Computing Solution Concepts of Normal-Form Games

NE in n-player, general-sum games

- Can no longer be represented even as an LCP
  - Impractical to solve exactly
- Unclear how to best formulate the problem as input to an algorithm

NE in n-player, general-sum games

- There has been some success approximating the solution using a sequence of linear complementarity problems (SLCP)
  - Each LCP is an approximation of the problem, and its solution is used to create the next approximation in the sequence
  - Can be thought of as a generalization to Newton's method of approximating the local maximum of a quadratic equation
  - Although not globally convergent, it is possible to try a number of different starting points because of its relative speed

NE in n-player, general-sum games

- Another approach is to formulate the problem as a minimum of a function
  - Starting from a strategy profile $\boldsymbol{s}$, let $c_i^j(s)$ be the change in utility to player $\boldsymbol{i}$ if he switches to playing action a_i^j as a pure strategy
  - Then, define $d_i^j(s)$ as $c_i^j(s)$ bounded from below by zero

$$c_i^j(s) = u_i(a_i^j, s_{-i}) - u_i(s)$$
$$d_i^j(s) = \max(c_i^j(s), 0)$$

NE in n-player, general-sum games

$$c_i^j(s) = u_i(a_i^j, s_{-i}) - u_i(s)$$
$$d_i^j(s) = \max(c_i^j(s), 0)$$

- Note that $d_i^j(s)$ is positive if and only if player i has an incentive to deviate to action $a_i^j(s)$
- Thus, strategy profile s is a Nash equilibrium if and only if $d_i^j(s) = 0$ for all players i, and all actions j for each player

NE in n-player, general-sum games

$$\text{minimize} \quad f(s) = \sum_{i \in N} \sum_{j \in A_i} \left(d_i^j(s)\right)^2$$

makes the function differentiable everywhere

$$\text{subject to} \quad \sum_{j \in A_i} s_i^j = 1 \qquad \forall i \in N$$

$$s_i^j \geq 0 \qquad \forall i \in N, \forall j \in A_i$$

- This function has one or more global minima at ***0***, and the set of all ***s*** such that ***f(s) = 0*** is exactly the set of Nash equilibria
- We can now apply any method for constrained optimization

NE in n-player, general-sum games

- For an unconstrained optimization method, we can roll the constraints into the objective function

$$\text{minimize} \quad \sum_{i \in N} \sum_{j \in A_i} \left(d_i^j(s)\right)^2 + \sum_{i \in N} \left(1 - \sum_{j \in A_i} s_i^j\right)^2 + \sum_{i \in N} \sum_{j \in A_i} \left(\min(s_i^j, 0)\right)^2$$

- We still have a differentiable function that is zero if and only if ***s*** is a Nash equilibrium

NE in n-player, general-sum games

- :(Both optimization problems have local minima which do not correspond to Nash equilibria
- Considering the commonly-used optimization methods
  - :(**hill-climbing** gets stuck in local minima
  - :(**simulated annealing** often converges globally only for parameter settings that yield an impractically long running time

NE in n-player, general-sum games

- Alternative algorithms
  - Scarf's algorithm
  - Govindan and Wilson
  - SEM

Identifying dominated strategies

- Computational uses for identifying dominated strategies
- Computational complexity of this process

Identifying dominated strategies

- Iterated removal of **strictly** dominated strategies
  - The same set of strategies will be identified regardless of the elimination order
  - All Nash equilibria of the original game will be contained in this set
  - Can be used to narrow down the set of strategies to consider before attempting to identify a sample Nash equilibrium
  - In the worst case this procedure will have no effect
  - In practice, it can make a big difference

Identifying dominated strategies

- Iterated removal of **weakly** dominated strategies
  - The elimination order does make a difference
    - the set of strategies that survive iterated removal can differ depending on the order in which dominated strategies are removed
    - Removing weakly dominated strategies can eliminate some equilibria of the original game
  - No new equilibria are ever created by this elimination
  - Since every game has at least one equilibrium, at least one of the original equilibria always survives
  - Will often produce a smaller game

Identifying dominated strategies

- Algorithm for determining whether s_i is **strictly** dominated by any pure strategy

forall *pure strategies* $a_i \in A_i$ *for player* i *where* $a_i \neq s_i$ **do**
  $dom \leftarrow true$
  forall *pure-strategy profiles* $a_{-i} \in A_{-i}$ *for the players other than* i **do**
    if $u_i(s_i, a_{-i}) \geq u_i(a_i, a_{-i})$ **then**
      $dom \leftarrow false$
      break
  if $dom = true$ **then**
    return $true$
return $false$

Identifying dominated strategies

- How to change this algorithm to determine whether $\boldsymbol{s_i}$ is **weakly** dominated by any pure strategy?

forall *pure strategies* $a_i \in A_i$ *for player* i *where* $a_i \neq s_i$ **do**
  $dom \leftarrow true$, *flag* = **false**
  forall *pure-strategy profiles* $a_{-i} \in A_{-i}$ *for the players other than* i **do**
    if $u_i(s_i, a_{-i})$ $\boxed{>}$ $u_i(a_i, a_{-i})$ **then**
      $dom \leftarrow false$
      break
    else if $u_i(s_i,a_{-i}) < u_i(a_i,a_{-i})$ **then**
      $flag \leftarrow true$
  if $dom = true$ **and** *flag* = **true then**
    return $true$
return $false$

Identifying dominated strategies

- For all of the definitions of domination, the complexity of the procedure is $O(|A|)$, linear in the size of the normal-form game

Domination by a mixed strategy

- We cannot use the previous simple algorithm to test whether a given strategy s_i is dominated by a mixed strategy because these strategies **cannot** be enumerated
- However, it turns out that we can still answer the question in polynomial time by solving a **linear program**
  - Each flavor of domination requires a somewhat different linear program

Another solution concept

- Reminder:

	L	***C***
U	3,1	0,3
M	1,5	1,1
B	0,½	4,2

Why is the action dominated?

Domination by a mixed strategy

- **Q:** What does this feasibility program give us for a given strategy ***s**$_i$*?

$$\sum_{j \in A_i} p_j u_i(a_j, a_{-i}) \geq u_i(s_i, a_{-i}) \qquad \forall a_{-i} \in A_{-i}$$

$$p_j \geq 0 \qquad \forall j \in A_i$$

$$\sum_{j \in A_i} p_j = 1$$

- **A:** Very weakly dominance!

Domination by a mixed strategy

- Matlab code:

```
A = [-3 0; 0 -4]
b = [-1 -1];
Aeq = [1 1];
beq = 1;
lb = [0 0];
f = [];
z = linprog(f,A,b,Aeq,beq,lb)
z = 0.3349
    0.6651
```

$$\sum_{j \in A_i} p_j u_i(a_j, a_{-i}) \geq u_i(s_i, a_{-i}) \qquad \forall a_{-i} \in A_{-i}$$

$$p_j \geq 0 \qquad \forall j \in A_i$$

$$\sum_{j \in A_i} p_j = 1$$

	L	C
U	3,1	0,3
M	1,5	1,1
B	0,½	4,2

Domination by a mixed strategy

- Then, let us consider strict domination by a mixed strategy
- This would seem to have the following straightforward LP formulation (indeed, a mere feasibility program)

$$\sum_{j \in A_i} p_j u_i(a_j, a_{-i}) > u_i(s_i, a_{-i}) \qquad \forall a_{-i} \in A_{-i}$$

$$p_j \geq 0 \qquad \forall j \in A_i$$

$$\sum_{j \in A_i} p_j = 1$$

Domination by a mixed strategy

- The problem is that the constraints in linear programs must be weak inequalities

$$\sum_{j \in A_i} p_j u_i(a_j, a_{-i}) > u_i(s_i, a_{-i}) \qquad \forall a_{-i} \in A_{-i}$$

$$p_j \geq 0 \qquad \forall j \in A_i$$

$$\sum_{j \in A_i} p_j = 1$$

cannot be written in LPs

Domination by a mixed strategy

- Instead, we must use the LP that follows

$$
\begin{aligned}
&\text{minimize} && \sum_{j \in A_i} p_j \\
&\text{subject to} && \sum_{j \in A_i} p_j u_i(a_j, a_{-i}) \geq u_i(s_i, a_{-i}) && \forall a_{-i} \in A_{-i} \\
& && p_j \geq 0 && \forall j \in A_i
\end{aligned}
$$

Domination by a mixed strategy

- This LP simulates the strict inequality of constraint through the objective function

$$
\begin{aligned}
\text{minimize} \quad & \sum_{j \in A_i} p_j \\
\text{subject to} \quad & \sum_{j \in A_i} p_j u_i(a_j, a_{-i}) \geq u_i(s_i, a_{-i}) && \forall a_{-i} \in A_{-i} \\
& p_j \geq 0 && \forall j \in A_i
\end{aligned}
$$

no constraints restrict the ***p_j***'s from above: this LP will always be feasible

In the optimal solution, the ***p_j***'s may be smaller or greater to ***1***

Domination by a mixed strategy

- A strictly dominating mixed strategy therefore exists **iff** the optimal solution to the LP has objective function ***< 1***
- In this case, we can add a positive amount to each $\boldsymbol{p_j}$ in order to cause constraint to hold in its strict version everywhere while achieving the condition $\boldsymbol{\Sigma_j p_j = 1}$

$$
\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{j \in A_i} p_j & \\
\text{subject to} & \displaystyle\sum_{j \in A_i} p_j u_i(a_j, a_{-i}) \geq u_i(s_i, a_{-i}) & \forall a_{-i} \in A_{-i} \\
 & p_j \geq 0 & \forall j \in A_i
\end{array}
$$

Domination by a mixed strategy

- Matlab code:

```
A = [-3 0; 0 -4]
b = [-1 -1];
Aeq = [];
beq = [];
lb = [0 0];
f = [1 1];
z = linprog(f,A,b,Aeq,beq,lb)
z =
   0.3333
   0.2500
```

$$
\begin{aligned}
&\text{minimize} && \sum_{j \in A_i} p_j \\
&\text{subject to} && \sum_{j \in A_i} p_j u_i(a_j, a_{-i}) \geq u_i(s_i, a_{-i}) && \forall a_{-i} \in A_{-i} \\
& && p_j \geq 0 && \forall j \in A_i
\end{aligned}
$$

	L	***C***
U	3,1	0,3
M	1,5	1,1
B	0,½	4,2

Domination by a mixed strategy

- How to identify if strategy s_i is weakly dominated by a mixed strategy?

$$\text{maximize} \quad \boxed{\sum_{a_{-i} \in A_{-i}} \left[\left(\sum_{j \in A_i} p_j \cdot u_i(a_j, a_{-i}) \right) - u_i(s_i, a_{-i}) \right]}$$

$$\text{subject to} \quad \sum_{j \in A_i} p_j u_i(a_j, a_{-i}) \geq u_i(s_i, a_{-i}) \qquad \forall a_{-i} \in A_{-i}$$

$$p_j \geq 0 \qquad \forall j \in A_i$$

$$\sum_{j \in A_i} p_j = 1$$

any feasible solution will have a non-negative objective value

Domination by a mixed strategy

- If the optimal solution > 0 (not ≥), the mixed strategy given by the p_j 's achieves strictly positive expected utility for at least one $a_{-i} \in A_{-i}$, meaning that s_i is weakly dominated by this mixed strategy

$$
\begin{aligned}
\text{maximize} \quad & \sum_{a_{-i} \in A_{-i}} \left[\left(\sum_{j \in A_i} p_j \cdot u_i(a_j, a_{-i}) \right) - u_i(s_i, a_{-i}) \right] \\
\text{subject to} \quad & \sum_{j \in A_i} p_j u_i(a_j, a_{-i}) \geq u_i(s_i, a_{-i}) && \forall a_{-i} \in A_{-i} \\
& p_j \geq 0 && \forall j \in A_i \\
& \sum_{j \in A_i} p_j = 1
\end{aligned}
$$

Iterated dominance

- For all three flavors of domination, it requires only polynomial time to iteratively remove dominated strategies until the game has been maximally reduced
- A single step of this process consists of checking whether every pure strategy of every player is dominated by any other mixed strategy, which requires us to solve at worst $\boldsymbol{\Sigma_{i \in N}|A_i|}$ linear programs
- Each step removes one pure strategy for one player, so there can be at most $\boldsymbol{\Sigma_{i \in N}(|A_i|-1)}$ steps

Iterated dominance

- Some forms of dominance can produce different reduced games
- Some computational questions
  - (**Strategy elimination**) Does there exist some elimination path under which the strategy $\boldsymbol{s_i}$ is eliminated?
  - (**Reduction identity**) Given action subsets $\boldsymbol{A_i' \subseteq A_i}$ for each player $\boldsymbol{i}$, does there exist a maximally reduced game where each player $\boldsymbol{i}$ has the actions $\boldsymbol{A_i'}$?
  - (**Reduction size**) Given constants $\boldsymbol{k_i}$ for each player $\boldsymbol{i}$, does there exist a maximally reduced game where each player $\boldsymbol{i}$ has exactly $\boldsymbol{k_i}$ actions?

Iterated dominance

- **Theorem**
  - (i) For iterated strict dominance, the strategy elimination, reduction identity, uniqueness and reduction size problems are in ***P***. (ii) For iterated weak dominance, these problems are ***NP-complete***
    - (i) it follows from the fact that iterated strict dominance always arrives at the same set of strategies regardless of elimination order
    - (ii) Gilboa et al. [1989] and Conitzer and Sandholm [2005]

Computing Correlated Equilibria

maximize: $$\sum_{a \in A} \boxed{p(a)} \sum_{i \in N} u_i(a)$$

probability of suggesting strategy profile ***a***

$$\sum_{a \in A | a_i \in a} p(a) u_i(a) \geq \sum_{a \in A | a_i \in a} p(a) u_i(a_i', a_{-i}) \quad \forall i \in N, \forall a_i, a_i' \in A_i$$

$$p(a) \geq 0 \quad \forall a \in A$$

$$\sum_{a \in A} p(a) = 1$$

Computing Correlated Equilibria

maximize: $\displaystyle \sum_{a \in A} p(a) \sum_{i \in N} u_i(a)$

$$\sum_{a \in A | a_i \in a} [u_i(a) - u_i(a'_i, a_{-i})] p(a) \geq$$

$$p(a) \geq 0 \qquad \forall a \in A$$

$$\sum_{a \in A} p(a) = 1$$

Computing Correlated Equilibria

- *max [6p(D,D) + 6p(D,H) + 6p(H,D) + 0p(H,H)]*
- *s.t.*
- $p(D,D)(3-5) + p(D,H)(1-0) \geq 0$
- $p(H,D)(5-3) + p(H,H)(0-1) \geq 0$
- $p(D,D)(3-5) + p(H,D)(1-0) \geq 0$
- $p(D,H)(5-3) + p(H,H)(0-1) \geq 0$
- $p(D,D) + p(D,H) + p(H,D) + p(H,H) = 1$
- $p(D,D), p(D,H), p(H,D), p(H,H) \geq 0$

	D	*H*
D	3,3	1,5
H	5,1	0,0

Computing Correlated Equilibria

- *max [6p(D,D) + 6p(D,H) + 6p(H,D)]*
- *s.t.*
- *-2 p(D,D) + p(D,H) ≥ 0*
- *2 p(H,D) - p(H,H) ≥ 0*
- *-2 p(D,D) + p(H,D) ≥ 0*
- *2 p(D,H) - p(H,H) ≥ 0*
- *p(D,D) + p(D,H) + p(H,D) + p(H,H) = 1*
- *p(D,D), p(D,H), p(H,D), p(H,H) ≥ 0*

	D	*H*
D	3,3	1,5
H	5,1	0,0

Computing Correlated Equilibria

- $\min -[6p(D,D) + 6p(D,H) + 6p(H,D)]$
- s.t.
- $2\,p(D,D) - p(D,H) \le 0$
- $-2\,p(H,D) + p(H,H) \le 0$
- $2\,p(D,D) - p(H,D) \le 0$
- $-2\,p(D,H) + p(H,H) \le 0$
- $p(D,D) + p(D,H) + p(H,D) + p(H,H) = 1$
- $p(D,D), p(D,H), p(H,D), p(H,H) \ge 0$

	D	H
D	3,3	1,5
H	5,1	0,0

Computing Correlated Equilibria

	D	H
D	3,3	1,5
H	5,1	0,0

- ***min -[6p(D,D) + 6p(D,H) + 6p(H,D)]***
- ***s.t.***
- ***2 p(D,D) - p(D,H) ≤ 0***
- ***-2 p(H,D) + p(H,H) ≤ 0***
- ***2 p(D,D) - p(H,D) ≤ 0***
- ***-2 p(D,H) + p(H,H) ≤ 0***
- ***p(D,D) + p(D,H) + p(H,D) + p(H,H) = 1***
- ***p(D,D), p(D,H), p(H,D), p(H,H) ≥ 0***

Matlab code:

```
f = [-6  -6 -6 0];

A = [ 2 -1 0 0 ;0 0 -2 1 ; 2 0 -1 0 ;
0 -2 0 1 ];

b = [0 0 0 0];

Aeq = [1 1 1 1];

beq = 1;

lb = [0 0 0 0];

z = linprog(f,A,b,Aeq,beq,lb)
z =

    0.1220
    0.4390
    0.4390
    0.0000
```

Computing Correlated Equilibria

- ***Theorem***
  - The following problems are in the complexity class ***P*** when applied to correlated equilibria: uniqueness, Pareto optimal, guaranteed payoff, subset inclusion, and subset containment

Computing Correlated Equilibria

- ***Problem:***
  - Interpersonal comparison of utility
    - Depends upon the ability to aggregate, or sum up, individual preferences into a combined social welfare function
  - In the previous example, if you make cell ***(H,D) = (1000, 1)***, the correlated equilibrium will give probability **1** to this state

Computing Correlated Equilibria

	D	H
D	3,3	1,6
H	5,1	0,0

- *min* $-[6p(D,D) + 7p(D,H) + 6p(H,D)]$
- *s.t.*
- $2\,p(D,D) - p(D,H) \leq 0$
- $-2\,p(H,D) + p(H,H) \leq 0$
- $3\,p(D,D) - p(H,D) \leq 0$
- $-3\,p(D,H) + p(H,H) \leq 0$
- $p(D,D) + p(D,H) + p(H,D) + p(H,H) = 1$
- $p(D,D), p(D,H), p(H,D), p(H,H) \geq 0$

Python code:

```
f = [-6,  -7, -6, 0]
A = [[2, -1, 0, 0] , [0, 0, -2, 1],
[3, 0, -1, 0 ], [0, -3, 0, 1 ]]

b = [0, 0, 0, 0]

Aeq = [[1, 1, 1, 1]]

beq = [1]
x0_bounds = (0, None)
x1_bounds = (0, None)
x2_bounds = (0, None)
x3_bounds = (0, None)

res = linprog(f, A_ub=A, b_ub=b,
bounds=(x0_bounds, x1_bounds,
x2_bounds, x3_bounds), A_eq=Aeq,
b_eq=beq)

x: array([0., 1., 0., 0.])
fun: -7.0
```

Computing Correlated Equilibria

	D	H
D	3,3	1,6
H	5,1	0,0

- *min -[6p(D,D) + 7p(D,H) + 6p(H,D)]*
- *s.t.*
- *2 p(D,D) - p(D,H) ≤ 0*
- *-2 p(H,D) + p(H,H) ≤ 0*
- *3 p(D,D) - p(H,D) ≤ 0*
- *-3 p(D,H) + p(H,H) ≤ 0*
- *p(D,D) + p(D,H) + p(H,D) + p(H,H) = 1*
- *p(D,D), p(D,H), p(H,D), p(H,H) ≥ 0*
- *p(D,H) - p(H,D) = 0*

Python code:

```
f = [-6,  -7, -6, 0]
A = [[2, -1, 0, 0] , [0, 0, -2, 1],
[3, 0, -1, 0 ], [0, -3, 0, 1 ]]

b = [0, 0, 0, 0]

Aeq = [[1, 1, 1, 1],[0, 1, -1, 0]]

beq = [1, 0]
x0_bounds = (0, None)
x1_bounds = (0, None)
x2_bounds = (0, None)
x3_bounds = (0, None)

res = linprog(f, A_ub=A, b_ub=b,
bounds=(x0_bounds, x1_bounds,
x2_bounds, x3_bounds), A_eq=Aeq,
b_eq=beq)

x: array([0. , 0.5, 0.5, 0. ])
fun: -6.5
```

Computing Correlated Equilibria

- **Q:** Why can we express the definition of a correlated equilibrium as a linear constraint, while we cannot do the same with the definition of a Nash equilibrium, even though both definitions are quite similar?
- **A:** The difference is that a correlated equilibrium involves a single randomization over action profiles, while in a Nash equilibrium agents randomize separately

Software

- GAMBIT [McKelvey et al., 2006] (http://www.gambit-project.org/) is a library of game-theoretic algorithms for finite normal-form and extensive-form games
- GAMUT [Nudelman et al., 2004] (http://gamut.stanford.edu) is a suite of game generators designed for testing game-theoretic algorithms
- NASHPY [Knight. 2017] (https://nashpy.readthedocs.io/en/stable/) is a Python library used for the computation of equilibria in 2 player strategic form games.