Computation Offloading in Mobile Edge Computing (MEC) Networks: A Multi-Task Learning Approach

Lijun Qian, Bo Yang, Joshua Bassey
CREDIT Center, Department of ECE, PVAMU

The 4th Mission Critical Big Data Analytics (MCBDA) Workshop

Contents

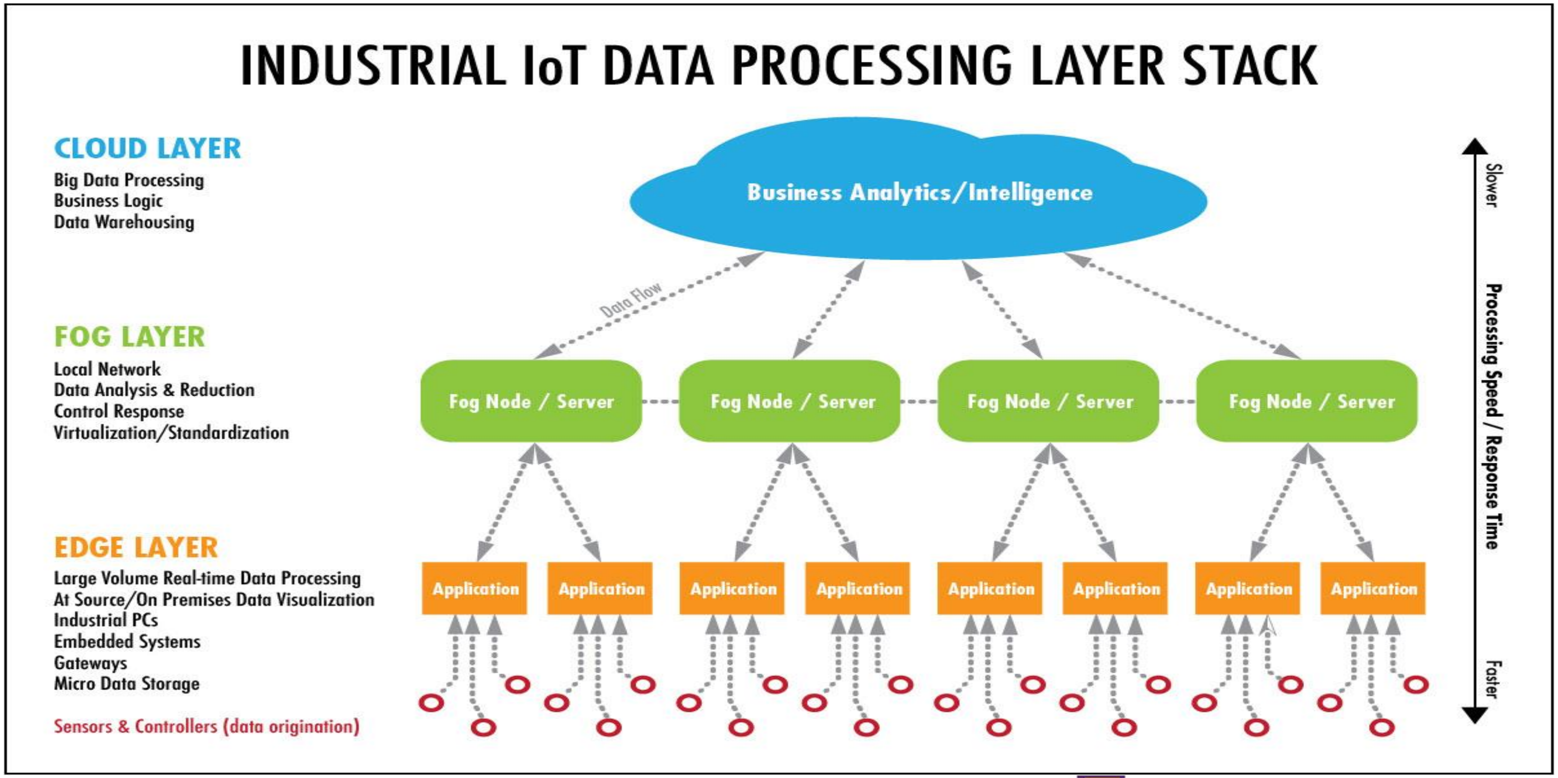
PRAIRIE VIEW A&M UNIVERSITY

Cloud Computing (CC)

- On-demand availability of computer system resources
- The resources cannot be managed by the users
- Wireless connection between users and central servers

Mobile Edge Computing (MEC)

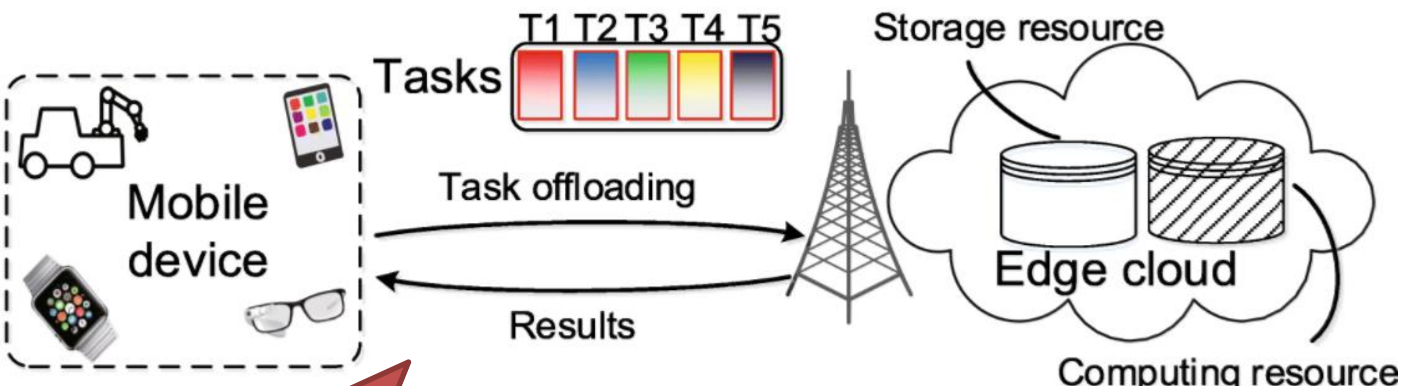
- MEC plays a key role in bringing cloud functionalities to the edge that in close proximity to mobile users or devices.



PRAIRIE VIEW A&M UNIVERSITY

Computation Offloading

- Computation offloading is the transfer of resource intensive computational tasks to an external platform, such as a cluster, grid, or a cloud.
- Offloading may be necessary due to hardware limitations of a devices, such as limited computational power, storage, and energy.



Offloading decision includes:
1) Whether offloading or not?
2) How much resources can be obtained?

Offloading decision should be OPTIMIZED in REAL-TIME!

PRAIRIE VIEW A&M UNIVERSITY

Challenges & motivations

- The authors generally formulated the joint optimization of computational resources and offloading decision as a mixed integer nonlinear programming (MINLP) problem, which is NP-hard.

Accuracy

- This NP-hard problem is usually solved by seeking an optimal (or sub-optimal) solution using traditional mathematical algorithms.

Efficiency

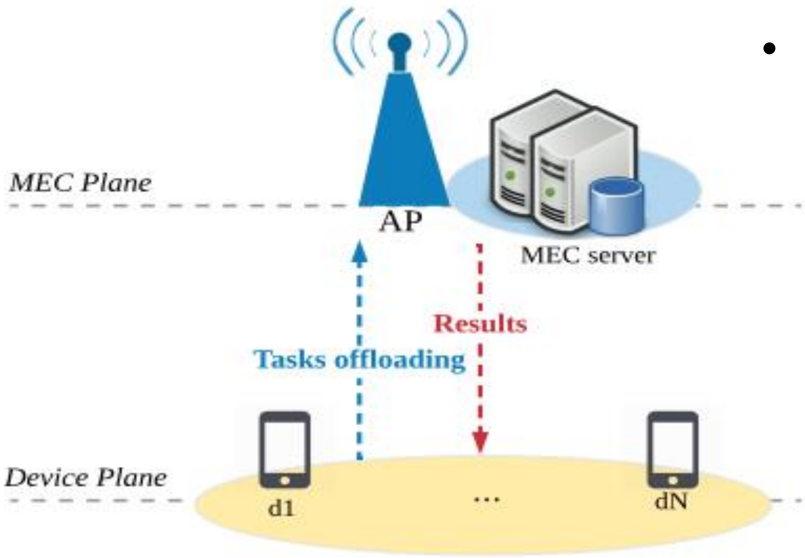
- Usually, huge computation overhead is introduced, which increases the delay obtaining the solutions.

Contents

Prairie View A&M University

Network scenario & assumptions

- Total N devices in the network, $\mathcal{N} = \{d_1, d_2, ..., d_N\}$.
- We assume that each device has only one computation-intensive task (denoted as $\mathcal{J}_i, \ \forall i \in \mathcal{N}$) to be processed at a time, which is atomic and cannot be further divided.
- Denote $D_i \in \{0, 1\}$ as the computation offloading decision of d_i, so the offloading decision vector for all devices can be expressed as $\mathbf{D} = \{D_i | i \in \mathcal{N}\}$, which is a N-dimensional binary vector.



- Let $\mathbf{F} = \{f_i | i \in \mathcal{N}\}$ be the allocated computational resource (i.e., central processing unit (CPU) cycles per second) vector by the MES. Hence, the offloading strategy for all devices can be defined as $\boldsymbol{S} = \{\mathbf{D}, \mathbf{F}\}$

System models

- Communication model:
  - ❖ The mobile devices are assumed to operate using non-orthogonal multiple access (NOMA) such that the data from different mobile devices can be decoded separately from the superposed signal using multiuser detection algorithms at the AP.
  - ❖ The received SINR of d_i served by the AP is $\mathrm{SINR}_i = \frac{P_t^i |h_i|^2}{\delta^2 + \sum_{j=i+1}^{N} P_t^j |h_j|^2}$
- Computing model:
  - ❖ Local execution delay: $\tau_l^i = \frac{c_i}{f_l^i}$
  - ❖ Local energy consumption: $\varepsilon_l^i = \kappa \left(f_l^i\right)^2 c_i$
  - ❖ Local weighted-cost: $\mathcal{O}_l^i = \alpha \tau_l^i + (1-\alpha)\varepsilon_l^i, \ \forall i \in \mathcal{N}$
  - ❖ Offloading delay and energy consumption:

$$\tau_o^i = \frac{s_i}{W log_2(1 + \mathrm{SINR}_i)} + \frac{c_i}{f_i},$$

$$\varepsilon_o^i = \frac{P_t^i s_i}{W log_2(1 + \mathrm{SINR}_i)} + \frac{P_I^i c_i}{f_i}.$$

  - ❖ Offloading weighted-cost:

$$\mathcal{O}_o^i = \alpha \tau_o^i + (1-\alpha)\varepsilon_o^i, \ \forall i \in \mathcal{N}$$

PRAIRIE VIEW A&M UNIVERSITY

Problem formulation and analysis

- We define the sum cost of the MEC system as the weighted-sum cost of all the devices: $\mathcal{O}_{total} = \sum_{i \in \mathcal{N}} (1 - D_i)\, \mathcal{O}_l^i + D_i \mathcal{O}_o^i$
- We formulate the joint offloading and computational resource allocation as a weighted-sum cost minimization problem:

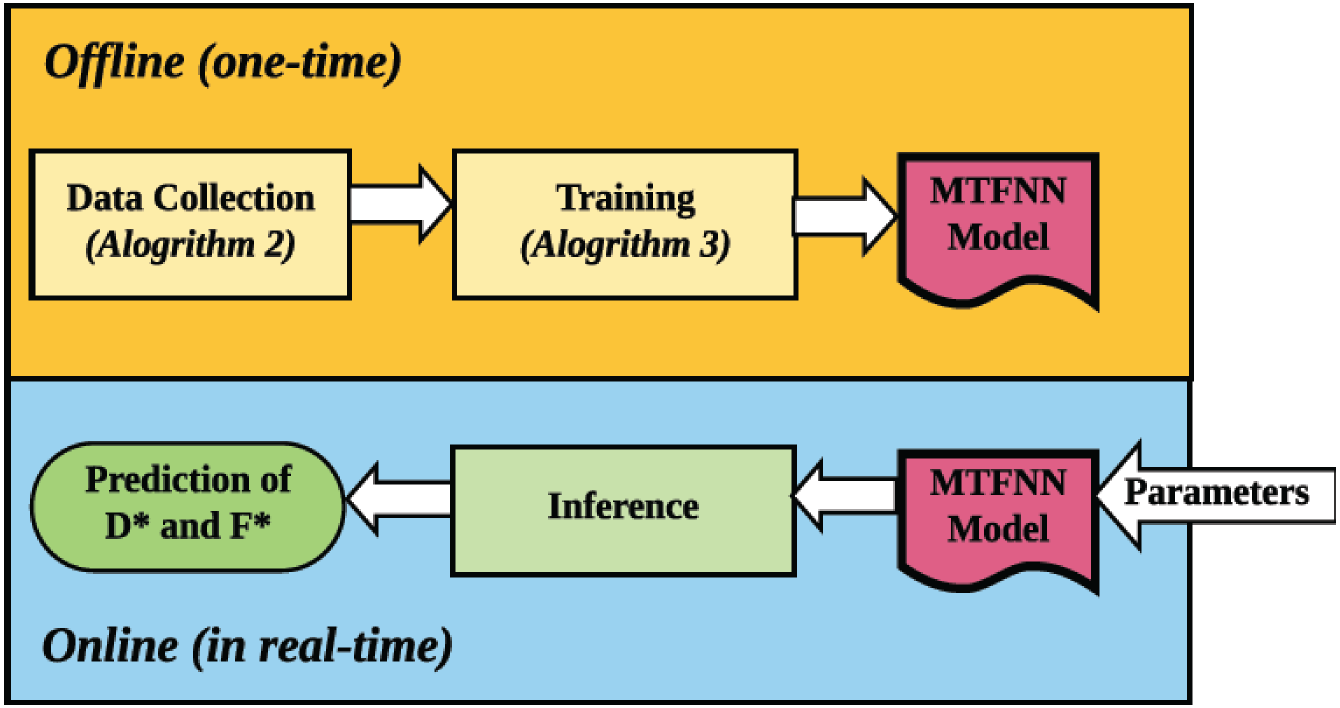
P1 (Original problem):

$$\begin{aligned} \underset{\{\mathbf{D},\mathbf{F}\}}{\text{minimize}} \;\; & \mathcal{O}_{total} \\ \text{s.t.} \;\; & \mathbf{C1}: \; D_i \in \{0,1\}, \; \forall i \in N, \\ & \mathbf{C2}: \; (1 - D_i)\, \tau_l^i + D_i \tau_o^i \le \vartheta_i, \\ & \mathbf{C3}: \; 0 \le f_i \le F, \; \forall i \in N, \\ & \mathbf{C4}: \; \sum_{i=1}^{N} D_i f_i \le F, \; \forall i \in N. \end{aligned}$$

$$\boldsymbol{S}^* = \underset{\{\mathbf{D},\mathbf{F}\}}{\text{argmin}} \; \mathcal{O}_{total}$$

Multi-task learning framework

- The offloading decision making is formulated as a multiclass classification problem and the computational resource allocation is formulated as a regression problem.



PRAIRIE VIEW
A&M UNIVERSITY

Data collection (1/2)

- Given the parameters in the table, we generate the dataset by traversing all the possible combinations of **D** and **S** with the exhaustive searching algorithm, to minimize the weighted-sum cost.

Parameters	**Value range**
The number of devices (N)	$[2 - 8]$
Data payload size (s)	$[1 - 500]$ kbits
CPU cycle required to process the data (c)	$[3 - 1500]$ Megacycles
CPU frequency of the device (f_l)	[1Hz – 1GHz]
Weights of delay and energy cost (α, β)	$[0.0 - 1.0]$

Data collection (2/2)

- We generate and collect training dataset in the MATLAB environment using a computer with NIVIDA GPU TITAN X (Pascal).
- The GPU can accelerate the matrix calculation in MATLAB.

Algorithm 2: Dataset Generation Algorithm

Initialization: $i = 0$, $S^* = \varnothing$;

Iteration:

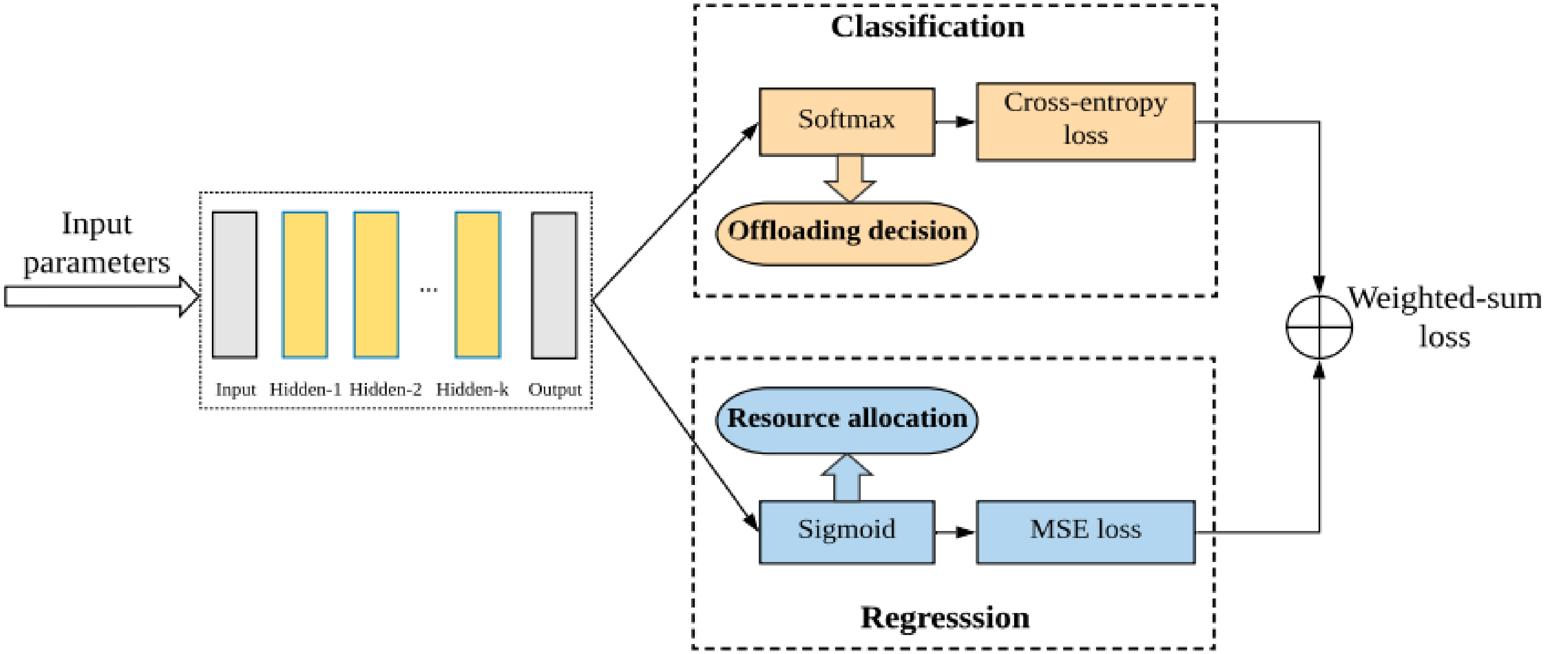
1: **while** $i <$ *dataset size* **do**
2: $\quad i \leftarrow i + 1$;
3: $\quad$ Generate input parameters set $(\mathbf{X}_i)$ for all devices;
4: $\quad$ Formulate the optimization problem $\mathbf{P1}$ as (18);
5: $\quad$ Solve $\mathbf{P1}$ with exhaustive searching method and record the optimal solution as $\mathbf{Y}_i = (\mathbf{D}^*, \mathbf{\Theta}^*)$;
6: $\quad$ Add an input/output pair $\mathcal{S}_i = \{\mathbf{X}_i, \mathbf{Y}_i\}$ to S^*;
7: **end while**

Prairie View A&M University

Offline training

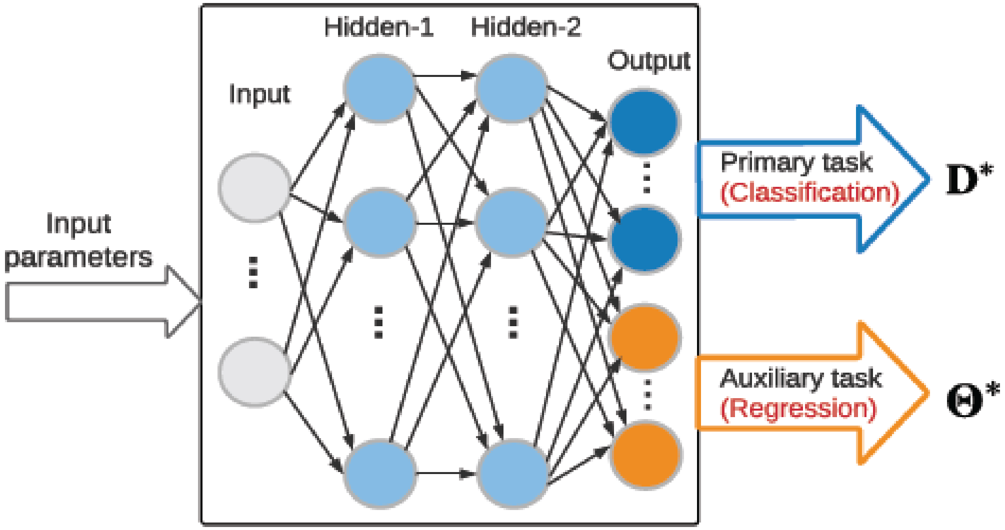
- For the classifier, $l_c = -\frac{1}{K}\sum_{i=1}^{K} \mathbf{Y}_i \ln f(\mathbf{X}_i),$
- For the regressor, $l_r = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{Y}_i - f(\mathbf{X}_i))^2$

$$l = \chi_c l_c + \chi_l l_r$$



The pre-trained model

- An example of the pre-trained MTL model with N = 3



Multi-task forward neural network model (trained offline)

Contents

PRAIRIE VIEW
A&M UNIVERSITY

Testing results (1/2)

- We compare with a benchmark scheme "sBB" which is implemented using the MATLAB toolbox of the APMonitor Optimization Suite (http://APMonitor.com).

- Performance index:
  - Computation complexity:

$$t = \frac{Total\ execution\ time}{Number\ of\ samples}$$

  - Computation accuracy:

The smaller the better

Classification: $$\eta = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

The larger the better

Regression: $$\varepsilon = \frac{1}{mN}\sum_{i=1}^{m}\sum_{j=1}^{N}(y_j^i - x_j^i)^2$$

The smaller the better

PRAIRIE VIEW A&M UNIVERSITY

Testing results (2/2)

η, ε, t \ S / N	sBB	MTFNN with $\chi_c = \chi_l = 1$
2	$70\%,\ 0.055,\ 14.1\ ms$	$96\%,\ 0.016,\ 2.5\ \mu s$
3	$62\%,\ 0.047,\ 14.2\ ms$	$89\%,\ 0.027,\ 2.5\ \mu s$
4	$58\%,\ 0.053,\ 14.5\ ms$	$83\%,\ 0.029,\ 2.2\ \mu s$
5	$47\%,\ 0.051,\ 15.2\ ms$	$50\%,\ 0.021,\ 2.0\ \mu s$

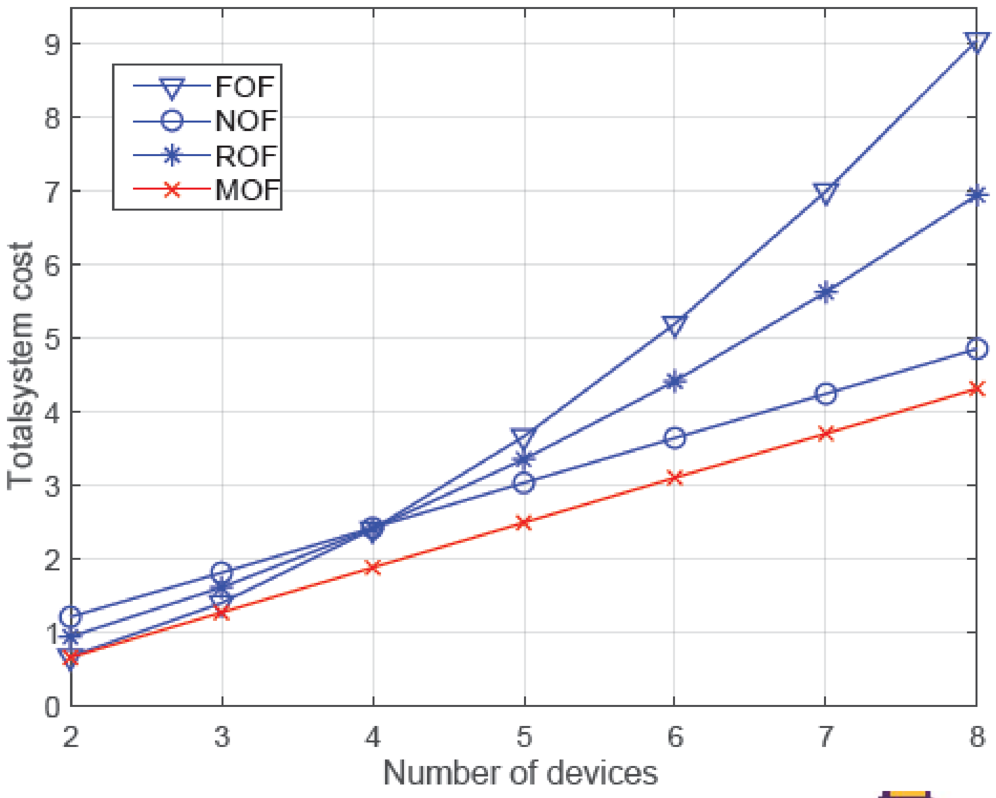
η, ε, t \ S / N	sBB	MTFNN with $\chi_c = 0,\ \chi_l = 1$
5	$47\%,\ 0.051,\ 15.2\ ms$	$78\%,\ 0.009,\ 5.0\ \mu s$
6	$42\%,\ 0.092,\ 15.8\ ms$	$82\%,\ 0.009,\ 5.7\ \mu s$
7	$38\%,\ 0.095,\ 16.6\ ms$	$81\%,\ 0.009,\ 3.6\ \mu s$
8	$34\%,\ 0.097,\ 16.9\ ms$	$78\%,\ 0.009,\ 3.9\ \mu s$

Simulation results (1/4)

- Three benchmark offloading approaches
  - Full offloading (FOF)
  - None offloading (NOF)
  - Random offloading (ROF): The ROF scheme denotes that all the tasks will be executed by the two ways above randomly.

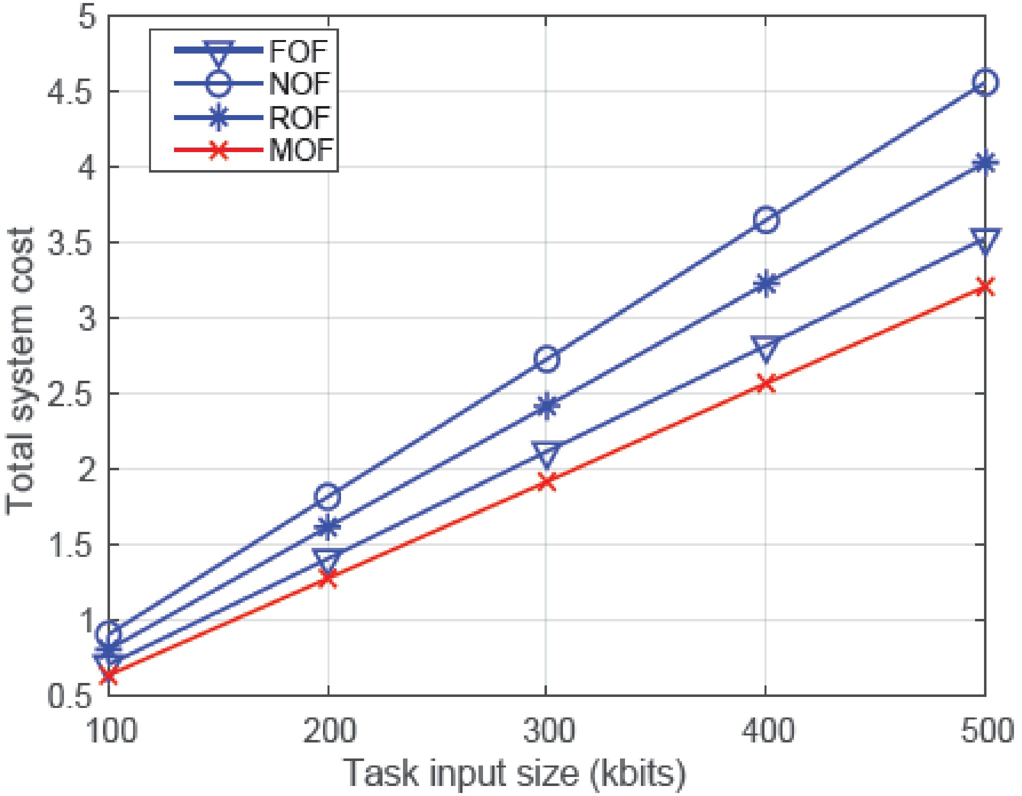
Simulation results (2/4)

$$f_l = 0.5 \text{ GHz} \quad \alpha = \beta = 0.5$$



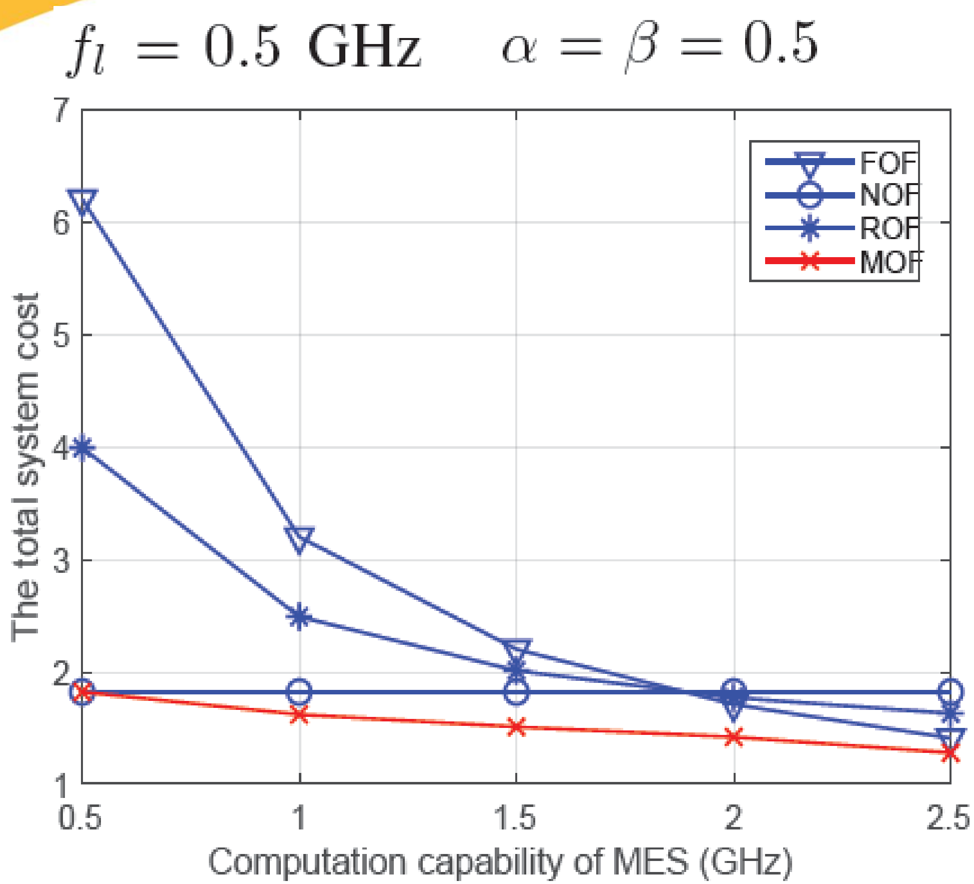
Simulation results (3/4)

$f_l = 0.5$ GHz $\quad \alpha = \beta = 0.5$



PRAIRIE VIEW A&M UNIVERSITY

Simulation results (4/4)



PRAIRIE VIEW A&M UNIVERSITY

Contents

PRAIRIE VIEW A&M UNIVERSITY

Contributions

- We propose a multi-task learning based solution that can adapt to the varying network conditions and the changing requirements of devices' applications.
- The MTL is trained offline and only one time. After the MTL model is trained, it can be directly used to generate the optimal solution of the MINLP problem with high accuracy in near-real-time.
- The proposed MTL model outperforms the conventional optimization algorithms significantly in terms of computation time (four orders of magnitude) and inference accuracy (up to two times better).

Q/A