Simple Search Methods for Finding a Nash Equilibrium

By Mahnoor Malik
PROBLEM

• How do you find a Nash Equilibrium?
• So far the best ways to do so have been:
  • Lemke-Howson for 2-player games
  • Govindan-Wilson for n-player games
  • Simplicial Subdivision for n-player games

• BUT the algorithms in this paper, when tested, are “better” in practice than existing ones
Contributions

- Heuristic search techniques in algorithms
- An extensive test suite to evaluate algorithms
- Focus on running empirical tests (i.e. using a testbed)
Notation

- n-player, normal form games: \( G = \langle N, (A_i), (u_i) \rangle \)
- \( A_i = \{a_{i1}, \ldots, a_{im_i}\} \) : Set of actions for player i
- \( a = (a_i, \ldots, a_n) \) : Profile of actions, one for every player
- \( a_{-i} = (a_i, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n) \) : Profile of actions of all players except i
- \( u_i : A_i x \ldots x A_n \rightarrow \mathbb{R} \) : Utility function for player i
- \( \mathcal{P}_i = \left\{ p_i : A_i \rightarrow [0, 1] \mid \sum_{a_i \in A_i} p_i(a_i) = 1 \right\} \) : Player i selects mixed strategy
- \( x = (x_1, \ldots, x_n) \) : Size of support of each player
- \( p = (p_i, \ldots, p_n) \) : Strategy Profile
- \( u_i(p) = \sum_{a \in A} p(a)u_i(a) \) where \( p(a) = \prod_{i \in N} p_i(a_i) \) : \( \mathbb{E}[u] \) of player i
- A strategy profile \( p^* \in \mathcal{P} \) is a Nash Equilibrium if:
  \( \forall i \in N, a_i \in A_i : u_i(a_i, p^*_{-i}) \leq u_i(p^*_i, p^*_{-i}) \)
Given a support profile, this gives you the Nash Equilibrium (mixed) strategy $p$.

**PS.** $v$ here is refers to profile of the expected utility of the players in equilibrium.
Algorithm 1

for all support size profiles $x = (x_1, x_2)$, sorted in increasing order of, first, $|x_1 - x_2|$ and, second, $(x_1 + x_2)$ do

for all $S_1 \subseteq A_1$ s.t. $|S_1| = x_1$ do

$A'_2 \leftarrow \{a_2 \in A_2$ not conditionally dominated, given $S_1\}$

if $\nexists a_1 \in S_1$ conditionally dominated, given $A'_2$ then

for all $S_2 \subseteq A'_2$ s.t. $|S_2| = x_2$ do

if $\nexists a_1 \in S_1$ conditionally dominated, given $S_2$ then

if Feasibility Program 1 is satisfiable for $S = (S_1, S_2)$ then

Return the found NE $p$
n-player games: Algorithm

Algorithm 2

for all \( x = (x_1, \ldots, x_n) \), sorted in increasing order of, first, \( \sum_i x_i \) and, second, \( \max_{i,j}(x_i - x_j) \) do

\( \forall i: S_i \leftarrow \text{NULL} \quad \text{//uninstantiate supports} \)

\( \forall i: D_i \leftarrow \{ S_i \subseteq A_i : |S_i| = x_i \} \quad \text{//domain of supports} \)

if Recursive-Backtracking\((S, D, 1)\) returns a NE \( p \) then

Return \( p \)
n-player games: Recursive Backtracking

Procedure 1 Recursive-Backtracking

Input: $S = (S_1, \ldots, S_n)$: a profile of supports
       $D = (D_1, \ldots, D_n)$: a profile of domains
       $i$: index of next support to instantiate

Output: A Nash equilibrium $p$, or failure

if $i = n + 1$ then
    if Feasibility Program 1 is satisfiable for $S$ then
        Return the found NE $p$
    else
        Return failure
else
    for all $d_i \in D_i$ do
        $S_i \leftarrow d_i$
        $D_i \leftarrow D_i - \{d_i\}$
        if IRSDS($([S_1], \ldots, \{S_i\}, D_{i+1}, \ldots, D_n)$) succeeds then
            if Recursive-Backtracking($S, D, i + 1$) returns NE $p$ then
                Return $p$
        Return failure
n-player games: IRSDS

Procedure 2 Iterated Removal of Strictly Dominated Strategies (IRSDS)

Input: $D = (D_1, \ldots, D_n)$: profile of domains
Output: Updated domains, or failure

repeat
  changed ← false
  for all $i \in N$ do
    for all $a_i \in \bigcup_{d_i \in D_i} d_i$ do
      for all $a_i' \in A_i$ do
        if $a_i$ is conditionally dominated by $a_i'$, given $\bigcup_{d_{-i} \in D_{-i}} d_{-i}$ then
          $D_i \leftarrow D_i - \{d_i \in D_i: a_i \in d_i\}$
          changed ← true
          if $D_i = \emptyset$ then
            Return failure
      end for
  end for
until changed = false
Return $D$
TestBed: Subsets tested

<table>
<thead>
<tr>
<th>Descriptions of GAMUT distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D1</strong> Bertrand Oligopoly</td>
</tr>
<tr>
<td><strong>D3</strong> Bidirectional LEG, Random Graph</td>
</tr>
<tr>
<td><strong>D5</strong> Covariance Game: $\rho = 0.9$</td>
</tr>
<tr>
<td><strong>D7</strong> Covariance Game: $\rho = 0$</td>
</tr>
<tr>
<td><strong>D9</strong> Graphical Game, Random Graph</td>
</tr>
<tr>
<td><strong>D11</strong> Graphical Game, Star Graph</td>
</tr>
<tr>
<td><strong>D13</strong> Minimum Effort Game</td>
</tr>
<tr>
<td><strong>D15</strong> Polymatrix Game, Random Graph</td>
</tr>
<tr>
<td><strong>D17</strong> Polymatrix Game, Small-World</td>
</tr>
<tr>
<td><strong>D19</strong> Travelers Dilemma</td>
</tr>
<tr>
<td><strong>D21</strong> Uniform LEG, Random Graph</td>
</tr>
<tr>
<td><strong>D23</strong> Location Game</td>
</tr>
</tbody>
</table>
Testing

- Experiments run on 12-dual-processor, 2.4GHz Premium machines running Linux 2.4.20
- Algorithms were set to timeout at 1800 seconds
- Unconditional: Where timeouts were counted as 1800 seconds
- Conditional: Where timeouts were excluded
Results: 2-player games

- L-H failed to solve any game > 600 actions whereas Algorithm 1 solved all instances
- Hardest distribution for both algorithms was ‘Covariance Games’
Results: n-player games

- 6-player 5-action games
- Best results for unconditional median-running time
- Fared well in other metrics too, but not by as large a magnitude
- In scaling behavior, i.e. varying number of players or actions, Algorithm 2 still managed to find solutions for *most* games
Caveat

The algorithms both first check for a pure strategy equilibrium most of the instances had these:
Discussion Questions

- Given the large proportion of games with pure strategy equilibria, is it still fair to argue that Algorithm 1 and 2 would be "better" or "faster" than the ones they're being compared to? The authors suggest they are.

- Is it enough to have an algorithm that performs well on a of specific "empirical tests" or do we need/want more mathematical proof?

- In cases where there is more than one NE, will there be any real life situations where we may need to know a different equilibrium than the one we got?