Deep Learning Computation for Economic Theory and Its Applications

Prof. Joongheon Kim
Korea University, School of Electrical Engineering
Artificial Intelligence and Mobility Laboratory
https://joongheon.github.io
joongheon@korea.ac.kr

Research Collaborators

Korea University Members

MyungJae Shin
(Alumnus, SNU-Hospital)
Soohyun Park (Ph.D. Course)
Yeongeun Kang (Ph.D. Course)
Junghyun Kim (Ph.D. Course)




Collaborators

Prof. Marco Levorato
(CS@UC-Irvine)
Prof. Minseok Choi
(TE@Jeju Nat'l University)


Related Projects

Hanyang-ITRC (5G/Unmanned Vehicle Research Center)

- [PI] Hanyang University
- [WP2] Ajou University

Artificial Intelligence and Mobility Lab

Economics and Distributed Optimization

Deep Learning Solutions to Economic Theory

Applications to Distributed Systems

Summary

Artificial Intelligence and Mobility Lab

Linear Programming

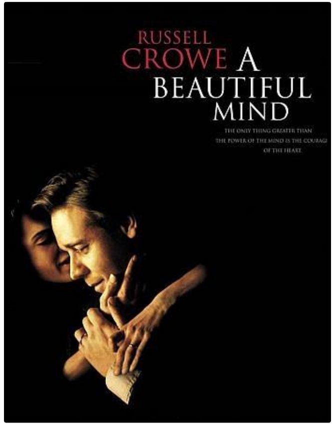
min: $c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$

subject to

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \le b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \le b_2$$
$$\cdots$$
$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \le b_m$$

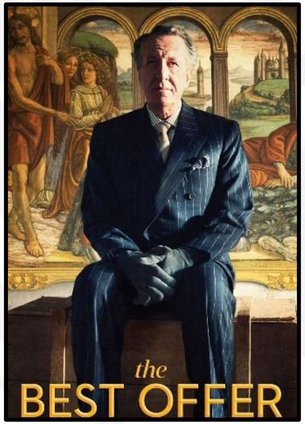
$$x_j \ge 0, j \in \{1, \cdots, n\}$$

Decision Making under Uncertainty



Game Theory

Stable Marriage



Auction

Game Theory

Pure Strategic Game (Strategic User)

MBC (KBS) Payoff Matrix		**KBS**		
		Movie	Opera	Comedy
MBC	Movie	35 (65)	15 (85)	60 (40)
	Opera	45 (55)	58 (42)	50 (50)
	Comedy	38 (62)	14 (86)	70 (30)

Game Theory

Pure Strategic Game (Strategic User)

MBC (KBS) Payoff Matrix		**KBS**		
		Movie	Opera	Comedy
MBC	Movie	35 (65)	15 (85)	60 (40)
	Opera	45 (55)	58 (42)	50 (50)
	Comedy	38 (62)	14 (86)	70 (30)

Betting ($2)?
Passing ($1)?

Player 1

High: 10, J, Q, K, A
Low: 2, …, 9

Player 2

Call ($2)?
Fold ($1)?

Betting ($2)?
Passing ($1)?

Player 1

High: 10, J, Q, K, A
Low: 2, …, 9

Player 2

Call ($2)?
Fold ($1)?

Player 1 Payoff Matrix		**Player 2**	
		Call	Fold
Player 1	PP	-1	-1
	PB	-21/13	3/13
	BP	2/13	-3/13
	BB	-6/13	1

Betting ($2)?
Passing ($1)?

Player 1

High: 10, J, Q, K, A
Low: 2, …, 9

Player 2

Call ($2)?
Fold ($1)?

Player 1 Payoff Matrix		**Player 2**	
		Call	Fold
Player 1	PP	-1	-1
	PB	-21/13	3/13
	BP	2/13	-3/13
	BB	-6/13	1

Player 1 Payoff Matrix		**Player 2**	
		Call	Fold
Player 1	~~PP~~	~~-1~~	~~-1~~
	~~PB~~	~~-21/13~~	~~3/13~~
	BP	2/13	-3/13
	BB	-6/13	1

Artificial Intelligence and Mobility Lab

Player 1 Payoff Matrix (Reduced)		Player 2 Call y_1	Player 2 Fold y_2
Player 1	BP x_1	2/13	-3/13
Player 1	BB x_2	-6/13	1

Maximizing the **profit** of Player 1

$$
\begin{aligned}
\text{maximize} \quad & \boxed{z} \\
\text{subject to} \quad & \frac{2}{13}x_1 - \frac{6}{13}x_2 \geq \boxed{z} \\
& -\frac{3}{13}x_1 + x_2 \geq \boxed{z} \\
& x_1 + x_2 = 1 \\
& x_1 \geq 0 \\
& x_2 \geq 0
\end{aligned}
$$

Expected **Profit** if Player 2 is doing **Call**

Expected **Profit** if Player 2 is doing **Fold**

Minimizing the **loss** of Player 2

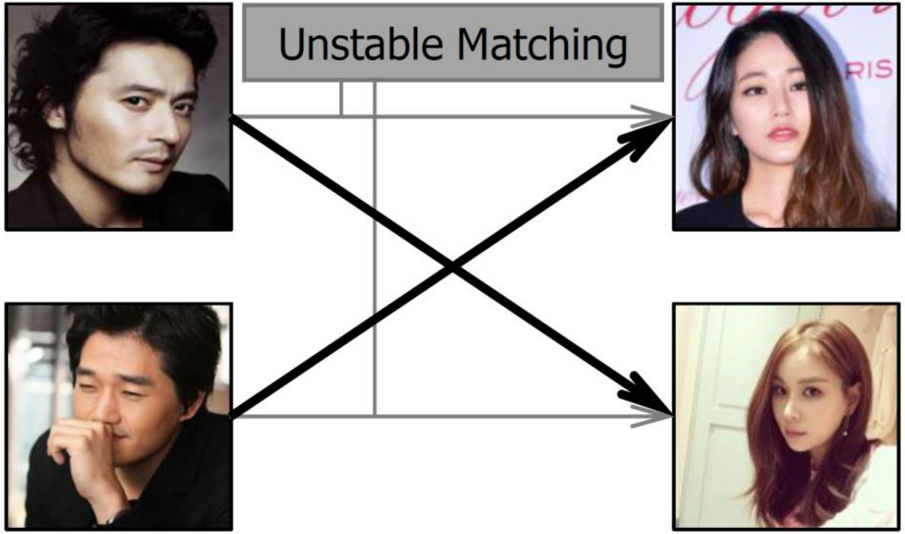
$$
\begin{aligned}
\text{minimize} \quad & \boxed{w} \\
\text{subject to} \quad & \frac{2}{13}y_1 - \frac{3}{13}y_2 \leq \boxed{w} \\
& -\frac{6}{13}y_1 + y_2 \leq \boxed{w} \\
& y_1 + y_2 = 1 \\
& y_1 \geq 0 \\
& y_2 \geq 0
\end{aligned}
$$

Expected **Loss** if Player 1 is doing **BP**

Expected **Loss** if Player 1 is doing **BB**

Stable Marriage

Introductory Example



Man	Preference List	
 Kim	 Jeon	 Jin
 Lee	 Jin	 Jeon

Woman	Preference List	
 Jeon	 Kim	 Lee
 Jin	 Kim	 Lee

Two Possible Matching

 Kim  Jeon	 Lee  Jin	**Stable**
 Lee  Jeon	 Kim  Jin	**Unstable**

Gale-Shapley Algorithm (GSA) Procedure

A	1 2 3 4
B	2 1 4 3
C	3 2 4 1
D	3 4 2 1

1	A C D B
2	C D A B
3	B A D C
4	A B C D

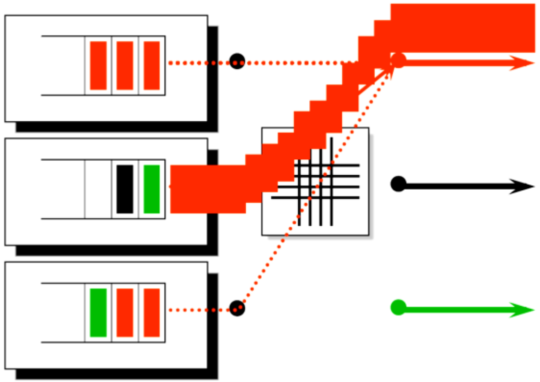
GSA Procedure

1. (A 1)
2. (A 1)(B 2)
3. (A 1)(B 2)(C 3)
4. 3 prefers D over C, i.e.,
   (A 1)(B 2)(D 3)
5. C’s next preference is 2.,
   2 prefers C over B, i.e., (A 1)(C 2)(D 3)
6. B’s next preference is 1.,
   The a does not want to switch.
7. B’s next preference is 4; and
   the 4 is free., i.e.,
   (A 1)(C 2)(D 3)(B 4)

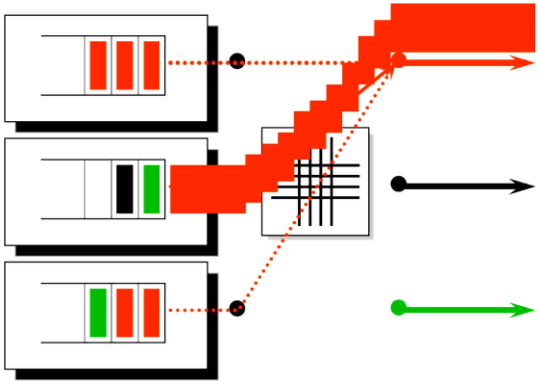
David Gale
PROFESSOR, UC BERKELEY

Lloyd Shapley
PROFESSOR EMERITUS, UCLA

Head-of-Line (HOL) Blocking

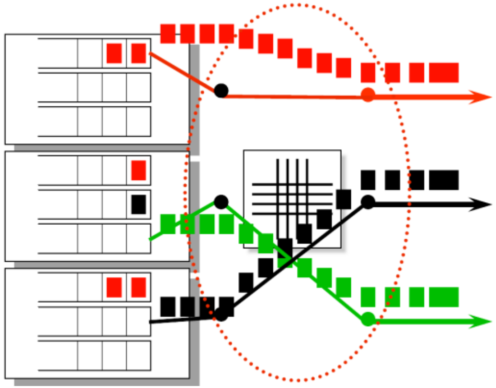


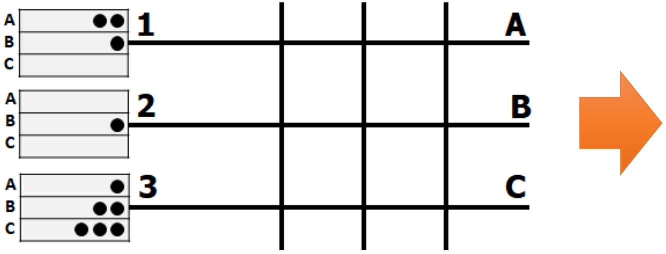
Head-of-Line (HOL) Blocking



Nick McKeown, Adisak Mekkittikul, Venkat Anantharam, and Jean Walrand, "**Achieving 100% Throughput in an Input-Queued Switch**," *IEEE Transactions on Communications*, Vol.47, No.8, August 1999.

Virtual Queue

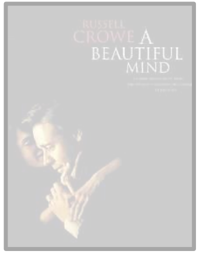




A	1 3 2
B	3 1 2
C	3 1 2

1	A B C
2	B A C
3	C B A

Auction



First Price Auction (FPA)

Payment: **$10**

Strategic (considering the others)

Truthful → No

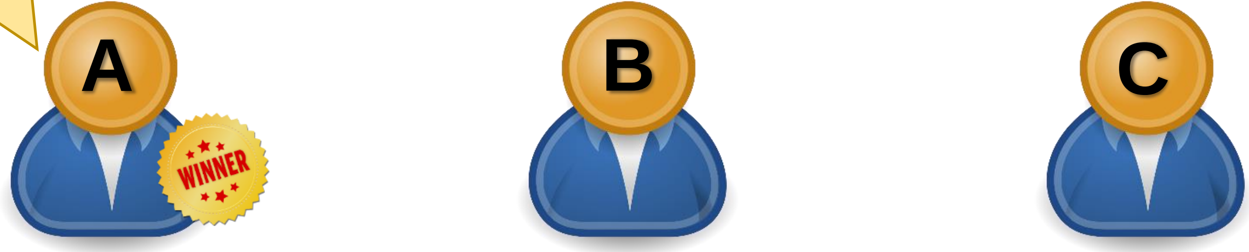
Auctioneer (Seller)
Revenue: $10

Utility (A) = Bid – Payment
= 10 - 10 = 0
Utility (B) = 8 - 0 = 8
Utility (C) = 6 - 0 = 6

Bid: $10

Bid: $8

Bid: $6



Second Price Auction (SPA)

Payment: **$8**

Truthful → Yes when all bids are truthful

Auctioneer (Seller)

Revenue: $8

Utility (A) = Bid – Payment = 10 - 8 = 2

Utility (B) = 8 - 0 = 8

Utility (C) = 6 - 0 = 6

Bid: $10

Bid: **$8**

Bid: $6

Payment

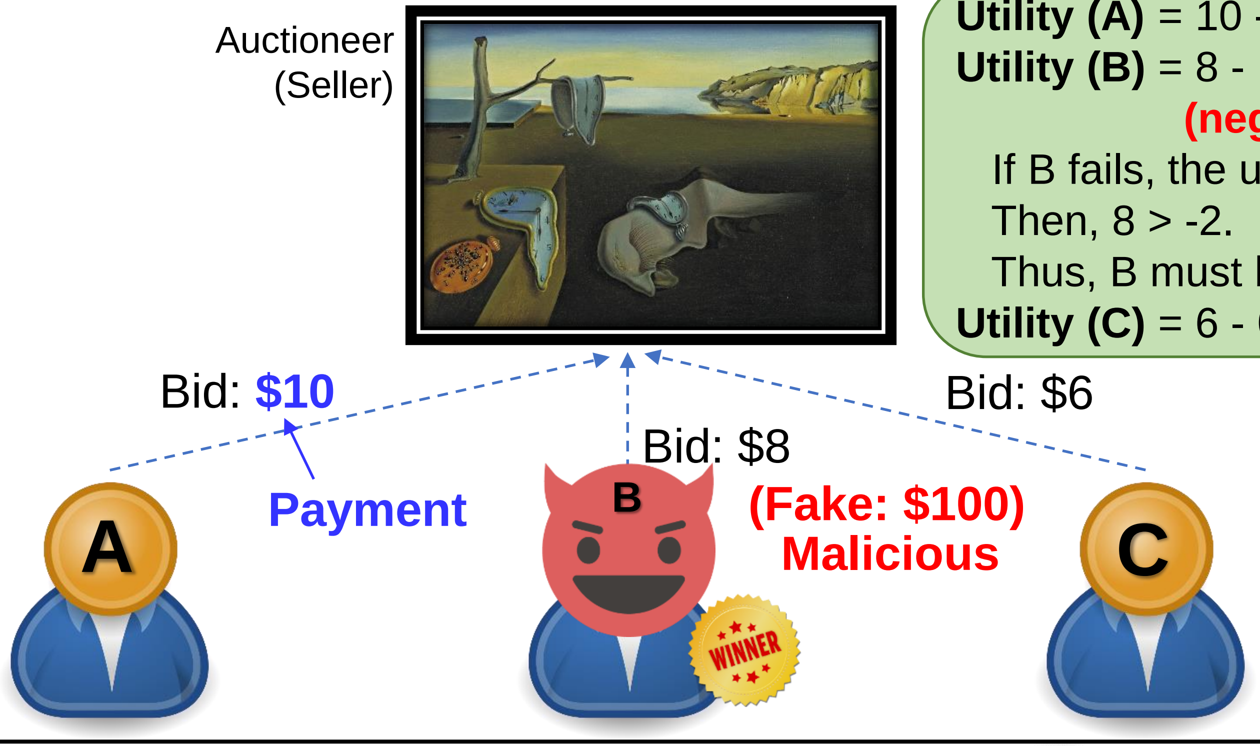
A

WINNER

B

C

Second Price Auction (SPA)



Utility (A) = 10 - 0 = 10
Utility (B) = 8 - 10 = **-2**
(negative)
If B fails, the utility is 8-0=8.
Then, 8 > -2.
Thus, B must be **truthful**.
Utility (C) = 6 - 0 = 6

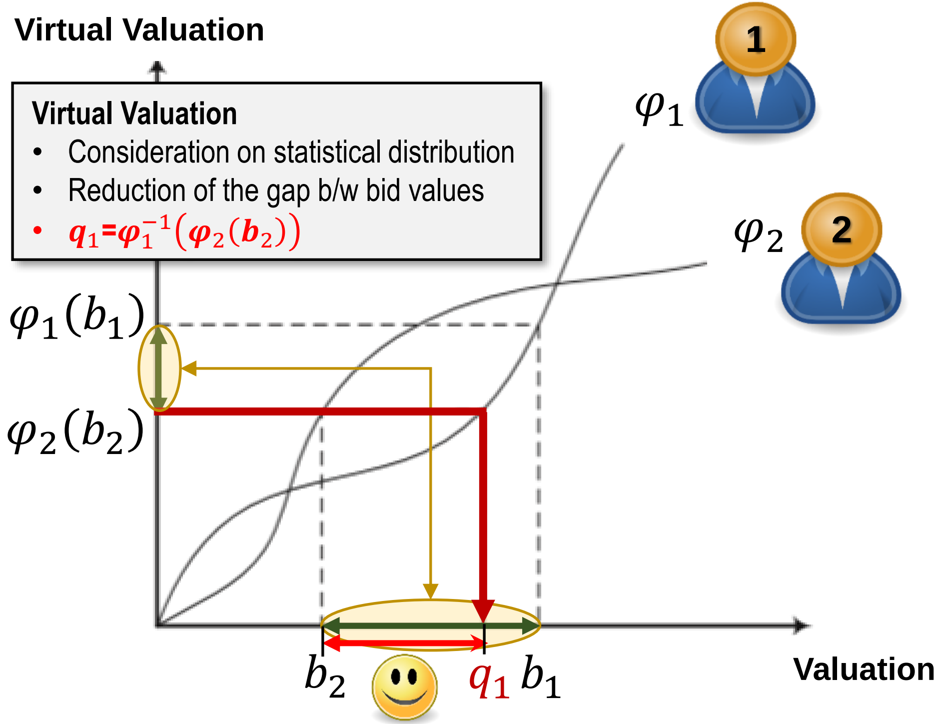
Auction (Basic Concepts)

- **Truthfulness:** The rule that induces true behaviors of players
- FPA vs. SPA
  - **FPA:** Revenue-Optimal (O) & Truthful (X) // Revenue: Payment to Auctioneer (Seller)
    - Revenue-Optimal: because the auctioneer will get the payment (the highest bid)
  - **SPA:** Revenue-Optimal (X) & Truthful (O)

DSIC, IR, and Optimal Auction

- **Dominant Strategy Incentive Compatibility (DSIC):** All participants should submit truthful bids. This guarantees the maximum utility. SPA.
- **Individual Rationality (IR):** All utilities should be non-negative.
- Optimal Auction → **Myerson Auction maximizes revenue in single-item auction.**
  - DSIC
  - IR
  - Revenue Maximization

Improving SPA (by increasing Revenue)



Definition (Virtual Valuation)

$$\varphi_i(b_i) = b_i - \frac{1 - F_i(b_i)}{f_i(b_i)}$$

- $f_i(b_i)$: probability for b_i
- $1 - F_i(b_i)$: probability for having a value higher than b_i

$\boldsymbol{v_i}$

- If it is the largest, $1 - F_i(b_i)$: It is 0, i.e., $\varphi_i(b_i) = b_i$
- If it is the smallest, $1 - F_i(b_i)$: It is 1, i.e., $\varphi_i(b_i) = b_i - \frac{1}{f_i(b_i)}$ and $0 < f_i(b_i) < 1$

Linear Programming

min: $c_1x_1 + c_2x_2 + \cdots + c_nx_n$

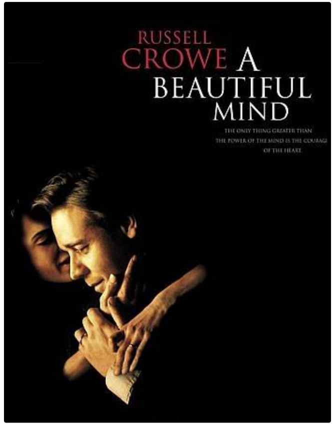
subject to

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \le b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \le b_2$$
$$\cdots$$
$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \le b_m$$

$$x_j \ge 0, j \in \{1, \cdots, n\}$$

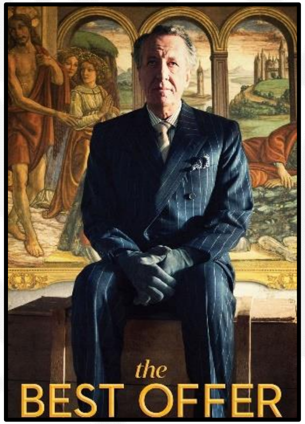
Decision Making under Uncertainty

Distributed Optimization



Game Theory

Stable Marriage



Auction

Economics and Distributed Optimization

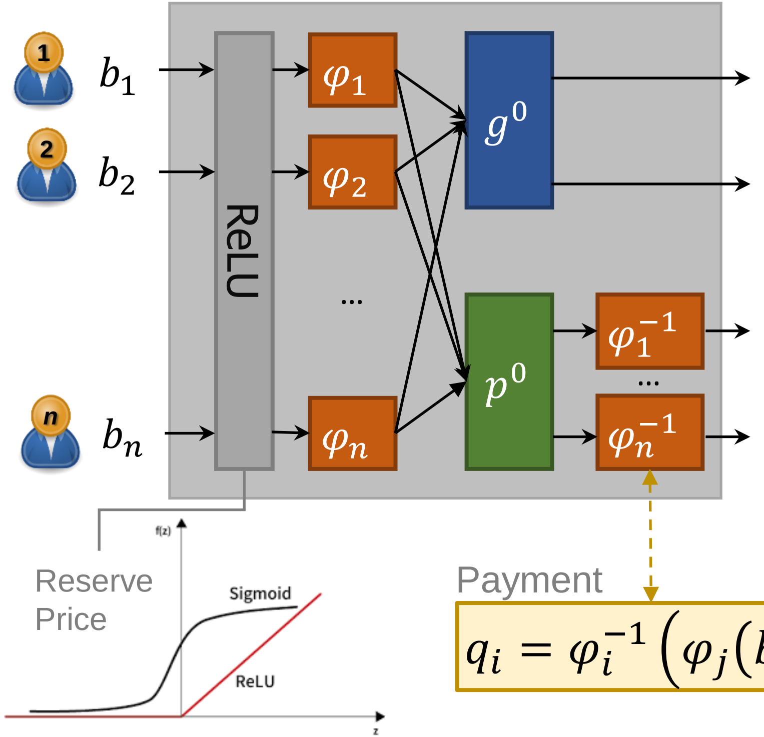
Deep Learning Solutions to Economic Theory

Applications to Distributed Systems

Summary

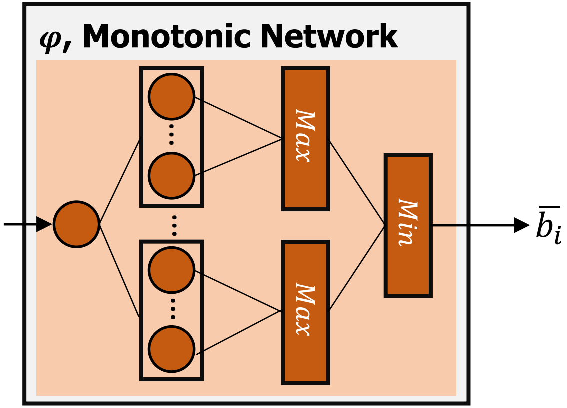
Artificial Intelligence and Mobility Lab

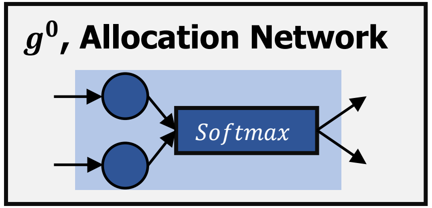
Auction Solution with Deep Learning Framework

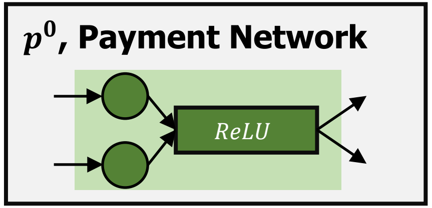


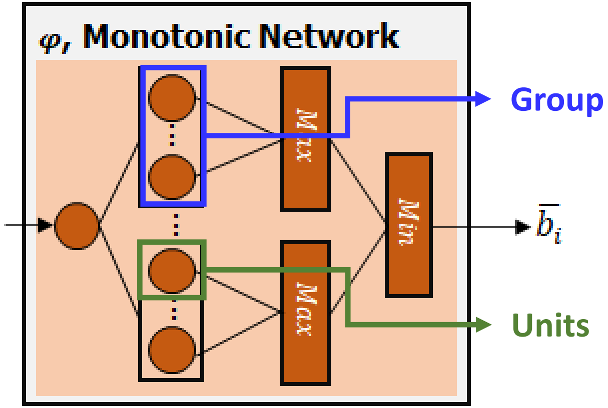
Payment

$$q_i = \varphi_i^{-1}\left(\varphi_j(b_j)\right)$$









$$\bar{\varphi}_{\mathrm{i}} = \min_{k \in K} \max_{j \in J} w_{k,j}^{i} b_i$$

Monotonic Network

- Monotone transformations $\varphi_1, \ldots, \varphi_n$ to the input bids $b_1, \ldots, b_n$
- Transformed bids $\overline{b_1}, \ldots, \overline{b_n}$ are fed into the SPA network.
- The monotonic network consists of K groups of J units.
- Each $\boldsymbol{w_{k,j}}$ has to be positive ($\boldsymbol{w_{k,j} > 0}$), i.e., $\boldsymbol{w_{k,j}^{i} = e^{\alpha_{k,j}^{i}}}$.
- **Monotonicity** of monotonic network
  - [Sill, 1998] proves that $\bar{\varphi}_{\mathrm{i}} = \min_{k \in K} \max_{j \in J} w_{k,j}^{i} b_i$ guarantees the monotonicity of monotonic network.
  - [Daniels, 2010] proves that $\bar{\varphi}_{\mathrm{i}} = \max_{k \in K} \min_{j \in J} w_{k,j}^{i} b_i$ guarantees the monotonicity of monotonic network.
  - [Yang, 1995] Min-max is used because it was proved that the min-max/max-min form guarantees monotonicity.
- The inverse transform φ^{-1} can be directly obtained from the parameters for the forward transform.

[1] J. Sill, "Monotonic Networks," In *Proc. NIPS*, pages 661–667, 1998.
[2] H. Daniels and M. Velikova, "Monotone and Partially Monotone Neural Networks," *IEEE Transactions on Neural Networks*, 21(6): 906-917 (2010).
[3] P.-F. Yang and P. Maragos, "Min-Max Classifiers: Learnability, Design and Application," *Pattern Recognition*, 28(6):879-899 (1995).

Monotonic Network

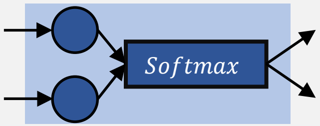
```
class monotonicNet(nn.Module):
    def __init__(self, numUser, numUnit, numGroup):
        super(monotonicNet, self).__init__()

        self.numUnit = numUnit
        self.numGroup = numGroup
        self.var = nn.Parameter(torch.randn((numUser, numGroup, numUnit), requires_grad=True))
        self.params = list(self.var)

    def forward(self, x):
        result = torch.mul(torch.exp(self.var), x)
        max = torch.max(result, dim=2)
        min = torch.min(max[0], dim=-1)
        return min[0]

    def getVars(self):
        return self.var
```

1. Makes variables ($\boldsymbol{w_{k,j}^i}$) of K groups of J units.
2. Transforms units $\boldsymbol{w_{k,j}^i}$ to $\boldsymbol{e^{\alpha_{k,j}^i}}$, and calculates $w_{k,j}^i b_i$.
3. Calculates $\bar{\varphi}_\mathrm{i}$ following $\min_{k\in K}\max_{j\in J} w_{k,j}^i b_i$.

Deep Learning Solutions to Economic Theory

Artificial Intelligence and Mobility Lab

g^0, Allocation Network



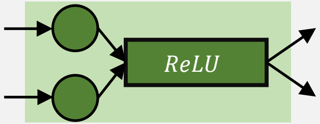
Allocation Network

- The SPA allocation rule g^0 can be approximated using a **softmax** function on transformed values $\bar{b}_1, \ldots, \bar{b}_n$

$$g_i = \frac{e^{\kappa \bar{b}_i}}{\sum_{j=1}^{N} e^{\kappa \bar{b}_j}}$$

```
class AllocNet(nn.Module):
    def __init__(self, numUser, k):
        super(AllocNet, self).__init__()
        self.numUser = numUser
        self.k = k
    def forward(self, x):
        x = torch.div(torch.exp(k*x),torch.sum(torch.exp(k*x)))
        return x
```

p^0, Payment Network



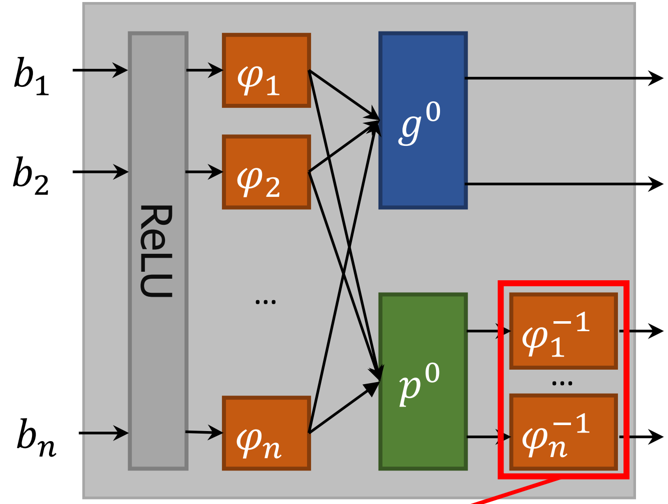
Payment Network

- The payment network consists of a **ReLU** function ensure positiveness of transformed values $\bar{b}_1, \ldots, \bar{b}_n$.

$$\mathrm{p}^0(\bar{b}_i) = ReLU(\bar{b}_1)$$

```
class PayNet(nn.Module):
    def __init__(self, numUser):
        super(PayNet, self).__init__()
        self.numUser = numUser
        self.rl = nn.ReLU()
    def forward(self,x):
        x = self.rl(x)
        return x
```

Actual Payment Computation (Inverse Monotonic Network)



Payment $\bar{\varphi}_i^{-1}(\bar{b}_i)$:

$$\bar{\varphi}_i^{-1}(\bar{b}_i) = \max_{j \in J} \min_{k \in K} \boldsymbol{e}^{-\alpha_{k,j}^{i}} \mathrm{p}^0(\bar{b}_i)$$

```
class BackMonotonicNet(nn.Module):
    def __init__(self, numUnit, numGroup, vars):
        super(BackMonotonicNet, self).__init__()

        self.numUnit = numUnit
        self.numGroup = numGroup
        self.vars = vars

    def forward(self, x):
        y = torch.ones(size=(numUser, numGroup, numUnit))
        for i in range(numUser):
            y[i, :, :] = x[i]
        result = torch.mul(torch.exp(-self.vars), y)
        min = torch.min(result, dim=1)
        max = torch.max(min[0], dim=-1)
        return max[0]
```

Allocation Network (Output)

Results (1st Auction Computation)					
Bids	-1.2201 negatives	-1.2892	-0.5140	**0.4430** Winner	**0.3734** SPA
Probs	0.0002	0.0002	0.0002	0.4972 Maximum	0.4822
Payment	0.0000 ReLU	0.0000	0.0000	**0.4428** SPA+DL	0.3329

- 0.4430 > **0.4428** **(~FPA)**
- **0.4428** > 0.3734 **(SPA+DL)**

Allocation Network (Output)

Results (2nd Auction Computation)					
Bids	**0.9578** Winner	0.3370	0.1259	-0.0669 negatives	**0.5899** SPA
Probs	0.4713 Maximum	0.0762	0.0115	0.0055	0.4355
Payment	**0.9514** SPA+DL	0.3825	0.0905	0.0000 ReLU	0.5269

- 0.9578 > **0.9514** **(~FPA)**
- **0.9514** > 0.5899 **(SPA+DL)**

- Deep learning auction removes the negative bids and determines the winner of the auction.
- The winner's payment does not exceed the winner's bid; and it is guaranteed to be higher than the second winning price.

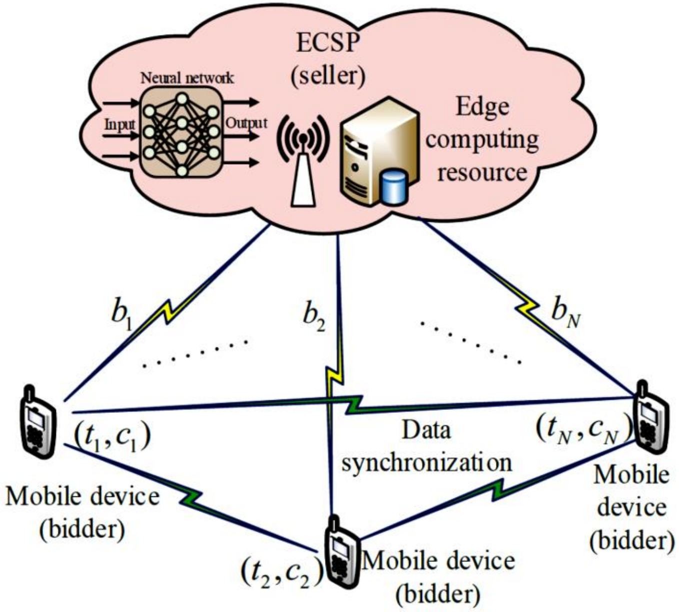
Economics and Distributed Optimization

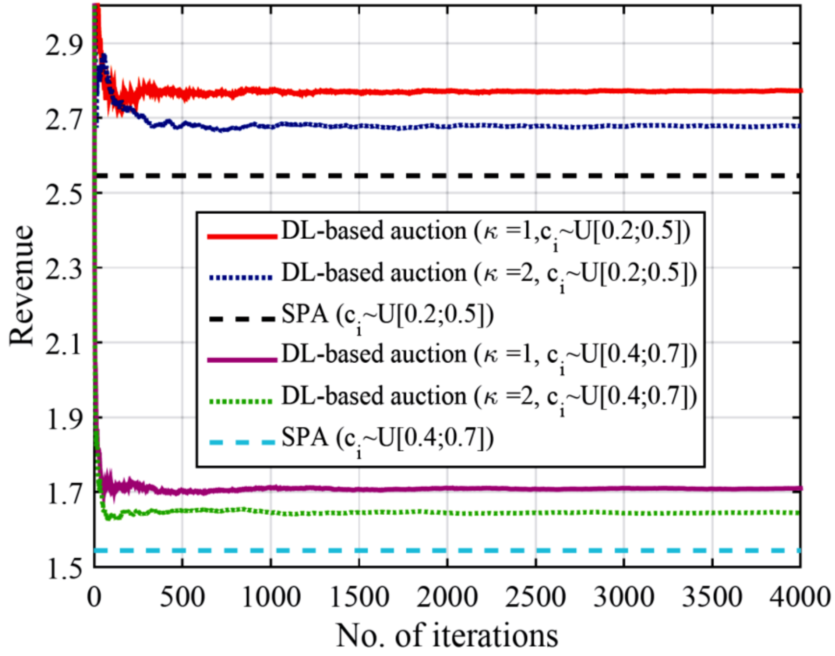
Deep Learning Solutions to Economic Theory

Applications to Distributed Systems

Summary

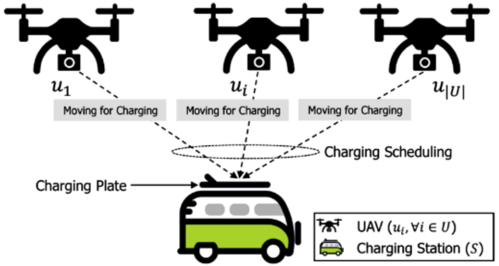
Mobile Blockchain Mining

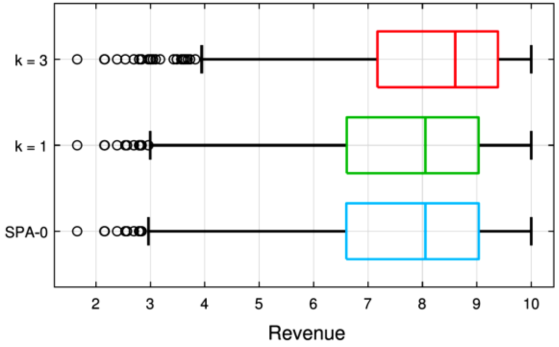


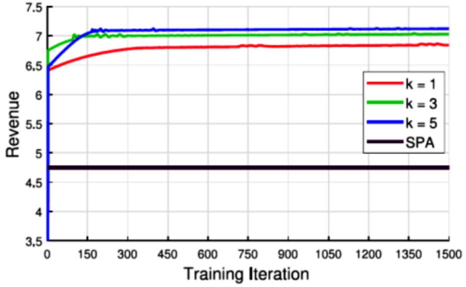


N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal Auction For Edge Computing Resource Management in Mobile Blockchain Networks: A Deep Learning Approach," *arXiv preprint*, *arXiv:1711.02844*, November 2017 (Preliminary version was presented at IEEE ICC 2018).

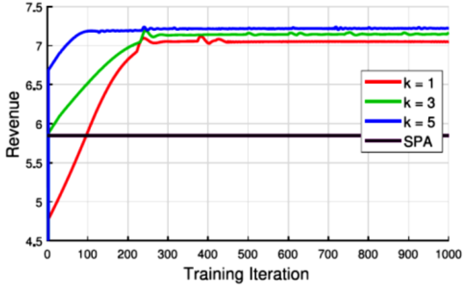
Multi-UAV Charging Scheduling



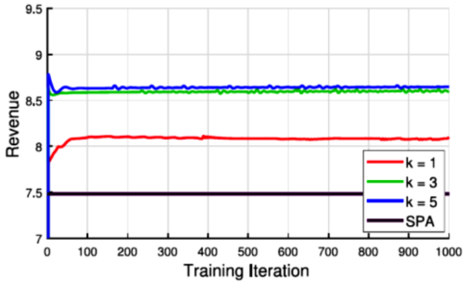




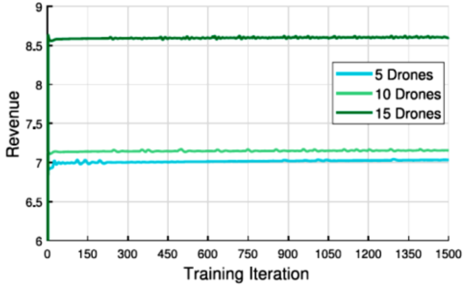
(a) Revenue statistics, 5 drones



(b) Revenue statistics, 10 drones



(c) Revenue statistics, 15 drones



(d) Revenue statistics, 5/10/15 drones, $k = 3$

M. Shin, J. Kim, and M. Levorato, "Auction-Based Charging Scheduling With Deep Learning Framework for Multi-Drone Networks," *IEEE Transactions on Vehicular Technology*, 68(5):4235-4248, May 2019.

Economics and Distributed Optimization

Deep Learning Solutions to Economic Theory

Applications to Distributed Systems

Summary

Summary

- **Economics and Distributed Optimization**
  - Game Theory
  - Stable Marriage
  - Auction
- **Deep Learning Solutions to Truthful/Revenue-Optimal Auction**
- **Applications to Distributed Platforms**
  - Mobile Blockchain Mining
  - Multi-UAV Charging Scheduling

- M. Shin, J. Kim, and M. Levorato, "**Auction-Based Charging Scheduling With Deep Learning Framework for Multi-Drone Networks**," ***IEEE Transactions on Vehicular Technology***, 68(5):4235-4248, May 2019., (Special Issue on Machine Learning-based Internet of Vehicle: Theory, Methodology, and Applications)
- L Park, S. Jeong, J. Kim, and S. Cho, "**Joint Geometric Unsupervised Learning and Truthful Auction for Local Energy Market**," ***IEEE Transactions on Industrial Electronics***, 66(2):1499-1508, February 2019.
- S. Jeong, W. Na, J. Kim, and S. Cho, "**Internet of Things for Smart Manufacturing System: Trust Issues in Resource Allocation**," ***IEEE Internet of Things Journal***, 5(6):4418-4427, December 2018.

Thank you for your attention!

- More questions?
  - joongheon@korea.ac.kr
- More details?
  - https://joongheon.github.io/ (Personal)
  - https://aimlab-kuee.github.io/ (Lab)
- Appendix
  - Auxiliary Parts of PyTorch Codes

Main

```
if __name__=="__main__":
    # Training Setting
    epoch = 100
    # Auction Setting
    numUser = 5
    numUnit = 3
    numGroup = 5
    k = 1

    Auction = AuctionNet(numUser, numUnit, numGroup, k)
    optimizer = optim.SGD(Auction.parameters(), lr=0.001, momentum=0.9)
    clipper = ZeroOneClipper()
```

Main

```
for i in range(epoch):
   optimizer.zero_grad()

   bid = np.random.randn(numUser)
   bids = np.ones(shape=(numUser,numGroup, numUnit))
   for k in range(numUser):
      bids[k, :, :] = bid[k]
   bids = torch.FloatTensor(bids)


   probs, pays, loss  = Auction(bids)
   loss.backward()
   optimizer.step()
   Auction.apply(clipper)
   print("Bids : ", bid, " / Probs : ", probs, " / Payment : ", pays)


   loss += loss.item()
   if i % 10 == 0:
      print('[{}] loss: {}.3f'.format(i, loss))
```

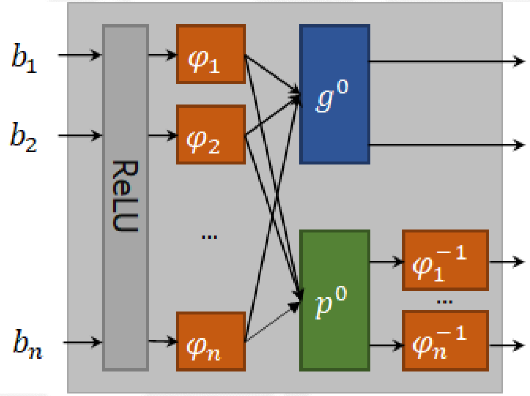
1
2
3

1. Makes Bids using Gaussian Distribution.
2. Executes auction algorithm.
3. Updates neural networks.

Overall Deep Learning Auction

```
class AuctionNet(nn.Module):
    def __init__(self, numUser, numUnit, numGroup, k):
        super(AuctionNet, self).__init__()

        self.numUser = numUser
        self.numUnit = numUnit
        self.numGroup = numGroup

        self.rl1 = nn.ReLU()
        self.monoNets = monotonicNet(numUser, numUnit, numGroup)
        self.backNets = BackMonotonicNet(numUnit, numGroup, self.monoNets.getVars())
        self.pay = PayNet(numUser)
        self.allocNet = AllocNet(numUser, k)
```

1



1. Call aforementioned models.

Weight Clipping

```
class ZeroOneClipper(object):

    def __init__(self, frequency=5):
        self.frequency = frequency

    def __call__(self, module):
        # filter the variables to get the ones you want
        if hasattr(module, 'weight'):
            w = module.weight.data
            w = w.clamp(0.01, 100)
```

- Pytorch's weight clipping method.
- Use proper weight clipping depending on the distribution of bid values.
- Disadvantages of deep learning based auction that have to rely on experimental results.

$$Loss = -g^0\left(\varphi_1(b_1), \dots, \varphi_N(b_N)\right)^T * [\overline{\varphi_1}^{-1}\left(p^0\left(\varphi_1(b_1)\right)\right), \dots, \overline{\varphi_N}^{-1}\left(p^0\left(\varphi_N(b_N)\right)\right)]$$

- The **incentive compatibility (IC)** constraint of auction can be restated as requiring the **expected ex post regret** for the auction to be 0 (Dutting, 2019).
- The **ex post regret** for each bidder is the extent to which an auction violates **incentive compatibility (IC).**
- we optimize the parameters using the negated revenue on bids $\boldsymbol{b} = (\boldsymbol{b_1}, \dots, \boldsymbol{b_N})$ as the error function.

[4] Dütting, Paul, et al. "Optimal auctions through deep learning." *International Conference on Machine Learning*. 2019.

Overall Deep Learning Auction

```
class AuctionNet(nn.Module):

    def forward(self, x):
        x = x.float()
        posBD = self.rl1(x)
        transBD = self.monoNets(posBD)
        probs = self.allocNet(transBD)
        pays = self.pay(torch.tensor(transBD))
        payment = self.backNets(pays)
        loss = - torch.sum(torch.mul(probs, torch.as_tensor(payment)))
        return probs, payment, loss
```

1

2

3

1. Calculates g^0 and p^0.
2. Calculates $\bar{\varphi}^{-1}(\bar{b}_i) = \max_{j\in J}\min_{k\in K} e^{-\alpha_{k,j}^i} p^0(\bar{b}_i)$
3. Calculates loss.

1. Calculates g^0 and p^0.
2. Calculates $\bar{\varphi}^{-1}(\bar{b}_i) = \max_{j\in J}\min_{k\in K} \boldsymbol{e}^{-\alpha_{k,j}^{i}} \mathrm{p}^0(\bar{b}_i)$
3. Calculates loss.