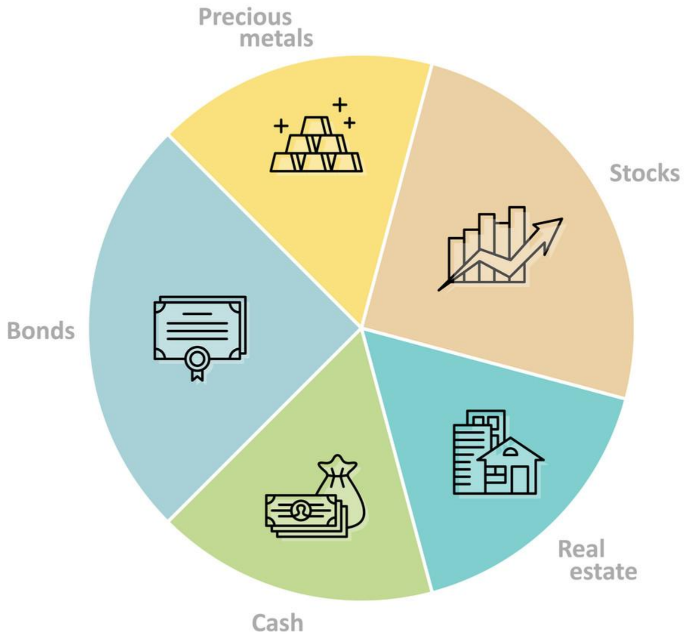
# Algorithmic Mechanism Design

Qianhui Wang

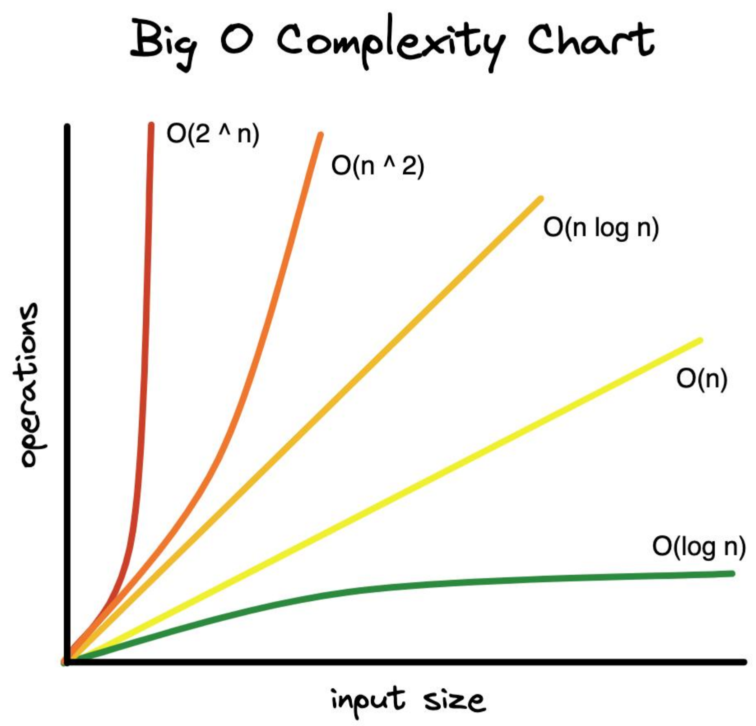
# Mechanism Design

“a field in economics and game theory that explores how businesses and institutions can achieve desirable social or economic outcomes given the constraints of individuals' self-interest and incomplete information.”



# Where Algorithm Comes In

| Economics | Computer Science |
|---|---|
| real-world Bayesian distributions<br>- Average Case Analysis | abstraction<br>- Worst Case Analysis |
| Solution exactness | Approximations |
| Ignore | Efficiency<br>- computation time<br>- memory<br>- communication |



# Strategic Models

- Rational Agents:
  - Utility Maximising  - Strategic Moves
  - Selfishness & Lying

- Utility Model:
  - Quasi-linear utility
    - Auction: Bidder i has valuation $v_i$
    - utility $u_i = v_i(m_i) - p_i$
  - Value transfer

# Combining Strategy and Computation

- Goal
  - Design truthful mechanisms
  - Run in polynomial time
  - Determine “optimal” social outcome


- Truthfulness:
  - aka **Incentive compatibility**
    - **Reporting false value is no better than true value**
  - Internalise “**Dominant strategy**” into the mechanism
    - Direct Revelation principle in Mechanism Design

# Multi-unit Auction


## Problem Definition

- Algorithmic Perspective:
  - m identical items
  - n bidders with private valuation functions (willingness to pay)
  - Give an efficient algorithm that outputs allocation $(m_1, m_2, \ldots, m_n)$
  - Maximize total social welfare $\sum v_i(m_i)$
  - Constraint $\sum m_i \leq m$

- Additional Strategic Considerations:
  - output payments $(p_1, p_2, \ldots, p_n)$
  - ensure truthfulness

UTILITY $u_i = v_i(m_i) - p_i$

# Memory Considerations

- Input Representation
  - valuations for m items

    $v_i(1), v_i(2), \ldots, v_i(m) \text{ for } \forall i$

m could be very large!

-> more succinct representations?

  - single value encompasses multiple meanings

## Examples

- step function $v(k) = \begin{cases} 0 \text{ for } k < k^* \\ p \text{ for } k \geq k^* \end{cases}$
  - only need **(k*, p)**
- piese-wise function $v(\text{item } k) = \begin{cases} p_1 \text{ for } k \leq k_1 \\ p_2 \text{ for } k_1 < k \leq k_2 \\ \ldots \\ p_t \text{ for } k_{t-1} < k \leq k_t \end{cases}$
  - **$t \ll m$** pairs of (k_i, p_i)

# Efficiency Requirements


- POLY time w.r.t. ?

  A theory perspective:
  - size of input encoding

Two data query models

1. concrete functions (**bidding language**)
   - length of all valuations
   - n, bidders
   - logm, #bits to represent #items we bid
   - t, #bits to represent a value
     - aka precision

2. black-box communication
   - no precise input representation

   - number of "**value-queries**"
   - length of answer bits

   - still require sublinear in m!

   eg, $\sqrt{m}, \ln m, \log m$

# General Allocation Algorithm

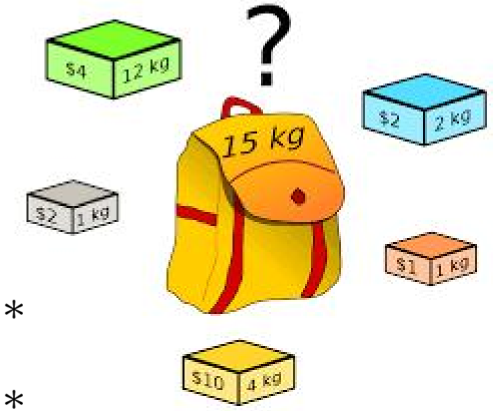
- Dynamic Programming (DP)
  - Divide-and-Conquer & Optimal Substructure
  - Subproblem:
    - Optimal allocation of first k items among the first i bidders
    - $s(i,k) = \max_{0 \le j \le k} v_i(j) + s(i-1, k-j)$

NOT computationally efficient!

$O(nm^2)$

| #bidders\ items | 0 | 1 | 2 | ... | m-1 | m |
|---|---|---|---|---|---|---|
| 0 | | | | | | |
| 1 | | | | | | |
| 2 | | | s(i, k) | | | |
| ... | | | | | | |
| n | | | | | | s(n, m) |

# Intractability

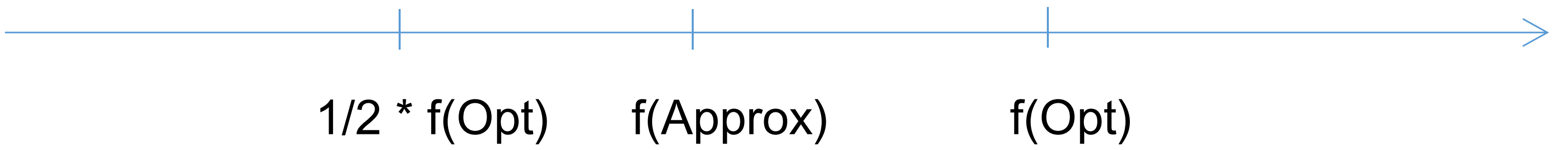



- Bidding language model
  - Consider the simplest step function $$v(k) = \begin{cases} 0 \text{ for } k < k^* \\ p \text{ for } k \geq k^* \end{cases}$$
  - "Knapsack problem"
    - player i = item i with value p;
    - m = knapsack capacity
    - allocate k items to player i = packing item i with weight k
    - $\sum k_i \leq m$
  - NP-complete
- Communication model O(m)
  - number of value queries
  - length of answer bits

# Approximate Optimality

- Fully polynomial-time approximation scheme (FPTAS)
  - additional parameter $\epsilon < 1$
    - specifies how close our approximation should be
  - additional polynomial time requirement $Poly(\frac{1}{\epsilon})$
  - for maximisation problem: $f(Approx) \geq (1-\epsilon) f(Opt)$

- $\alpha$ - approximation
  - for maximisation problem: $f(Approx) \geq \alpha * f(Opt)$



# Truncation

| #bidders/ total value constraint | 0 | 1 | ... | ... | n^2/e |
|---|---|---|---|---|---|
| 0 | | | | | |
| 1 | look at previous n/e cells in the row above | | | | |
| ... | | | | (i, w) | |
| n | | | | | |

- Previously, m columns in the table
  - O(m) tries for each subproblem
- A **FPTAS** by value truncation:
  - number of possible values $w \leq n/\epsilon$
- New subproblem:
  - minimum number of items that yields total value $\geq w\delta$
- carefully choose truncation precision $\delta = \epsilon * \max v/n$
  - total relative error less than $\epsilon$
  - $(1-\epsilon)$ approximation

# Truthful Auctions

## - VCG Mechansim

- Choice of payment function
  - Align players' max utility with total social welfare

$$u_i = v_i(m_i) - p_i = v_i(m_i) + \sum_{j \neq i} v_j(m_j) - \sum_{j \neq i} v_j(m'_j)$$

$$p_i = \sum_{j \neq i} v_j(m'_j) - \sum_{j \neq i} v_j(m_j)$$

optimal allocation that maximises other players' welfare

1. no positive transfer
2. individual rationality

- Sadly,
  - $O(nm^2)$ General Allocation Algorithm

# Truthful Approximation?

## - Counter Example

allocation that "approximately" maximises total welfare

- Idea:

$$p_i = \sum_{j \neq i} v_j(m'_j) - \sum_{j \neq i} v_j(m_j)$$

- 2 players, 2 items; v(1) = 1.9 and v(2) = 3.1
- Truncation scheme: $p_i = \sum_{j \neq i} v_j(m'_j) - \sum_{j \neq i} v_j(m_j)$
  - v(1) = 1 and v(2) = 3
  - Optimal allocation: 2 items to single player
    - payment = 3.1-0 = 3.1; net utility = 3.1-3.1 = 0!
  - false reporting v(1) = v(2) = 3
  - Optimal allocation: 1 item to each
    - payment = 3.1-1.9=1.2; net utlity = 1.9-1.2 > 0!

Not incentive compatible!

# Restricted VCG

## - Maximum in Range

- 2-approximation scheme
- Idea:
  - maximise over a restricted range of possible allocations
  - use General Allocation Algorithm
- bundles of items
  - manipulate m into some form in n
  - eg. $m/n^2$ items in a bundle -> $n^2$ items in total

the only approximation mechanisms that become truthful with the VCG payment rule.

# Proof

$m/n^2$

cannot be better than 2!

- bidder who gets the most number of items
  - Case 1: at least half total value from him

    -> give everything to him

$\geq \frac{1}{2} \sum v_i(m_i)$

  - Case 2: less than half from him

    -> take everything from him; distribute to all others

- our allocation always ≥ half total optimal

$\geq \frac{1}{2} \sum v_i(m_i)$

# Truthful Auctions
## - non- VCG Mechansim (Single Parameter)

- Two properties for Truthfulness
  - Monotonicity
    - win with bid (ki, vi) => win with bid (ki’ < ki, vi’ > vi)
  - Critical payment
    - the minimum value for a win with k items

$$v(k) = \begin{cases} 0 \text{ for } k < k^* \\ p \text{ for } k \geq k^* \end{cases}$$

- Idea:
  - still use $\delta$ scale truncation to keep range of values small -> efficiency
  - monotone way of choosing $\delta$ -> truthfulness
    - simultaneously try multiple values of $\delta$
    - independent of bids (k*, m) -> but trim down search space to $\frac{2n^2}{\epsilon}$
- **FPTAS!**

# Beyond

- Randomisation
  - maximise **expected** social welfare
  - maximum-in-distributional-range
  - FPTAS
    - reduce the number of values to consider for each player

- Combinatorial Auctions
  - m heterogenous items
  - allocation of a subset of items
  - valuation of all subsets

# References

"Algorithmic mechanism design", (Nisan, A. Ronen), Proceedings of IEEE Symp. on Foundations of Computer Science (FOCS), 2001

# Central Issue

with Algorithmic Mechanism Design

Algorithm Efficiency

CLASH

Truthfulness

# Beyond Multi-unit Auctions