed Integer) Linear Programming
2. Dominated Strategies
3. Minmax and Maxmin Strategies
4. Nash Equilibrium
5. Correlated NE

NE of Two-player, Zero-sum Games

- Maxmin value for agent 1

NE of Two-player, Zero-sum Games

- Maxmin value for agent 1

$$
\begin{aligned}
\text{max.} \quad & U_1 \\
\text{s.t.} \quad & \sum_{a_1 \in A_1} p_{a_1} u_1(a_1, a_2) \geq U_1 \quad && \forall a_2 \in A_2 \\
& \sum_{a_1 \in A_1} p_{a_1} = 1 \\
& p_{a_1} \geq 0, && \forall a_1 \in A_1
\end{aligned}
$$

NE of Two-player, Zero-sum Games

- Maxmin value for agent 1

$$
\begin{aligned}
\text{max.} \quad & U_1 \\
\text{s.t.} \quad & \sum_{a_1 \in A_1} p_{a_1} u_1(a_1, a_2) \geq U_1 && \forall a_2 \in A_2 \\
& \sum_{a_1 \in A_1} p_{a_1} = 1 \\
& p_{a_1} \geq 0, && \forall a_1 \in A_1
\end{aligned}
$$

- Minmax value for agent 1

NE of Two-player, Zero-sum Games

- Maxmin value for agent 1

$$
\begin{aligned}
\text{max.} \quad & U_1 \\
\text{s.t.} \quad & \sum_{a_1 \in A_1} p_{a_1} u_1(a_1, a_2) \geq U_1 \quad && \forall a_2 \in A_2 \\
& \sum_{a_1 \in A_1} p_{a_1} = 1 \\
& p_{a_1} \geq 0, && \forall a_1 \in A_1
\end{aligned}
$$

- Minmax value for agent 1

$$
\begin{aligned}
\text{min.} \quad & U_1 \\
\text{s.t.} \quad & \sum_{a_2 \in A_2} p_{a_2} u_1(a_1, a_2) \leq U_1 \quad && \forall a_1 \in A_1 \\
& \sum_{a_2 \in A_2} p_{a_2} = 1 \\
& p_{a_2} \geq 0, && \forall a_2 \in A_2
\end{aligned}
$$

NE of Two-player, Zero-sum Games

- Maxmin value for agent 1

$$\begin{aligned} \text{max.} \quad & U_1 \\ \text{s.t.} \quad & \sum_{a_1 \in A_1} p_{a_1} u_1(a_1, a_2) \geq U_1 \quad && \forall a_2 \in A_2 \\ & \sum_{a_1 \in A_1} p_{a_1} = 1 \\ & p_{a_1} \geq 0, && \forall a_1 \in A_1 \end{aligned}$$

- Minmax value for agent 1

$$\begin{aligned} \text{min.} \quad & U_1 \\ \text{s.t.} \quad & \sum_{a_2 \in A_2} p_{a_2} u_1(a_1, a_2) \leq U_1 \quad && \forall a_1 \in A_1 \\ & \sum_{a_2 \in A_2} p_{a_2} = 1 \\ & p_{a_2} \geq 0, && \forall a_2 \in A_2 \end{aligned}$$

- NE is expressed as LP $\Rightarrow$ NE can be computed in polynomial time

Maxmin Strategy for General-sum Games

- Agents could still play minmax strategy in general-sum games

Maxmin Strategy for General-sum Games

- Agents could still play minmax strategy in general-sum games
  - I.e., pretend that the opponent is only trying to hurt you

Maxmin Strategy for General-sum Games

- Agents could still play minmax strategy in general-sum games
  - I.e., pretend that the opponent is only trying to hurt you
- But this might not be rational:

Maxmin Strategy for General-sum Games

- Agents could still play minmax strategy in general-sum games
  - I.e., pretend that the opponent is only trying to hurt you
- But this might not be rational:

		Agent 2: Left	Agent 2: Right
Agent 1	Up	$0, 0$	$3, 1$
	Down	$1, 0$	$2, 1$

Maxmin Strategy for General-sum Games

- Agents could still play minmax strategy in general-sum games
  - I.e., pretend that the opponent is only trying to hurt you
- But this might not be rational:

Agent 2

Agent 1		Left	Right
	Up	$0, 0$	$3, 1$
	Down	$1, 0$	$2, 1$

  - If A2 was trying to hurt A1, she would play Left, so A1 should play Down

Maxmin Strategy for General-sum Games

- Agents could still play minmax strategy in general-sum games
  - I.e., pretend that the opponent is only trying to hurt you
- But this might not be rational:

		Agent 2	
		Left	Right
Agent 1	Up	$0, 0$	$3, 1$
	Down	$1, 0$	$2, 1$

  - If A2 was trying to hurt A1, she would play Left, so A1 should play Down
  - In reality, A2 will play Right (strictly dominant), so A1 should play Up

Hardness of Computing NE for General-sum Games

- Complexity was open for long time
  - "together with factoring [. . .] the most important concrete open question on the boundary of P today" [Papadimitriou STOC'01]

Hardness of Computing NE for General-sum Games

- Complexity was open for long time
  - "together with factoring [. . .] the most important concrete open question on the boundary of P today" [Papadimitriou STOC'01]
- Sequence of papers showed that computing any NE is PPAD-complete (even in 2-player games) [Daskalakis, Goldberg, Papadimitriou 2006; Chen, Deng 2006]

Hardness of Computing NE for General-sum Games

- Complexity was open for long time
  - "together with factoring [...] the most important concrete open question on the boundary of P today" [Papadimitriou STOC'01]
- Sequence of papers showed that computing any NE is PPAD-complete (even in 2-player games) [Daskalakis, Goldberg, Papadimitriou 2006; Chen, Deng 2006]
- All known algorithms require exponential time (in worst case)

Hardness of Computing NE for General-Sum Games (cont.)

- What about computing NE with specific property?
  - NE that is not Pareto-dominated
  - NE that maximizes expected social welfare (i.e., sum of all agents' utilities)
  - NE that maximizes expected utility of given agent
  - NE that maximizes expected utility of worst-off player
  - NE in which given pure strategy is played with positive probability
  - NE in which given pure strategy is played with zero probability
  - ...

Hardness of Computing NE for General-Sum Games (cont.)

- What about computing NE with specific property?
  - NE that is not Pareto-dominated
  - NE that maximizes expected social welfare (i.e., sum of all agents' utilities)
  - NE that maximizes expected utility of given agent
  - NE that maximizes expected utility of worst-off player
  - NE in which given pure strategy is played with positive probability
  - NE in which given pure strategy is played with zero probability
  - ...
- All of these are NP-hard (and the optimization questions are inapproximable assuming P != NP), even in 2-player games

  [Gilboa, Zemel 89; Conitzer & Sandholm IJCAI-03/GEB-08]

Search-based Approaches (for Two-player Games)

- We can use LP, if we know support X_i of each player i's mixed strategy

Search-based Approaches (for Two-player Games)

- We can use LP, if we know support X_i of each player i's mixed strategy

$$
\begin{aligned}
\text{find} \quad & (U_1, U_2) \\
\text{s.t.} \quad & p_{a_i} \geq 0, && \forall i, a_i \in A_i \\
& \sum_{a_i \in A_i} p_{a_i} = 1, && \forall i \\
& p_{a_i} = 0, && \forall i, a_i \in A_i / X_i \\
& \sum_{a_{-i} \in A_{-i}} p_{a_{-i}} u_i(a_i, a_{-i}) = U_i, && \forall i, a_i \in X_i \\
& \sum_{a_{-i} \in A_{-i}} p_{a_{-i}} u_i(a_i, a_{-i}) \leq U_i, && \forall i, a_i \in A_i / X_i
\end{aligned}
$$

Search-based Approaches (for Two-player Games)

- We can use LP, if we know support X_i of each player i's mixed strategy

$$
\begin{aligned}
\text{find} \quad & (U_1, U_2) & \\
\text{s.t.} \quad & p_{a_i} \geq 0, & \forall i, a_i \in A_i \\
& \sum_{a_i \in A_i} p_{a_i} = 1, & \forall i \\
& p_{a_i} = 0, & \forall i, a_i \in A_i / X_i \\
& \sum_{a_{-i} \in A_{-i}} p_{a_{-i}} u_i(a_i, a_{-i}) = U_i, & \forall i, a_i \in X_i \\
& \sum_{a_{-i} \in A_{-i}} p_{a_{-i}} u_i(a_i, a_{-i}) \leq U_i, & \forall i, a_i \in A_i / X_i
\end{aligned}
$$

- Thus, we can search over possible supports, which is basic idea underlying methods in [Dickhaut & Kaplan 91; Porter, Nudelman, Shoham AAAI04/GEB08]

NE using MILP (for Two-player Games)

[Sandholm, Gilpin, Conitzer AAAI05]

$$
\begin{aligned}
\text{max.} \quad & \text{whatever you like (e.g., social welfare)} & \\
\text{s.t.} \quad & p_{a_i} \geq 0, & \forall i, a_i \in A_i \\
& \sum_{a_i \in A_i} p_{a_i} = 1, & \forall i \\
& \sum_{a_{-i} \in A_{-i}} p_{a_{-i}} u_i(a_i, a_{-i}) = u_{a_i}, & \forall i, a_i \in A_i \\
& u_{a_i} \leq u_i, & \forall i, a_i \in A_i \\
& p_{a_i} \leq b_{a_i}, & \forall i, a_i \in A_i \\
& u_i - u_{a_i} \leq M(1 - b_{a_i}), & \forall i, a_i \in A_i \\
& b_{a_i} \in \{0, 1\}, & \forall i, a_i \in A_i
\end{aligned}
$$

NE using MILP (for Two-player Games)

[Sandholm, Gilpin, Conitzer AAAI05]

$$
\begin{aligned}
\text{max.} \quad & \text{whatever you like (e.g., social welfare)} & \\
\text{s.t.} \quad & p_{a_i} \geq 0, & \forall i, a_i \in A_i \\
& \sum_{a_i \in A_i} p_{a_i} = 1, & \forall i \\
& \sum_{a_{-i} \in A_{-i}} p_{a_{-i}} u_i(a_i, a_{-i}) = u_{a_i}, & \forall i, a_i \in A_i \\
& u_{a_i} \leq u_i, & \forall i, a_i \in A_i \\
& p_{a_i} \leq b_{a_i}, & \forall i, a_i \in A_i \\
& u_i - u_{a_i} \leq M(1 - b_{a_i}), & \forall i, a_i \in A_i \\
& b_{a_i} \in \{0, 1\}, & \forall i, a_i \in A_i
\end{aligned}
$$

- b_{a_i} indicates whether a_i is in support of i's mixed strategy, and M is large number

Outline

1. Brief Overview of (Mixed Integer) Linear Programming
2. Dominated Strategies
3. Minmax and Maxmin Strategies
4. Nash Equilibrium
5. Correlated NE

Correlated Equilibrium (N-player Games!)

- Variables are now p_a for all action profiles a (i.e., outcome)

Correlated Equilibrium (N-player Games!)

- Variables are now p_a for all action profiles a (i.e., outcome)

$$
\begin{aligned}
\text{max.} \quad & \text{whatever you like (e.g., social welfare)} \\
\text{s.t.} \quad & \sum_{a_{-i} \in A_{-i}} p_a u_i(a) \geq \sum_{a_{-i} \in A_{-i}} p_a u_i(t_i, a_{-i}) \quad \forall i, a_i, t_i \in A_i \\
& \sum_{a \in A} p_a = 1 \\
& p_a \geq 0, \qquad \forall a \in A
\end{aligned}
$$

Acknowledgment

- This lecture is a slightly modified version of ones prepared by
  - Vincent Conitzer [Duke CPS 590.4]
- Xiaoliang Zhou helped with importing slides from PowerPoint to LaTeX