DRL-based policies for joint resource scheduling and computation offloading of Energy Harvesting Devices

Mireille Sarkiss

Joint work with: Ibrahim Fawaz (Telecom SudParis), Philippe Ciblat (Telecom Paris) and Deniz Gündüz (Imperial College)

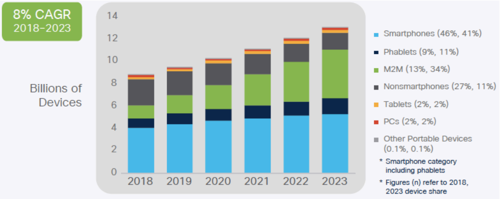
INSTITUT POLYTECHNIQUE DE PARIS

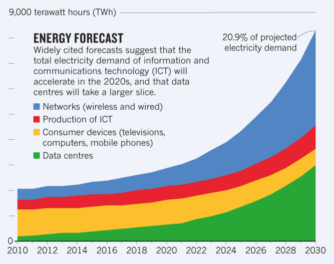
Electrical Engineering Artificial Intelligence Day

November 19, 2020

5G Mobile Networks

- Mobile networks will have to deal with
  - Enormous number of connected devices
  - Resource-hungry applications
  - Exponential growth of mobile traffic





- Global ICT ecosystem consumes more than 2000 TWh of electricity annually
  - predicted to grow to 20% of global electricity demand by 2030
  - greatly increased emitted carbon footprint

Challenges and Promising Solutions

Mobile terminals limitations:

- Processing capacity
- Storage
- Energy

Promising solutions:

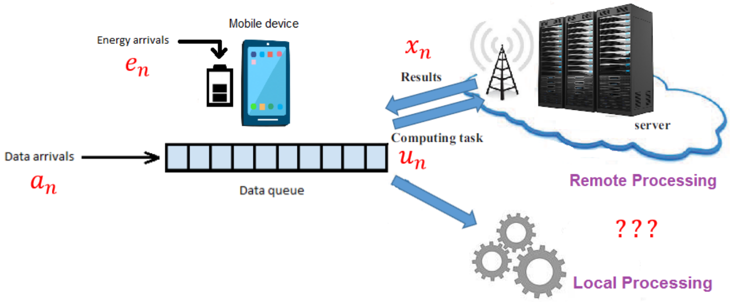
- Energy Harvesting (EH)
- Computation offloading

Work Objective

- Design efficient policies for resource scheduling and computation offloading under EH constraints

- Optimize transmission policies taking into account:
  - Random data arrivals statistics
  - Sporadic energy arrivals statistics
  - Channel conditions
  - Packet queue status
  - Battery energy level

Joint Resource Scheduling and Computation Offloading



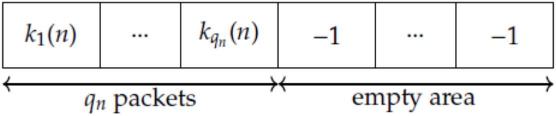
- Data arrival $\sim$ Poisson distribution with mean λ_d
- Energy arrival $\sim$ Poisson distribution with mean λ_e
- Constant channel during a time slot with perfect CSIT
- At the beginning of each time slot, mobile device decides:
  - Type of processing: locally or remotely
  - Number of packets to be processed

Strict Delay Constraint

Previous works: Average delay constraint

- Little's law: convert average delay constraint into average queue length constraint
- Drawback: packets can stay in the buffer for long time

Proposed scheme: Strict delay constraint



- $k_i(n)$ is the age of the i-th packet at the beginning of time slot n
- A packet can be discarded due to
  - **Delay violation** The i-th packet is discarded if $k_i(n) > K_0$
  - **Buffer overflow:** New arrivals are discarded if $q_n = B_d$

Energy Cost

At the beginning of each time slot, 3 possible processing decisions:

- **Local processing:** Mobile device executes u packets

$$E_\ell(u) = \left\lceil u.P_\ell.\frac{T_s}{\mathcal{E}_U} \right\rceil$$

- **Remote processing:** Mobile device transmits u packets to be executed at BS

$$E_o(x,u) = \left\lceil \frac{u}{\mathcal{E}_U}\left(\frac{L.P_t}{W_{UL}.\log_2\left(1+\frac{P_t.x}{W_{UL}.N_0}\right)} + T_w.P_w + \frac{L_{DL}.P_r}{W_{DL}.\log_2\left(1+\frac{P_s.x}{W_{DL}.N_0}\right)}\right)\right\rceil$$

- **Idle:** Mobile device waits for the next time slot

$$E_I = 0$$

Markov Decision Process

- **State space:** $\mathcal{S} = (\mathbf{k}, b, x)$
  - $\mathbf{k} = [k_1, \cdots, k_{B_d}]$: age of each packet in the data buffer
  - b: battery level
  - x: channel gain (quantized value)
- **Action space:** Type of processing and number of packets u
- **Cost:** Average number of discarded packets due to
  - Delay:

$$\varepsilon_d(\mathbf{s}_n, \nu_n) = \begin{cases} 0 & \text{if } m_n = 0 \text{ or } m_n \leqslant u_{\nu_n} \\ m_n - u_{\nu_n} & \text{otherwise.} \end{cases}$$

  - Overflow:

$$\varepsilon_o(\mathbf{s}_n, \nu_n) = \sum_{a=B_d-q_n+w_n+1}^{+\infty} (q_n - w_n + a - B_d).e^{-\lambda_d}.\frac{(\lambda_d)^a}{a!}$$

Dynamic Programming Approach

- Fully-known system states and transitions
- Optimal Deterministic Offline policy using Policy Iteration algorithm
- **Policy Iteration (PI)**
  - Policy Evaluation

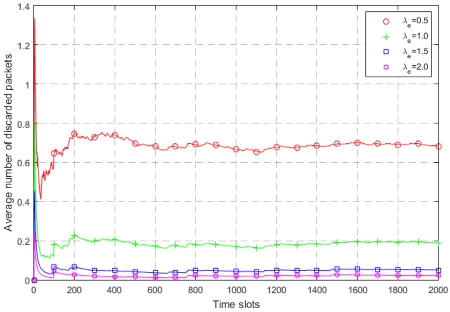
$$\beta^{n-1}\mathbf{1} + (\mathbf{Id} - \mathbf{P})\mathbf{v}^{n-1} = \mathbf{c}^{n-1}$$

$$\sum_{\mathbf{s}\in\mathcal{S}} v^{n-1}(s) = 0$$

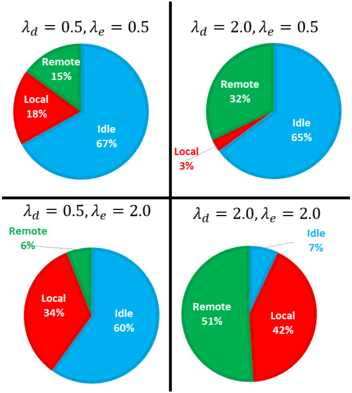
  - Policy Improvement

$$\mu^n(s) = \arg\min_{u\in\mathcal{U}} \left[c(s,u) + \sum_{s'\in\mathcal{S}} p(s'|s,u)v^{n-1}(s') \right]$$

Numerical Results - Convergence and Processing Decisions



Convergence of average number of discarded packets for different energy arrival rates



Percentage of processing decisions

- Only few hundreds of slots are needed for the system to achieve the long-term cost
- $\lambda_e \nearrow \quad \Rightarrow \quad$ Average number of discarded packets $\searrow$
- $\lambda_d \nearrow \quad \Rightarrow \quad$ Idle mode $\searrow$

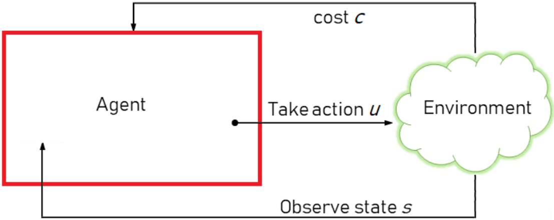
Reinforcement Learning (RL)

DP Solution:

- **Advantage:** Optimal Solution
- **Drawback:** Only applicable when the environment model is known

Alternative Solution: Reinforcement Learning (RL)

- Learn the state-action function: $Q(s, u)$ while interacting with the environment



Q-Learning Algorithm

Algorithm 1 Q-Learning Algorithm

Set learning rate α
Initialize $Q(s,u)$ for all $s \in \mathcal{S}$ and $u \in \mathcal{U}(s)$ randomly
for t = 1, T **do**
    Generate random state s_0
    for n = 0, N **do**
        $u_n = \arg\min_u Q(s_n, u)$ with probability $1-\epsilon$
        Otherwise, u_n is selected randomly
        Execute u_n and observe $c(s_n, u_n)$ and s_{n+1}
        Update $Q(s,u)$ with $(1-\alpha)Q(s,u) + \alpha(c(s_n,u_n) + \min_u Q(s_{n+1}, u))$
    end for
end for

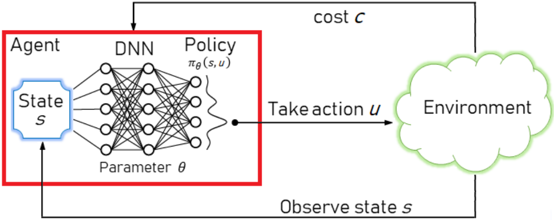
Deep Reinforcement Learning (DRL)

DP and RL Solutions:

- **Drawback:** Impractical and very complex with large system states

Alternative Solution: Function Approximation

- Estimate of the state-action function: $Q(s, u, \theta) \approx Q^{\star}(s, u)$
- Non-linear function: Neural-Network (NN) $\rightarrow$ Deep Q-Network (DQN)



Training the NN

- Learn θ by minimizing the MSE between:
  - **Target** $= c(s_n, u_n) + \min_u Q(s_{n+1}, u; \theta)$
  - **Prediction** $= Q(s_n, u_n; \theta)$
- Ensure stable learning by applying:
  - **Experience Replay:** Store the experience $[s_n, u_n, c(s_n, u_n), s_{n+1}]$ in replay memory $\mathcal{M}$, and train using random mini-batches from $\mathcal{M}$.
  - **Fixed target Network:** Use a second network where its weights θ' are fixed, and only periodically or slowly updated to the primary network values for $Q(s_{n+1}, u; \theta')$
  - **Double DQN:** Use a second network to decouple the action selection from the target Q value generation, i.e. $Q(s_{n+1}, \arg\min_u Q(s_{n+1}, u; \theta); \theta')$
- Ensure adequate exploration of the state space by using ϵ-greedy strategy:
  - Choose **best** action $u_n = \min_u Q(s_n, u; \theta)$ with probability $1 - \epsilon$
  - Select **random** action with probability ϵ

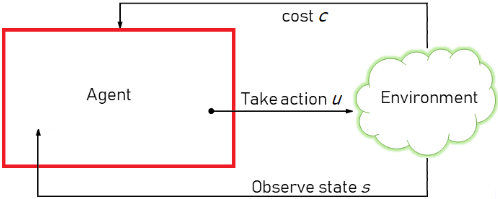
Double Deep Q-Learning Algorithm

Algorithm 2 Double Deep Q-Learning algorithm

Initialize replay memory $\mathcal{M}$ to capacity M
Initialize Q-network with random weights θ
Initialize target Q-network with random weights $\theta' = \theta$
for t = 1, T **do**
  Generate random state s_0
  for n = 0, N **do**
    $u_n = \arg\min_u Q(s_n, u; \theta)$ with probability $1 - \epsilon$
    Otherwise, u_n is selected randomly
    Execute u_n and observe $c(s_n, u_n)$ and s_{n+1}
    Store experience $[s_n, u_n, c(s_n, u_n), s_{n+1}]$ in $\mathcal{M}$
    Sample random mini-batch of B_m transitions from $\mathcal{M}$
    Set the target to $c(s_n, u_n) + Q(s_{n+1}, \arg\min_u Q(s_{n+1}, u; \theta); \theta')$
    Perform *Adam* update on θ
  end for
  Update target network, i.e. $\theta' = \theta$
end for

1-Step Training example (1)

1. Interact/Explore



$\Downarrow$

Memory $\mathcal{M}$

$$[s_0, u_0, c_0, s_1]$$
$$[s_1, u_1, c_1, s_2]$$
$$\vdots$$
$$[s_n, u_n, c_n, s_{n+1}]$$

1-Step Training example (2)

2. Prepare data/Train the network

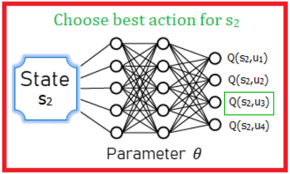
Memory $\mathcal{M}$

$[s_0, u_0, c_0, s_1]$

$\boxed{[s_1, u_1, c_1, s_2]}$

$\vdots$

$[s_n, u_n, c_n, s_{n+1}]$



$$\arg\min_u Q(s_{n+1}, u; \theta)$$

1-Step Training example (2)

2. Prepare data/Train the network

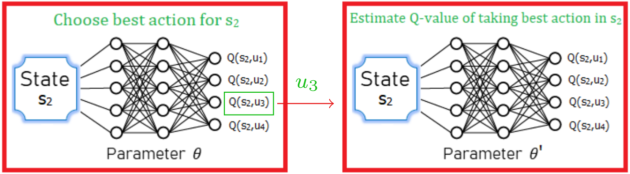
Memory $\mathcal{M}$

$[s_0, u_0, c_0, s_1]$

$\boxed{[s_1, u_1, c_1, s_2]}$

$\vdots$

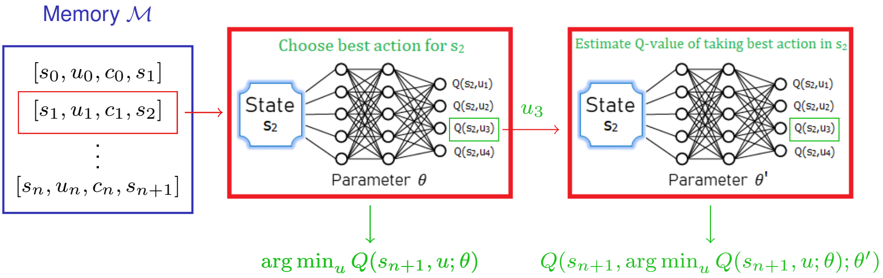
$[s_n, u_n, c_n, s_{n+1}]$



$$\arg\min_u Q(s_{n+1}, u; \theta)$$

1-Step Training example (2)

2. Prepare data/Train the network



1-Step Training example (2)

2. Prepare data/Train the network

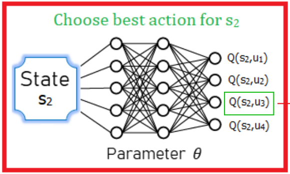
Memory $\mathcal{M}$

$[s_0, u_0, c_0, s_1]$

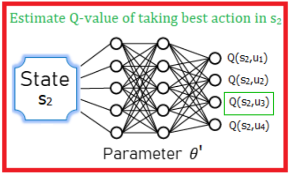
$\boxed{[s_1, u_1, c_1, s_2]}$

$\vdots$

$[s_n, u_n, c_n, s_{n+1}]$

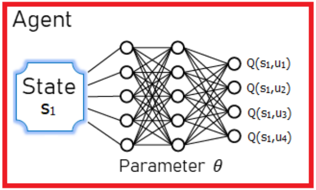


u_3



$\arg\min_u Q(s_{n+1}, u; \theta)$

$Q(s_{n+1}, \arg\min_u Q(s_{n+1}, u; \theta); \theta')$



1-Step Training example (2)

2. Prepare data/Train the network

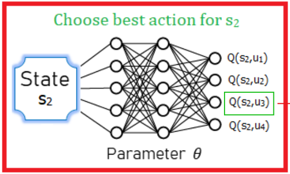
Memory $\mathcal{M}$

$[s_0, u_0, c_0, s_1]$

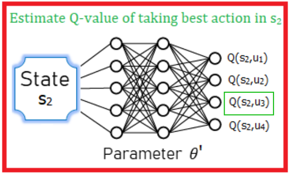
$\boxed{[s_1, u_1, c_1, s_2]}$

$\vdots$

$[s_n, u_n, c_n, s_{n+1}]$

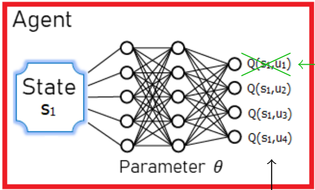


u_3



$\arg\min_u Q(s_{n+1}, u; \theta)$

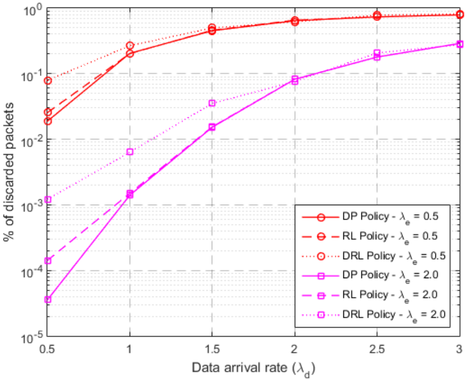
$Q(s_{n+1}, \arg\min_u Q(s_{n+1}, u; \theta); \theta')$



$$\underbrace{c_1 + Q(s_{n+1}, \arg\min_u Q(s_{n+1}, u; \theta); \theta')}_{\text{Target}}$$

Prediction

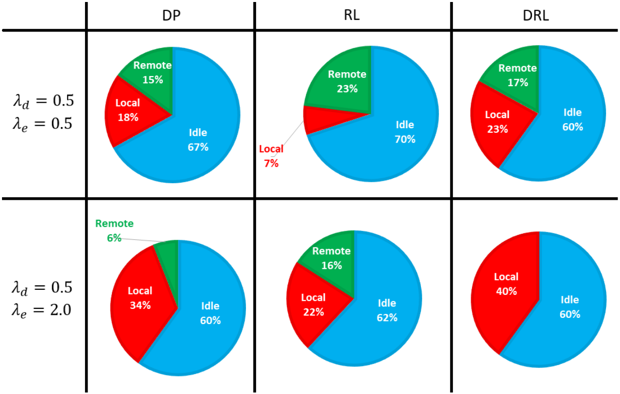
Numerical Results - Discarded Packets



Percentage of discarded packets versus data arrival rate for different energy arrival rates

- RL policy is almost optimal in most of the cases.
- DRL policy achieve optimal performance for high λ_d.

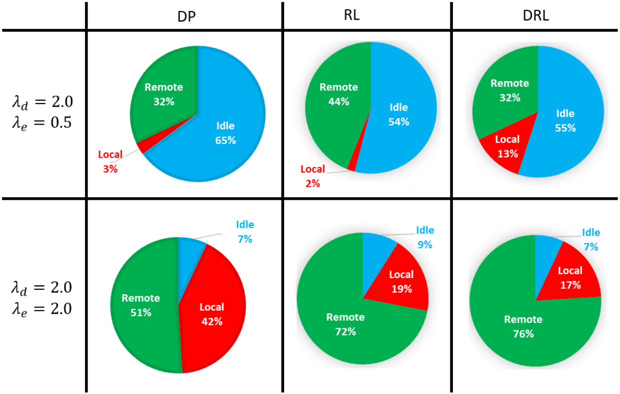
Numerical Results - Processing Decisions



Percentage of processing decisions

- Small $\lambda_d, \lambda_e \nearrow \quad \Rightarrow \quad$ local mode $\nearrow$

Numerical Results - Processing Decisions



Percentage of processing decisions

- High $\lambda_d, \lambda_e \nearrow \quad \Rightarrow \quad$ remote and local modes $\nearrow$

Conclusion

- **Main objective:** Propose policies for 5G mobile system with
  - Offloading capabilities
  - Energy harvesting
  - Strict delay

- Investigate resource scheduling and computation offloading for EH mobile device
  - Optimal policy outperforms other policies by adapting the number of executed packets to the system states
  - DRL-based policy can be improved by improving training
    - with larger training set
    - using multistep learning algorithms