Computation Offloading and Activation of Mobile Edge Computing Servers: Minority Game Models

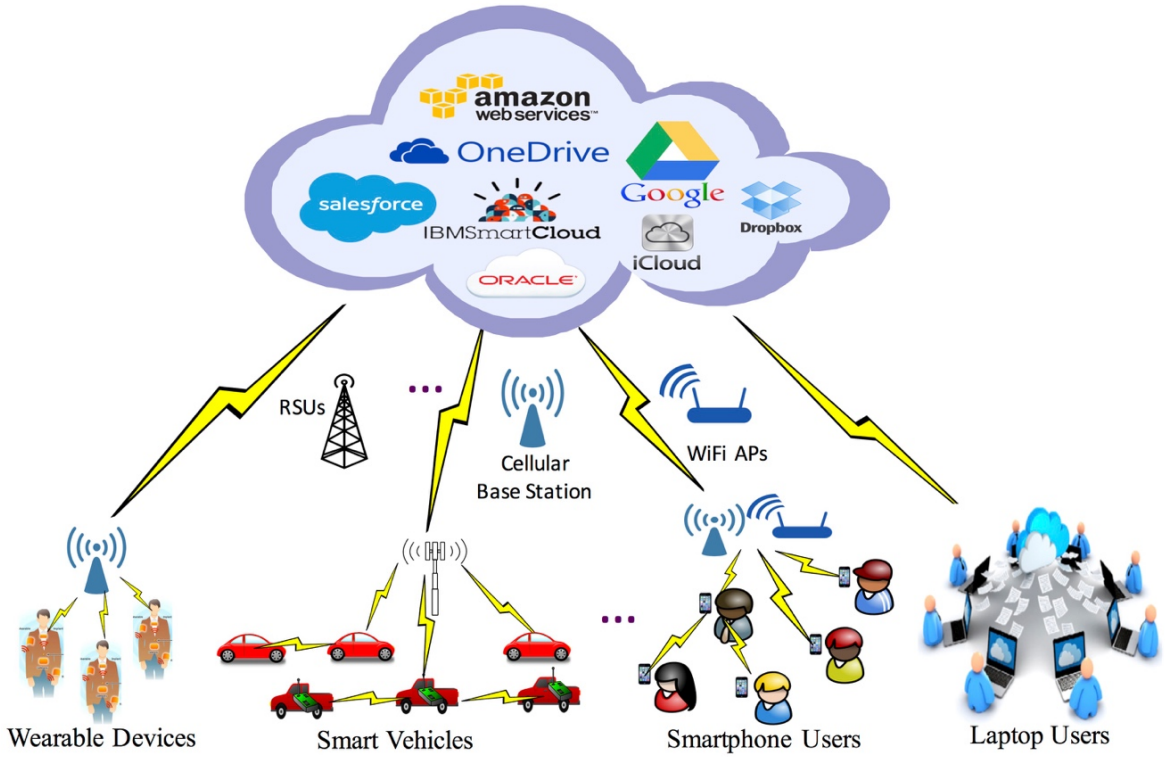
Ekram Hossain

21 October 2020

Outline

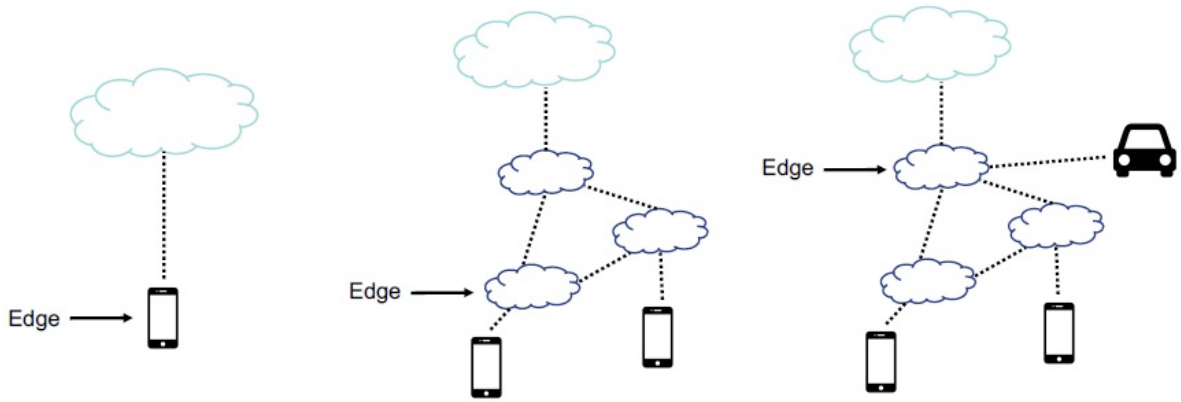
- Introduction
- Basics of Minority Game (MG)
- Applications of MG
  - Computation Offloading
  - Activation of MEC Servers
- Open Problems and Future Research Directions

Cloud computing



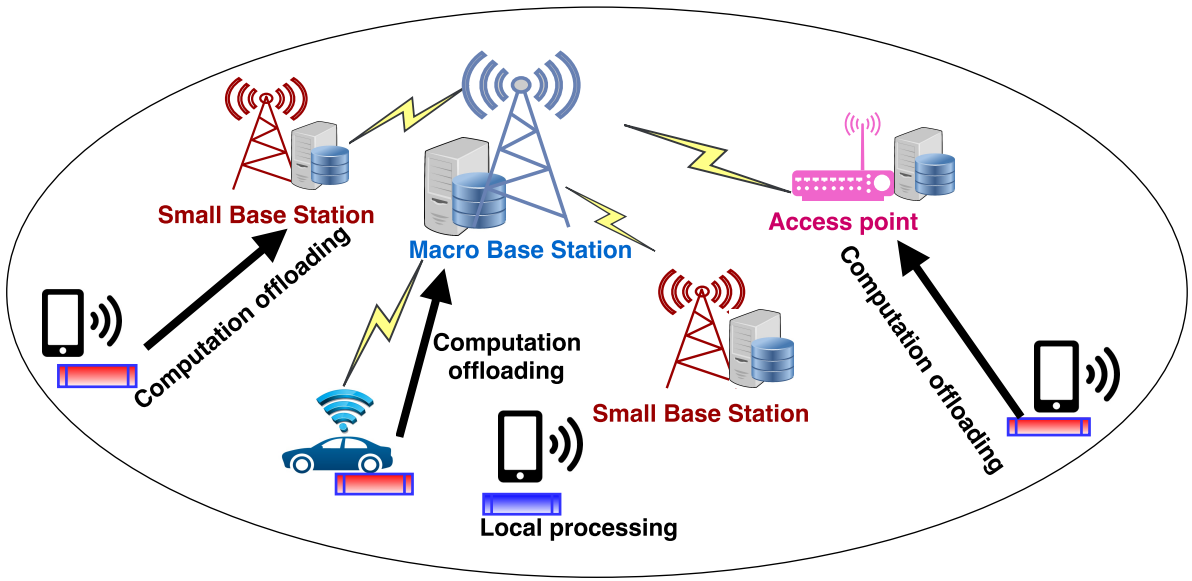
- Computing services through Internet
- Resources reside in the “cloud" and use them as needed

Edge computing



- Bring the computation facilities closer to the source of data
- Extends the cloud to the network edge
- Not a replacement of the cloud

Mobile edge computing



Computation offloading in small cell networks (SCNs)

Problem description

- Users have computationally-intensive tasks, e.g. image processing, augmented reality
- User devices have *limited computational capability and limited battery capacity*.
- **Computational offloading**: transfer expensive tasks to a remote server with higher computational capability.
- Saves the energy cost spent for local execution, thus the battery life of user devices.

S. Ranadheera, S. Maghsudi, and E. Hossain, “Minority games with applications to distributed decision making and control in wireless networks," *IEEE Wireless Communications*.

Computation offloading in SCNs

Problem description

- Computation offloading using cloud imposes challenges.
- Cloud servers
  - Typically located outside the local network
  - Large communication costs, high latency
  - Transmission energy cost
- Dense SCNs with large number of users cause large number of offloading requests.
  - Huge delays, increased backhaul traffic
- **Computational offloading to nearby SBSs (mobile-edge offloading) can be a better alternative**.
- SBSs located in close vicinity to users $\Longrightarrow$ minimizes latency, relieves the backhaul

Computation offloading in SCNs

Problem description

- Computation offloading is a dynamic resource allocation problem.
- Users *non-cooperatively* and *selfishly* try to utilize the limited computing resource.
- Users may not be able to communicate with each other.
- Users do not know about others' actions.
- Centralized control for scheduling offloading requests is *computationally expensive, causes large signaling overhead.*
- Hence, solutions should be **self-organizing, distributed and able to work with minimal external information**.

Activation of Mobile Edge Computing Servers

Problem description

- Small scale computing servers located in network edge for computation offloading, known as Mobile Edge Computing (MEC) servers
- Provides reduced latency and energy cost, compared to remote cloud
- **Efficient utilization of MEC servers is imperative, since MEC servers have limited computational resources and power**.
- **Activate only a specific number of servers while keeping the rest in the energy saving mode**.
- **Users' latency requirements** should also be met.

P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, 3rd quarter 2017.

Origin of minority game: Santa Fe ("El Farol") bar problem

- MG originates from *El Farol bar problem*
  - A college bar
  - Plenty of students like to attend the bar once every week
  - Every student enjoys the bar only if the attendance is 60 percent or less.
  - What should the student do?

Minority game: Basics

- Originally proposed in by Challet and Zhang*
- **Odd number** of anonymous players
- Players select between **two alternatives**
- Players selecting the **minority option** receive a pay-off
- Players do not communicate or cooperate

*D. Challet and Y.C. Zhang, "Emergence of cooperation and organization in an evolutionary game," *Physica A: Statistical Mechanics and Its Applications*, vol. 246, no. 3, pp. 407–418, 1997.

Minority game: Basics

- A simple non-cooperative congestion game
- Selfish players want to maximize their own pay-off.
- Players are *boundedly rational* thus use *inductive reasoning* to make decisions.
- Individual player interacts with *aggregate behavior* of all other players.

Minority game: Basics

Everyday examples:

- Drivers selecting less crowded roads
- People selecting less crowded restaurants
- Market traders deciding whether or not to join a market

“Wireless" examples:

- S-ALOHA based channel access
- Interference management in cognitive radio networks:
  - Secondary users play an MG, selecting between two options; *transmit* or *not transmit*.
  - Winning group is determined based on the interference experienced by the primary user.
- Transmission mode selection problem: D2D mode or cellular mode

Minority game: Basics

Players, Actions, Attendance, Utility

- A repeated game with N number of players
- *Cut-off* value (ϕ) or *minority* can be arbitrary, e.g. for odd N, minority can be defined at $(N-1)/2$
- Players choose between two actions (say, 1 and 0) in each round t.
- **Attendance** $n(t)$: Sum of actions of all players
- Players selecting the minority action will have higher utility.
- **Example utility of a player for actions '1' an '0'**:

$$U_1(t) = \begin{cases} 1, & \text{if } n(t) \leq \phi \\ 0, & \text{if } n(t) > \phi. \end{cases} \qquad U_0(t) = \begin{cases} 1, & \text{if } n(t) > \phi \\ 0, & \text{if } n(t) \leq \phi. \end{cases}$$

- After each round t, winning action is broadcast to all players.

Minority game: Basics

Example

- 5 players and minority threshold = 2

Go to bar (1)	Stay home (0)	Attendance	Total utility
5	0	5	0
4	1	4	1
3	2	3	2
2	3	2	2
1	4	1	1
0	5	0	0

- When attendance is equal to cut-off value, total utility is maximized.

Minority game: Basics

Example

- 6 players and minority threshold = 4

Go to bar (1)	Stay home (0)	Attendance	Total utility
6	0	6	0
5	1	5	1
4	2	4	4
3	3	3	3
2	4	2	2
1	5	1	1
0	6	0	0

- When attendance is equal to cut-off value, total utility is maximized.

Minority game: Basics

Strategies, brain size, history string

- Players use **strategies** to make decisions.
- A strategy predicts the winning action of the next round based on winning actions of previous m rounds ($m =$ **brain size**)
- A strategy maps m-bit **history string** to an action.
- An example strategy table for an agent:

History string	Predicted winning action
0 0	1
0 1	1
1 0	0
1 1	0

- $2^{(2^2)} = 16$ possible strategies
- For $m = 3$, $2^{(2^3)} = 256$ possible strategies

Minority game: Basics

Strategy space

- Universal strategy space (USS)
  - Has 2^{2^m} total strategies (very large even for small m)
- Reduced strategy space (RSS)
  - e.g. 2^{m+1} total strategies ($2^{m+1} \ll 2^{2^m}$)
  - May not have significant impact on MG dynamics

Minority game: Basics

- **Volatility σ**
  - Measures the fluctuation of *Attendance* around cut-off
  - An inverse measure of the system's performance
  - **Smaller volatility $\rightarrow$ smaller fluctuations $\rightarrow$ larger size of the minority $\rightarrow$ larger number of winners**

Minority game: Basics

Strategy reinforcement learning: Seminal MG learning algorithm*

- At the outset, each player draws a set $\mathcal{S}$ of S strategies from RSS.
- There is no a priori best strategy.
- Players evaluate their strategies as game is played.
  - Strategies are given points for predicting accurately.
  - Poorly performing strategies are penalized.
- On each play, the players use the strategy with the **highest score**.

*C. H. Yeung and Y. C. Zhang, "Minority games," in *Encyclopedia of Complexity and Systems Science*, A. R. Meyers, Ed. New York, NY: Springer New York, 2009, pp. 5588–5604.

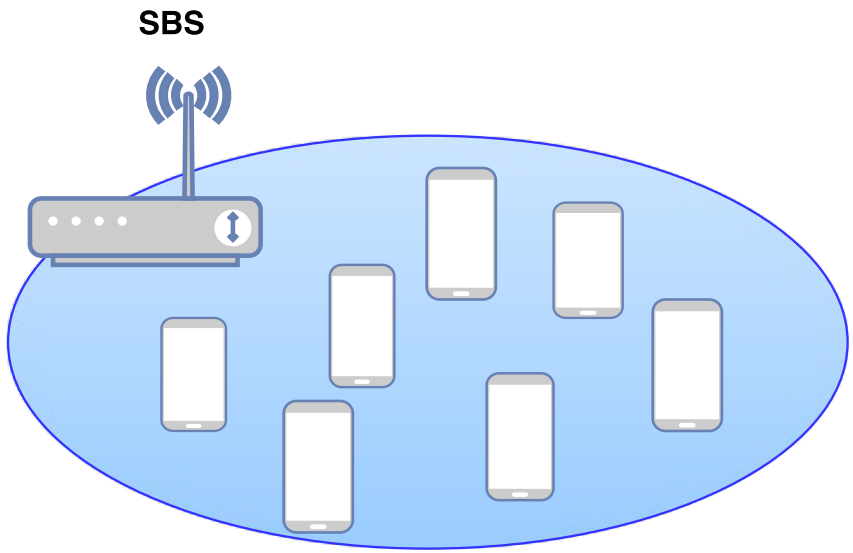
*D. Challet and Y. C. Zhang, "Emergence of cooperation and organization in an evolutionary game," *Physica A: Statistical Mechanics and its Applications*, vol. 246, no. 3, pp. 407–418, 1997

Motivations for applying MG in wireless problems

- Can accommodate a **large number of agents**.
- Essentially a **congestion game**, thus users competing for shared resources can be easily modeled.
- Involves **self-organized decision making** with **minimal external information**.
- Involves **anonymous** players.
- Has **no pairwise interactions** among the agents.

Computation offloading in SCNs

System model



- A small cell with N **homogeneous** users (i.e. all user devices have similar computational capability C_u).
- All users are assumed to have **homogeneous** tasks to be offloaded to SBS.

Computation offloading

System model and MG formulation

- **Objectives**:
  - Maximize users' utilities (latency minimization)
- Each computation offloading period = one round of MG
- Small cell users = players of MG
- Users have **two options**:
  - **Offload the task to the local SBS** (denote by '1')
  - **Execute the task locally** (denote by '0')
- SBS processes all offloading requests in a round-robin manner.

Computation offloading

Defining the cut-off value

- $n(t)$ = Attendance (number of offloading users at t)
- $L(t)$ = Latency experienced by an offloading user at t (worst-case)
- C_b = Computation capability of SBS (# of CPU cycles per unit time)
- C_u = Computation capability of user device (# of CPU cycles per unit time)
- M = Number of CPU cycles required to complete the task
- L_{th}= Local computation latency (maximum tolerable latency)

Computation offloading

Defining the cut-off value

- **Cut-off value of MG**, ϕ
  - ϕ = number of offloading requests SBS can handle before $L(t) = L_{th}$
  - Since $L(t) = n(t) \cdot M/C_b$ and $L_{th} = M/C_u$,

    $\frac{n(t) \cdot M}{C_b} \leq \frac{M}{C_u} \implies n(t) \leq C_b/C_u (= \phi)$.
  - For a user to benefit from offloading, $n(t) \leq \phi$.
- **Note**: ϕ affects outcome of the game and the payoffs of the players, and only the outcome of the game is observable to the users

Computation offloading

MG model

- **MG outcomes**
  - **If minority offloads and majority computes locally**: Minority wins since SBS is uncrowded.
  - **If minority computes locally and majority offloads**: Majority loses since SBS is crowded.
- **Control information**: SBS broadcasts the winning choice in each t by broadcasting $b(t)$,

$$b(t) = \begin{cases} 1, & \text{if } n(t) \leq \phi \text{ // offloaders win} \\ 0, & \text{if } n(t) > \phi \text{ // non-offloaders win} \end{cases}$$

Computation offloading

MG model

- **Utility of users**:

$U_o(t)$ = Utility for offloading and $U_l(t)$ = utility for local computing

$$U_o(t) = \begin{cases} 1, & \text{if } n(t) \leq \phi \\ 0, & \text{if } n(t) > \phi. \end{cases} \qquad U_l(t) = \begin{cases} 0, & \text{if } n(t) \leq \phi \\ 1, & \text{if } n(t) > \phi. \end{cases}$$

- **Solution method**
  - A basic strategy reinforcement technique is used (**Algorithm 1**).
  - MG based method is compared with a **random selection scenario** where users make their decisions randomly.

Reinforcement learning algorithm

Algorithm 1 Distributed learning algorithm

Initialization Each user i randomly draws S strategies. For every $s \in \mathcal{S}$, set the score $V_{i,s}(0) = 0$.

for t=1:T **do**

If $t = 1$, select strategy $s_i(1)$, at random from set $\mathcal{S}$. Otherwise, select strategy $s_i(t)$ with the highest score.

Each user i selects action $a_i(t)$ predicted by $s_i(t)$.

SBS broadcasts control information $b(t)$ to all users.

for s=1:S **do**

Each user i updates the score of the strategy s, $V_{i,s}$

if prediction of $s = b(t)$ **then**

$$V_{i,s}(t+1) = V_{i,s}(t) + 1$$

else

$$V_{i,s}(t+1) = V_{i,s}(t)$$

end if

end for

end for

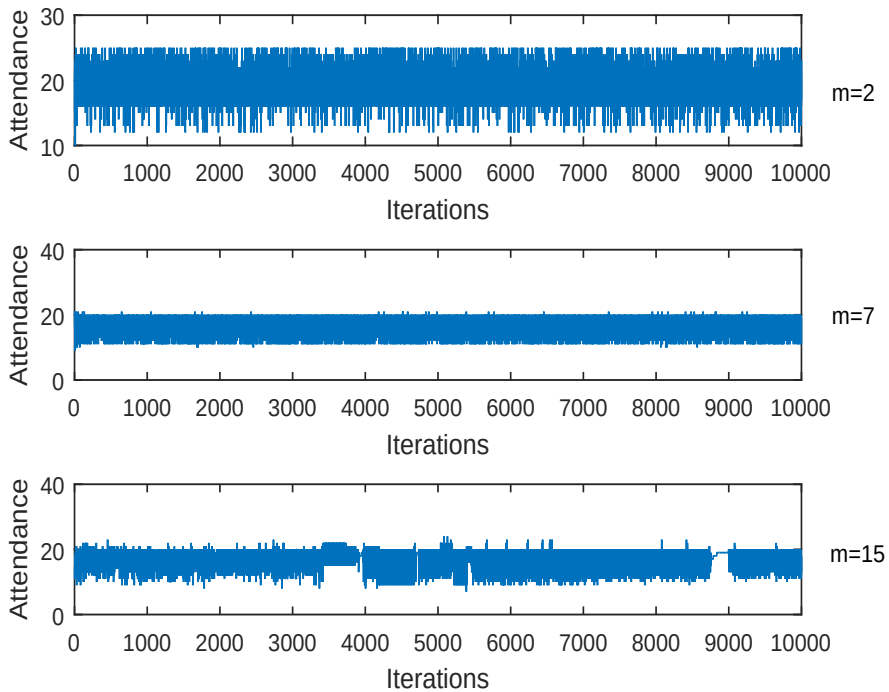
Computation offloading in SCNs

Simulation parameters

- Number of users associated to SBS, $N = 31$
- $C_u = 0.5$ MHz
- $C_b = 10$ MHz
- System's cut-off value, $\phi = C_b/C_u = 20$

Computation offloading

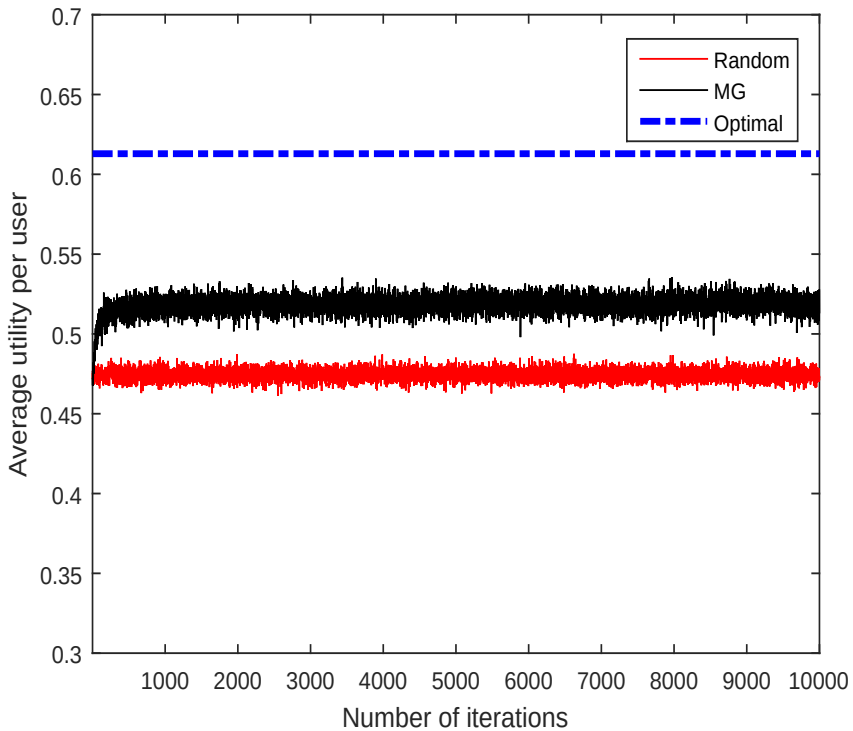
Simulation results: Time evolution of attendance



- When users play an MG, $n(t)$ always fluctuates near ϕ.
- System self-organizes into a state where # of users who experience offloading latency less than threshold is always near its maximum.
- Implies better SBS utilization

Computation offloading

Simulation results: Average utility received by users



- Average utility achieved in MG-based method is better than that of random choice game.
- Still, utility received by a user in MG-based method is lower than that of the optimal case.
- This is due to lack of coordination among agents and use of minimal external information in MG.

Activation of MEC servers

Contribution

- Most of the existing work address user-centric objectives (meeting delay constraints, minimizing users' energy consumption).
- We consider from both servers' and users' perspectives.
- Based on MG, we develop a mode selection mechanism to guarantee minimal server activation while meeting the users' delay constraints.

Activation of MEC servers

System model

- A MEC system consisting of a virtual pool of M computational servers (e.g. small cell base stations)
- Users (e.g. mobile devices) have delay sensitive computational jobs to be completed in consecutive offloading periods.
- Each offloading period is considered one *time slot*.
- K_T = Total # of jobs arriving at server pool
- Prior to task arrival, every server independently decides whether to
  - **accept computation jobs (*active* mode)**; or
  - **not to accept any computation job (*inactive* mode)**.

Activation of MEC servers

- **Servers' perspective**:
  - Every active server requires at least $k_{\min}$ jobs to earn a threshold revenue R_{th}.
  - For servers, it is beneficial to be active if $c(t) \leq c_{\max}$, where $c_{\max} = \frac{K_{\mathrm{T}}}{k_{\min}}$.
- **Users' perspective**:
  - Users prefer servers to process at most $k_{\max}$ jobs so that their desired QoE is fulfilled.
  - For users, $c(t) \geq c_{\min}$, where $c_{\min} = \frac{K_{\mathrm{T}}}{k_{\max}}$
- For efficient operation, $c_{\min} \leq c(t) \leq c_{\max}$
- E.g. $K_{\mathrm{T}} = 100$ jobs, $k_{\min} = 10$ jobs, and $k_{\max} = 20$ jobs $\Longrightarrow c_{\max} = 10$ and $c_{\min} = 5$

Activation of MEC servers

- **Selection of MG cutoff value ϕ:**
  - If $c_{\min} \leq c_{\max}$, choose $\phi = c_{\max}$ or $\phi = c_{\min}$.
  - If $c_{\min} > c_{\max}$, choose $\phi = c_{\min}$ and **increase payment to the sever** to make $c_{\max} = c_{\min}$
- **Example**: $K_{\mathsf{T}} = 100$ jobs, $k_{min} = 10$ jobs, $k_{max} = 5$ jobs

$\Longrightarrow c_{\min} = 20$ and $c_{\max} = 10$

$\Longrightarrow$ Choose $\phi = 20$ and increase payment/job fo the server

Activation of MEC servers

MG formulation

- The challenge is to activate ϕ servers in a self-organized manner.
- **Servers play a mode selection MG.**
- M servers = players of MG
- ϕ = MG cut-off value for the number of active servers.
- One offloading period = one round in MG
- In each t, servers decide between the two actions,
  - **active** (denote by '1')
  - **inactive** (denote by '0')
- $a_i(t)$ = action of a given player i in the time slot t
- Attendance = number of active servers $c(t)$
- Each player has S strategies.

Activation of MEC servers

MG formulation

- **MG outcomes**
  - If $c(t) \leq \phi$, minority (active servers) earns a reward $R \geq R_{\text{th}}$
  - If $c(t) > \phi$, $c(t)$ majority (active servers) cannot earn R_{th}. Thus minority action is the winning choice.
- **Control Information**
  - After each round, a central unit broadcasts the winning choice to all servers by sending

$$b(t) = \begin{cases} 1, & \text{if } c(t) \leq \phi \\ 0, & \text{otherwise.} \end{cases}$$

Activation of MEC servers

MG formulation

- **Utility**
  - $U_{i,\mathrm{a}}(t)$ = Utility of active servers, $U_{i,\mathrm{p}}(t)$ = utility of inactive servers

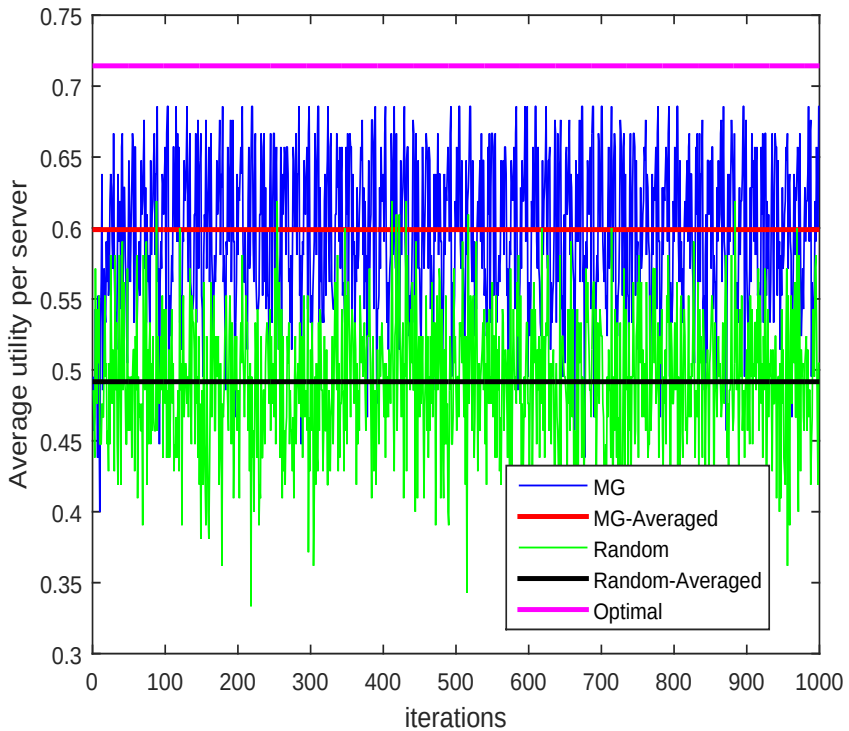
$$U_{i,\mathrm{a}}(t) = \begin{cases} 1, & \text{if } c(t) \leq \phi \\ 0, & \text{otherwise} \end{cases} \qquad U_{i,\mathrm{p}}(t) = \begin{cases} 1, & \text{if } c(t) > \phi \\ 0, & \text{otherwise.} \end{cases}$$

- **Distributed learning method**

Every player applies a basic strategy reinforcement technique to solve the formulated MG.

Activation of MEC servers

Numerical results: Average utility



- Utility of MG-based strategy is higher than that of random selection
- Yet, it is below the average utility of the optimal scenario.
- In MG-based method, servers make decisions under minimal external information, and without any coordination.

Open problems

- **Computation offloading**
  - Multiple computational resources
    - **Multi-option MG** can be used to model this.
  - Heterogeneous users (different computational and battery capacity)
    - e.g. smart phones, tablets, laptops
    - Require MG models that can model **heterogeneous users**.
  - For more realistic scenarios, consider
    - **Stochastic arrival** of offloading jobs at the server pool.
    - Energy cost per job to be proportional to processing time.

Open problems

- **Evolutionary variations of MG**
  - In basic MG, strategies *do not evolve*, only scored to be selected as the best.
  - Many complex dynamic systems require agents to both **learn the best strategy and dynamically *adjust* the strategies**.
  - Evolutionary MG, exponential learning, etc. can be used.

Thank you!