Game Theory Explorer - Software for the Applied Game Theorist

Rahul Savani
University of Liverpool

Bernhard von Stengel
London School of Economics

May 2016
Overview

Explain and demonstrate GTE (Game Theory Explorer), open-source software, under development, for creating and analyzing non-cooperative games

in strategic form:

and extensive form:
Intended users

Applied game theorists:
- experimental economists (analyze game before running experiment)
- game-theoretic modelers in biology, political science, . . .
- in general: non-experts in equilibrium analysis
⇒ design goal: ease of use

Researchers in game theory:
- testing conjectures about equilibria
- as contributors: designers of game theory algorithms

Educators:
- interactive tool to explain solution concepts and algorithms
History: Gambit

GTE now part of the **Gambit** open-source software development, http://www.gambit-project.org


Gambit software started \(\sim\) 1990 with **Richard McKelvey** (Caltech) to analyze games for **experiments**, developed since 1994 with **Andy McLennan** into C++ package, since 2001 maintained by **Ted Turocy** (UEA, Norwich, UK).

- Gambit must be **installed** on PC/Mac/Linux, with GUI (graphical user interface) using platform-independent **wxWidgets**
- has collection of algorithms for computing Nash equilibria
- offers **scripting language**, now developed using Python
Features of GTE

GTE independent browser-based development:
• no software installation needed, low barrier to entry
• nicer GUI than Gambit
• export to graphical formats
Features of GTE

GTE independent browser-based development:

- no software installation needed, low barrier to entry
- nicer GUI than Gambit
- export to graphical formats

Disadvantages:

- manual storing / loading of files for security reasons
- long computations require local server installation (same GUI)

Other Contributors:

Features of GTE

GTE independent browser-based development:

- no software installation needed, low barrier to entry
- nicer GUI than Gambit
- export to graphical formats

Disadvantages:

- manual storing / loading of files for security reasons
- long computations require local server installation (same GUI)

Example of a game

$2 \times 2$ game in strategic form:

\[
\begin{array}{c|cc}
 & l & r \\
\hline
T & 2 & 1 \\
B & 3 & 4 \\
\end{array}
\]
Example of a game

2 × 2 game in strategic form:

with pure best responses
Example of a game

$2 \times 2$ game in strategic form:

\[
\begin{array}{ccc}
 & 0 & 1 \\
 I & l & r \\
 0 & 2 & 1 \\
 1 & 3 & 4 \\
\end{array}
\]

with pure best responses and equilibrium probabilities
Extensive (= tree) form of the game

Players move sequentially,

*information sets* show lack of information about game state:

```
I
  T   B
  l   r
  5   3
   l   r
  2   1
II
  6   4
   3   4
```
Commitment (leadership) game

Changed game when player I can commit:

```
          I
           /\  
         T  B
     /     / \
 II  l   a  II
     \   /  \
      5  3  6  4
          /   /  \
         2   1  3  4
```

Subgame perfect equilibrium: (T, l - b)

Other equilibria: 8
Commitment (leadership) game

Changed game when player I can commit:

Subgame perfect equilibrium: \((T, l-b)\)
Commitment (leadership) game

Changed game when player I can commit:

Subgame perfect equilibrium: \((T, l-b)\)
Commitment (leadership) game

Changed game when player I can commit:

Subgame perfect equilibrium: $(T, l-b)$
Commitment (leadership) game

Changed game when player I can commit:

Subgame perfect equilibrium: \((T, l-b)\)
Commitment (leadership) game

Changed game when player I can commit:

Subgame perfect equilibrium: \((T, l-b)\)

Other equilibria: \((B, r-b)\)
Commitment (leadership) game

Changed game when player I can commit:

Subgame perfect equilibrium: \((T, l-b)\)

Other equilibria: \((B, r-b), (B, \frac{1}{2}l-b \frac{1}{2}r-b)\)
Commitment (leadership) game

Changed game when player I can commit:

Subgame perfect equilibrium: \((T, l-b)\)

Other equilibria: \((B, r-b)\), \((B, \frac{1}{2}l-b \frac{1}{2}r-b)\), \((T, \frac{1}{2}l-a \frac{1}{2}l-b)\)
GTE output for the commitment game

2 x 4 Payoff player 1
1-a  l-b  r-a  r-b
T  5  5  3  3
B  6  4  6  4

2 x 4 Payoff player 2
1-a  l-b  r-a  r-b
T  2  2  1  1
B  3  4  3  4

EE = Extreme Equilibrium, EP = Expected Payoffs

Rational:
EE 1 P1: (1) 0 1 EP= 4 P2: (1) 0 1/2 0 1/2 EP= 4
EE 2 P1: (1) 0 1 EP= 4 P2: (2) 0 0 0 1 EP= 4
EE 3 P1: (2) 1 0 EP= 5 P2: (3) 0 1 0 0 EP= 2
EE 4 P1: (2) 1 0 EP= 5 P2: (4) 1/2 1/2 0 0 EP= 2

Connected component 1:
\{1\} x \{1, 2\}

Connected component 2:
\{2\} x \{3, 4\}
Demonstration of GTE

Preceding games:

- $2 \times 2$ game in strategic form
- extensive form of that game
- commitment game, extensive and strategic form
Demonstration of GTE

Preceding games:
- $2 \times 2$ game in strategic form
- extensive form of that game
- commitment game, extensive and strategic form

Next: create from scratch a more complicated extensive game
- imperfectly observable commitment
Game with imperfectly observable commitment
Game tree drawn left to right

Purpose
Usage
Client/Server
Algorithms
Future
## GTE output for imperfectly observable commitment

<table>
<thead>
<tr>
<th></th>
<th>2 x 4 Payoff player 1</th>
<th>2 x 4 Payoff player 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>l-a</td>
<td>l-b</td>
</tr>
<tr>
<td>T</td>
<td>5 249/50</td>
<td>151/50</td>
</tr>
<tr>
<td>B</td>
<td>6 201/50</td>
<td>299/50</td>
</tr>
</tbody>
</table>

EE = Extreme Equilibrium, EP = Expected Payoffs

**Decimal:**

EE 1 P1: (1) 0.01 0.99 EP= 4.0102 P2: (1) 0 0.5102 0 0.4898 EP= 3.97

EE 2 P1: (2) 0 1.0 EP= 4.0 P2: (2) 0 0 0 1.0 EP= 4.0

EE 3 P1: (3) 0.99 0.01 EP= 4.9898 P2: (3) 0.4898 0.5102 0 0 EP= 2.01

**Connected component 1:**

\{1\} \times \{1\}

**Connected component 2:**

\{2\} \times \{2\}

**Connected component 3:**

\{3\} \times \{3\}
More complicated signaling game, 5 equilibria
Some more strategic-form games

For studying more complicated games:

generate game matrices as text files, copy and paste into strategic-form input.

Future extension:

Automatic generation via command lines or “worksheets” for scripting, connection with Python and Gambit
GTE software architecture

**Client** (your computer with a browser):
- GUI: JavaScript (Flash’s variant called ActionScript)
- store and load game described in XML format
- export to graphic formats (.png or XFIG → EPS, PDF)
- for algorithm: send XML game description to server
GTE software architecture

**Client** (your computer with a browser):
- GUI: JavaScript (Flash’s variant called ActionScript)
- store and load game described in XML format
- export to graphic formats (.png or XFIG → EPS, PDF)
- for algorithm: send XML game description to server

**Server** (hosting client program and equilibrium solvers):
- converts XML to Java data structure (similar to GUI)
- solution algorithms as binaries (e.g. C program `lrs`), send results as text back to client
High usage of computation resources

Finding all equilibria takes exponential time
⇒ for large games, server should run on your computer, not a public one

achieved by local server installation (“Jetty”), requires installation, but offers same user interface.
Algorithm: Finding all equilibria

For two-player games in strategic form, all Nash equilibria can be found as follows:

- payoffs define inequalities for “best response polyhedra”
- compute all vertices of these polyhedra (using Lrs by David Avis, requires arbitrary precision integers)
- match vertices for complementarity (LCP)
- find maximal cliques of matching vertices for equilibrium components
Example
Best response polyhedron of player I

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

payoff player I

\[ \text{prob}(r) \]
Best response polyhedron of player I

payoff player I
Best response polyhedron of player I

payoff player I

\[ \text{prob}(r) \]

\[ B \]
Best response polyhedron of player I

\[
\begin{array}{ccc}
&T & l & r \\
B & 0 & 0 & 2 \\
T & 1 & 3 & 3 \\
\end{array}
\]

payoff player I

\[
\text{prob}(r) \quad 0 \quad 1
\]
Best response polyhedron of player II

Payoff player I

Payoff player II

Purpose
Usage
Client/Server
Algorithms
Future
Best response polyhedron of player II
Best response polyhedron of player II

\begin{array}{c|cc|c}
 & l & r & \text{prob}(B) \\
\hline
T & 1 & 3 & 10 \\
B & 0 & 2 & 10 \\
\end{array}

\begin{array}{c|cc|c}
 & l & r & \text{prob}(r) \\
\hline
T & 3 & 3 & \text{payoff player I} \\
B & 0 & 2 & \text{payoff player II} \\
\end{array}
Label with best responses and unplayed strategies

payoff player I

payoff player II

B=0
T=0
prob(B)
prob(r)
Equilibrium = all labels $T$, $B$, $l$, $r$ present

- Payoff player I
  - $r=0$ payoff region
  - $l=0$ payoff region

- Payoff player II
  - $B=0$ payoff region
  - $T=0$ payoff region

Equilibrium conditions:

- $r = 0$ (Player II chooses $B$)
- $l = 0$ (Player I chooses $T$)
- $B = 0$
- $T = 0$
Equilibrium with multiple label $r$ (degeneracy)
Equilibrium with multiple label $B$ (degeneracy)

payoff player I

$B=0$

payoff player II

$T=0$
equilibrium component with labels $T$ and $B, l, r$
Equilibrium components via cliques

In degenerate games (= vertices with zero basic variables, occur for game trees), get convex combinations of “exchangeable” equilibria. Recognized as cliques of matching vertex pairs:

\[ \begin{array}{cccc}
  v_1 & v_2 & v_3 & v_4 \\
  u_1 & & & \\
  u_2 & & & \\
  u_3 & & & \\
  u_4 & & & \\
\end{array} \]

Table of extreme equilibria
Algorithm: Sequence form for game trees

Example of game tree:
Exponentially large strategic form

**Strategy** of a player:
specifies a move for every information set of that player
(except for unspecified moves * at unreachable information sets)

⇒ **exponential** number of strategies

<table>
<thead>
<tr>
<th></th>
<th>ap*</th>
<th>aq*</th>
<th>b**</th>
<th>c*s</th>
<th>c*t</th>
<th>d**</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L*C</strong></td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>20</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td><strong>L*D</strong></td>
<td>5</td>
<td>5</td>
<td>20</td>
<td>30</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td><strong>RUC</strong></td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td><strong>RUD</strong></td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>30</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td><strong>RVC</strong></td>
<td>15</td>
<td>–5</td>
<td>10</td>
<td>20</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td><strong>RVD</strong></td>
<td>15</td>
<td>–5</td>
<td>20</td>
<td>30</td>
<td>15</td>
<td>5</td>
</tr>
</tbody>
</table>
Sequences instead of strategies

**Sequence** specifies moves only along **path** in game tree

⇒ **linear** number of sequences, sparse payoff matrix $A$

<table>
<thead>
<tr>
<th></th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$d$</th>
<th>$ap$</th>
<th>$aq$</th>
<th>$cs$</th>
<th>$ct$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$L$</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R$</td>
<td></td>
<td>10</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$RU$</td>
<td></td>
<td>15</td>
<td>-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C$</td>
<td>10</td>
<td>20</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D$</td>
<td>20</td>
<td>30</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Expected payoff $x^\top Ay$, play **rows** with $x \geq 0$ subject to $Ex = e$,

play **columns** with $y \geq 0$ subject to $Fy = f$. 
Given: $x \geq 0$ with $Ex = e$.

Move $L$ is last move of unique sequence, say $PQL$, where one row of $Ex = e$ says $x_{PQL} + x_{PQR} = x_{PQ}$

$\Rightarrow$ behavior-probability($L$) = $\frac{x_{PQL}}{x_{PQ}}$
Play as behavior strategy

Given: \( x \geq 0 \) with \( Ex = e \).

Move \( L \) is last move of unique sequence, say \( PQL \), where one row of \( Ex = e \) says \( x_{PQL} + x_{PQR} = x_{PQ} \)

\[ \Rightarrow \text{behavior-probability}(L) = \frac{x_{PQL}}{x_{PQ}} \]

Required assumption of perfect recall [Kuhn 1953, Selten 1975]:
Each node in an information set is preceded by same sequence, here \( PQ \), of the player’s own earlier moves.
Linear-sized sequence form

**Input:** Two-person game tree with perfect recall.

**Theorem** [Romanovskii 1962, von Stengel 1996]

The equilibria of a **zero-sum** game are the solutions to a Linear Program (LP) of **linear** size in the size of the game tree.
Linear-sized sequence form

**Input:** Two-person game tree with perfect recall.

**Theorem** [Romanovskii 1962, von Stengel 1996]

The equilibria of a **zero-sum** game are the solutions to a Linear Program (LP) of **linear** size in the size of the game tree.


The equilibria of a **non-zero-sum** game are the solutions to a Linear Complementarity Problem (LCP) of linear size.

A sample equilibrium is found by **Lemke’s algorithm**.

This algorithm mimicks the Harsanyi–Selten tracing procedure and finds a normal-form perfect equilibrium.
Three GSoC students currently working on:

- Improve and convert GUI to pure JavaScript
- Advanced game tree layout e.g. drawing information sets in games without time structure
- Educational features (example next)
Example of educational feature

payoff player I

payoff player II

best responses player I

best responses player II

purposes usage client/server algorithms future
# Planned Extensions

**Further solution algorithms:**
- EEE [Audet/Hansen/Jaumard/Savard 2001]
- Path-following algorithms (Lemke-Howson, variants of Lemke)
- $n$-player games: simplicial subdivision, polynomial inequalities

**Scripting features:**
- connect with Gambit and Python
- database of reproducible computational experiments

**Educational features:**
- teaching algorithms interactively
Summary

GTE – Game theory explorer

- helps **create**, **draw**, and **analyze** game-theoretic models
- user-friendly, browser-based, low barriers to entry
- open-source, work in progress, **welcomes contributors**

https://github.com/gambitproject/gte/
https://github.com/gambitproject/jsgte/

Rahul Savani and Bernhard von Stengel (2016)
*Game Theory Explorer – Software for the Applied Game Theorist*
*Computational Management Science* 12, 5-33.