Automated Mechanism Design

EC-08 tutorial

Vincent Conitzer &

Yevgeniy (Eugene) Vorobeychik

First half (Vince): overview

- Part I: *Mechanism design review*
- Part II: *Automated mechanism design: the basic approach*
- Part III: *Some variants and applications*

Part I: Mechanism design review

- *Preference aggregation settings*
- *Mechanisms*
- *Solution concepts*
- *Revelation principle*
- *Vickrey-Clarke-Groves mechanism(s)*
- *Impossibility results*

Introduction

- Often, decisions must be taken based on the preferences of multiple, self-interested agents
  - Allocations of resources/tasks
  - Joint plans
  - …
- Would like to make decisions that are “good” with respect to the agents’ preferences
- But, agents may lie about their preferences if this is to their benefit
- Mechanism design = creating rules for choosing the outcome that get good results nevertheless

Preference aggregation settings

- Multiple agents…
  - humans, computer programs, institutions, …
- … must decide on one of multiple outcomes…
  - joint plan, allocation of tasks, allocation of resources, president, …
- … based on agents' preferences over the outcomes
  - Each agent knows only its own preferences
  - "Preferences" can be an ordering $\geq_i$ over the outcomes, or a real-valued utility function u_i
  - Often preferences are assumed to be drawn from a commonly known distribution

Elections

Outcome space = {  ,  ,  }

Resource allocation 

Outcome space = {  ,  ,  }

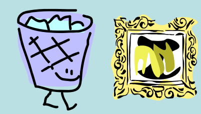
v() = $55

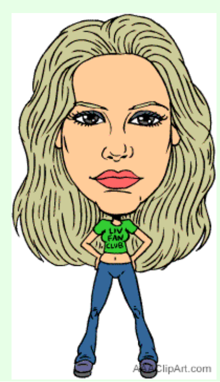
v() = $0

v() = $0

v() = $0

v() = $32

v() = $0

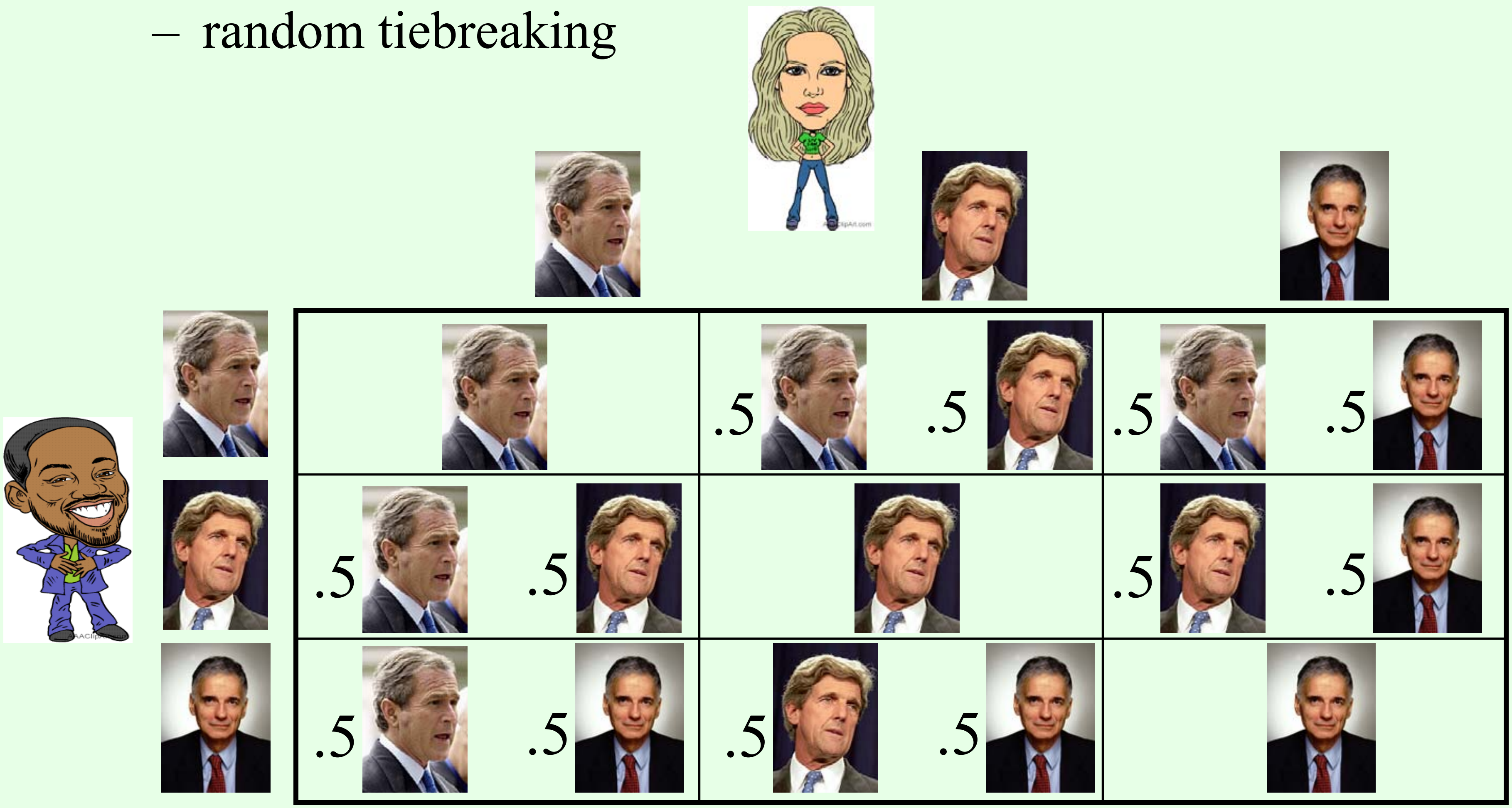


So, what is a mechanism?

- A mechanism prescribes:
  - actions that the agents can take (based on their preferences)
  - a mapping that takes all agents' actions as input, and outputs the chosen outcome
    - the "rules of the game"
    - can also output a probability distribution over outcomes
- Direct revelation mechanisms are mechanisms in which action set = set of possible preferences

Example: plurality voting

- Every agent votes for one alternative
- Alternative with most votes wins
  - random tiebreaking



Some other well-known voting mechanisms

- In all of these rules, each voter ranks all m candidates (direct revelation mechanisms)
- Other scoring mechanisms
  - Borda: candidate gets m-1 points for being ranked first, m-2 for being ranked second, …
  - Veto: candidate gets 0 points for being ranked last, 1 otherwise
- Pairwise election between two candidates: see which candidate is ranked above the other more often
  - Copeland: candidate with most pairwise victories wins
  - Maximin: compare candidates by their worst pairwise elections
  - Slater: choose overall ranking disagreeing with as few pairwise elections as possible
- Other
  - Single Transferable Vote (STV): candidate with fewest votes drops out, those votes transfer to next remaining candidate in ranking, repeat
  - Kemeny: choose overall ranking that minimizes the number of disagreements with some vote on some pair of candidates

The "matching pennies" mechanism 

- Winner of "matching pennies" gets to choose outcome

Mechanisms with payments

- In some settings (e.g. auctions), it is possible to make payments to/collect payments from the agents
- Quasilinear utility functions: $u_i(o, \pi_i) = v_i(o) + \pi_i$
- We can use this to modify agents' incentives

A few different 1-item auction mechanisms

- English auction:
  - Each bid must be higher than previous bid
  - Last bidder wins, pays last bid
- Japanese auction:
  - Price rises, bidders drop out when price is too high
  - Last bidder wins at price of last dropout
- Dutch auction:
  - Price drops until someone takes the item at that price
- Sealed-bid auctions (direct revelation mechanisms):
  - Each bidder submits a bid in an envelope
  - Auctioneer opens the envelopes, highest bid wins
    - First-price sealed-bid auction: winner pays own bid
    - Second-price sealed bid (or Vickrey) auction: winner pays second highest bid

What can we expect to happen?

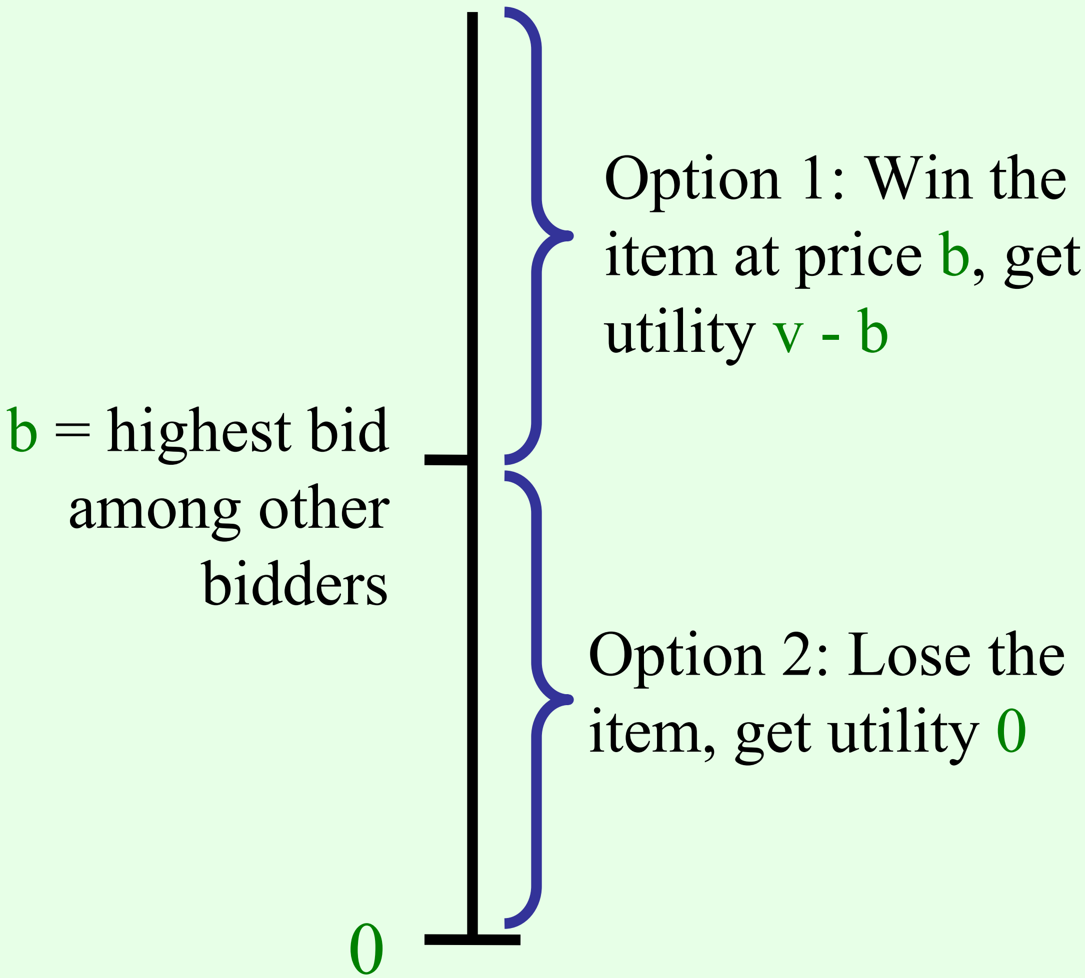
- In direct revelation mechanisms, will (selfish) agents tell the truth about their preferences?
  - Voter may not want to "waste" vote on poorly performing candidate (e.g. Nader)
  - In first-price sealed-bid auction, winner would like to bid only ε above the second highest bid
- In other mechanisms, things get even more complicated…

A little bit of game theory

- Θ_i = set of all of agent i's possible preferences ("types")
  - Notation: $u_i(\theta_i, o)$ is i's utility for o when i has type θ_i
- A strategy s_i is a mapping from types to actions
  - $s_i: \Theta_i \rightarrow A_i$
  - For direct revelation mechanism, $s_i: \Theta_i \rightarrow \Theta_i$
  - More generally, can map to distributions, $s_i: \Theta_i \rightarrow \Delta(A_i)$
- A strategy s_i is a dominant strategy if for every type θ_i, *no matter what the other agents do*, $s_i(\theta_i)$ maximizes i's utility
- A direct revelation mechanism is strategy-proof (or dominant-strategies incentive compatible) if telling the truth ($s_i(\theta_i) = \theta_i$) is a dominant strategy for all players
- (Another, weaker concept: Bayes-Nash equilibrium)

The Vickrey auction is strategy-proof!

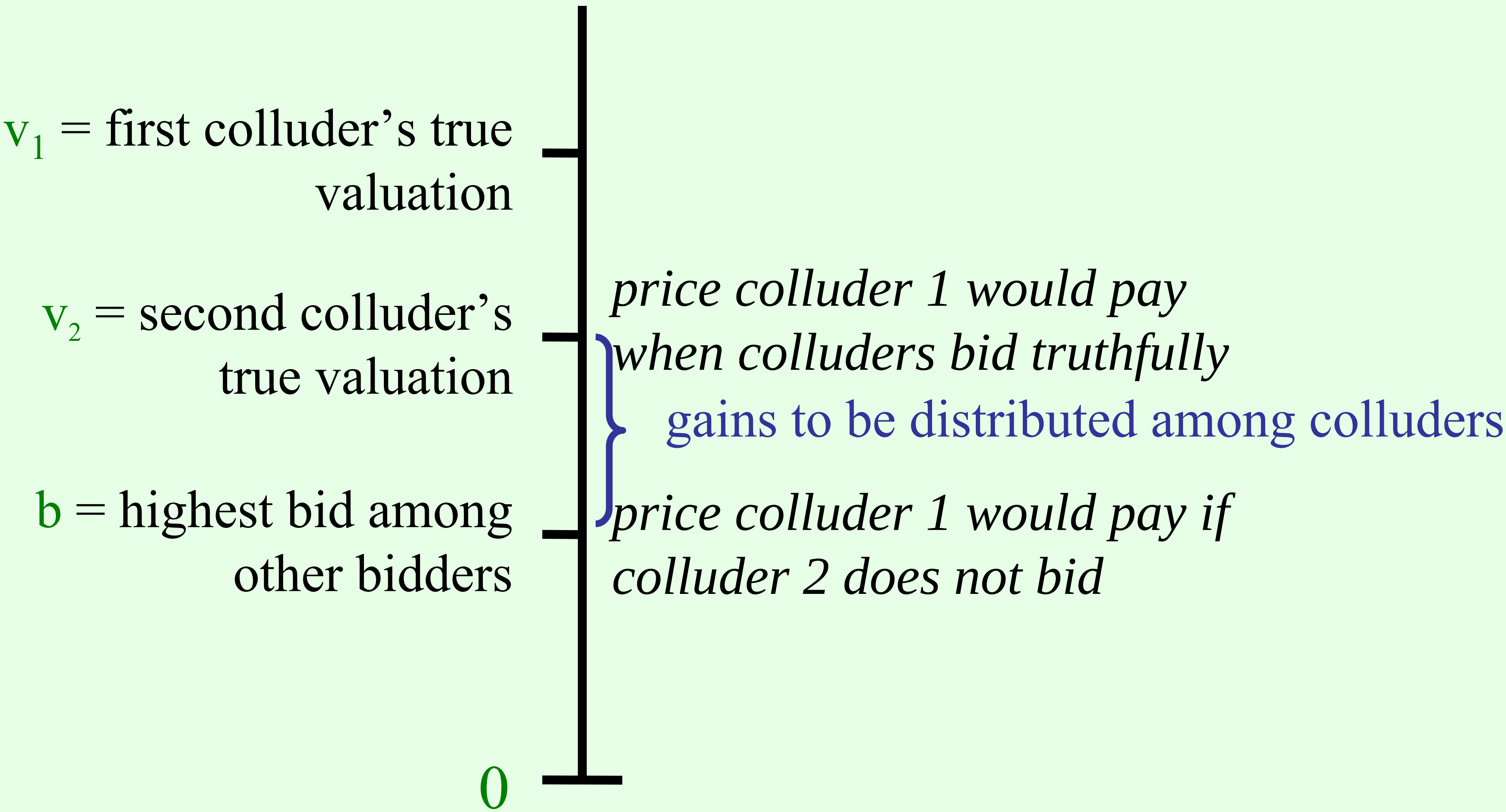
- What should a bidder with value v bid?



Would like to win if and only if v - b > 0 – but bidding truthfully accomplishes this!

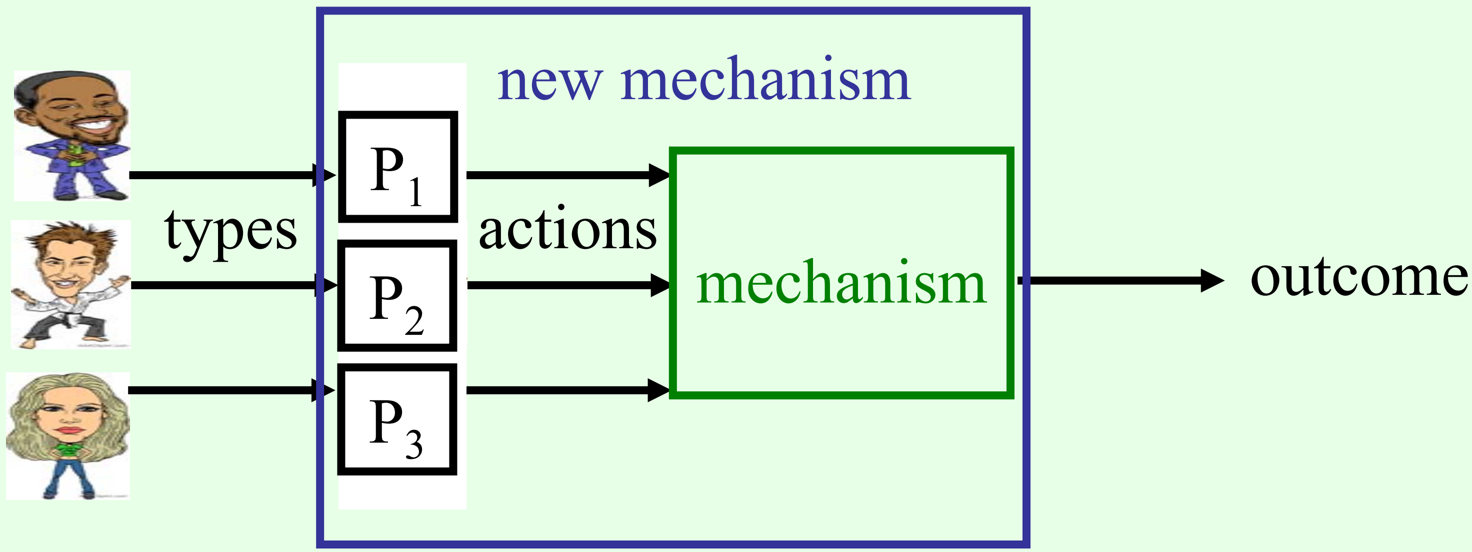
Collusion in the Vickrey auction

- Example: two colluding bidders



The revelation principle

- For any (complex, strange) mechanism that produces certain outcomes under strategic behavior…
- … there exists an incentive compatible direct revelation mechanism that produces the same outcomes!
  - "strategic behavior" = some solution concept (e.g. dominant strategies)



The Clarke mechanism [Clarke 71]

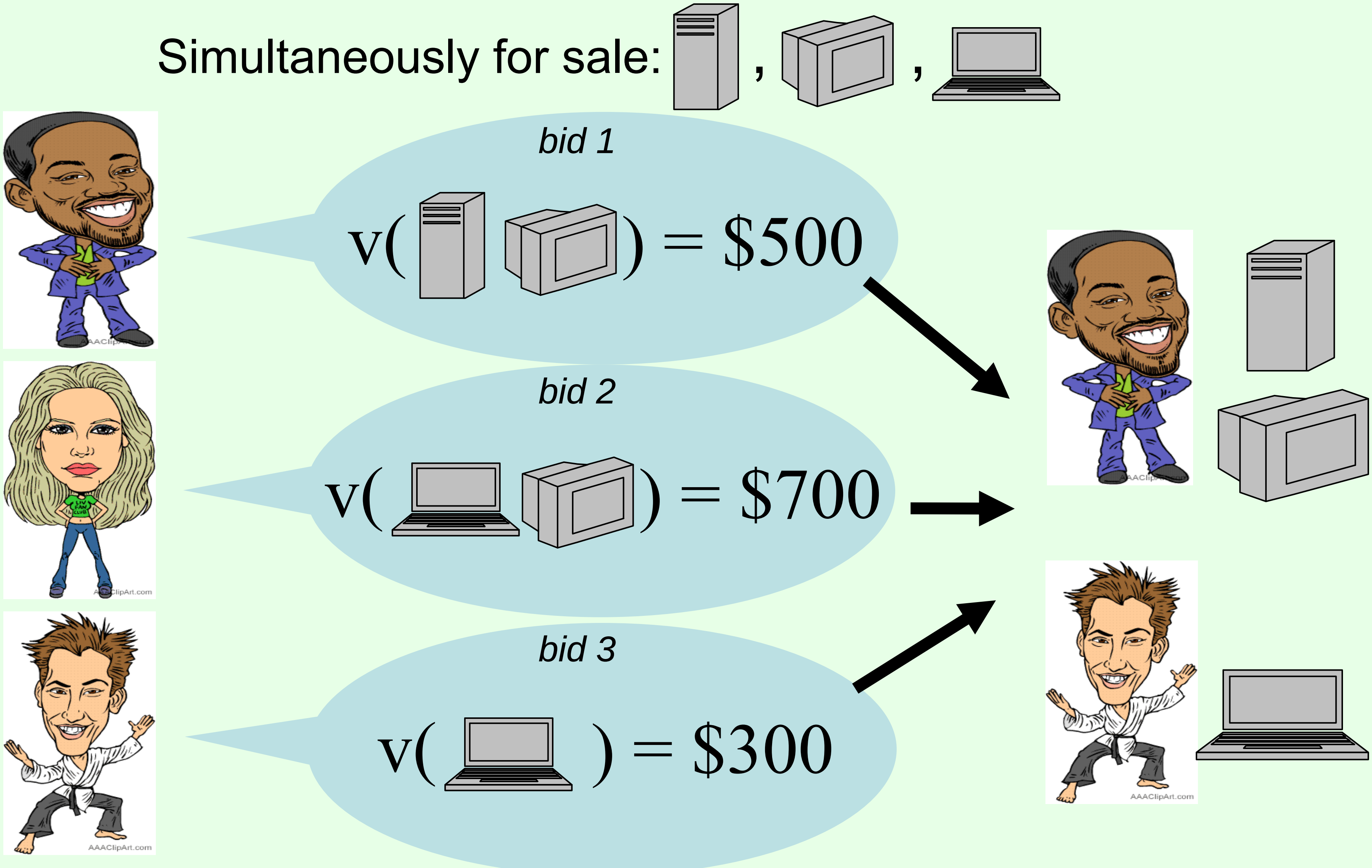
- Generalization of the Vickrey auction to arbitrary preference aggregation settings
- Agents reveal types directly
  - θ_i' is the type that i reports, θ_i is the actual type
- Clarke mechanism chooses some outcome o that maximizes $\Sigma_i u_i(\theta_i', o)$
- To determine the payment that agent j must make:
  - Choose o' that maximizes $\Sigma_{i\neq j} u_i(\theta_i', o')$
  - Make j pay $\Sigma_{i\neq j} (u_i(\theta_i', o') - u_i(\theta_i', o))$
- Clarke mechanism is:
  - individually rational: no agent pays more than the outcome is worth to that agent
  - (weak) budget balanced: agents pay a nonnegative amount

Why is the Clarke mechanism strategy-proof?

- Total utility for agent j is
  $u_j(\theta_j, o) - \Sigma_{i \neq j} (u_i(\theta_i', o') - u_i(\theta_i', o)) =$
  $u_j(\theta_j, o) + \Sigma_{i \neq j} u_i(\theta_i', o) - \Sigma_{i \neq j} u_i(\theta_i', o')$
- But agent j cannot affect the choice of o'
- Hence, j can focus on maximizing $u_j(\theta_j, o) + \Sigma_{i \neq j} u_i(\theta_i', o)$
- But mechanism chooses o to maximize $\Sigma_i u_i(\theta_i', o)$
- Hence, if $\theta_j' = \theta_j$, j's utility will be maximized!

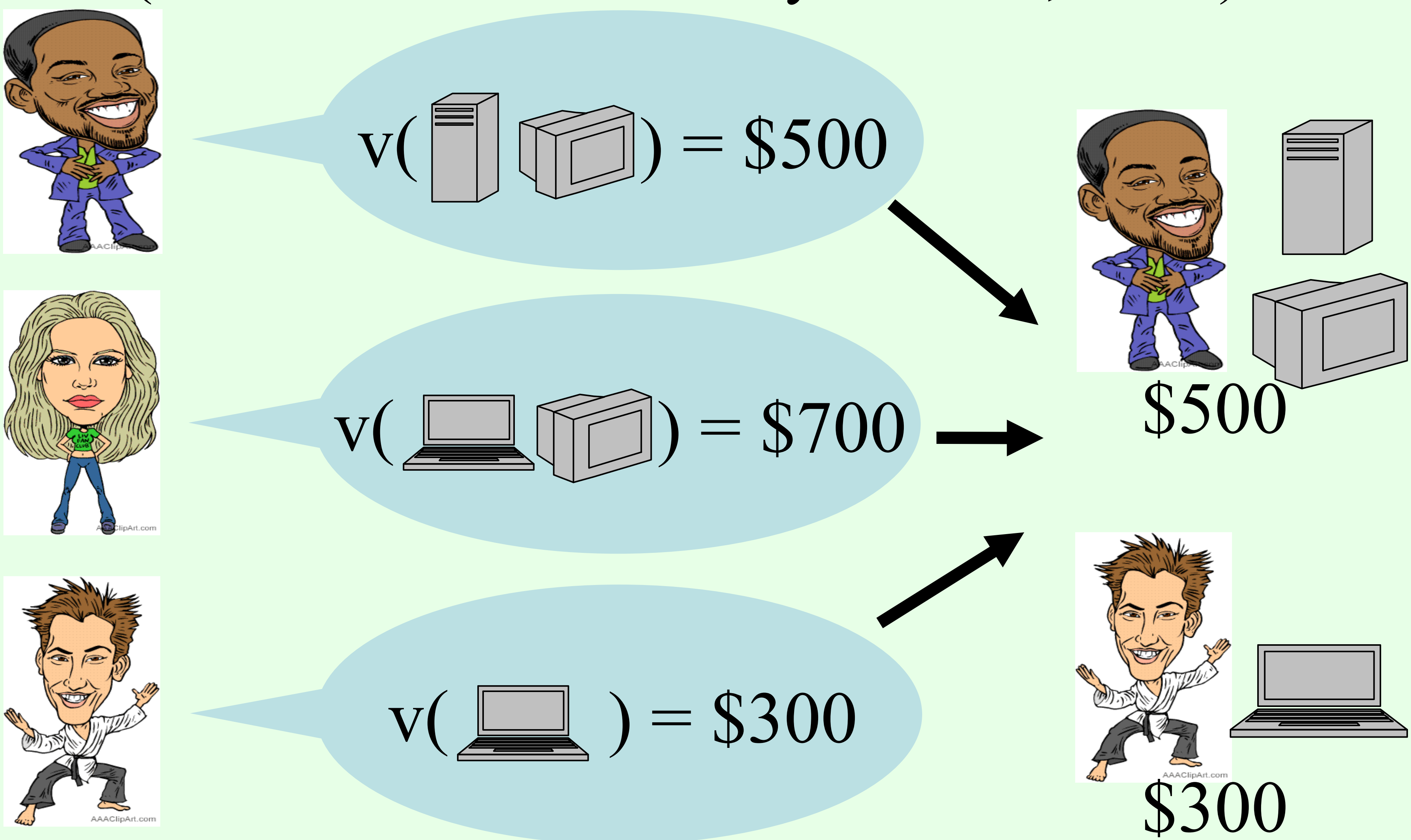
- Extension of idea: add any term to player j's payment that does not depend on j's reported type
- This is the family of Groves mechanisms [Groves 73]
- "The VCG mechanism" usually refers to Clarke, "VCG mechanisms" usually refers to Groves

Combinatorial auctions

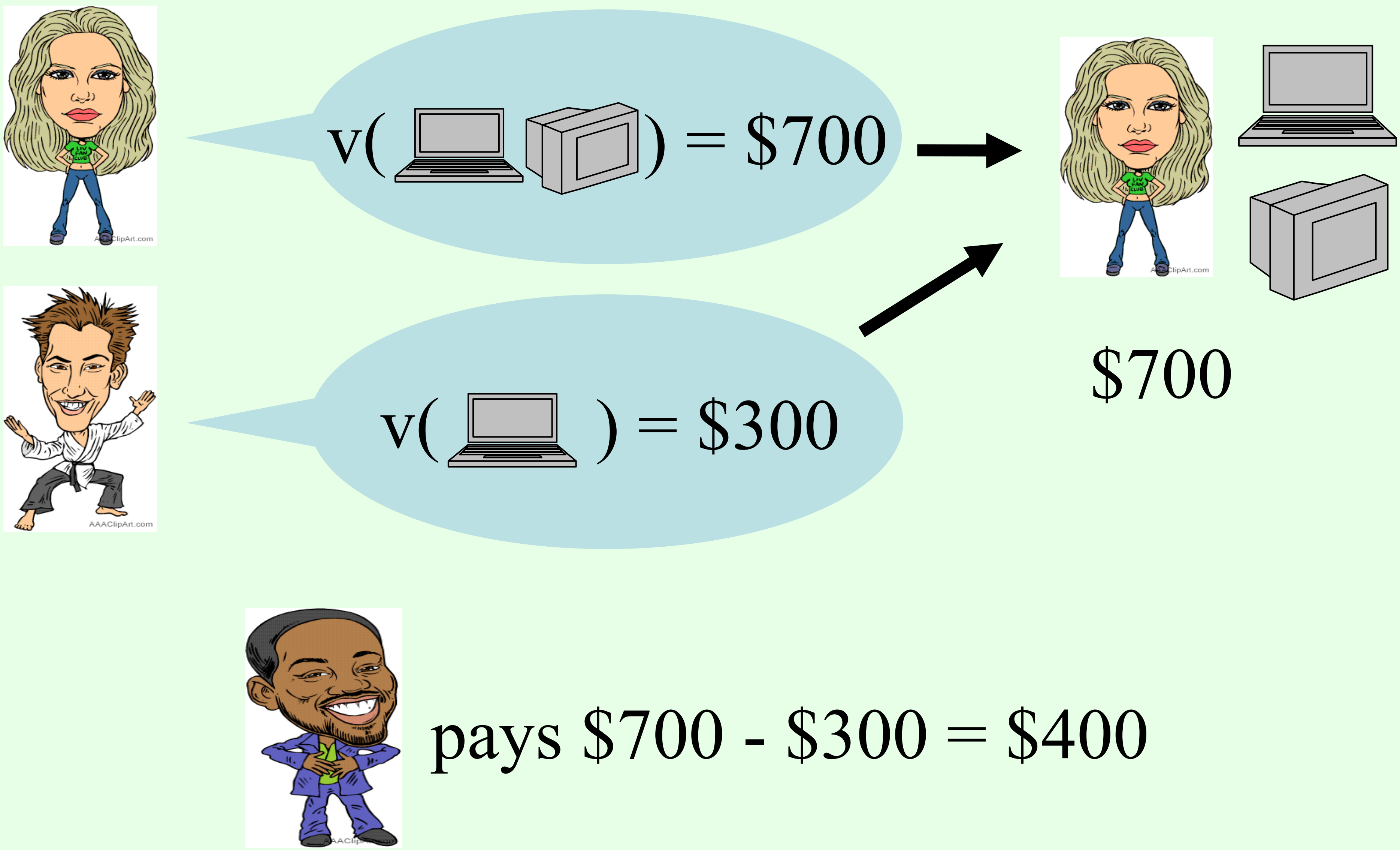


used in truckload transportation, industrial procurement, radio spectrum allocation, …

Clarke mechanism in CA
(aka. Generalized Vickrey Auction, GVA)



Clarke mechanism in CA…



The Clarke mechanism is not perfect

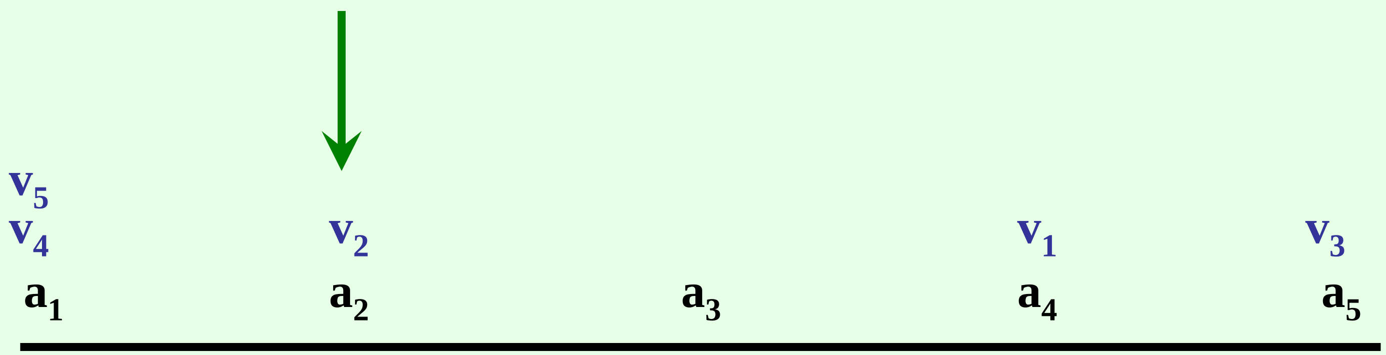
- Requires payments + quasilinear utility functions
- In general money needs to flow away from the system
- Vulnerable to collusion, false-name manipulation
- Maximizes sum of agents' utilities (not counting payments), but sometimes we are not interested in this
  - E.g. want to maximize revenue

Impossibility results without payments

- Can we do without payments (voting mechanisms)?
- Gibbard-Satterthwaite [Gibbard 73, Satterthwaite 75] impossibility result: with three or more alternatives and unrestricted preferences, no voting mechanism exists that is
  - deterministic
  - strategy-proof
  - non-imposing (every alternative can win)
  - non-dictatorial (more than one voter can affect the outcome)
- Generalization [Gibbard 77]: a randomized voting rule is strategy-proof only if it is a randomization over unilateral and duple rules
  - unilateral = at most one voter affects the outcome
  - duple = at most two alternatives have a possibility of winning

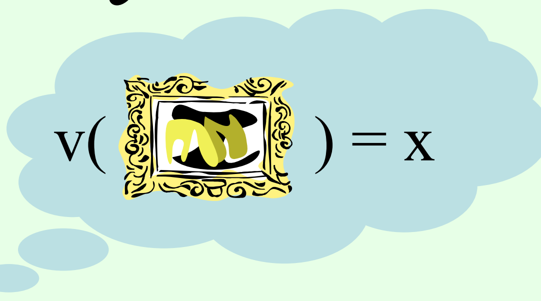
Single-peaked preferences [Black 48]

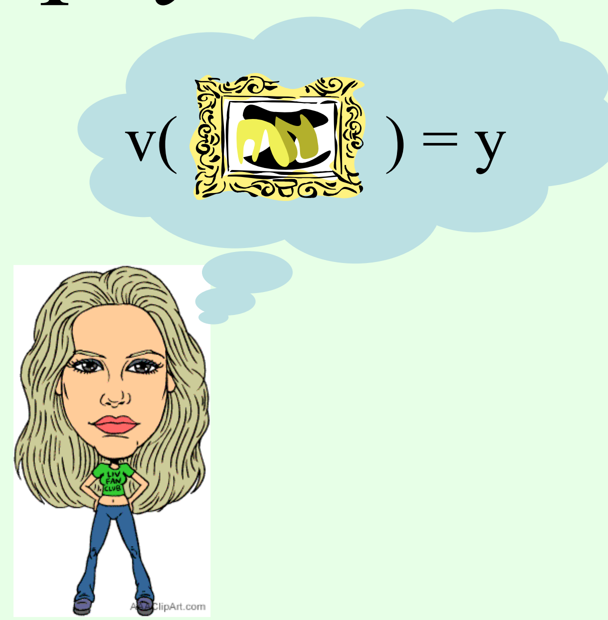
- Suppose alternatives are ordered on a line
- Every voter prefers alternatives that are closer to her most preferred alternative
- Let every voter report only her most preferred alternative ("peak")
- Choose the median voter's peak as the winner
- Strategy-proof!



Impossibility result with payments

- Simple setting:





- We would like a mechanism that:
  - is efficient (trade iff y > x)
  - is budget-balanced (seller receives what buyer pays)
  - is strategy-proof (or even weaker forms of incentive compatible)
  - is individually rational (even just in expectation)
- This is impossible! [Myerson & Satterthwaite 83]

Part II: Automated mechanism design: the basic approach

- *General vs. specific mechanisms*
- *Motivation*
- *Examples*
- *Linear/integer programming approaches*
- *Computational complexity*
- *More examples*

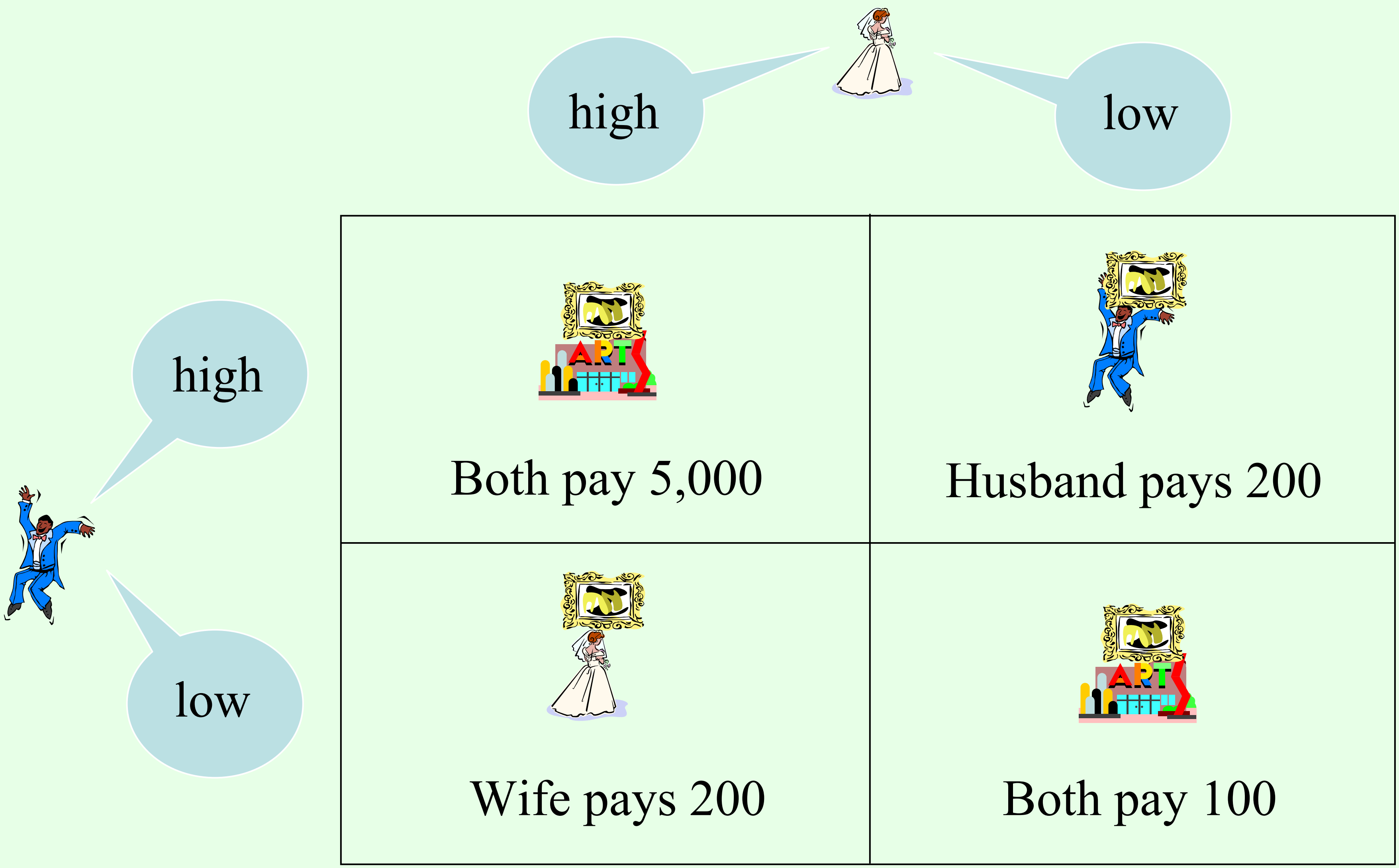
General vs. specific mechanisms

- Mechanisms such as Clarke (VCG) mechanism are very general…
- … but will instantiate to something specific in any specific setting
  - This is what we care about

Example: Divorce arbitration

- Outcomes:    
- Each agent is of *high* type w.p. .2 and *low* type w.p. .8
  - Preferences of *high* type:
    - u(get the painting) = 11,000
    - u(museum) = 6,000
    - u(other gets the painting) = 1,000
    - u(burn) = 0
  - Preferences of *low* type:
    - u(get the painting) = 1,200
    - u(museum) = 1,100
    - u(other gets the painting) = 1,000
    - u(burn) = 0

Clarke (VCG) mechanism



Expected sum of divorcees’ utilities = 5,136

"Manual" mechanism design has yielded

- some positive results:
  - "Mechanism x achieves properties P in any setting that belongs to class C"
- some impossibility results:
  - "There is no mechanism that achieves properties P for all settings in class C"

Difficulties with manual mechanism design

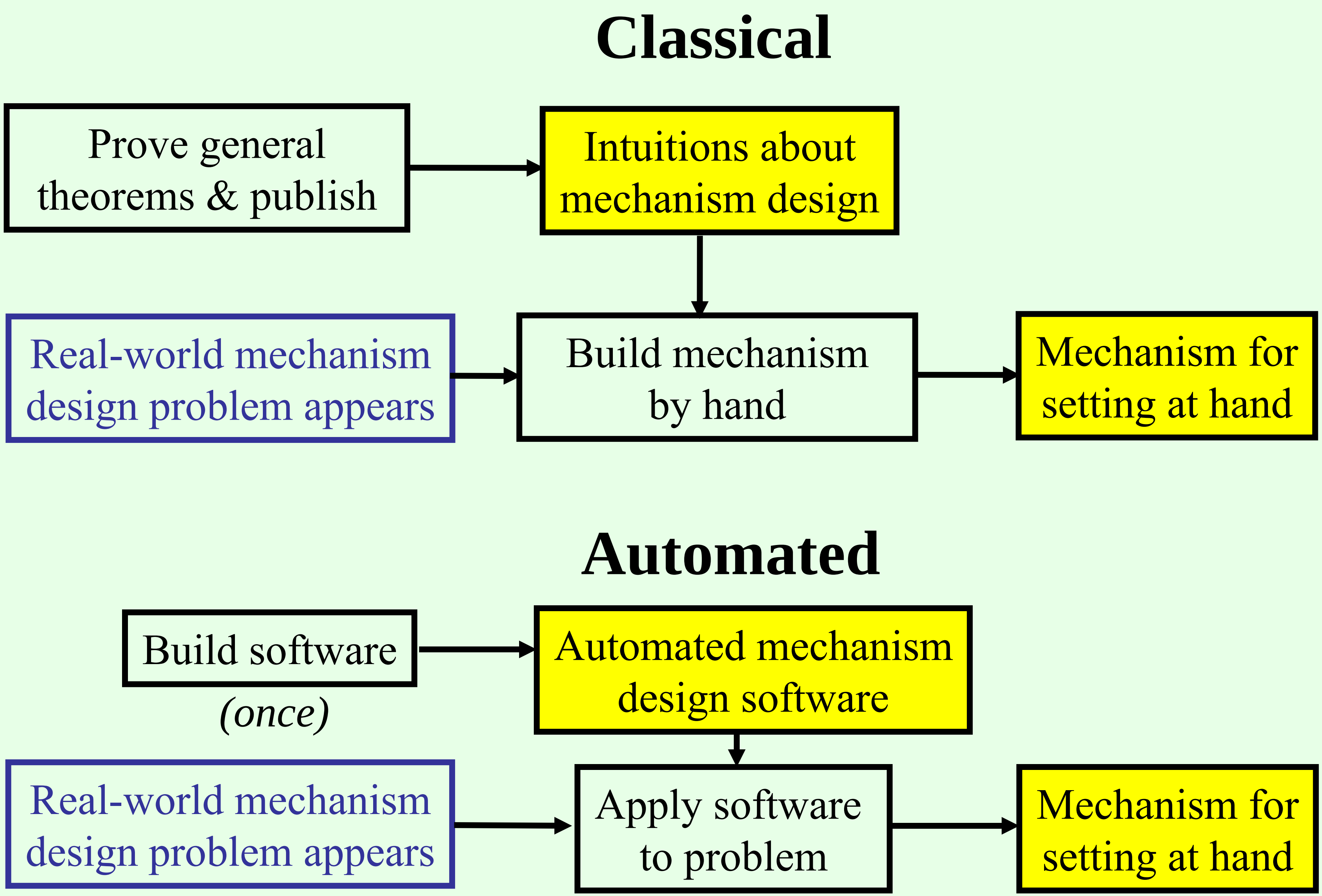
- Design problem instance comes along
  - Set of outcomes, agents, set of possible types for each agent, prior over types, …
- What if no canonical mechanism covers this instance?
  - Unusual objective, or payments not possible, or …
  - Impossibility results may exist for the general class of settings
    - But instance may have additional structure (restricted preferences or prior) so good mechanisms exist (but unknown)
- What if a canonical mechanism does cover the setting?
  - Can we use instance's structure to get higher objective value?
  - Can we get stronger nonmanipulability/participation properties?
- Manual design for every instance is prohibitively slow

Automated mechanism design (AMD)

[C. & Sandholm UAI-02, later papers; for overview, see either Sandholm CP-03 overview or (more up to date) Chapter 6 of C.'s thesis (2006)]

- Idea: Solve mechanism design as optimization problem automatically
- Create a mechanism for the specific setting at hand rather than a class of settings
- Advantages:
  - Can lead to greater value of designer's objective than known mechanisms
  - Sometimes circumvents economic impossibility results & always minimizes the pain implied by them
  - Can be used in new settings & for unusual objectives
  - Can yield stronger incentive compatibility & participation properties
  - Shifts the burden of design from human to machine

Classical vs. automated mechanism design



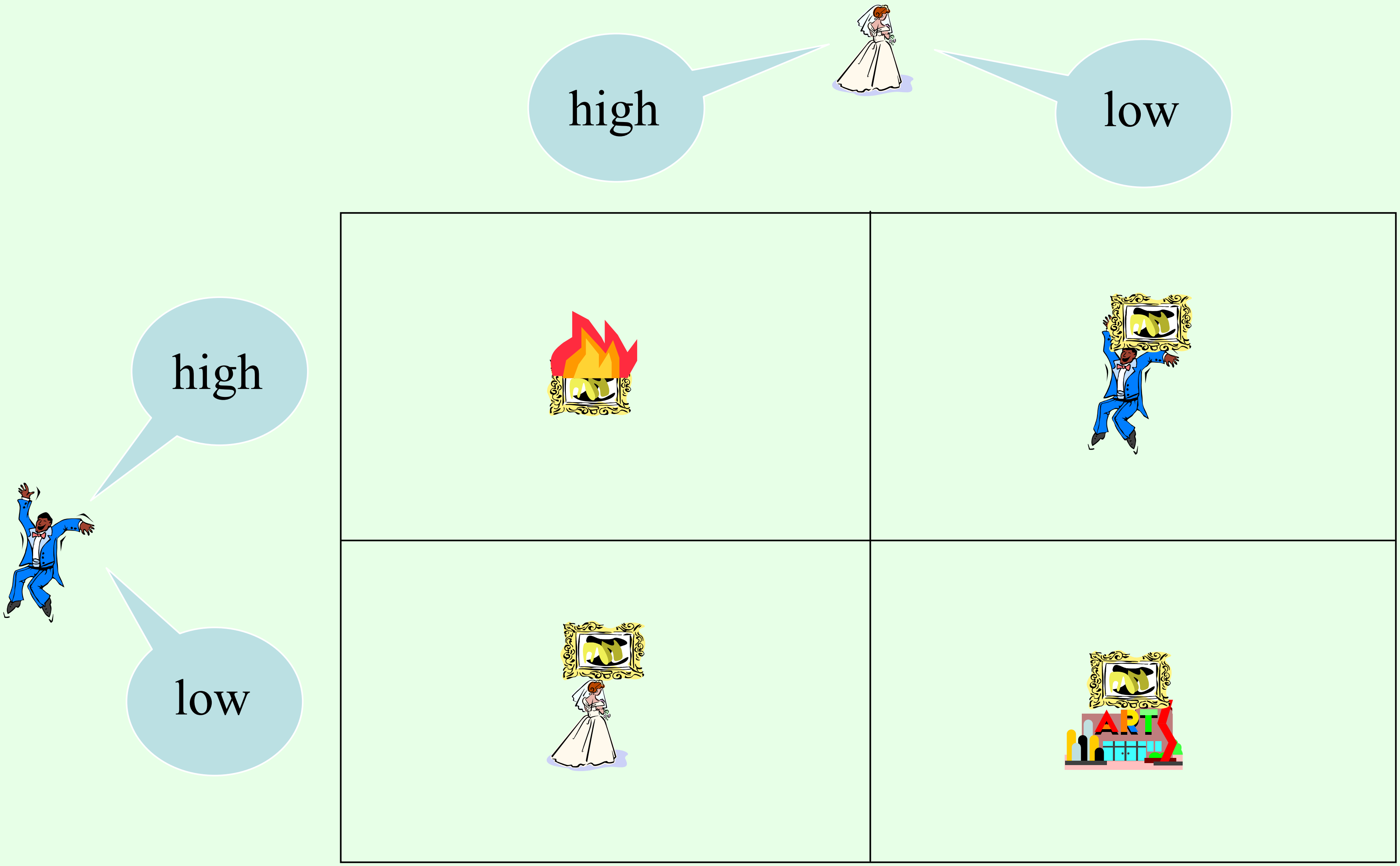
Input

- Instance is given by
  - Set of possible *outcomes*
  - Set of *agents*
    - For each agent
      - set of possible *types*
      - *probability distribution* over these types
  - *Objective function*
    - Gives a value for each outcome for each combination of agents' types
    - E.g. social welfare, payment maximization
  - Restrictions on the mechanism
    - Are payments allowed?
    - Is randomization over outcomes allowed?
    - What versions of incentive compatibility (IC) & individual rationality (IR) are used?

Output

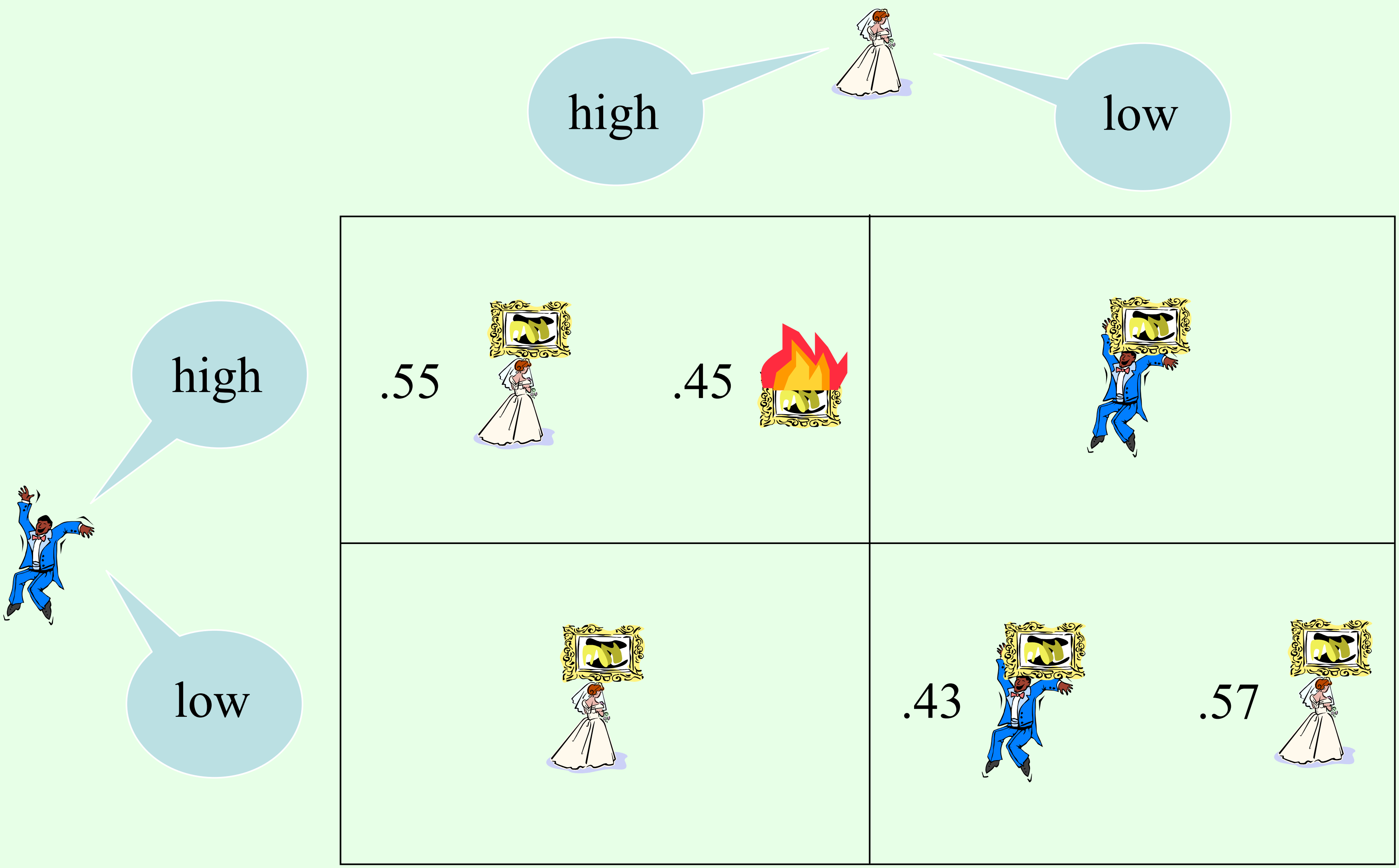
- *Mechanism*
  - A mechanism maps combinations of agents' revealed types to outcomes
    - *Randomized mechanism* maps to probability distributions over outcomes
    - Also specifies payments by agents (if payments allowed)
- *…* which
  - satisfies the IR and IC constraints
  - maximizes the expectation of the objective function

Optimal BNE incentive compatible deterministic mechanism without payments for maximizing sum of divorcees’ utilities



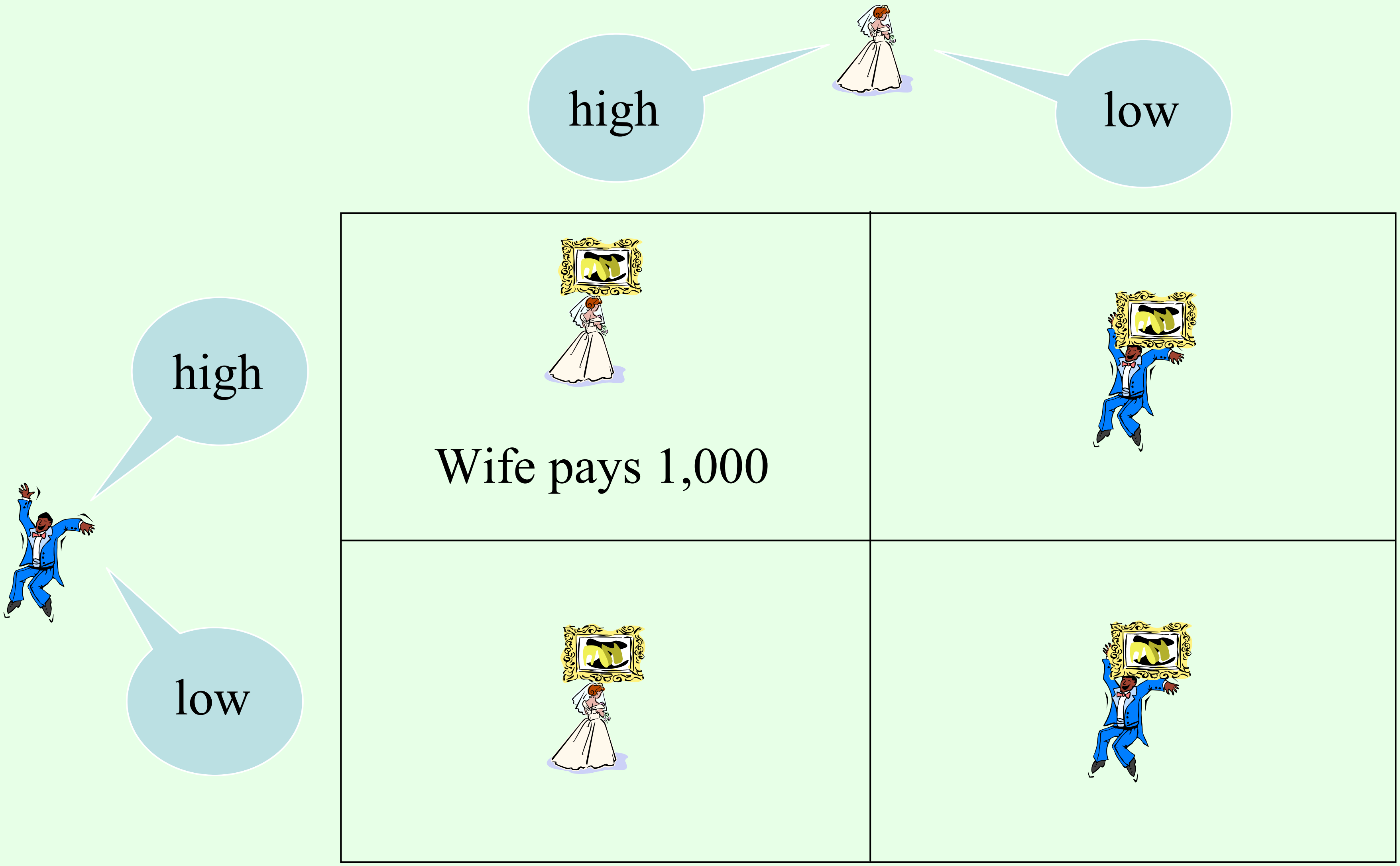
Expected sum of divorcees’ utilities = 5,248

Optimal BNE incentive compatible *randomized* mechanism without payments for maximizing sum of divorcees’ utilities



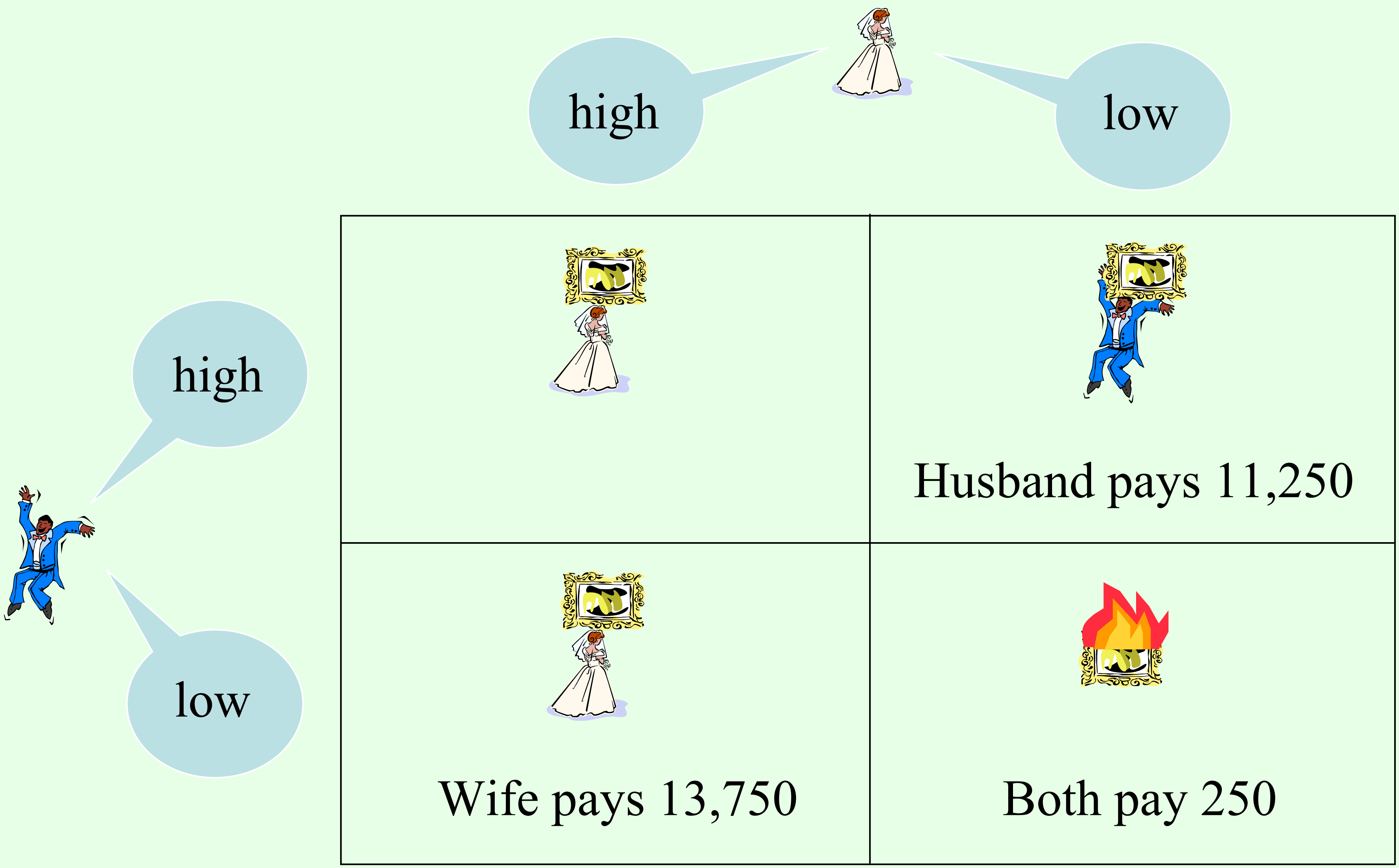
Expected sum of divorcees’ utilities = 5,510

Optimal BNE incentive compatible randomized mechanism *with payments* for maximizing sum of divorcees’ utilities



Expected sum of divorcees’ utilities = 5,688

Optimal BNE incentive compatible randomized mechanism with payments for *maximizing arbitrator’s revenue*

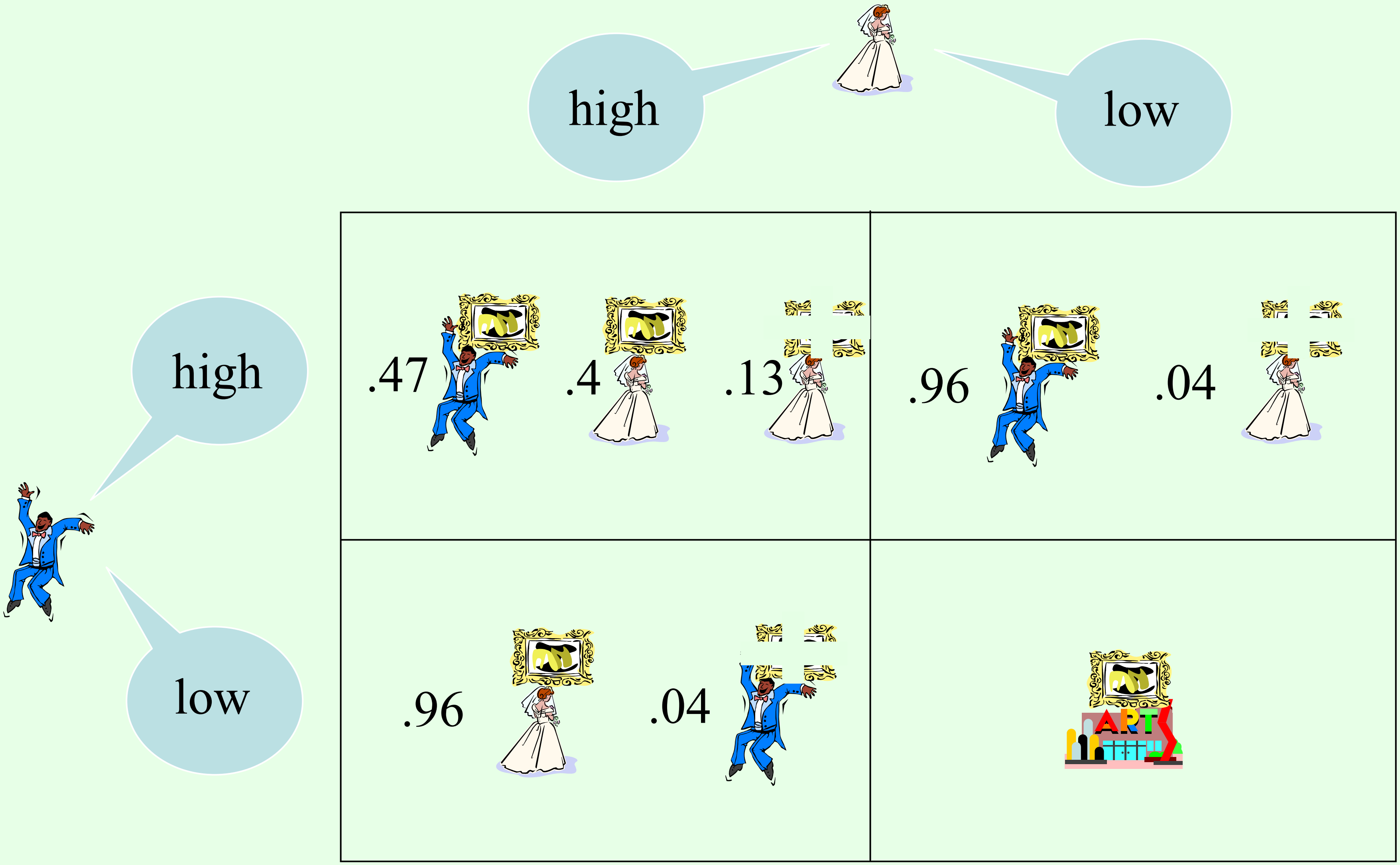


Expected sum of divorcees’ utilities = 0   Arbitrator expects 4,320

Modified divorce arbitration example

- Outcomes: 
- Each agent is of *high* type with probability 0.2 and of *low* type with probability 0.8
  - Preferences of *high* type:
    - u(get the painting) = 100
    - u(other gets the painting) = 0
    - u(museum) = 40
    - u(get the pieces) = -9
    - u(other gets the pieces) = -10
  - Preferences of *low* type:
    - u(get the painting) = 2
    - u(other gets the painting) = 0
    - u(museum) = 1.5
    - u(get the pieces) = -9
    - u(other gets the pieces) = -10

Optimal *dominant-strategies* incentive compatible randomized mechanism for maximizing expected sum of utilities



How do we set up the optimization?

- Use linear programming
- Variables:
  - $p(o \mid \theta_1, \ldots, \theta_n)$ = probability that outcome o is chosen given types $\theta_1, \ldots, \theta_n$
  - (maybe) $\pi_i(\theta_1, \ldots, \theta_n)$ = i's payment given types $\theta_1, \ldots, \theta_n$
- Strategy-proofness constraints: for all $i, \theta_1, \ldots \theta_n, \theta_i'$:

  $\Sigma_o p(o \mid \theta_1, \ldots, \theta_n) u_i(\theta_i, o) + \pi_i(\theta_1, \ldots, \theta_n) \geq$

  $\Sigma_o p(o \mid \theta_1, \ldots, \theta_i', \ldots, \theta_n) u_i(\theta_i, o) + \pi_i(\theta_1, \ldots, \theta_i', \ldots, \theta_n)$
- Individual-rationality constraints: for all $i, \theta_1, \ldots \theta_n$:

  $\Sigma_o p(o \mid \theta_1, \ldots, \theta_n) u_i(\theta_i, o) + \pi_i(\theta_1, \ldots, \theta_n) \geq 0$
- Objective (e.g. sum of utilities)

  $\Sigma_{\theta_1, \ldots, \theta_n} p(\theta_1, \ldots, \theta_n) \Sigma_i (\Sigma_o p(o \mid \theta_1, \ldots, \theta_n) u_i(\theta_i, o) + \pi_i(\theta_1, \ldots, \theta_n))$
- Also works for BNE incentive compatibility, ex-interim individual rationality notions, other objectives, etc.
- For deterministic mechanisms, use mixed integer programming (probabilities in $\{0, 1\}$)
  - Typically designing the optimal deterministic mechanism is NP-hard

Computational complexity of automatically designing deterministic mechanisms

- Many different variants
  - Objective to maximize: Social welfare/revenue/designer's agenda for outcome
  - Payments allowed/not allowed
  - IR constraint: ex interim IR/ex post IR/no IR
  - IC constraint: Dominant strategies/Bayes-Nash equilibrium
- The above already gives 3 * 2 * 3 * 2 = 36 variants
- Approach: Prove hardness for the case of only 1 type-reporting agent
  - results imply hardness in more general settings

DSE & BNE incentive compatibility constraints coincide when there is only 1 (reporting) agent

Dominant strategies:

Reporting truthfully is optimal for *any* types the others report

	t_{21}	t_{22}
t_{11}	o_5	o_9
t_{12}	o_3	o_2

$u_1(t_{11},o_5) \geq u_1(t_{11},o_3)$

AND

$u_1(t_{11},o_9) \geq u_1(t_{11},o_2)$

Bayes-Nash equilibrium:

Reporting truthfully is optimal *in expectation* over the other agents' (true) types

	t_{21}	t_{22}
t_{11}	o_5	o_9
t_{12}	o_3	o_2

$P(t_{21})u_1(t_{11},o_5) + P(t_{22})u_1(t_{11},o_9) \geq P(t_{21})u_1(t_{11},o_3) + P(t_{22})u_1(t_{11},o_2)$

With only 1 reporting agent, the constraints are the same

	t_{21}
t_{11}	o_5
t_{11}	o_3

$u_1(t_{11},o_5) \geq u_1(t_{11},o_3)$

is equivalent to

$P(t_{21})u_1(t_{11},o_5) \geq P(t_{21})u_1(t_{11},o_3)$

Ex post and *ex interim* individual rationality constraints coincide when there is only 1 (reporting) agent

Ex post:

Participating never hurts (for *any* types of the other agents)

	t_{21}	t_{22}
t_{11}	o_5	o_9
t_{12}	o_3	o_2

$u_1(t_{11},o_5) \geq 0$

AND

$u_1(t_{11},o_9) \geq 0$

Ex interim:

Participating does not hurt *in expectation* over the other agents' (true) types

	t_{21}	t_{22}
t_{11}	o_5	o_9
t_{12}	o_3	o_2

$P(t_{21})u_1(t_{11},o_5) +$

$P(t_{22})u_1(t_{11},o_9) \geq 0$

With only 1 reporting agent, the constraints are the same

	t_{21}
t_{11}	o_5
t_{11}	o_3

$u_1(t_{11},o_5) \geq 0$

is equivalent to

$P(t_{21})u_1(t_{11},o_5) \geq 0$

How hard is designing an optimal *deterministic* mechanism?

[C. and Sandholm UAI02, ICEC03, EC04]

NP-hard (even with 1 reporting agent):	**Solvable in polynomial time (for any constant number of agents):**
1. Maximizing social welfare (no payments) 2. Designer's own utility over outcomes (no payments) 3. General (linear) objective that doesn't regard payments 4. Expected revenue	1. Maximizing social welfare (not regarding the payments) (VCG)

1 and 3 hold even with no IR constraints

AMD can create small optimal (expected-revenue maximizing) combinatorial auctions

- Instance 1
  - 2 items, 2 bidders, 4 types each (LL, LH, HL, HH)
  - H=utility 2 for that item, L=utility 1
  - But: utility 6 for getting both items if type HH (complementarity)
  - Uniform prior over types
  - Optimal *ex-interim* IR, BNE mechanism (0 = item is burned):
  - Payment rule not shown
  - Expected revenue: 3.94 (VCG: 2.69)
- Instance 2
  - 2 items, 3 bidders
  - Complementarity and substitutability
  - Took 5.9 seconds
  - Uses randomization

	LL	LH	HL	HH
LL	0,0	0,2	2,0	2,2
LH	0,1	1,2	2,1	2,2
HL	1,0	1,2	2,1	2,2
HH	1,1	1,1	1,1	1,1

Optimal mechanisms for a public good

- AMD can design optimal mechanisms for public goods, taking money burning into account as a loss
- Bridge building instance
  - Agent 1: High type (prob .6) values bridge at 10. Low: values at 1
  - Agent 2: High type (prob .4) values bridge at 11. Low: values at 2
  - Bridge costs 6 to build
- Optimal mechanism (*ex-post* IR, BNE):

Outcome rule

	Low	High
Low	Don't build	Build
High	Build	Build

Payment rule

	Low	High
Low	0, 0	0, 6
High	4, 2	.67, 5.33

- There is no general mechanism that achieves budget balance, *ex-post* efficiency, and *ex-post* IR
- However, for this instance, AMD found such a mechanism

Combinatorial public goods problems

- AMD for interrelated public goods
- Example: building a bridge and/or a boat
  - 2 agents each uniform from types: {None, Bridge, Boat, Either}
    - Type indicates which of the two would be useful to the agent
    - If something is built that is useful to you, you get 2, otherwise 0
  - Boat costs 1 to build, bridge 3
- Optimal mechanism (*ex-post* IR, dominant strategies):

Outcome rule

(P(none), P(boat), P(bridge), P(both))

	None	Boat	Bridge	Either
None	(1,0,0,0)	(0,1,0,0)	(1,0,0,0)	(0,1,0,0)
Boat	(.5,.5,0,0)	(0,1,0,0)	(0,.5,0,.5)	(0,1,0,0)
Bridge	(1,0,0,0)	(0,1,0,0)	(0,0,1,0)	(0,0,1,0)
Either	(.5,.5,0,0)	(0,1,0,0)	(0,0,1,0)	(0,1,0,0)

- Again, no money burning, but outcome not always efficient
  - E.g., sometimes nothing is built while boat should have been

Additional & future directions

- Scalability is a major concern
  - Can sometimes create more concise LP formulations
    - Sometimes, some constraints are implied by others
  - In restricted domains faster algorithms sometimes exist
    - Can sometimes make use of partial characterizations of the optimal mechanism (e.g. [C. and Sandholm AAMAS04])
  - More heuristic approaches (e.g. [C. and Sandholm IJCAI07])
- Automatically generated mechanisms can be complex/hard to understand
  - Can we make automatically designed mechanisms more intuitive? Do we need to?
- Settings where communicating entire type (preferences) is undesirable
  - AMD with partial types [Hyafil & Boutilier IJCAI07, AAAI07, Hyafil thesis proposal]
  - AMD for multistage mechanisms [Sandholm, C., Boutilier IJCAI07]
- Using AMD to create conjectures about general mechanisms

Part III: Some variants and applications

(AMD as a "philosophy")

- *Truthful feedback mechanisms with minimal payments*
- *Auctions with revenue redistribution*

Designing truthful feedback mechanisms [Jurca & Faltings EC06]

- Say we have a buyer of a product; would like her to give feedback
- Quality of her experience is represented by signal s_j
- She submits a signal s_h as her feedback
- If she reports truthfully, her signal is likely to match/be close to a “reference” reviewer’s feedback s_k
  - Assume reference reviewer reports truthfully (equilibrium reasoning)
- We pay the reviewer $\tau(s_h, s_k)$

Linear program for designing truthful feedback mechanism with minimal expected payment [Jurca & Faltings EC06]

- $\Delta(s_j, s_h)$ is the maximum external incentive to misreport s_h given true experience s_j
- C is the maximum cost for reporting at all

$$min \quad W = \sum_{j=1}^{M} Pr[s_j]\Big(\sum_{k=1}^{M} Pr[s_k|s_j]\tau(s_j, s_k)\Big)$$

$$s.t. \quad \sum_{k=1}^{M} Pr[s_k|s_j]\Big(\tau(s_j, s_k) - \tau(s_h, s_k)\Big) > \Delta(s_j, s_h);$$

$$\forall s_j, s_h \in \mathcal{S}, s_j \neq s_h;$$

$$\sum_{k=1}^{M} Pr[s_k|s_j]\tau(s_j, s_k) > C; \quad \forall s_j \in \mathcal{S}$$

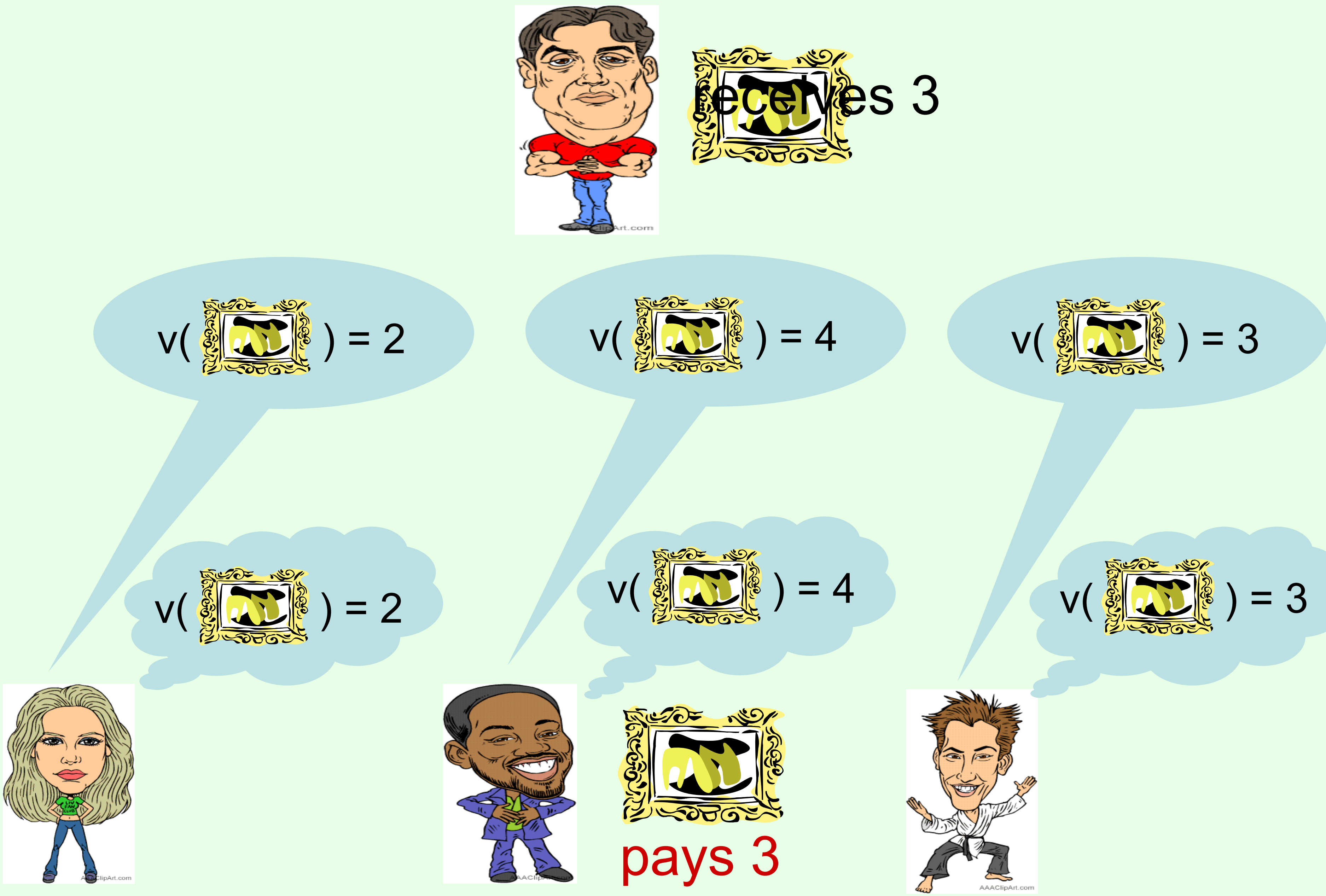
$$\tau(s_j, s_k) \geq 0; \forall s_j, s_k \in \mathcal{S}$$

- Jurca and Faltings study a number of related problems [EC06, WWW07,EC07, Jurca's thesis, SIGecom Exchanges 08]

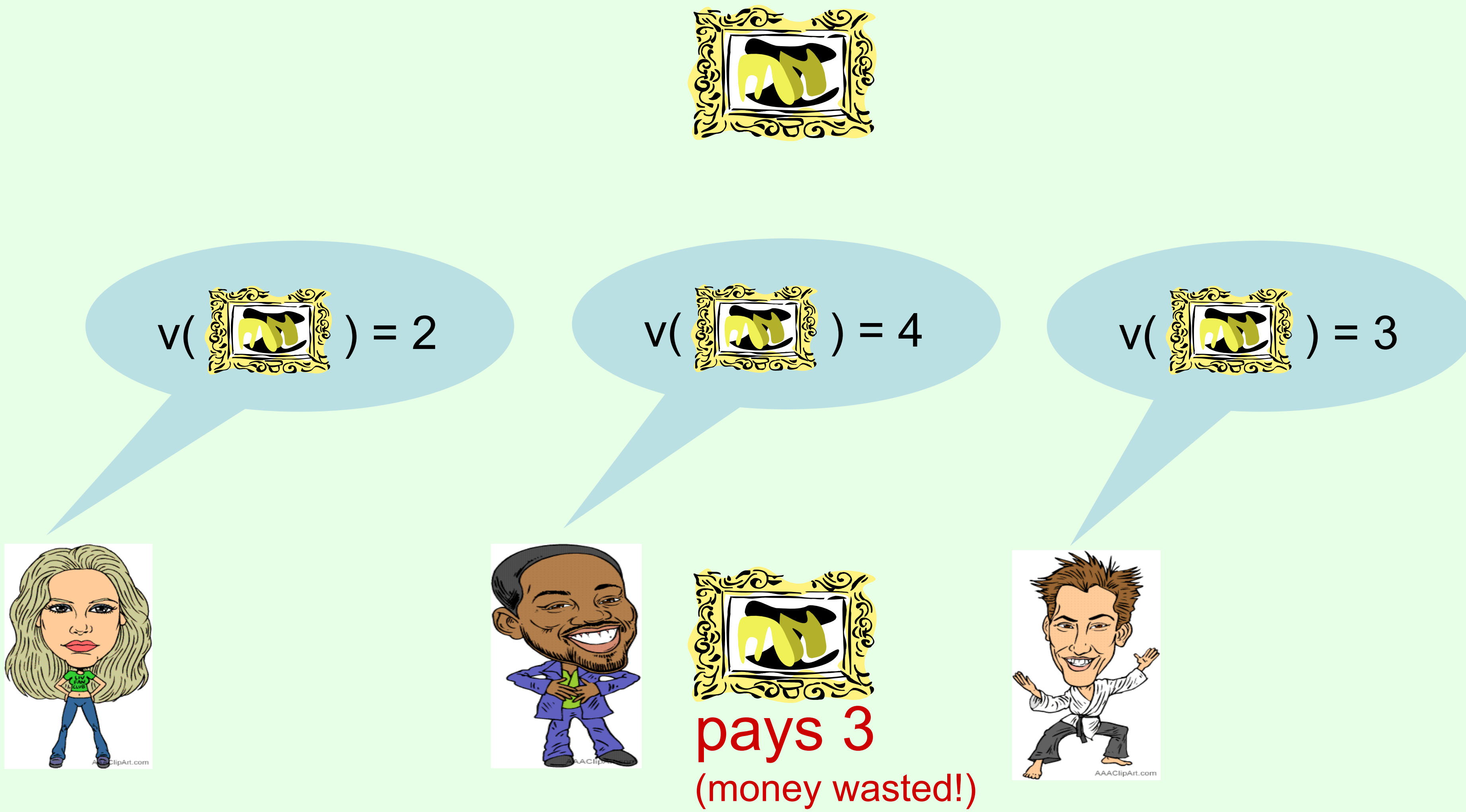
Auctions with revenue redistribution

Guo and C.,
EC 07,
Games and Economic Behavior forthcoming

Second-price (Vickrey) auction

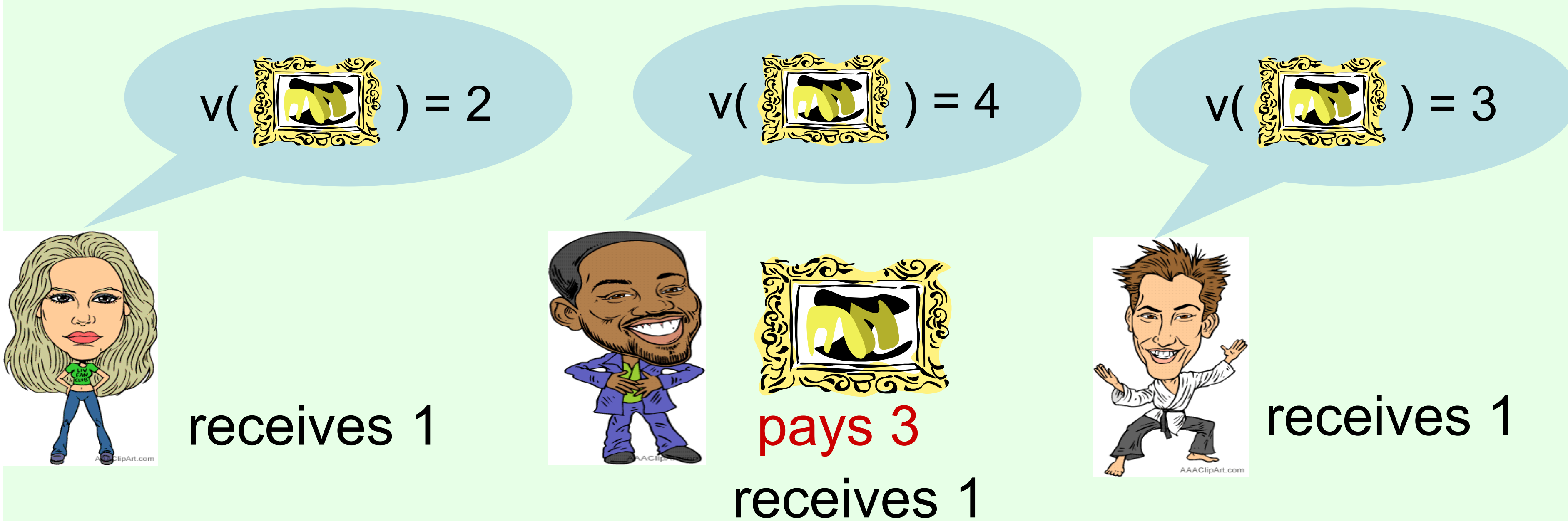


Vickrey auction without a seller



Can we redistribute the payment?

Idea: give everyone 1/n of the payment



not strategy-proof

Bidding higher can increase your redistribution payment

Strategy-proof redistribution

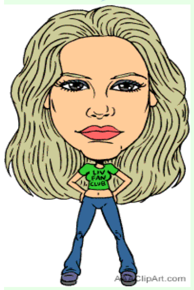
[Bailey 97, Porter et al. 04, Cavallo 06]

Idea: give everyone 1/n of second-highest **other** bid

v( ) = 2

v( ) = 4

v( ) = 3



receives 1

pays 3

receives 2/3

receives 2/3

2/3 wasted (22%)

strategy-proof

Your redistribution does not depend on your bid;

incentives are the same as in Vickrey

Bailey-Cavallo mechanism…

- Bids: $V_1 \geq V_2 \geq V_3 \geq \ldots \geq V_n \geq 0$
- First run Vickrey auction
- Payment is V_2
- First two bidders receive V_3/n
- Remaining bidders receive V_2/n
- Total redistributed: $2V_3/n+(n-2)V_2/n$

$R_1 = V_3/n$
$R_2 = V_3/n$
$R_3 = V_2/n$
$R_4 = V_2/n$
...
$R_{n-1}= V_2/n$
$R_n = V_2/n$

Can we do better?

Desirable properties

- Strategy-proofness
- Voluntary participation: bidder's utility always nonnegative
- Efficiency: bidder with highest valuation gets item
- Non-deficit: sum of payments is nonnegative
  - i.e. total Vickrey payment ≥ total redistribution
- (Strong) budget balance: sum of payments is zero
  - i.e. total Vickrey payment = total redistribution
- **Impossible to get all**
- We sacrifice budget balance
  - Try to get approximate budget balance
- Other work sacrifices: strategy-proofness [Parkes 01], efficiency [Faltings 04], non-deficit [Bailey 97], budget balance [Cavallo 06]

Another redistribution mechanism

- Bids: $V_1 \geq V_2 \geq V_3 \geq V_4 \geq \ldots \geq V_n \geq 0$
- First run Vickrey
- Redistribution:

  Receive 1/(n-2) * second-highest **other** bid,
  - 2/[(n-2)(n-3)] third-highest **other** bid
- Total redistributed:

  $V_2 - 6V_4/[(n-2)(n-3)]$
- Efficient & strategy-proof
- Voluntary participation & non-deficit (for large enough n)

$R_1 = V_3/(n-2) - 2/[(n-2)(n-3)]V_4$

$R_2 = V_3/(n-2) - 2/[(n-2)(n-3)]V_4$

$R_3 = V_2/(n-2) - 2/[(n-2)(n-3)]V_4$

$R_4 = V_2/(n-2) - 2/[(n-2)(n-3)]V_3$

...

$R_{n-1} = V_2/(n-2) - 2/[(n-2)(n-3)]V_3$

$R_n = V_2/(n-2) - 2/[(n-2)(n-3)]V_3$

Comparing redistributions

- Bailey-Cavallo: $\sum R_i = 2V_3/n + (n-2)V_2/n$
- Second mechanism: $\sum R_i = V_2 - 6V_4/[(n-2)(n-3)]$
- Sometimes the first mechanism redistributes more
- Sometimes the second redistributes more
- Both redistribute 100% in some cases
- What about the **worst** case?
- Bailey-Cavallo worst case: $V3=0$
  - fraction redistributed: $1-2/n$
- Second mechanism worst case: $V2=V4$
  - fraction redistributed: $1-6/[(n-2)(n-3)]$
- For large enough n, $1-6/[(n-2)(n-3)] \geq 1-2/n$, so second is better (in the worst case)

Generalization: linear redistribution mechanisms

- Run Vickrey
- Amount redistributed to bidder i:

$$C_0 + C_1 V_{-i,1} + C_2 V_{-i,2} + \ldots + C_{n-1} V_{-i,n-1}$$

  where $V_{-i,j}$ is the j-th highest **other** bid for bidder i
- Bailey-Cavallo: $C_2 = 1/n$
- Second mechanism: $C_2 = 1/(n-2)$, $C_3 = -2/[(n-2)(n-3)]$

- Bidder's redistribution does not depend on own bid, so strategy-proof
- Efficient
- Other properties?

Redistribution to each bidder

Recall: $R = C_0 + C_1 V_{-i,1} + C_2 V_{-i,2} + \dots + C_{n-1} V_{-i,n-1}$

$R_1 = C_0+C_1V_2+C_2V_3+C_3V_4+\dots+C_iV_{i+1}+\dots+C_{n-1}V_n$

$R_2 = C_0+C_1V_1+C_2V_3+C_3V_4+\dots+C_iV_{i+1}+\dots+C_{n-1}V_n$

$R_3 = C_0+C_1V_1+C_2V_2+C_3V_4+\dots+C_iV_{i+1}+\dots+C_{n-1}V_n$

$R_4 = C_0+C_1V_1+C_2V_2+C_3V_3+\dots+C_iV_{i+1}+\dots+C_{n-1}V_n$

...

$R_{n-1} = C_0+C_1V_1+C_2V_2+C_3V_3+\dots+C_iV_i +\dots+C_{n-1}V_n$

$R_n = C_0+C_1V_1+C_2V_2+C_3V_3+\dots+C_iV_i +\dots+C_{n-1}V_{n-1}$

Voluntary participation & non-deficit

- Voluntary participation:

  equivalent to

$$R_n = C_0 + C_1V_1 + C_2V_2 + C_3V_3 + \dots + C_iV_i + \dots + C_{n-1}V_{n-1} \geq 0$$

$$\text{for all } V_1 \geq V_2 \geq V_3 \geq \dots \geq V_{n-1} \geq 0$$

- Non-deficit:

  $\sum R_i \leq V_2$ for all $V_1 \geq V_2 \geq V_3 \geq \dots \geq V_{n-1} \geq V_n \geq 0$

Worst-case optimal (linear) redistribution

Try to maximize worst-case redistribution %

Variables: C_i, K

Maximize K

Subject to:

$R_n \geq 0$ for all $V_1 \geq V_2 \geq V_3 \geq \ldots \geq V_{n-1} \geq 0$

$\sum R_i \leq V_2$ for all $V_1 \geq V_2 \geq V_3 \geq \ldots \geq V_n \geq 0$

$\sum R_i \geq K V_2$ for all $V_1 \geq V_2 \geq V_3 \geq \ldots \geq V_n \geq 0$

R_i as defined in previous slides

Transformation into linear program

- **Claim**: $C_0=0$
- **Lemma**: $Q_1X_1+Q_2X_2+Q_3X_3+...+Q_kX_k\geq 0$ for all $X_1\geq X_2\geq ...\geq X_k\geq 0$

  is equivalent to

  $Q_1+Q_2+...+Q_i\geq 0$ for $i=1$ to k
- Using this lemma, can write all constraints as linear inequalities over the C_i

Worst-case optimal **remaining** %

n=5: 27% (40%)

n=6: 16% (33%)

n=7: 9.5% (29%)

n=8: 5.5% (25%)

n=9: 3.1% (22%)

n=10: 1.8% (20%)

n=15: 0.085% (13%)

n=20: 3.6 e-5 (10%)

n=30: 5.4 e-8 (7%)

The data in the parentheses are for the Bailey-Cavallo mechanism

Average-case remaining %

(uniform distribution)

n=5: 8.9% (6.7%)

n=6: 6.9% (4.8%)

n=7: 3.6% (3.6%)

n=8: 2.5% (2.8%)

n=9: 1.3% (2.2%)

n=10: 0.8% (1.8%)

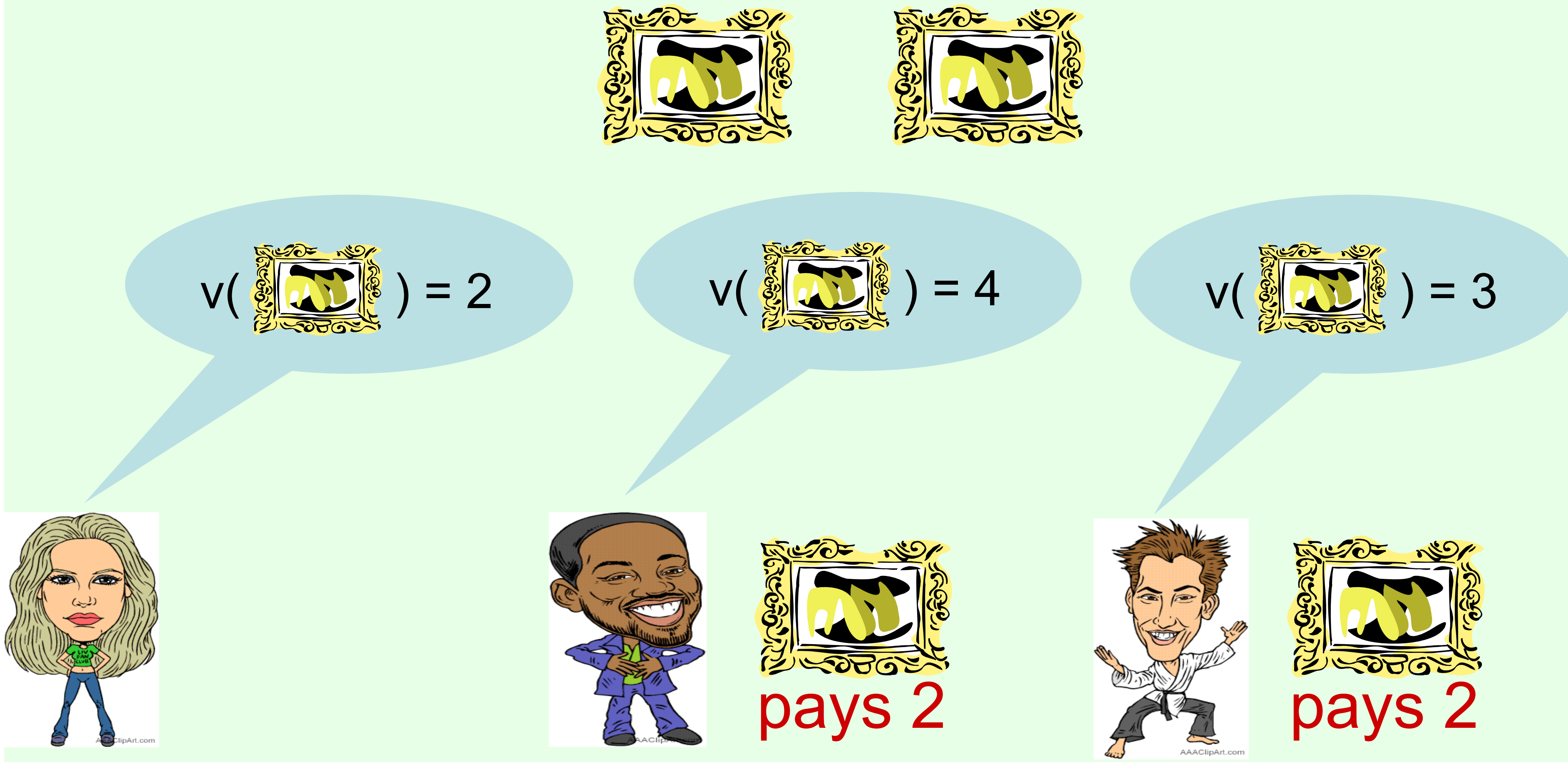
n=15: 3.7 e-4 (0.8%)

n=20: 1.7 e-5 (0.5%)

n=30: 2.6 e-8 (0.2%)

The data in the parentheses are for the Bailey-Cavallo mechanism

m-unit auction with unit demand: VCG (m+1th price) mechanism



strategy-proof

Our techniques can be generalized to this setting

m+1th price mechanism

Variables: C_i ,K

Maximize K

subject to:

$R_n \geq 0$ for all $V_1 \geq V_2 \geq V_3 \geq \ldots \geq V_{n-1} \geq 0$

$\sum R_i \leq V_2$ for all $V_1 \geq V_2 \geq V_3 \geq \ldots \geq V_n \geq 0$

$\sum R_i \geq K\, V_2$ for all $V_1 \geq V_2 \geq V_3 \geq \ldots \geq V_n \geq 0$

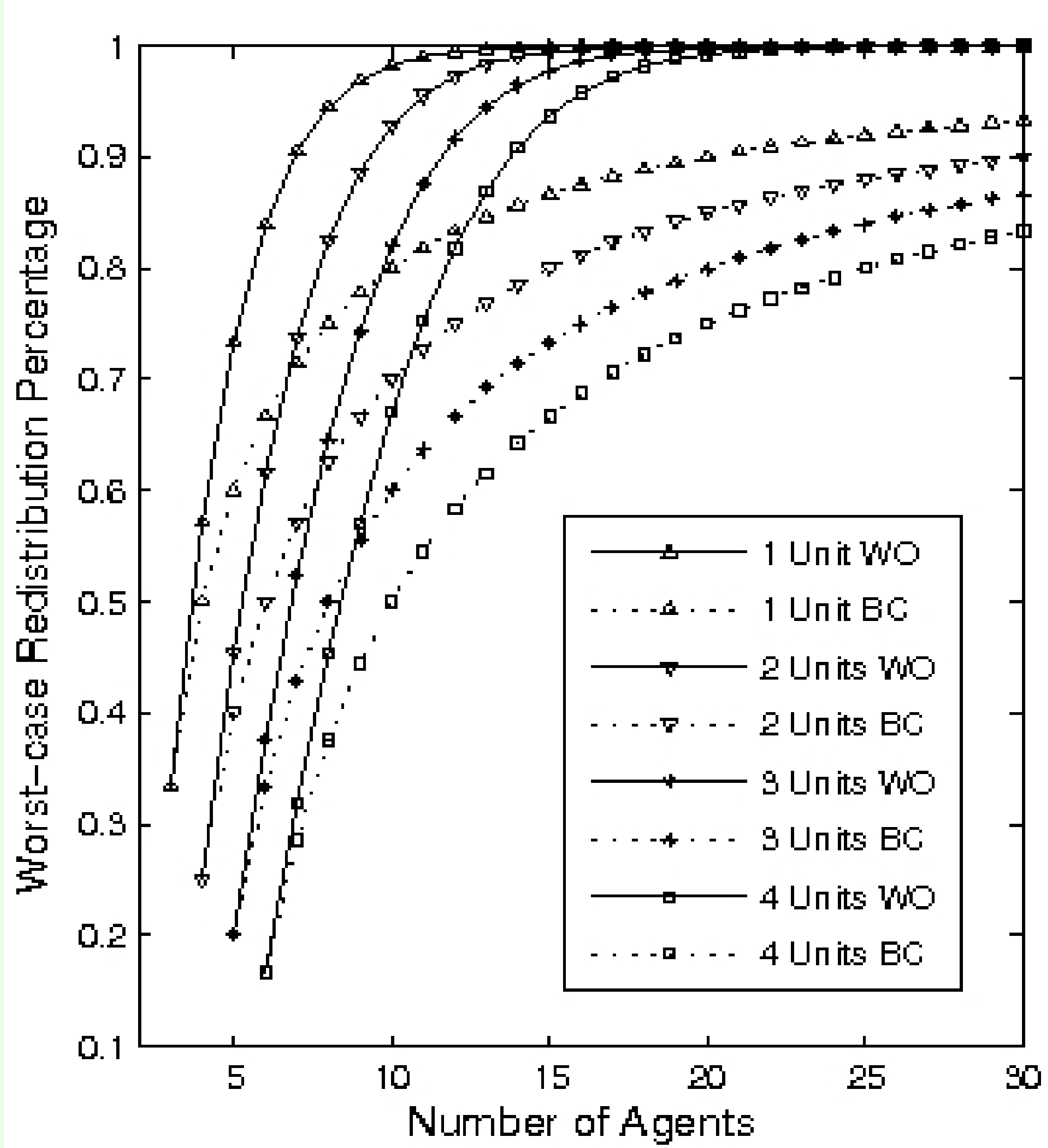
R_i as defined in previous slides

Only need to change V_2 into mV_{m+1}

Results

BC = Bailey-Cavallo

WO = Worst-case Optimal



Analytical characterization of WO mechanism

$$k^* = 1 - \frac{\binom{n-1}{m}}{\Sigma_{j=m}^{n-1}\binom{n-1}{j}}$$

$$c_i^* = \frac{(-1)^{i+m-1}(n-m)\binom{n-1}{m-1}}{i\Sigma_{j=m}^{n-1}\binom{n-1}{j}} \frac{1}{\binom{n-1}{i}} \Sigma_{j=i}^{n-1}\binom{n-1}{j}$$

$$for\ i = m+1, \ldots, n-1$$

- Unique optimum
- Can show: for fixed m, as n goes to infinity, worst-case redistribution percentage approaches 100% with rate of convergence 1/2

Worst-case optimality outside the linear family

- **Theorem:** The worst-case optimal **linear** redistribution mechanism is also worst-case optimal among **all** VCG redistribution mechanisms that are
  - deterministic,
  - anonymous,
  - strategy-proof,
  - efficient,
  - non-deficit

- Voluntary participation is not mentioned
  - Sacrificing voluntary participation does not help
- Not **uniquely** worst-case optimal

Related paper

- Moulin's working paper “Efficient, strategy-proof and almost budget-balanced assignment”
  - pursues different worst-case objective (minimize waste/efficiency)
  - results in same mechanism in the unit-demand setting (!)
  - different mechanism results after removing voluntary participation requirement

Additional results on redistribution

- We generalized the above to multi-unit auctions with nonincreasing marginal values [Guo & C. GEB forthcoming]
- Maximizing expected redistribution given a prior [Guo & C. AAMAS-08a]
- Redistribution mechanisms that are not "dominated" by other redistribution mechanisms [Guo & C. AAMAS-08b; Apt, C., Guo, Markakis <under construction>]
- Sacrificing efficiency to increase redistribution (and, thereby, overall welfare)

  [Guo & C. EC-08! Saturday 10am]

Some additional special-purpose AMD directions

- Sequences of take-it-or-leave-it-offers [Sandholm & Gilpin AAMAS06]
- Revenue-maximizing combinatorial auctions [Likhodedov & Sandholm AAAI04, AAAI05]
- Online mechanisms [Hajiaghayi, Kleinberg, Sandholm AAAI07]

- And: more in the second half of this tutorial…

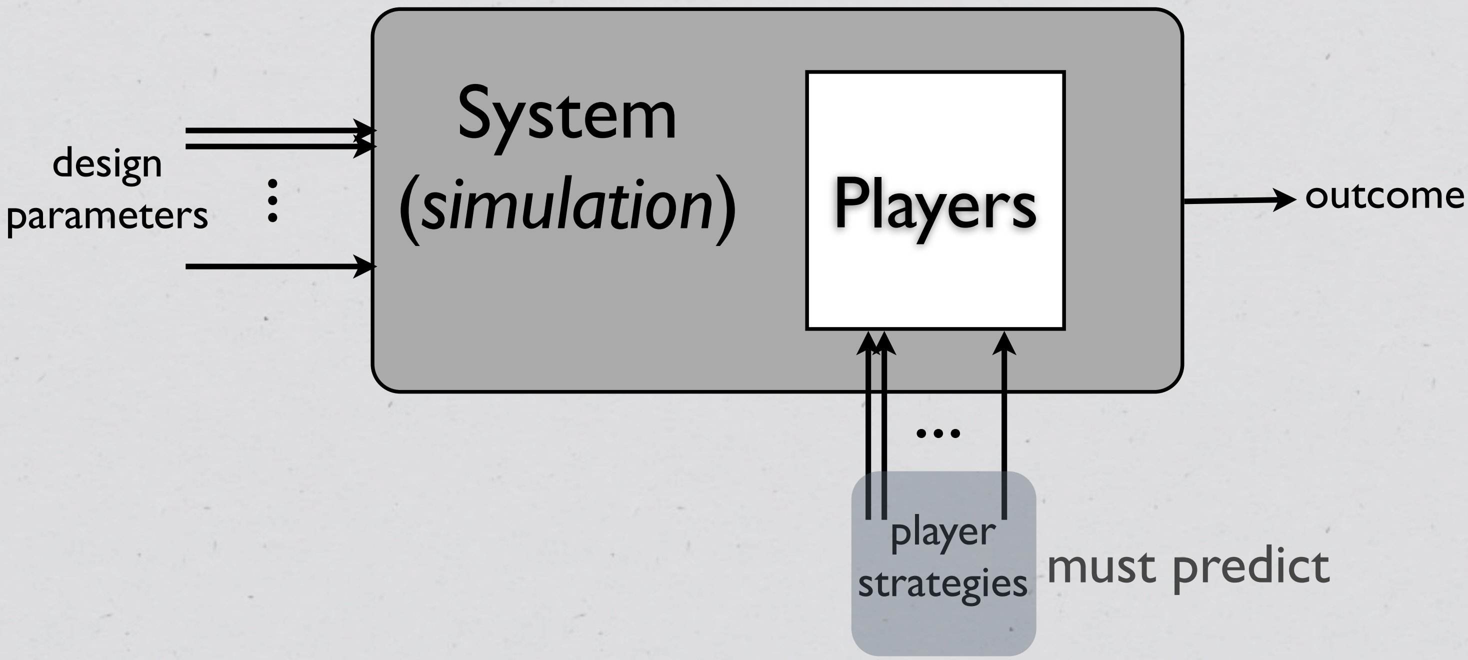
Automted Mechanism Design: Approaches & Applications PART II

Vincent Conitzer and Yevgeniy Vorobeychik

A Computational Approach to Constrained MD

- *Input*: designer objective, design parameters, constraints, game model
- *Output*: (nearly) optimal mechanism with respect to the specified objective
- This is a constrained optimization problem if predictions of the strategic choices of players are readily available

Mechanism Design for Simulation-Based Games



Outline

- Motivating example: strategic procurement in a supply-chain game
- The two-stage model of mechanism design
- Mechanism design in a supply-chain game
- Automated mechanism design on constrained design spaces
- Solving simulation-based games
- Evolutionary and learning approaches to automated mechanism design

Outline

* Motivating example: strategic procurement in a supply-chain game
* The two-stage model of mechanism design
* Mechanism design in a supply-chain game
* Automated mechanism design on constrained design spaces
* Solving simulation-based games
* Evolutionary and learning approaches to automated mechanism design

Supply-Chain Game (TAC/SCM)

- TAC/SCM: supply-chain management (SCM) scenario of the international Trading Agent Competition (TAC)
- Autonomous agents (developed by teams) act as PC manufacturers
  - buy components from suppliers (simulator)
  - bid on orders from customers (simulator)
- In TAC/SCM 2003 agents were observed to make excessive component purchases on day 0 (the first simulation day)

Game Master Response

- Designers introduced storage cost, charged daily for component inventory, to reduce incentives for excessive day-0 procurement
- **MD Question**: how to set the storage cost parameter?
  - agent behavior extremely complex
  - payoffs uncertain
- **My approach**: systematic exploration of the parameter and agent strategy spaces

Outline

- Motivating example: strategic procurement in a supply-chain game
- The two-stage model of mechanism design
- Mechanism design in a supply-chain game
- Automated mechanism design on constrained design spaces
- Solving simulation-based games

Some Notation

- Mechanism parameters: θ
- Strategic choice by player i : r_i
  - strategy profile: r
- Objective function: $W(\theta, r)$
- Player utility functions: $u_i(\theta, r)$

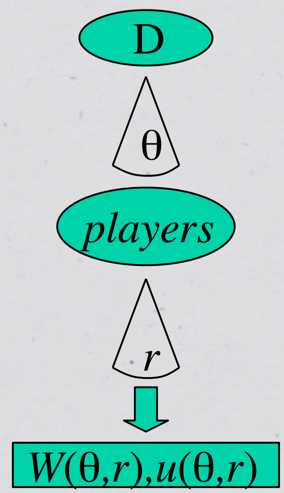
TAC/SCM Example

- Mechanism parameter θ is storage cost
- A strategic choice r_i is the day-0 procurement decision of player i
- Player utility functions, $u_i(\theta, r)$, are expected profits at the end of simulation
- Objective function, $W(\theta, r)$, is an indicator function:
  - 1 if total day-0 procurement (sum of individual procurement choices) is below a fixed threshold

Mechanism Design: The Model

stage 1: designer chooses the mechanism

stage 2: players choose strategies

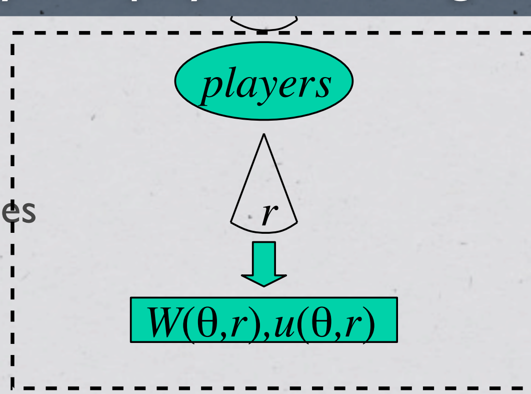


Mechanism Design: The Model

* *Given* the mechanism, stage 2 is a *game*
* Designer must predict joint strategic choices by the players in this game

stage 1: designer c

stage 2: players choose strategies

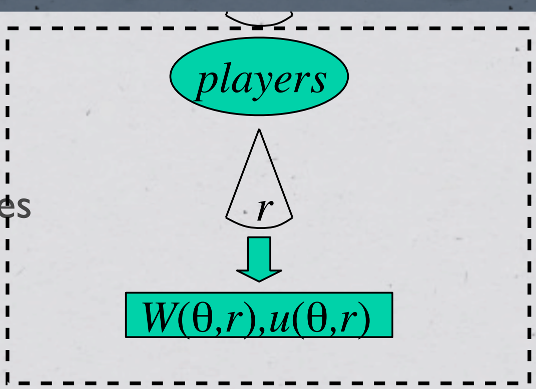


Mechanism Design: The Model

* *Given* the mechanism, stage 2 is a *game*
* Designer must predict joint strategic choices by the players in this game

stage 1: designer c

stage 2: players choose strategies



Formally, can solve this by *backwards induction:*
1. Obtain solutions to games in stage 2, $r^*(\theta)$
2. Find θ that maximizes $W(\theta, r^*(\theta))$

Outline

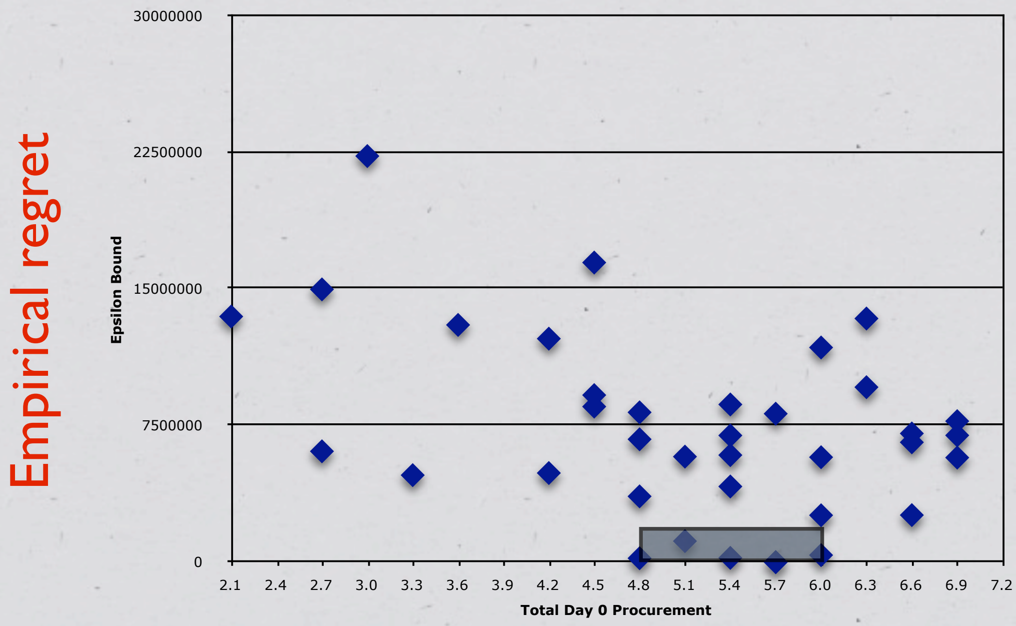
- Motivating example: strategic procurement in a supply-chain game
- The two-stage model of mechanism design
- Mechanism design in a supply-chain game
- Automated mechanism design on constrained design spaces
- Solving simulation-based games
- Evolutionary and learning approaches to automated mechanism design

General Approach

- For each (of a small set of) θ:
  - Collect payoff samples for a set of strategy profiles *r*
  - Approximate (ranges of) N.E. outcomes based on collected data
    - In TAC/SCM, we can summarize N.E. outcomes as *total day-0 procurement:* $\varphi(r,\theta) = \sum_i r_i(\theta)$
- Generalize solution correspondence to θ outside of the data set

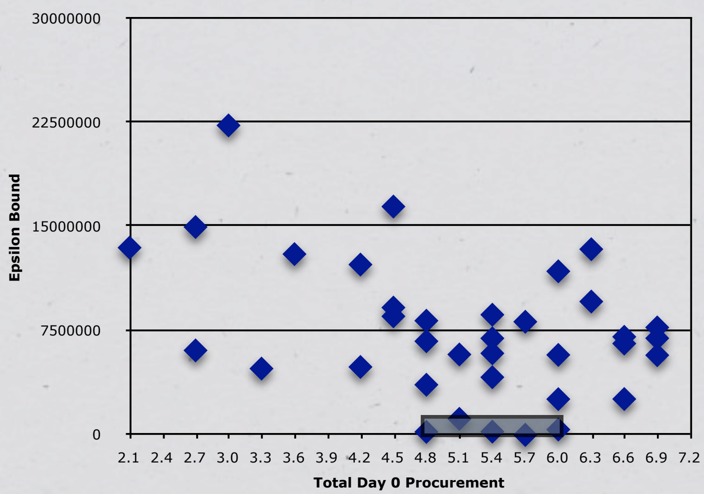
Obtaining N.E. Outcome Correspondence

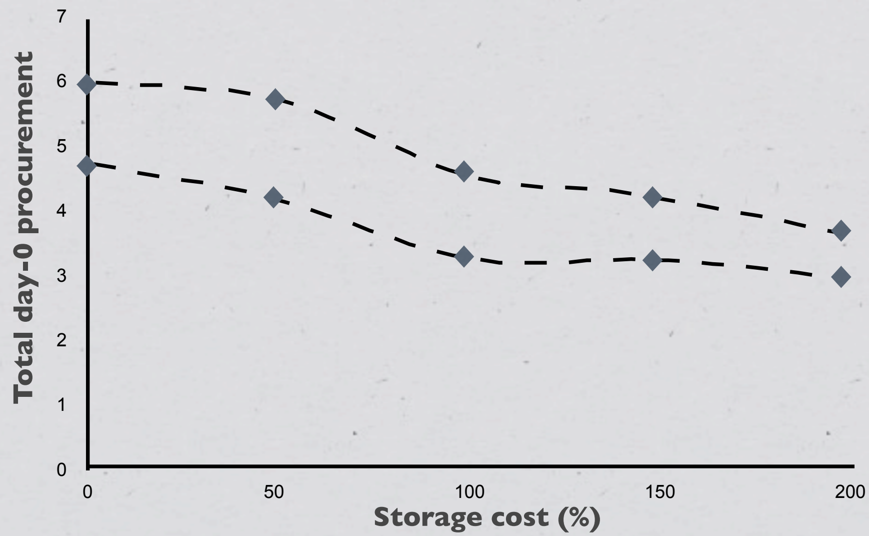
Storage Cost = 0



Total day-0 procurement , $\varphi(r,0)$

Obtaining N.E. Outcome Correspondence



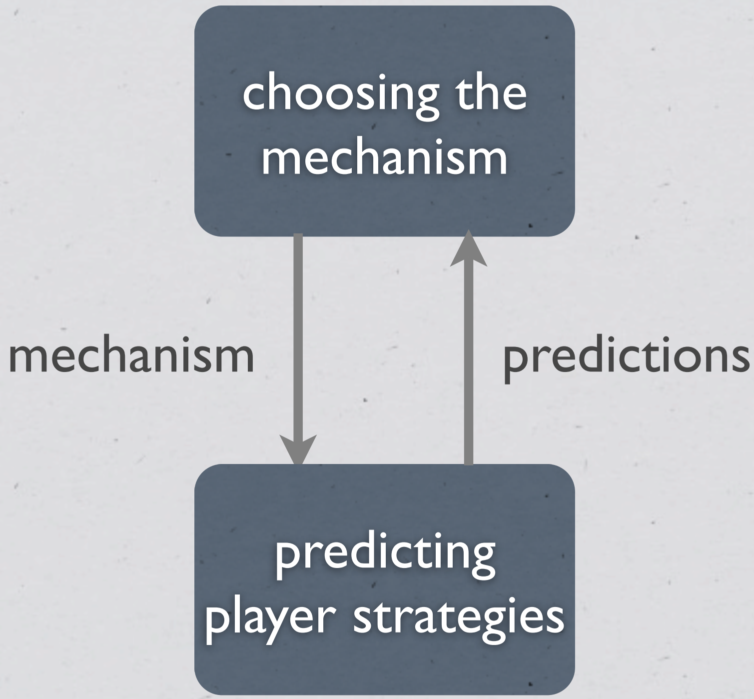


No reasonable setting of storage cost likely to achieve the designer's objective

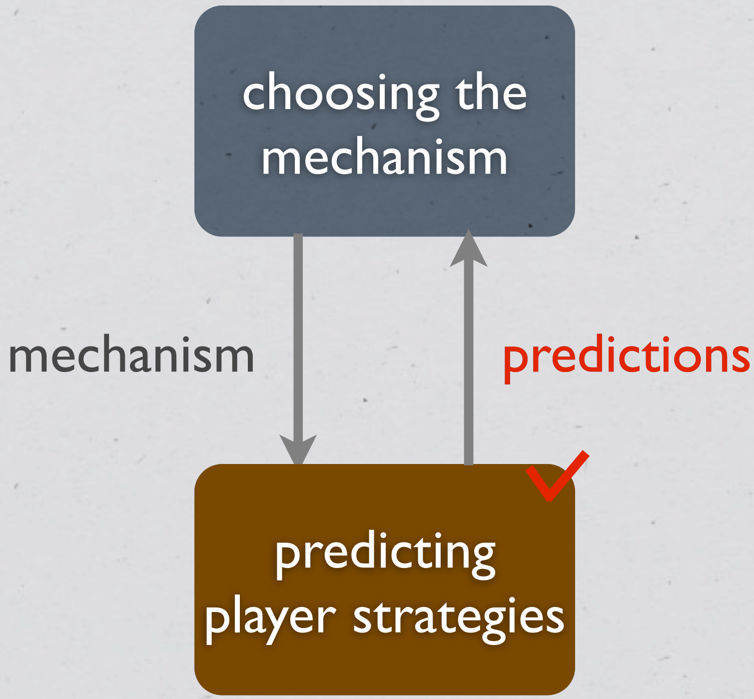
Stepping back...

- Able to take advantage of structure in the TAC/SCM application
  - The designer only cares about *total* day-0 procurement
  - Can plot the approximate N.E. outcome correspondence in 2D
  - Very simple objective function
- How can we do simulation-based mechanism design *in general*?

The Mechanism Design Process



The Mechanism Design Process



Outline

- Motivating example: strategic procurement in a supply-chain game
- The two-stage model of mechanism design
- Mechanism design in a supply-chain game
- Automated mechanism design on constrained design spaces
- Solving simulation-based games
- Evolutionary and learning approaches to automated mechanism design

Problem Specification and Inputs

- Constrained, *n*-dimensional design space, Θ
- Each mechanism induces an infinite game (possibly specified using simulations)
- Black-box specification of the objective and constraints
- *Suppose we are given a solver for a class of games induced by* Θ
  - SOLVER: $\mathbf{S}(\theta)$ is a mapping from $\theta \in \Theta$ to a solution $r^*(\theta)$

Solution Strategy: Stochastic Optimization

- Iterative algorithm that explores the mechanism design space
  - *Simulated annealing*:
    - move to the next mechanism if it is better than current
    - probabilistically explore inferior mechanism choices
    - local search algorithm with global convergence properties
  - Many other alternatives (stochastic approximation, genetic algorithms, etc)

Application to Mechanism Design in Bayesian Games

- Each mechanism induces infinite games of incomplete information (i.e., infinite sets of player choices and *types*)
- Joint space of player types T, a profile of types is $t \in T$
- Black-box specification of the distribution over T
- The solution concept is Bayes-Nash Equilibrium
  - Thus, $\mathbf{S}(\theta)$ produces $r^*(t, \theta)$ s.t. for every player i, $r_i^*(t_i, \theta)$ is a best response to the strategies of other players

Mechanism Design Problems

- Consider two types of mechanism design problems:
  - *Bayesian mechanism design:* $\max_{\theta} \mathbf{E}_t \left[W(r^*(t, \theta), t, \theta)\right]$
  - *Robust mechanism design:* $\max_{\theta} \mathbf{inf}_t \left[W(r^*(t, \theta), t, \theta)\right]$
- Caveat with Robust MD: cannot computationally take inf (or min) of a black box objective function over an infinite type space
- Relaxation: *probably approximately robust mechanism design*
  - Estimate worst case w.r.t. "large" set of types using *n* samples

Probably Approximately Robust MD

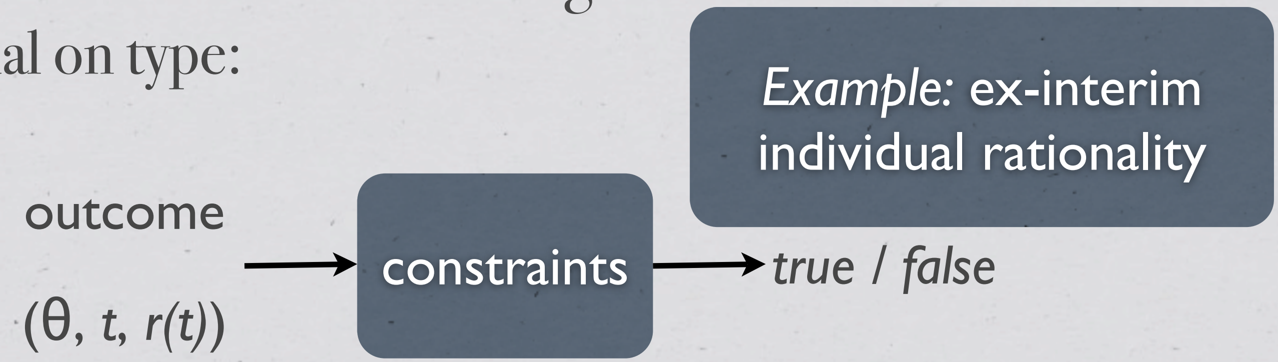
- Suppose we select the best of L candidate mechanisms using n samples from the type distribution to estimate the worst-case outcomes. In order to attain confidence of at least 1 - α that we "ignore" a set of types no larger than a measure p, we need at least

$$n \geq \frac{\log\left(1 - (1 - \alpha)^{\frac{1}{L}}\right)}{\log(1 - p)}$$

samples

Evaluating Constraints

- A similar caveat exists in evaluating constraints which are conditional on type:



- Cannot computationally evaluate such constraints for every type
- Relaxation: ensure constraint holds on a "large" set of types (*p-strong constraints* hold for a type set with measure at least 1 - p)

Verifying *p-strong constraints*

- Let B be a set on which a probabilistic constraint is violated and suppose that there is a uniform prior over [0,1] on the measure of B. We need

$$n \geq \frac{\log \alpha}{\log (1-p)} - 1$$

samples to verify with confidence at least 1 - α that the constraint holds for a set of types with probability at least 1 - p

Applications to Computational Auction Design

- Several examples of 2-player 1-item auctions
  - Games solved using the Reeves-Wellman solver (Reeves and Wellman, 2004)
- Objectives:
  - Fairness, revenue, welfare
- Constraints:
  - Ex-interim individual rationality

1. Shared-Good Auction

$$u(t,a,t',a') = \begin{cases} t - f(a,a') & \text{if } a > a' \\ \frac{t - f(a,a') + f(a',a)}{2} & \text{if } a = a' \\ f(a',a) & \text{if } a < a' \end{cases}$$

- Design space: $f(a,a') = ha + ka'$ (two parameters h and k)
  - Notation: SGA(h, k)
- For 2 players, U[A,B] types, have an analytic expression for BNE
  - Remark: all SGA(h,k) mechanisms are efficient

Searching for Sharing Mechanisms

start at a random point (h_0, k_0)

in iteration k, probabilistically select the next point

(h_k, k_k)

evaluate (predict player strategies)

(h_{k+1}, k_{k+1})

...

select the best mechanism seen thus far

Objective: Expected "Fairness"

* Minimize difference in expected utility between winner and loser
* Theory: SGA(0, k) optimal for $k > 0$

Applying AMD — Finds the optimal mechanism

Maximize Ex Ante Fairness

- Minimize expected difference in utility
- No analytic characterization

Applying AMD

SGA(0.49,1) with value 0.176 could not improve on this even with known BNE

Robust Fairness

* Minimize nearly-maximal difference in utility
* Theory: SGA(h, 0) is optimal for $h > 0$

Applying AMD — Finds the optimal mechanism

2. "Myerson" Auctions

$$u(t,a,t',a') = \begin{cases} U_1 & \text{if } a > a' \\ 0.5(U_1 + U_2) & \text{if } a = a' \\ U_2 & \text{if } a < a' \end{cases}$$

$U_1 = q\,t - p_1(t)$ $\qquad$ $p_1(t) = k_1\,a(t) + k_2\,a'(t) + K_1$

$U_2 = (1 - q)\,t - p_2(t)$ $\qquad$ $p_2(t) = k_3\,a(t) + k_4\,a'(t) + K_2$

all parameters in [0,1]

winner gets the good with probability q, pays $p_1(t)$

Maximizing Expected Revenue

- Theory: optimal incentive compatible* mechanism in this design space yields expected revenue of 1/3

Applying AMD finds an auction with expected revenue of 0.3

*Incentive compatible = it is a Bayes-Nash Equilibrium to bid actual type (value for the item)

Maximize Expected Welfare

- Welfare = sum of player utilities
- Theory: monotone strategies suffice; optimal welfare is 2/3
- first-price and second-price sealed-bid auctions are efficient

Applying AMD | Finds the optimal mechanism

Robust Revenue Maximization

* Theory: any auction with no *fixed* transfers and monotone increasing BNE strategies yields at most 0 robust revenue (e.g., first-price and second-price auctions)

Applying AMD finds an auction with minimum revenue > 0

3. "Anti-Social" Auctions

$$u(t, a, t', a') = \begin{cases} U_1 & \text{if } a > a' \\ 0.5(U_1 + U_2) & \text{if } a = a' \\ U_2 & \text{if } a < a' \end{cases}$$

U_1 and U_2 are like in "Myerson" auctions, but include a parameter for the amount of disutility to one agent from the other's utility

Same set of parameters as before

Maximizing Expected Revenue

- Theory:
  - No known optimum; expected revenue from Vicious Vickrey (VV) auction (the only one previously studied) is 0.48
  - Vicious Vickrey is *not* ex-interim individually rational
  - After adjustment for individual rationality, expected revenue of VV falls to 0.438

Applying AMD

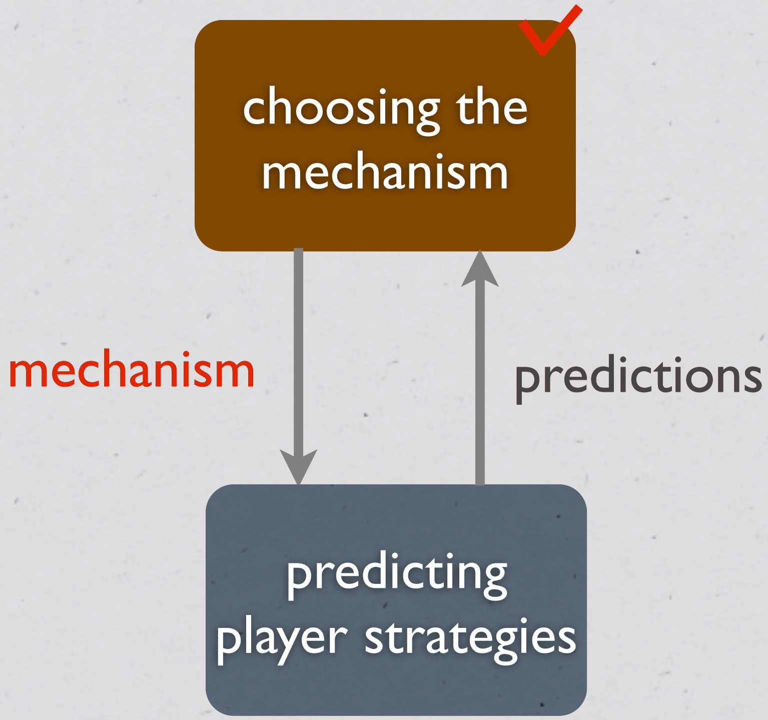
Using VV as a starting point, finds IR auction with revenue = 0.49

From a random starting point, finds IR auction with revenue = 0.44

Takeaways

- The mechanism design problem can be modeled as a one-shot two-stage game
  - This game can in principle be solved using backward induction
  - **In practice, we can use an iterative improvement algorithm**, which runs a game solver as a subroutine, evaluating the objective function and constraints on the obtained solutions
  - This is a **practical approach for a parameterized design of auctions** in various settings: a series of positive examples

The Mechanism Design Process



Outline

- Motivating example: strategic procurement in a supply-chain game
- The two-stage model of mechanism design
- Mechanism design in a supply-chain game
- Automated mechanism design on constrained design spaces
- Solving simulation-based games
- Evolutionary and learning approaches to automated mechanism design

The "Small" Game Setting

- The simplest setting is when we have at least *n* samples available for every joint strategic choice of players in the game (define a game comprised of sample mean payoffs)
- An "obvious" approach is to compute a Nash equilibrium on this *empirical game* (e.g., using GAMBIT, etc)
- *Analysis question 1*: sensitivity analysis (probabilistic bounds on the Nash equilibrium approximation quality)
- *Analysis question 2*: the sequence of sets of equilibria converges in several senses to the set of equilibria on the underlying game

The “Small” Game Setting

		Column	
		$a_{C,1}$	$a_{C,2}$
Row	$a_{R,1}$	(5.5, 5.5)	(0.5, 6.3)
	$a_{R,2}$	(6.3, 0.5)	(2.1, 2.1)

$(a_{R,1}, a_{C,1})$

↓

(5, 5)

(6, 5)

(5, 6)

(6, 6)

Convergence Results for "Small" Games

- Result 1: regrets of all mixed strategy profiles converge a.s.
- Result 2: The set of N.E. points w.r.t. the estimated game converges to the set of actual N.E. in *directed* Hausdorff distance
  - Every N.E. of the estimated game is eventually close to *some* N.E. of the underlying game
- Result 3: Every N.E. is an approximate N.E. of the estimated game for a large enough number of payoff samples

The “Small” Game Setting

- Mechanism design result:
  - *IF*
    - Finite mechanism design space
    - Each mechanism induces a finite game with a unique N.E.
  - *THEN*
    - Mechanism design choices w.r.t. estimated game converge to optimal choices

The “Large” Game Setting

- Impossible to take samples for every strategy profile: must approximate Nash equilibria based on limited information
- *Fundamental question*: how do we guide the sampling process to obtain a set of payoff samples which yields a good Nash approximation?
- One answer to this is by appealing again to *stochastic search* techniques

Stochastic Search Methods For Infinite Games

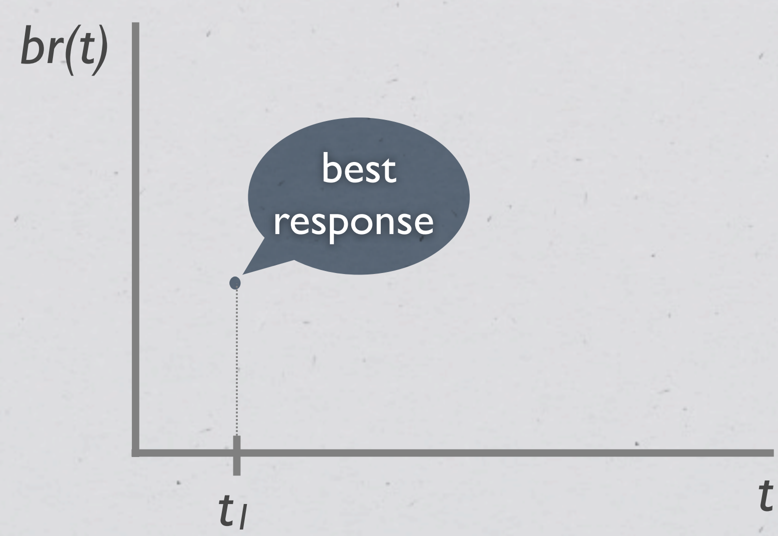
- Let's focus on some player, *i*, and **fix the strategies of the others**
- Given the simulation-based game and a fixed r_{-i}, computing a *best response* for player *i* is a *stochastic optimization problem*, that is, we need to maximize *i*'s utility given the simulation
  - *We will see that approximating a best response is a key step towards Nash equilibrium approximation*

Approximating Best Response Directly in Bayesian Games

- Parameterize the strategy function: $r(t_i) = f(\mathbf{k}, t_i)$, where **k** is a vector of parameters
- Find the setting of **k** which maximizes $u_i(\,f(\mathbf{k}, t_i)\,,\; r_{-i}(t_{-i})\,)$
  - $br(t_i) \approx \text{argmax}_{\mathbf{k}}\; u_i(\,f(\mathbf{k}, t_i)\,,\; r_{-i}(t_{-i})\,)$
  - Can use stochastic search to find an approximately maximizing vector **k** (e.g., *simulated annealing* is globally convergent)
- Call this the “direct” method

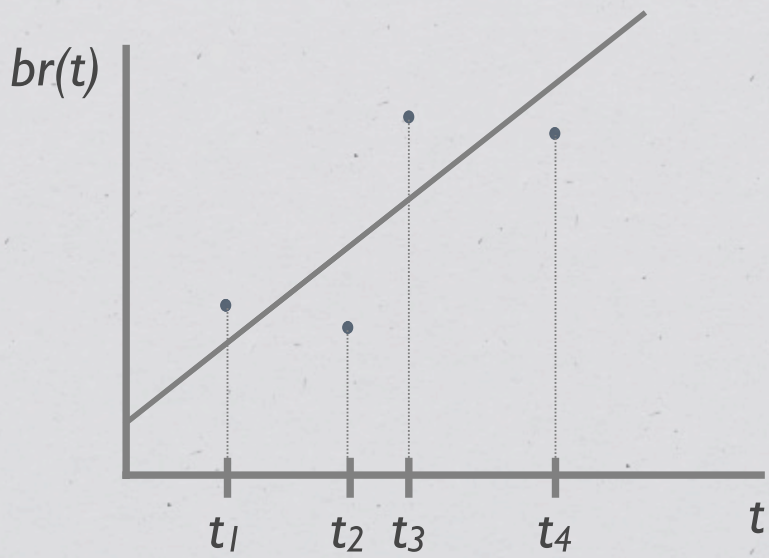
Learning Best Response in Bayesian Games

Use machine learning techniques to approximate the best response strategy as a function of type (value)

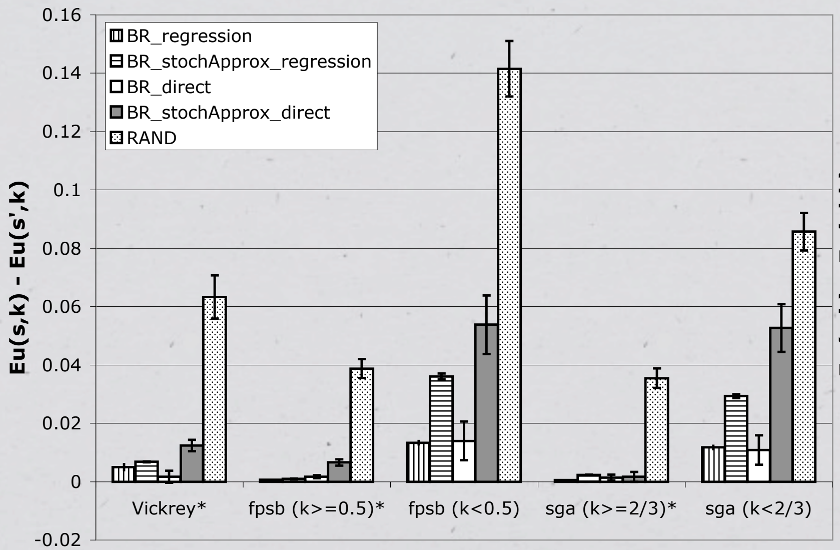


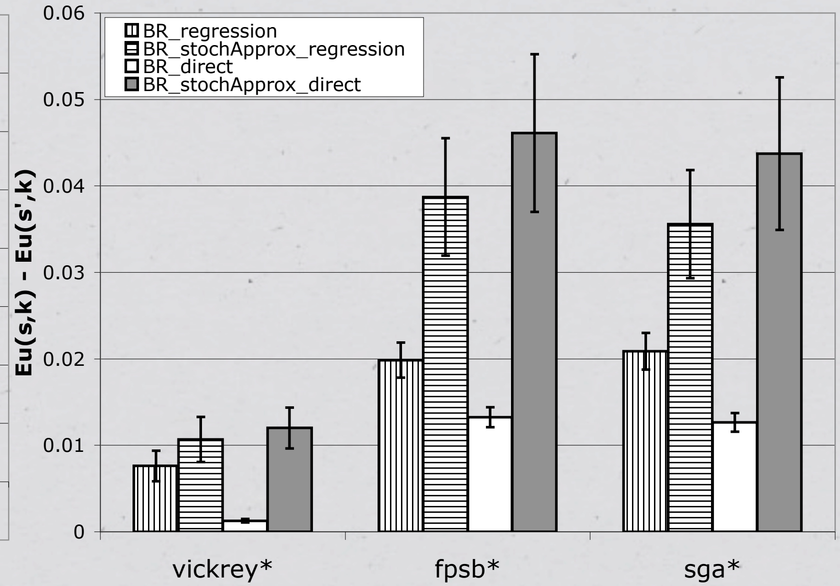
Learning Best Response in Bayesian Games

Use machine learning techniques to approximate the best response strategy as a function of type (value)



Comparison of Best Response Approximation Methods





all methods perform very well (small error relative to calibration)
“direct” method (search in function space) > learning-based method
simulated annealing > stochastic gradient-descent

From Best Response to a Nash Equilibrium

- To go from best response to a Nash equilibrium, we can follow *iterated best response dynamics*

  1. Start with a profile *r*

  2. Find best response, *br(r)* to *r* for all players

  3. Set *r* to *br(r)* in the next iteration

  4. Repeat

- Poor convergence properties but performs well in practice

From Best Response to a Nash Equilibrium

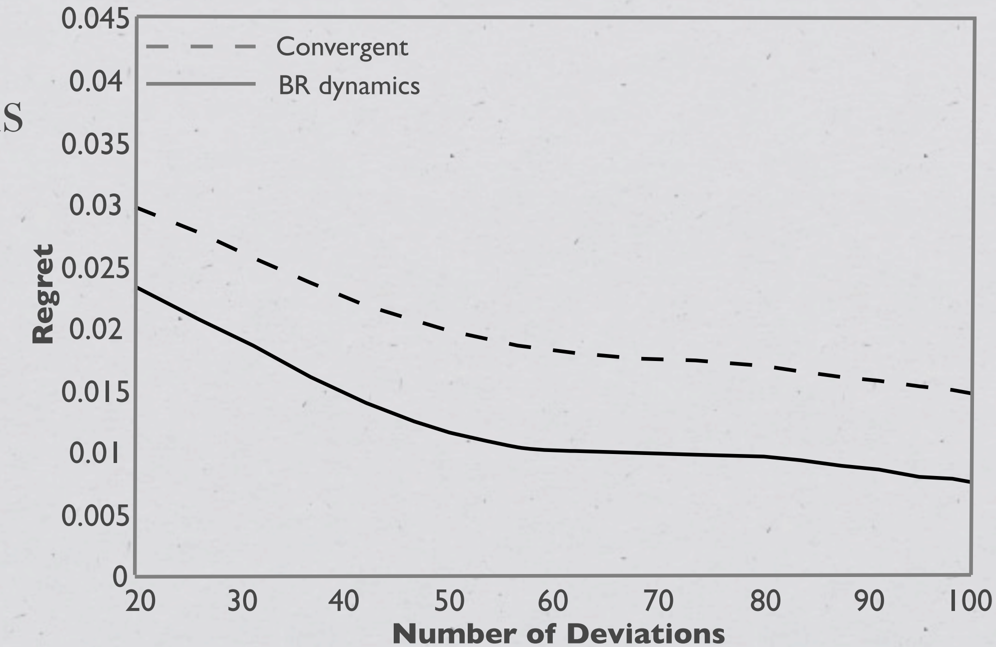
- We propose an alternative algorithm based on minimizing *game theoretic regret*
- Game theoretic regret of a strategy profile r (denote $\epsilon(r)$):
  - the most utility any agent i can gain by deviating from r_i to another strategy
  - in a Nash equilibrium, no such gain can be obtained; thus, if r is Nash equilibrium, $\epsilon(r) = 0$

Regret Minimization

- If a Nash equilibrium r^* exists, the function $\epsilon(r)$ has r^* as its global minimum: $\epsilon(r^*) = \min_r \epsilon(r)$
  - Thus, if we actually know the regret function, we could use non-linear minimization to approximate a Nash equilibrium
- Suppose we have an estimate of $\epsilon(r)$, $\hat{e}(r)$
  - Finding the minimum of $\hat{e}(r)$ gives us an approximate Nash equilibrium
- *Globally convergent if we use simulated annealing*

Approximate Regret Minimization vs. Iterative BR

- The approximate regret minimization algorithm is provably convergent
- Best response need not converge
- Best response often very effective in practice



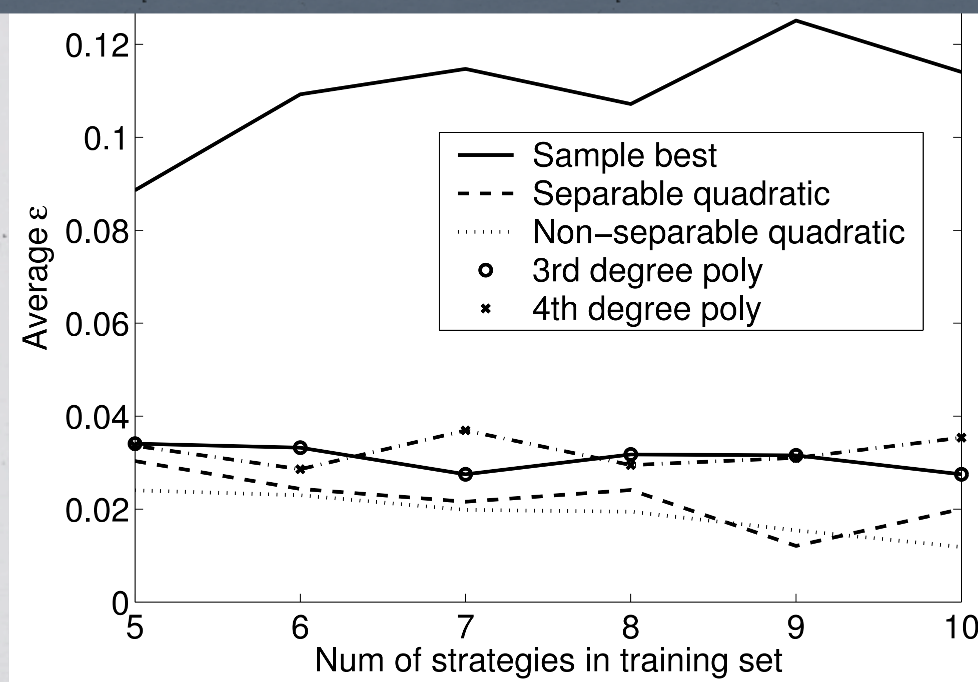
In practice, BR dynamics may be better

What Can We Do With Monte-Carlo Samples?

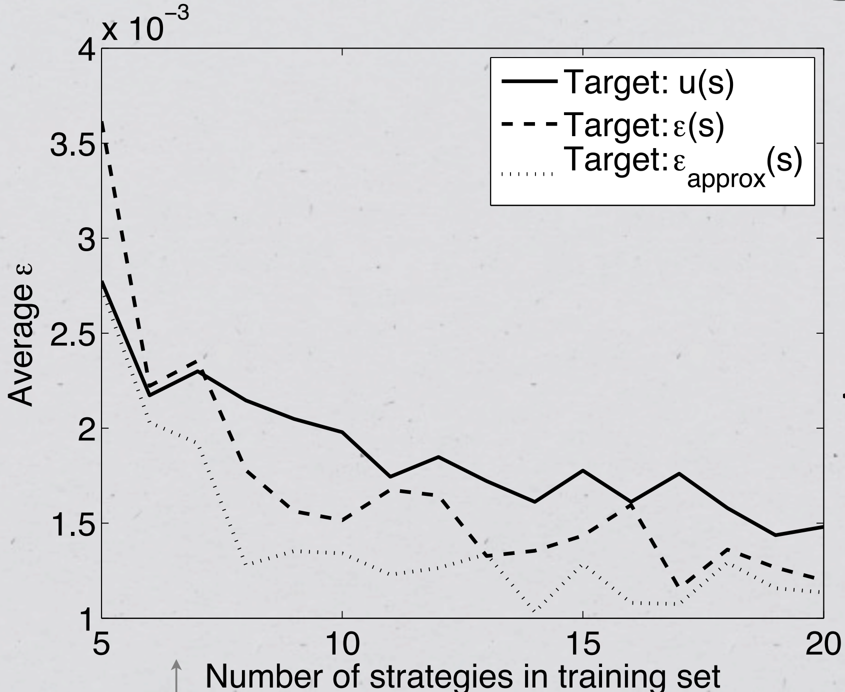
- Estimate a Nash equilibrium from profile - payoff tuples directly (use the profile in the data set with lowest game theoretic regret)
- Machine learning
  - Learn the payoff function and compute Nash equilibria based on the learned game model
  - Learn the regret function and compute Nash equilibria as its global minima

Empirical Analysis: Learning Payoffs vs. Direct Estimation

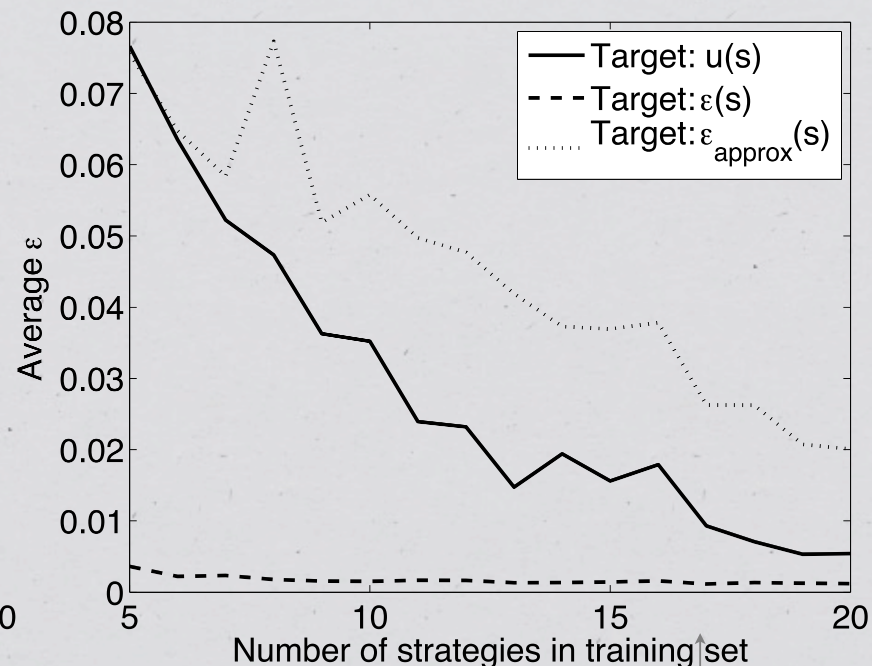
the method which learns payoff functions from data is substantially better than direct estimation



Learning Payoffs vs. Learning Regret



without noise



with noise

* when payoffs obtained from the payoff simulation contain *no noise*, using regret as the learning target is better
* when simulation payoffs *do* contain noise, using payoff function as the target is better

Takeaways

- Use of stochastic search techniques from OR can be very effective in estimating best responses and Nash equilibria in infinite games (e.g., in infinite Bayesian games)
  - Approximate regret minimization is provably convergent
  - Stochastic best response dynamics has very good empirical performance
- Once payoff data is obtained, can use machine learning to obtain better Nash equilibrium estimates than those obtained from data directly

Outline

- Motivating example: strategic procurement in a supply-chain game
- The two-stage model of mechanism design
- Mechanism design in a supply-chain game
- Automated mechanism design on constrained design spaces
- Solving simulation-based games
- Evolutionary and learning approaches to automated mechanism design

Evolution of Market Mechanisms (Cliff)

- Single continuous parameter based on continuous double auction market rules
- Uses GA as the parameter optimization routine
- Mechanism is evaluated based on the performance of ZIP agents
  - ZIP agent behavior is co-evolved together with the market using a GA
- Design objective: minimize deviation of transaction prices from competitive equilibrium

Evolution of CDA Pricing Rules (Phelps et al.)

- Search in the space of pricing rules in CDAs using genetic programming
- Approach 1: co-evolve bidder strategies together with auction rules
  - Design objective: a notion of economic efficiency
- Approach 2: mechanisms are evaluated w.r.t. the outcome of reinforcement learning strategies (Erev-Roth)
  - Objective: maximize efficiency, minimize trader market power

AMD Using "Evolutionary Game Theory" (Byde)

- Search in a one-dimensional space of 1-item auction mechanisms
- Design objective: revenue (although, in principle, can be generalized)
- Mechanisms evaluated using evolutionary game theory
  - Parametrized bid function evolved using a GA using utility from repeated play joint (with randomly generated types) as fitness
  - Breeding between genomes proportional to fitness

Metalearning (Pardoe, et al.)

- Assume a fixed population (distribution) of bidders
- No bidder participates more than once
- Design objective: revenue
- Adapt auction design to bidder behavior over a series of single-item auctions
- *Learn the parameters of the adaptive learning algorithm using bidding simulations*

Summary

- Stochastic search methods effective in parametrized mechanism design
- The key problem is predicting player *for a given mechanism choice*
  - Equilibria can define or form predictions, but are difficult to compute / approximate (above, one general method for infinite games is suggested)
  - No truly principled approach to the prediction problem besides Nash equilibria