Attracting Students to Computer Science Using Artificial Intelligence, Economics, and Linear Programming

Vincent Conitzer
Duke University
AI & Economics

• AI has always used techniques from economics
  – Herbert Simon
  – Probabilities, beliefs, utility, discounting, …

• Last ~decade: AI research increasingly focused on multiagent systems, economics
  – Auctions, voting
  – Game theory
  – Mechanism design

• Conferences
  – Conference on Autonomous Agents and Multiagent Systems (AAMAS)
  – ACM Conference on Electronic Commerce (EC)
  – Also lots of work at IJCAI, AAAI, …
  – Some at UAI, ICML, …
What is Economics?

- “the social science that studies the production, distribution, and consumption of valuable goods and services” [Wikipedia, Jan. 07]

- Some key concepts:
  - Economic agents or players (individuals, households, firms, …)
  - Agents’ current endowments of goods, money, skills, …
  - Possible outcomes ((re)allocations of resources, tasks, …)
  - Agents’ preferences or utility functions over outcomes
  - Agents’ beliefs (over other agents’ utility functions, endowments, production possibilities, …)
  - Agents’ possible decisions/actions
  - Mechanism that maps decisions/actions to outcomes
An economic picture

$v(\text{server}) = 200$

$\$800$

$v(\text{desktop}) = 100$

$v(\text{laptop}) = 400$

$v(\text{TV}) = 200$

$v(\text{computer}) = 400$

$\$600$

$\$200$
After trade (a more efficient outcome)

\[ v( ) = 200 \]
\[ v( ) = 100 \]
\[ v( ) = 400 \]

$ 1100$

... but how do we get here?
Auctions?
Exchanges?
Unstructured trade?

\[ v( ) = 400 \]
\[ v( ) = 200 \]

$ 400$

$ 100$
Economic mechanisms

“true” input

\[ v(\text{agent 1}) = 400 \]
\[ v(\text{agent 2}) = 600 \]

agents’ bids

\[ v(\text{agent 1}) = 500 \]
\[ v(\text{agent 2}) = 501 \]

result

\[ \text{Exchange mechanism (algorithm)} \]
\[ v(\text{agent 1}) = 451 \]
\[ v(\text{agent 2}) = 450 \]

Exchange mechanism designer does not have direct access to agents’ private information

Agents will selfishly respond to incentives
Teaching an introductory course

• Goals:
  – Expose computer science students to basic concepts from microeconomics, game theory
  – Expose economics students to basic concepts in programming, algorithms
  – Show how to increase economic efficiency using computation

• Cannot include whole intro programming course

• Solution: focus strictly on linear/integer programming
  – Can address many economics problems
  – Nice modeling languages that give flavor of programming
  – Computer science students have generally not been exposed to this either
Example linear program

- We make reproductions of two paintings
  - maximize $3x + 2y$
  - subject to
    - $4x + 2y \leq 16$
    - $x + 2y \leq 8$
    - $x + y \leq 5$
    - $x \geq 0$
    - $y \geq 0$

- Painting 1 sells for $30, painting 2 sells for $20
- Painting 1 requires 4 units of blue, 1 green, 1 red
- Painting 2 requires 2 blue, 2 green, 1 red
- We have 16 units blue, 8 green, 5 red
Solving the linear program graphically

maximize $3x + 2y$

subject to

$4x + 2y \leq 16$

$x + 2y \leq 8$

$x + y \leq 5$

$x \geq 0$

$y \geq 0$

optimal solution: $x = 3$, $y = 2$
Modified LP

maximize $3x + 2y$

subject to

$4x + 2y \leq 15$

$x + 2y \leq 8$

$x + y \leq 5$

$x \geq 0$

$y \geq 0$

Optimal solution: $x = 2.5, y = 2.5$

Solution value = $7.5 + 5 = 12.5$

Half paintings?
Integer (linear) program

maximize $3x + 2y$
subject to
$4x + 2y \leq 15$
$x + 2y \leq 8$
$x + y \leq 5$
$x \geq 0$, integer
$y \geq 0$, integer

optimal LP solution: $x=2.5$, $y=2.5$ (objective 12.5)

optimal IP solution: $x=2$, $y=3$ (objective 12)
Mixed integer (linear) program

\[
\text{maximize } 3x + 2y
\]

subject to

\[
\begin{align*}
4x + 2y & \leq 15 \\
x + 2y & \leq 8 \\
x + y & \leq 5 \\
x & \geq 0 \\
y & \geq 0, \text{ integer}
\end{align*}
\]

optimal LP solution: \(x=2.5, y=2.5\) (objective 12.5)

optimal IP solution: \(x=2, y=3\) (objective 12)

optimal MIP solution: \(x=2.75, y=2\) (objective 12.25)
The MathProg modeling language

set PAINTINGS;
set COLORS;

var quantity_produced{j in PAINTINGS}, >=0, integer;
param selling_price{j in PAINTINGS};
param paint_available{i in COLORS};
param paint_needed{i in COLORS, j in PAINTINGS};

maximize revenue: sum{j in PAINTINGS}
  selling_price[j]*quantity_produced[j];

s.t. enough_paint{i in COLORS}: sum{j in PAINTINGS}
  paint_needed[i,j]*quantity_produced[j] <=
  paint_available[i];

...
The MathProg modeling language

... data;
set PAINTINGS := p1 p2;
set COLORS := blue green red;
param selling_price := p1 3 p2 2;
param paint_available := blue 15 green 8 red 5;
param paint_needed :
       p1  p2 :=
blue   4  2
green  1  2
red    1  1;
end;
A knapsack-type problem

- We arrive in a 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
  - There are 3 units available
- Unit of object B: 4kg, 4 liters, sells for $4
  - There are 4 units available
- Unit of object C: 6kg, 3 liters, sells for $9
  - Only 1 unit available
- What should we take?
Knapsack-type problem instance…

\[
\begin{align*}
\text{maximize} & \quad 11x + 4y + 9z \\
\text{subject to} & \\
16x + 4y + 6z & \leq 30 \\
3x + 4y + 3z & \leq 20 \\
x & \leq 3 \\
y & \leq 4 \\
z & \leq 1 \\
x, y, z & \geq 0, \text{ integer}
\end{align*}
\]
Knapsack-type problem instance in MathProg modeling language

set OBJECT;
set CAPACITY_CONSTRAINT;
param cost{i in OBJECT, j in CAPACITY_CONSTRAINT};
param limit{j in CAPACITY_CONSTRAINT};
param availability{i in OBJECT};
param value{i in OBJECT};
var quantity{i in OBJECT}, integer, >= 0;
maximize total_value: sum{i in OBJECT} quantity[i]*value[i];
s.t. capacity_constraints {j in CAPACITY_CONSTRAINT}: sum{i in OBJECT} cost[i,j]*quantity[i] <= limit[j];
s.t. availability_constraints {i in OBJECT}: quantity[i] <= availability[i];
...

Knapsack-type problem instance in MathProg modeling language...

...
data;
set OBJECT := a b c;
set CAPACITY_CONSTRAINT := weight volume;
param cost: weight volume :=
  a 16 3
  b 4 4
  c 6 3;
param limit:= weight 30 volume 20;
param availability:= a 3 b 4 c 1;
param value:= a 11 b 4 c 9;
end;
Combinatorial auctions

Simultaneously for sale:  

\[ v(\text{server, monitor}) = $500 \]

\[ v(\text{computer, monitor}) = $700 \]

\[ v(\text{computer}) = $300 \]

used in truckload transportation, industrial procurement, radio spectrum allocation, …
The **winner determination problem (WDP)**

- Choose a subset $A$ (the accepted bids) of the bids $B$,
- to maximize $\sum_{b \in A} v_b$,
- under the constraint that every item occurs at most once in $A$
  - This is assuming **free disposal**, i.e. not everything needs to be allocated
An integer program formulation

\( x_b \) equals 1 if bid \( b \) is accepted, 0 if it is not

\[
\text{maximize} \quad \sum_b v_b x_b \\
\text{subject to} \\
\quad \text{for each item } j, \quad \sum_{b: j \in b} x_b \leq 1 \\
\quad \text{for each bid } b, \quad x_b \text{ in } \{0,1\}
\]
WDP in the modeling language

set ITEMS;
set BIDS;

var accepted\{j in BIDS\}, binary;

param bid_amount\{j in BIDS\};

param bid_on_object\{i in ITEMS, j in BIDS\}, binary;

maximize revenue: sum\{j in BIDS\} accepted[j]*bid_amount[j];

s.t. at_most_once\{i in ITEMS\}: sum\{j in BIDS\} accepted[j]*bid_on_object[i,j] <= 1;
Game theory

• Game theory studies settings where agents each have
  – different preferences (utility functions),
  – different actions that they can take

• Each agent’s utility (potentially) depends on all agents’ actions
  – What is optimal for one agent depends on what other agents do
    • Very circular!

• Game theory studies how agents can rationally form beliefs over what other agents will do, and (hence) how agents should act
  – Useful for acting as well as predicting behavior of others
Penalty kick example

Is this a “rational” outcome? If not, what is?
### Rock-paper-scissors

The rock-paper-scissors game is a simple example of a **zero-sum game**. The utilities in each entry sum to 0 (or a constant)

<table>
<thead>
<tr>
<th></th>
<th>Row player’s utility</th>
<th>Column player’s utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rock</td>
<td>0, 0</td>
<td>-1, 1</td>
</tr>
<tr>
<td>Paper</td>
<td>1, -1</td>
<td>0, 0</td>
</tr>
<tr>
<td>Scissors</td>
<td>-1, 1</td>
<td>1, -1</td>
</tr>
</tbody>
</table>

A row or column is called an action or (pure) strategy.

Row player’s utility is always listed first, column player’s second.

- **Zero-sum game**: the utilities in each entry sum to 0 (or a constant)
- Three-player game would be a 3D table with 3 utilities per entry, etc.

**Row player** aka. player 1 chooses a row

**Column player** aka. player 2 (simultaneously) chooses a column
Minimax strategies

• A conservative approach:
• We (Row) choose a distribution over rows
  – $p_r$ is probability on row $r$
• To evaluate quality of a distribution, pessimistically assume that Column will choose worst column for us:
  $$\arg \min_c \sum_r p_r u_R(r, c)$$
• Try to optimize for this worst case:
  $$\arg \max_{p_r} \min_c \sum_r p_r u_R(r, c)$$
• Theoretically very well-motivated in zero-sum
Solving for minimax strategies using linear programming

- maximize $u_R$
- subject to
  - for any column $c$, $\sum_r p_r u_R(r, c) \geq u_R$
  - $\sum_r p_r = 1$
Game playing & AI

**perfect information games:** no uncertainty about the state of the game (e.g. tic-tac-toe, chess, Go)

- Optimal play: value of each node = value of optimal child for current player (backward induction, minimax)
- For chess and Go, tree is too large
  - Use other techniques (heuristics, limited-depth search, alpha-beta, …)
- Top computer programs (arguably) better than humans in chess, not yet in Go

**imperfect information games:** uncertainty about the state of the game (e.g. poker)

- Player 2 cannot distinguish nodes connected by dotted lines
  - Backward induction fails; need more sophisticated game-theoretic techniques for optimal play
- Small poker variants can be solved optimally
- Humans still better than top computer programs at full-scale poker
- Top computer (heads-up) poker players are based on techniques for game theory
Solving the tiny poker game

<table>
<thead>
<tr>
<th></th>
<th>2/3</th>
<th>0</th>
<th>1/3</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cc</td>
<td>cf</td>
<td>fc</td>
<td>ff</td>
</tr>
<tr>
<td>1/3</td>
<td>bb</td>
<td>0,0</td>
<td>1, -1</td>
<td>1, -1</td>
</tr>
<tr>
<td>2/3</td>
<td>bs</td>
<td>.5,-.5</td>
<td>1.5,-1.5</td>
<td>0,0</td>
</tr>
<tr>
<td>0</td>
<td>sb</td>
<td>-.5,.5</td>
<td>-.5,.5</td>
<td>1, -1</td>
</tr>
<tr>
<td>0</td>
<td>ss</td>
<td>0,0</td>
<td>1, -1</td>
<td>0,0</td>
</tr>
</tbody>
</table>

Player 1 decisions:
- Bet
- Fold

Player 2 decisions:
- Call
- Fold

Nature outcomes:
- 1 gets King
- 1 gets Jack

Player 1's strategy: choose bet or fold based on player 2's decision and nature outcome.
Mechanism design

- **Mechanism** = rules of auction, exchange, ...
- A **function** that takes **reported preferences** (bids) as input, and produces **outcome** (allocation, payments to be made) as output

\[ f(\text{preferences}) = \text{outcome} \]

- The **entire function** \( f \) is one mechanism
- E.g.: find allocation that maximizes (reported) utilities, distribute (reported) gains evenly
- Other mechanisms choose different allocations, payments
Example: (single-item) auctions

- **Sealed-bid** auction: every bidder submits bid in a sealed envelope
- **First-price** sealed-bid auction: highest bid wins, pays amount of own bid
- **Second-price** sealed-bid auction: highest bid wins, pays amount of second-highest bid

0

<table>
<thead>
<tr>
<th>Bid</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bid 1</td>
<td>$10</td>
</tr>
<tr>
<td>Bid 2</td>
<td>$5</td>
</tr>
<tr>
<td>Bid 3</td>
<td>$1</td>
</tr>
</tbody>
</table>

- **First-price**: bid 1 wins, pays $10
- **Second-price**: bid 1 wins, pays $5
Which auction generates more revenue?

- Each bid depends on
  - bidder’s **true valuation** for the item (utility = valuation - payment),
  - bidder’s **beliefs** over what others will bid (→ game theory),
  - and... the **auction mechanism** used

- In a first-price auction, it does not make sense to bid your true valuation
  - Even if you win, your utility will be 0…

- In a second-price auction, it turns out that it always makes sense to bid your true valuation

---

Are there other auctions that perform better? How do we know when we have found the best one?
Other settings/applications
Financial securities

• Tomorrow there must be one of ☀️ ⛈️ ⚡️

• Agent 1 offers $5 for a security that pays off $10 if ⛈️ or ⚡️

• Agent 2 offers $8 for a security that pays off $10 if ☀️ or ⚡️

• Agent 3 offers $6 for a security that pays off $10 if ☀️

• Can we accept some of these at offers at no risk?
How to incentivize a weather forecaster

- Forecaster’s bonus can depend on
  - Prediction
  - Actual weather on predicted day
- Reporting true beliefs should maximize expected bonus

\[
P(\text{Sun}) = 0.5 \\
P(\text{Cloudy}) = 0.3 \\
P(\text{Rain}) = 0.2 \\
P(\text{Thunderstorm}) = 0.1 \\
P(\text{Snow}) = 0.8 \\
P(\text{Sleet}) = 0.1
\]
Kidney exchange

- Patient 1
  - Donor 1 (patient 1’s friend)

- Patient 2
  - Donor 2 (patient 2’s friend)

- Patient 3
  - Donor 3 (patient 3’s friend)

- Patient 4
  - Donor 4 (patient 4’s friend)

compatibilities
Conclusion

• Students enjoyed course
  – Overall quality rating of 4.78 out of 5
  – No complaints about missing prerequisites
  – One economics student took intro programming the next semester

• Downsides w.r.t. attracting new students:
  – Tends to attract economics students who already have some cs background anyway
  – Tends to attract juniors, seniors

Thank you for your attention!