SVM-Based Task Admission Control and Computation Offloading Using Lyapunov Optimization in Heterogeneous MEC Network

Nadine Abbas, *Member, IEEE*, Wissam Fawaz, *Senior Member, IEEE*, Sanaa Sharafeddine, *Senior Member, IEEE*, Azzam Mourad, *Senior Member, IEEE*, and Chadi Abou-Rjeily, *Senior Member, IEEE*

***Abstract*—Integrating device-to-device (D2D) cooperation with mobile edge computing (MEC) for computation offloading has proven to be an effective method for extending the system capabilities of low-end devices to run complex applications. This can be realized through efficient computing data offloading and yet enhanced while simultaneously using multiple wireless interfaces for D2D, MEC and cloud offloading. In this work, we propose user-centric real-time computation task offloading and resource allocation strategies aiming at minimizing energy consumption and monetary cost while maximizing the number of completed tasks. We develop dynamic partial offloading solutions using the Lyapunov drift-plus-penalty optimization approach. Moreover, we propose a task admission solution based on support vector machines (SVM) to assess the potential of a task to be completed within its deadline, and accordingly, decide whether to drop from or add it to the user's queue for processing. Results demonstrate high performance gains of the proposed solution that employs SVM-based task admission and Lyapunov-based computation offloading strategies. Significant increase in number of completed tasks, energy savings, and cost reductions are resulted as compared to alternative baseline approaches.**

***Index Terms*—Lyapunov optimization, mobile edge computing, partial offloading, computation resource allocation, admission control, D2D communication, multi-RAT.**

I. Introduction

With the rapid development of the Internet-of-Things (IoT), new innovative services are unfolding, many of which are extremely sensitive to delay and yet require tremendous computation capabilities [1], [2]. Mobile edge computing is one of the key designs of the future networks aiming at assisting devices with little computing capabilities to perform computation-intensive tasks at mobile edge devices or servers. In conventional MEC networks, tasks are completely offloaded to the server or cloud by adopting binary offloading as in [3]–[5]. Further enhancements are realized through integrating D2D cooperation with Het-MEC for D2D offloading to provide additional computation capabilities with low monetary cost and energy consumption. Partial offloading was adopted in [6]–[9], where a task can be partitioned into multiple subtasks to be executed at different nodes simultaneously. However, the number of subtasks allowed was limited; e.g.,: two subtasks to be executed locally or remotely as in [7], and three subtasks as in [8] and [9]. Unlike conventional networks, we aim at providing a comprehensive heterogeneous MEC framework while taking advantage of all the possible computation and communication resources and allow efficient utilization of all the available computation and radio resources.

Previous studies focused mainly on achieving one or two objectives such as maximizing computation capacity as in [8], fog profit as in [10] and a trade-off between energy and latency as in [7]. Some works adopted Lyapunov optimization to reduce queue congestion by stabilizing the queue backlog. They mainly aimed at maximizing throughput as in [11], minimizing energy as in [12] and minimizing cost as in [13]. However, these objectives are interdependent which creates the need for efficient solutions providing a trade-off between multiple objectives. Hence, we aim at maximizing the number of completed tasks by stabilizing the requester queue backlog while minimizing energy consumption and monetary cost subject to delay, computation and radio resources constraints.

Moreover, admission control schemes were proposed to verify the availability of computation resources before a generated task can be admitted [10], [14], [15]. Mainly, these approaches tend to be more network-centric, implemented at the fog node as in [10] and [11], and use binary offloading as in [14], or limited partial offloading as in [15]. To our knowledge, considering queue congestion and admission control schemes simultaneously with partial offloading to multiple cooperating nodes in D2D-enabled Het-MEC networks is still not yet well investigated. Hence, we leverage the use of Support Vector Machines to propose a task admission control scheme to decide on whether to admit or reject newly generated tasks.

Manuscript received 28 June 2021; revised 29 November 2021 and 16 February 2022; accepted 11 March 2022. Date of publication 14 March 2022; date of current version 12 October 2022. The associate editor coordinating the review of this article and approving it for publication was F. Paganelli. *(Corresponding author: Nadine Abbas.)*

Nadine Abbas and Sanaa Sharafeddine are with the Department of Computer Science and Mathematics, Lebanese American University, Beirut 1102, Lebanon (e-mail: nadine.abbas@lau.edu.lb; sanaa.sharafeddine@lau.edu.lb).

Wissam Fawaz and Chadi Abou-Rjeily are with the Department of Electrical and Computer Engineering, Lebanese American University, Beirut 1102, Lebanon (e-mail: wissam.fawaz@lau.edu.lb; chadi.abourjeily@lau.edu.lb).

Azzam Mourad is with Department of Computer Science and Mathematics, Lebanese American University, Beirut 1102, Lebanon, and also with the Division of Science, New York University, Abu Dhabi, UAE (e-mail: azzam.mourad@lau.edu.lb).

Digital Object Identifier 10.1109/TNSM.2022.3159478

In summary, existing research works are still limited in terms of scalability and performance. Previous studies focus mainly on achieving one or two objectives. Moreover, these studies mainly used binary offloading while some used partial offloading with limited number of nodes without considering task admission control. Complementing the existing literature, we present a comprehensive user-centric, real-time, partial computation offloading and resource allocation in D2D-enabled Het-MEC networks and develop an intelligent admission control scheme to decide on the tasks with high potential to be executed within their deadlines. The main contributions of this work can be summarized as follows:

1. We adopt multi-layer D2D-enabled heterogeneous MEC networks where the requester can partially offload its bitwise independent computation data to multiple nodes for parallel processing including one peer mobile terminal, edge server and cloud.
2. We take advantage of the coexistence of multiple heterogeneous networks and allow the requester to offload different parts of its computational data task using different wireless technologies, simultaneously, such as Bluetooth to the peer MT, WiFi to the edge server, and cellular network to the cloud.
3. We develop real-time multi-objective computation offloading and resource allocation solutions based on Lyapunov optimization to meet a tradeoff between three objectives: (1) stabilizing the requester queue backlog leading to a higher number of completed tasks, (2) minimizing the energy consumption and (3) minimizing the monetary cost subject to delay and resources constraints. Accordingly, we decide on the best offloading strategy, the amount of computation data to be offloaded, and the computation resource allocation providing the best tradeoff between the considered objectives.
4. We propose an SVM-based admission control scheme to decide on whether to admit or reject newly generated computation tasks by assessing their potential to be completed within their deadline under the dynamic variation of the system parameters. The SVM-based admission control scheme is embedded into the proposed Lyapunov-based computation offloading approach to reduce the requester queue congestion, increase the percentage of completed tasks, and minimize the energy consumption and monetary cost by allowing the less congested queue to serve tasks with higher potential to be completed.

II. Related Work

In a heterogeneous MEC networks, a requester mobile terminal can either perform its computation locally or can offload its data to be executed remotely at the edge server or cloud. Binary offloading was adopted in [16]–[20], where the computation data of a task is completely offloaded to one node. Partial offloading consisting of dividing the task into multiple portions to be executed simultaneously at different nodes enhances the system performance and allows the task to be completed within its deadline. Partial offloading was adopted in [21], [22] where computation tasks can be executed locally, as well as, remotely at the edge server or cloud.

In addition to multi-layer edge servers, D2D communication was integrated in MEC for D2D computation offloading to peer mobile terminals to further enhance system performance. In D2D-enabled Het-MEC, the proposed approaches in [3] and [4] considered binary offloading, while [6]–[9] adopted partial offloading. The authors in [6] used partial offloading to multiple peer devices aiming at minimizing energy consumption and execution time. The authors in [7] adopted five modes including local execution, complete and partial D2D, MEC and mobile cloud computing (MCC) offloading. The task can be divided into two parts to be executed locally and remotely, while minimizing latency and energy consumption. In [8] and [9], the task is divided into three parts to be executed locally, at one peer MT and MEC, simultaneously, aiming at maximizing the number of supported devices while meeting delay and power constraints. Many studies in the literature used the Lyapunov method for real-time computation offloading decisions providing near-optimal performance with low computational complexity. An energy-efficient task assignment, wireless and computation resource allocation were addressed in [23]. The authors in [24] proposed partial offloading while decomposing the problem into multiple Lyapunov optimization sub-problems while minimizing the energy consumption. The authors in [13] consider price-aware Lyapunov-based offloading strategies to decide at the edge server on the number of tasks to be offloaded and resources to be purchased while minimizing the total offloading cost.

Moreover, some work addressed admission control, which plays a major role in reducing the network congestion and enhancing the system performance. The authors in [10] formulated the admission control policies as a sequential game. In [14], the authors adopted binary offloading and proposed a task admission approach aiming at minimizing the total energy consumption while meeting the tasks latency constraints. In [11], the authors addressed admission control, computation resource allocation, and power control at the fog node while applying Lyapunov optimization for the different sub-problem aiming at maximizing system throughput.

In summary, the existing D2D-enabled Het-MEC works are still limited in terms of scalability, system performance and computation capacity. They mainly used binary offloading while some used partial offloading with limited number of nodes. Moreover, previous studies are in general network-centric and focus on one or two objective functions, while not considering queue congestion. In this work, we propose a user-centric real-time Lyapunov-based computation offloading decision and computation resource allocation in D2D-enabled Het-MEC system. The novelty of the work mainly lies in (1) accommodating for different offloading modes including local execution, complete and partial offloading strategies to multiple cooperating nodes in D2D-enabled heterogeneous MEC networks, (2) providing real-time optimized solutions based on the Lyapunov-drift-plus-penalty optimization that was customized to meet the desired multi-objectives of stabilizing the requester queue and achieving a balance between high number of completed tasks, low energy consumption

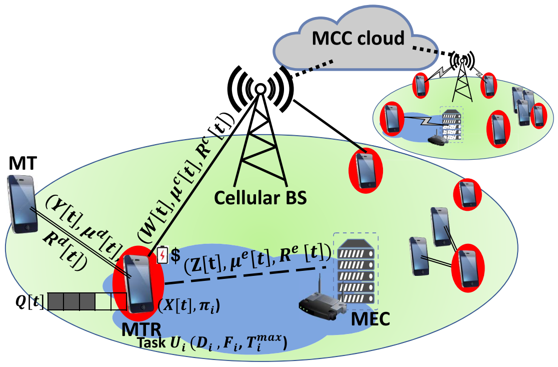
MCC cloud
Cellular BS
MT
$(Y[t], \mu^d[t], R^d[t])$
$(W[t], \mu^c[t], R^c[t])$
$\$$
$(Z[t], \mu^e[t], R^e[t])$
$Q[t]$
$(X[t], \pi_i)$
MTR
MEC
Task $U_i\ (D_i, F_i, T_i^{max})$

Fig. 1. Heterogeneous MEC network formed by mobile terminals, edge servers and cloud.

and monetary cost, and (3) leveraging the use of SVM to provide admission control scheme to further enhance the system performance.

III. System Model

As presented in Figure 1, we consider a network composed of mobile terminals, edge servers, in addition to cloud computing infrastructure. A mobile terminal (MT) is considered a requester mobile terminal (MTR) when it has computation tasks to be executed, which are often exhausting and require high computation capabilities with limited latency constraints. Hence, cooperating nodes with higher computation capacity are needed to assist in executing these computation tasks within their deadline. In our work, we adopt partial offloading and allow computation tasks to be performed locally or offloaded simultaneously to multiple nodes. Accordingly, a MTR can communicate with a peer MT using short range wireless technologies such as Bluetooth, with the edge servers and cloud through WiFi access points and cellular base stations.

A. Main System Parameters

In our work, we target low-latency and data partitioned bitwise independent oriented applications such as virus scan, file compression, face recognition and vision applications [25]. This type of applications can be abstracted as a profile with different requirements of a task U_i as follows: (1) D_i the task computation data size, (2) $T_i^{\max}$ the maximum delay tolerance, and (3) F the computation intensity in terms of number of CPU cycles required for computing 1-bit of data [26].

A requester then generates multiple computation tasks with an average task arrival rate λ_{N} reflecting the number of tasks requested per second with an average data size λ_{d} bits per task. We denote by T_i^0 the time of arrival of task U_i, i.e.,: the data of the task U_i is added at the MTR i queue at T_i^0 to be processed. We denote by $\mathcal{T}_i$, the deadline of task U_i, which is equal to $T_i^0 + T_i^{\max}$. We denote by $\mathcal{T}_i^d[t]$ the dynamic delay tolerance of the task which represents at every time slot the remaining time for a task to be completed. Hence, $\mathcal{T}_i^d[t]$ is set to be equal to $\mathcal{T}_i - T_{\mathrm{s}} \cdot t$; i.e.,: $\mathcal{T}_i^d[t]$ is initialized to $T_i^{\max}$ at T_i^0, and decreases by the time slot duration T_{s} every time slot. Accordingly, the computation data of task U_i should be fully executed before time slot $\mathcal{T}_i$ and the dynamic deadline

TABLE I
Main System Parameters and Decision Variables

Parameters	
U_i	computation task i
D_i	size of computation data (in bits) required to accomplish task U_i
F	number of CPU cycles required for computing 1-bit data
$T_i^{\max}$	maximum delay tolerance for task U_i
$\widehat{\mu}^d[t]$	available fractions of computation resource of a peer MT at time slot t
$\widehat{\mu}^e[t]$	available fractions of computation resource at the edge server MEC at time slot t
$\widehat{\mu}^c[t]$	available fractions of computation resource at the cloud MCC at time slot t
$\widehat{X}[t]$	estimated allowed data to be processed locally within T_{s} at time slot t
$\widehat{Y}[t]$	estimated allowed data to be processed at the peer MT within T_{s} at time slot t
$\widehat{Z}[t]$	estimated allowed data to be processed at the MEC within T_{s} at time slot t
$\widehat{W}[t]$	estimated allowed data to be processed at the MCC within T_{s} at time slot t
Decision Variables	
$\ell\ell'$	offloading strategy $\ell \in L[t]$ with specific computation resources allocation ℓ' at time slot t
$X[t]$	size of computation data executed locally by the MTR at time slot t
$Y[t]$	size of computation data offloaded to the peer MT at time slot t
$Z[t]$	size of computation data offloaded to MEC at time slot t
$W[t]$	size of computation data offloaded to the MCC at time slot t
$\mu^d[t]$	fraction of computation resources allocated by the peer MT at time slot t
$\mu^e[t]$	fraction of computation resources allocated by MEC at time slot t
$\mu^c[t]$	fraction of computation resources allocated by MCC at time slot t

$\mathcal{T}_i^d[t]$ reaches zero, to signal the task deadline has expired. Otherwise, the task data left to be processed will remain in the queue to be processed. When the task deadline is reached, the queued task U_i will be dropped and considered not completed. In this case, the remaining data belonging to the task U_i will also be removed from the queue. Let π_i denote whether a task U_i is completed, i.e.,: π_i is set to one if the data of task U_i is processed within its deadline, and zero otherwise.

We denote by F^l, F^d, F^e and F^c the total computation resources (in CPU cycles/s) of a MTR, a peer MT, MEC and the MCC, respectively. We assume that the computation resources are divided into equal-size chunks of size S^c. We denote by $\widehat{\mu}^d[t]$, $\widehat{\mu}^e[t]$ and $\widehat{\mu}^c[t]$, real variables varying between 0 and 1, indicating the available fractions of computation resources at MT, MEC, and MCC, respectively, at a time slot t. The main system parameters are summarized in Table I.

Our proposed method runs at the requester side and decides on the offloading mode and the computation resources assigned at every time slot based on system parameters such as MTR queue backlog size, transmission throughput, energy consumption, monetary cost, available computation resources and tasks' deadlines.To obtain the needed information, the requester is assumed to exchange control information over dedicated control channels with the remote nodes. The amount of data being exchanged is minimal, that leads to a negligible delay. Moreover, our proposed approach does not require any changes in the cellular/WiFi/Bluetooth standards and can accommodate for any multi-user communication resource management scheme. Accordingly, at every time slot *t*, the best computation offloading mode is selected, as well as the best computation resources needed to provide the best balance between MTR queue stability, energy consumption and monetary cost while meeting tasks' deadlines.

B. Decision Variables

- *Computation offloading and resource allocation modes (L[t]):* is the set of possible computation offloading modes. In our work, we consider a set of modes that tend to save the cloud and edge servers computation resources while encouraging more cost effective modes such as local execution and D2D offloading. *L*[*t*] includes up to 9 offloading modes: (1) local execution, (2) complete offloading to a peer MT, (3) complete offloading to a MEC, (4) complete offloading to MCC, (5) partial offloading (PO) between local execution and one peer MT, (6) PO between local execution and MEC, (7) PO between local execution and MCC, (8) PO between local execution, one MT and MEC, and (9) PO between local execution, one peer MT, MEC and MCC. At every time slot, the proposed approach decides on one offloading mode providing the best performance for computation offloading. We note that our approach can be easily extended to accommodate for all the possible combination offloading modes at the expense of increased complexity.
- *Computation offloading mode (ℓ):* The index $\ell \in L[t]$ indicates the possible offloading modes represented by *L*[*t*]. Based on the selected offloading mode ℓ, the allowed data size to be offloaded to every node, during time slot duration T_{s}, can be determined.
- *Resource allocation (ℓ'):* For every offloading mode ℓ, there will be different possibilities for allocating the computation resources of the cooperating nodes. Therefore, we denote by $\ell\ell'$, the offloading mode ℓ with specific computation resource allocation ℓ' assigned by the MT, MEC and cloud.
- *Computation data offloading and resources:* The computation offloading mode ℓ and resource allocation strategy ℓ' will decide on the size of data to be locally executed *X*[*t*], or offloaded *Y*[*t*], *Z*[*t*] and *W*[*t*] to the peer MT, MEC server and cloud, respectively. The selected mode will also indicate the fraction of computation resources allocated $\mu^d[t]$, $\mu^e[t]$ and $\mu^c[t]$ by the peer MT, MEC server and cloud, respectively, at every time slot *t*.

C. General Parameters

- *Time slot duration (T_{s})*: is the time slot duration, in seconds, representing how often the decision is taken.
- *Arrival data (A[t]):* represents the amount of computation data, in bits, that arrives to the user's queue within time slot *t*.
- *Processed data ($\mathcal{D}[t]$):* represents the amount of computation data, in bits, actually processed at time slot *t*, i.e.,; $\mathcal{D}[t] = X[t] + Y[t] + Z[t] + W[t]$, based on the selected offloading mode ℓ and resource allocation ℓ'.
- *Queue backlog (Q[t]):* is located at the requester end and represents the amount, in bits, of unfinished work as computation data not being processed yet at the beginning of time slot *t* and can be expressed as follows:

$$Q[t+1] = Q[t] - \mathcal{D}[t] + A[t] \tag{1}$$

- *Cost* ($C_{\ell\ell'}[t]$): represents the estimated normalized penalty function in terms of energy consumption $E_{\ell\ell'}[t]$ and monetary cost $\phi_{\ell\ell'}[t]$ when using offloading mode ℓ with computation resource allocation ℓ' at time slot *t*.

D. Computing Models

In our work, we adopt four computing models: (1) local computing, (2) D2D computing, (3) edge computing, and (4) cloud computing. We consider the energy consumed for transmission and computation execution, as well as monetary cost for transmission and computation data offloading. We assume a usage-based pricing, which charges users proportionally with respect to the amount of data consumed. In general, some interfaces have much less cost than others. In addition, transmission over D2D connectivity such as Bluetooth may be free of charge while charging for data computation to provide incentives for the peer MT to share its resources.

1) Local Computing: The computation delay T^l of processing *X* data locally can be computed as follows: $T^l = \frac{X \cdot F}{F^l}$. Accordingly, within a time slot duration of T_{s}, the maximum possible data to be processed locally at time slot *t* can be estimated as follows:

$$\widehat{X}[t] = \frac{T_{\text{s}} \cdot F^l}{F} \tag{2}$$

The energy consumed by MTR to execute locally *X*[*t*] bits can be expressed as follows:

$$E^l[t] = \mathfrak{C}^l \cdot X[t] \cdot \left(F^l\right)^2 \tag{3}$$

where $\mathfrak{C}_i^l$ is the local effective switched capacitance of MTR, reflecting the energy consumption coefficient related to its CPU performance [7], [16].

2) D2D Device Computing: For D2D offloading, we consider the transmission T^{dt} and computation T^{dc} delays. The total delay T^d of processing *Y* data by a peer MT can be expressed as follows:

$$T^d[t] = T^{dt}[t] + T^{dc}[t] = \frac{Y[t]}{R^d[t]} + \frac{Y[t] \cdot F}{\mu^d[t] \cdot F^d} \tag{4}$$

where $R^d[t]$ represents the transmission rate of the D2D link at every time slot *t*. Hence, the transmission rate will vary on a time slot basis to consider the communication channel quality. However, since we consider a small time slot duration, we assume the transmission rate won't change within a time slot duration [23]. The assigned computation resources $\mu^d[t] \cdot F^d$ are less than the available resources $\widehat{\mu^d}[t] \cdot F^d$. In our work, we neglect the downlink transmission time since the output data of the computation task is normally much smaller than that of the input data [7]–[9]. Accordingly, the maximum possible data to be processed at the peer MT, at time slot *t*, with $\mu^d[t]$, within T_{s}, can be estimated as follows:

$$\widehat{Y}[t] = \frac{T_{\mathrm{s}}}{\frac{1}{R^d[t]} + \frac{F}{\mu^d[t]\cdot F^d}} \tag{5}$$

The total energy E^d includes transmission E^{dt}, reception E^{dr} and computation processing E^{dc} energy consumption, and can be expressed as follows:

$$\begin{aligned} E^d[t] &= E^{dt} + E^{dr} + E^{dc} \\ &= P^{dt} \cdot \frac{Y[t]}{R^d[t]} + P^{dr} \cdot \frac{Y[t]}{R^d[t]} + \mathfrak{C}^d \\ &\quad \times Y[t] \cdot \left(\mu^d[t] \cdot F^d\right)^2 \end{aligned} \tag{6}$$

where P^{dt} and P^{dr} are the transmit and receive power of a MT using D2D connectivity, respectively. $\mathfrak{C}^d$ is the effective switched capacitance of the peer MT [7], [16].

As an incentive for MTs to cooperate, MTR is charged by ϕ^{dc} USD per Hz per time slot for computation processing at the peer MT. We denote by ϕ^{dt} the amount of USD charged per bit per time slot for transmission over Bluetooth, which may be typically free of charge. However, we included it for completeness to account for other short range connections that might be charged. Therefore, the total monetary cost $\phi^d[t]$ of offloading *Y*[*t*] (bits) to the peer MT is composed of communication charges expressed as $Y[t] \cdot \phi^{dt}$ and computation charges expressed as $\mu^d[t] \cdot F^d \cdot \phi^{dc}$ for sharing $\mu^d[t]$ of their computation capabilities F^d.

$$\phi^d[t] = Y[t] \cdot \phi^{dt} + \mu^d[t] \cdot F^d \cdot \phi^{dc}. \tag{7}$$

3) Edge Computing: Similar to (5), the maximum possible data to be processed at the MEC, at time slot *t*, within a time slot duration of T_{s}, can be estimated as follows:

$$\widehat{Z}[t] = \frac{T_{\mathrm{s}}}{\frac{1}{R^e[t]} + \frac{F}{\mu^e[t]\cdot F^e}} \tag{8}$$

where $R^e[t]$ represents the WiFi transmission rate, and the assigned computation resources $\mu^e[t] \cdot F^e$ is less than the available resources $\widehat{\mu^e}[t] \cdot F^e$. In our work, we focus on minimizing the energy consumption of the MTs, hence, the total energy E^e at time slot *t* can be expressed as follows:

$$E^e[t] = P^{et} \cdot \frac{Z[t]}{R^e[t]} \tag{9}$$

where P^{et} is the power consumed by requester MTR to transmit *Z*[*t*] data bits to MEC over WiFi link. The total monetary cost for offloading *Z*[*t*] to the MEC includes communication and computation charges, and can be expressed as follows:

$$\phi^e[t] = Z[t] \cdot \phi^{et} + \mu^e[t] \cdot F^e \cdot \phi^{ec} \tag{10}$$

where ϕ^{et} represents the amount of USD charged per bit for transmission over WiFi, and ϕ^{ec}, the amount of USD charged per Hz per time slot for computation processing at MEC.

4) Cloud Computing: Similar to edge computing, the maximum possible data to be processed at the MCC, at time slot *t* within T_{s}, can be estimated as follows:

$$\widehat{W}[t] = \frac{T_{\mathrm{s}}}{\frac{1}{R^c[t]} + \frac{F}{\mu^c[t]\cdot F^c}} \tag{11}$$

where $R^c[t]$ represents the transmission rate of the cellular link between the MTR and the MCC, and the assigned computation resources $\mu^c[t] \cdot F^c$ is less than the available resources $\widehat{\mu^c}[t] \cdot F^c$. The total energy E^c can be expressed as follows:

$$E^c[t] = P^{ct} \cdot \frac{W[t]}{R^c[t]} \tag{12}$$

where P^{ct} is the power consumed by requester MTR to transmit computation data to MCC over cellular link. The total monetary cost for MCC offloading of *W*[*t*] bits can be expressed as follows:

$$\phi^c[t] = W[t] \cdot \phi^{ct} + \mu^c[t] \cdot F^c \cdot \phi^{cc} \tag{13}$$

where ϕ^{ct} represents the amount of USD charged per bit for transmission over cellular link, and ϕ^{cc}, the amount of USD charged per Hz per time slot for processing at MCC.

IV. Lyapunov-Based Computation Offloading

In our work, we aim at selecting the best strategy that maximizes over time the total number of completed tasks Π while minimizing the total cost in terms of energy consumption Ψ and monetary cost Φ subject to system constraints. Accordingly, the objective function can be expressed as follows:

$$\underset{\substack{\mathbf{L},\boldsymbol{\ell},\boldsymbol{\ell}',\mathbf{X},\mathbf{Y},\mathbf{Z} \\ \mathbf{W},\boldsymbol{\mu}^d,\boldsymbol{\mu}^e,\boldsymbol{\mu}^c}}{\mathrm{maximize}} \quad \sum_{t}^{\infty} \Pi[t] - V_1 \cdot \Psi[t] - V_2 \cdot \Phi[t] \tag{14}$$

where $\Psi[t]$ is the total energy consumed when using strategy $\ell\ell'$ and is composed of the energy consumed locally $E^l[t]$, for D2D offloading $E^d[t]$, edge offloading $E^e[t]$, and cloud offloading $E^c[t]$. Similarly, the total monetary cost is composed of $\phi^d[t]$, $\phi^e[t]$ and $\phi^c[t]$. The weights V_1 and V_2 determine the impact of minimizing the energy and monetary cost, respectively. The problem can be shown to be a mixed-integer non-linear program (MINLP) and is NP-hard. To provide real-time optimized solutions, we suggest solving our optimization problem by selecting the most efficient strategy at every time slot *t* that will greedily provide the balance between the different objectives and achieve optimized solutions at the long run. For this reason, we take advantage of the Lyapunov Drift-Plus-Penalty optimization to provide optimized real time solutions aiming at minimizing the queue length at every time slot which leads to maximizing the total number of completed tasks while minimizing the penalty function expressed in terms of energy consumption and monetary cost.

We assume that tasks with specific arrival rate and different data sizes are queued to be processed within their deadline. The queue length may grow indefinitely when the processing rate is less than the arrival rate, hence, computation data may stay in a congested queue longer, and tasks may not be

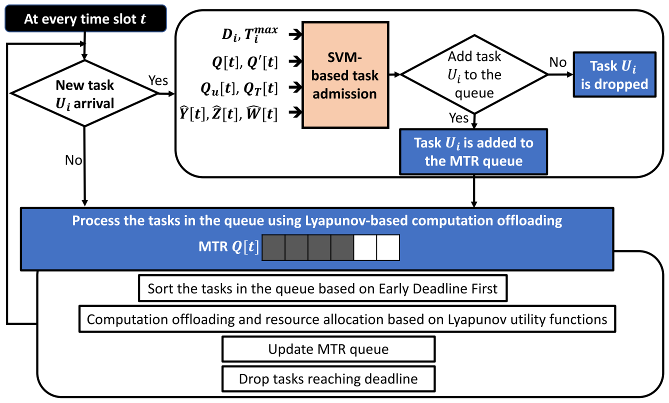
At every time slot t
New task U_i arrival
Yes
No
D_i, T_i^{max}
$Q[t], Q'[t]$
$Q_u[t], Q_T[t]$
$\hat{Y}[t], \hat{Z}[t], \hat{W}[t]$
SVM-based task admission
Add task U_i to the queue
No
Task U_i is dropped
Yes
Task U_i is added to the MTR queue
Process the tasks in the queue using Lyapunov-based computation offloading
MTR $Q[t]$
Sort the tasks in the queue based on Early Deadline First
Computation offloading and resource allocation based on Lyapunov utility functions
Update MTR queue
Drop tasks reaching deadline

Fig. 2. The proposed SVM-based task admission with Lyapunov-based computation offloading and resource allocation.

completed within their deadline. Therefore, our goal is to stabilize the user queue backlog to keep it from building up and causing tasks to be dropped, and hence, completing the maximum number of tasks within their deadlines. Therefore, we use Lyapunov drift-plus-penalty optimization, which enables real-time decisions and provides near-optimal performance for the chosen objectives with low computational complexity [27].

As presented in Figure 2, we first present a user-centric Lyapunov-based computation offloading (LCO) approach operating under real-time network conditions. To achieve our main objectives, we customize the Lyapunov penalty function to provide a balance between stabilizing requester queue and minimizing energy consumption and monetary cost. Lyapunov-based utility functions are then computed for the set of possible offloading modes $L[t]$ including local, complete or partial D2D, MEC and MCC offloading. The mode ℓ providing the maximum utility function will be selected with determined resource allocation ℓ' at every time slot t.

A. Lyapunov Drift-Plus-Penalty

In general, the Lyapunov optimization aims at minimizing the change in the MTR queue backlog size $Q[t]$ at every time slot t resulting in a scheduling algorithm that stabilizes the queue over time and enhances the system performance in terms of number of completed tasks [27]. The Lyapunov function measures the MTR queue congestion using a quadratic function $\zeta(Q[t])$ expressed as follows:

$$\zeta(Q[t]) = \frac{1}{2}(Q[t])^2 \tag{15}$$

The Lyapunov drift function $\Delta(Q[t])$ measures the expected difference in the Lyapunov function between two consecutive time slots given the queue state at time slot t, as follows:

$$\Delta(Q[t]) = \mathbb{E}\{\zeta(Q[t+1]) - \zeta(Q[t])|Q[t]\} \tag{16}$$

Minimizing the Lyapunov drift function at every time slot t consistently pushes the user queue towards a lower congestion state, and thereby maintains queue stability [27]. To handle our multi-objective problem, we use the Lyapunov drift-plus-penalty method, where the base Lyapunov optimization is extended to include a penalty cost function $C_{\ell\ell'}[t]$, in our case, expressed in terms of monetary cost and energy consumption. The penalty function is weighted by a positive coefficient V that determines the significance of the penalty function compared to stabilizing the queue. The objective function of the Lyapunov drift-plus-penalty approach will be:

$$\underset{\ell\ell' \in L[t]}{\text{argmin}} \quad \Delta(Q[t]) + V \cdot \mathbb{E}\{C_{\ell\ell'}[t]|Q[t]\} \tag{17}$$

where $\mathbb{E}\{C_{\ell\ell'}[t]|Q[t]\}$ is the expected offloading cost using offloading mode ℓ and resource allocation ℓ'. The multi-objective function in (17) can be upper bounded as follows:

$$\begin{aligned}
&\Delta(Q[t]) + V \cdot \mathbb{E}\{C_{\ell\ell'}[t]|Q[t]\} \\
&\quad \leq \frac{1}{2}\mathbb{E}\Big\{\mathcal{D}_{\ell\ell'}[t]^2 + A[t]^2|Q[t]\Big\} - Q[t] \cdot \mathbb{E}\{\mathcal{D}_{\ell\ell'}[t]|Q[t]\} \\
&\qquad + Q[t] \cdot \mathbb{E}\{A[t]|Q[t]\} + V \cdot \mathbb{E}\{C_{\ell\ell'}[t]|Q[t]\}
\end{aligned} \tag{18}$$

Minimizing our target multi-objective function (17) can thus be achieved by minimizing the upper bound in (18). We define $B[t]$ and λ as follows:

$$B[t] = \frac{1}{2}\mathbb{E}\Big\{\mathcal{D}_{\ell\ell'}[t]^2 + A[t]^2|Q[t]\Big\} \tag{19}$$

$$\lambda = \mathbb{E}\{A[t]|Q[t]\} = \mathbb{E}\{A[t]\} \tag{20}$$

where $B[t]$, assumed to be bounded by a fixed value B, is the sum of the variances of the service and arrival rate, which are non controllable parameters. λ represents the expected data arrival $A[t]$ defined by the application. Thus, minimizing (17) can be achieved by minimizing the controllable part of the upper bound in (18) which is equivalent to maximizing $\mathbb{E}\{Q[t] \cdot \mathcal{D}_{\ell\ell'}[t] - V \cdot C_{\ell\ell'}[t]|Q[t]\}$. Using the concept of opportunistically maximizing an expectation, the upper bound expression can be maximized at every time slot t, as follows:

$$\underset{\ell\ell' \in L[t]}{\text{argmax}} \quad Q[t] \cdot \mathbb{E}\{\mathcal{D}_{\ell\ell'}[t]|S[t]\} - V \cdot C_{\ell\ell'}[t] \tag{21}$$

where $\mathbb{E}\{\mathcal{D}_{\ell\ell'}[t]|S[t]\}$ is the expected amount of data to be processed using offloading mode ℓ and resource allocation ℓ' considering the system state $S[t]$ including the transmission rates over the multiple wireless interfaces and the computation resources availability at time slot t. Accordingly, the objective function in (14) can be achieved by maximizing the Lyapunov-based objective function presented in (21). The optimization problem can then be reformulated to select the best offloading mode that maximizes the Lyapunov-based utility function which can be shown to be NP-hard. However, due to the limited number of possible offloading modes, the optimized solutions can be achieved by cycling over the set of all possible actions $L[t]$ and selecting the best offloading mode ℓ that ensures queue stability while reducing the offloading cost at every time slot t. The selected strategy will then determine the size of data $\mathcal{D}[t]$ to be processed locally and remotely.

B. Lyapunov-Drift-Plus-Penalty Performance Bounds

We denote by C^* the desired infimum time average penalty function and assume that the expected penalty is lower bounded by C_{min}. We assume that the arrival process is strictly within the network capacity region, hence,

$\mathbb{E}\{\mathcal{D}_{\ell\ell'}[t]|S[t]\} = \lambda + \epsilon$, where $\epsilon > 0$. Therefore, replacing (19) and (20) in (18) and using the iterative expectation law results in: $\mathbb{E}\{\zeta(Q[t]) + V \cdot C_{\ell\ell'}[t]|Q[t]\} \leq B - \epsilon \cdot \mathbb{E}\{Q[t]\} + VC^*$. Summing on the time domain, the performance bounds on time average penalty and queue can be derived as follows:

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=1}^{t} \mathbb{E}\{C_{\ell\ell'}[\tau]\} \leq C^* + \frac{B}{V} \tag{22}$$

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=1}^{t} \mathbb{E}\{Q[\tau]\} \leq \frac{B + V\cdot(C^* - C_{min})}{\epsilon} \tag{23}$$

The performance bounds on the time average penalty and queue backlog hold for all $V > 0$. The theorem shows that increasing the value of V will result in B/V decreasing leading to a time average penalty being arbitrarily near the optimal value. However, increasing V will increase the time average queue leading to a lower number of completed tasks. Accordingly, the Lyapunov-drift-plus-penalty approach provides $[O(1/V), O(V)]$ performance-delay tradeoff resulting in a time average penalty that is within $O(1/V)$ of optimality, with $O(V)$ tradeoff in average queue size [27].

C. Penalty Function in Terms of Energy Consumption and Monetary Cost

We define the cost $C_{\ell\ell'}[t]$ in terms of energy consumption and monetary cost as follows:

$$C_{\ell\ell'}[t] = \beta\frac{E_{\ell\ell'}[t]}{E^{max}[t]} + (1-\beta)\frac{\phi_{\ell\ell'}[t]}{\phi^{max}[t]} \tag{24}$$

where β is a real variable varying between 0 and 1, defining the relative importance of minimizing the energy consumption compared to minimizing the monetary cost. $E_{\ell\ell'}[t]$ and $\phi_{\ell\ell'}[t]$ are the energy and monetary costs consumed when using offloading mode ℓ with resource allocation ℓ'. For instance, for strategy $\ell = 1$, the local execution consumes $E_{\ell\ell'}[t] = E^l[t]$. For $\ell = 9$, the data is divided between local execution and remote execution at a peer MT, MEC and MCC, the total energy and monetary cost can then be expressed as follows: $E_{\ell\ell'}[t] = E^l[t] + E^d[t] + E^e[t] + E^c[t]$ and $\phi_\ell[t] = \phi^l[t] + \phi^d[t] + \phi^e[t] + \phi^c[t]$. In (24), $E^{max}[t]$ and $\phi^{max}[t]$ represent the maximum energy consumption and monetary cost of all the offloading modes $\ell \in L[t]$ with full computation resources utilization. More details about utility computations are presented in Section IV-D (Algorithm 2). In our formulation, we divide $E_{\ell\ell'}[t]$ and $\phi_{\ell\ell'}[t]$ by their maximum values $E^{max}[t]$ and $\phi^{max}[t]$, respectively, to normalize both entities and scale their values within the range of 0 to 1. The Lyapunov-based utility function can then reflect a weighted sum of the three different objectives where the normalized penalty cost function in terms of monetary cost and energy consumption is weighted by V and the expected amount of computation data to be processed reflecting the service rate is weighted by the observed queue size $Q[t]$.

D. Lyapunov-Based Computation Offloading Approach

The proposed LCO approach executes in real-time, autonomously at the user end following the steps shown in Algorithm 1. At every time slot t, the cycle starts by sorting the tasks in the queue in ascending order based on their deadline $\mathcal{T}_i$ following the Earliest Deadline First (EDF) scheduling algorithm where the task with the earliest deadline, is served first [28]. The proposed LCO approach then considers the set of all the possible actions $L[t]$ and computes the utility functions for the different offloading modes presented in Section III considering different resource allocation ℓ' as presented in Algorithm 2. The offloading mode ℓ with computation resource allocation ℓ' providing the maximum utility function will be selected. If the maximum utility is negative, there is no benefit of processing the data since the device will be consuming more energy and monetary cost than benefiting from processing the computation data; in this case, no processing is recommended. Otherwise, the offloading mode ℓ is used with the determined resource allocation ℓ'. The data in the queue is then processed depending on the estimated allowed data to be processed within time slot t. If the data of a task is fully processed, the task is removed from the queue and considered completed, otherwise some of its data is processed and the remaining part stays in the queue to be processed in the next time slots. If the task reaches its deadline and is not yet fully processed, the task is dropped and removed from the queue. The tasks sorted in the queue are handled consecutively until the estimated allowed data to be processed within time slot t is fully used or the queue becomes empty. The process is repeated until all the tasks are considered.

Algorithm 1 The Proposed Lyapunov-Based Computation Offloading Approach (LCO)

```
Input:
- System parameters presented in Table I
- Set of possible offloading strategies: $L[t]$
- Cost weight: $V$
- Initial queue backlog size: $Q[0] = 0$, initial data processed: $\mathcal{D}[0] = 0$
Output:
- Computation offloading strategy $\ell \in L[t]$
- Task data offloading decision: $X[t]$, $Y[t]$, $Z[t]$, $W[t]$
- Computation resource allocation: $\mu^d[t]$, $\mu^e[t]$, $\mu^c[t]$
- Tasks completed: $\pi_i$
At every time slot $t$:
1: Sort tasks in the queue based on their deadlines $\mathcal{T}_i$, hence, tasks with
   lower deadlines will be processed first following the EDF methodology
2: Compute the utility functions for the $\ell \in L[t]$ offloading modes with
   different resource allocation strategies $\ell'$ based on Algorithm 2
3: Select the computation offloading strategy $\ell$ with resource allocation $\ell'$
   providing the maximum utility function
4: If the maximum utility is negative, then defer processing any computation
   task since the offloading cost is much greater than the benefit of processing
   the data; i.e.,:$\mathcal{D}[t] = 0$
5: else Process the tasks sorted in the backlog queue consecutively
   until $\widehat{\mathcal{D}}_{\ell\ell'}[t]$ is fully processed or the queue is empty as follows:
     • Keep the data of task $U_i$ in the queue if the task did not reach is
       deadline at time slot $t$
     • Remove task $U_i$ from the queue if its data is fully processed within
       its deadline, and set $\pi_i$ to 1 to indicate $U_i$ is completed
     • Remove the task $U_i$ and its remaining data from the queue if it
       reaches its deadline and set $\pi_i$ to 0 to indicate $U_i$ is dropped
     • Update the data processed to be $\mathcal{D}[t] = \min\{\widehat{\mathcal{D}}_{\ell\ell'}[t], Q[t]\}$
6: Update the actual data processed locally, or remotely at the MT, MEC,
   and MCC $X[t]$, $Y[t]$, $Z[t]$ and $W[t]$, respectively.
7: Update queue when a new task $U_i$ is admitted to the queue, $Q[t] =$
   $Q[t-1] - \mathcal{D}[t] + A[t]$
8: Repeat process (1)-(7) until all the tasks are considered
```

Algorithm 2 Lyapunov-Based Utility Functions Computation for Different Offloading Strategies $\ell \in L[t]$ and Resource Allocation ℓ'

```
Output: Utility functions for the $\ell \in L[t]$ offloading modes based on (21) with different computation resource allocation $\ell'$
1: Compute the local computing utility function for $\ell = 1$:
     • Estimate the maximum data to be processed $\widehat{X}[t]$ within $T_{\mathrm{s}}$
     • Set the estimated data to be processed $\widehat{\mathcal{D}}_{\ell}[t] = \widehat{X}[t]$
     • Estimate $E_{\ell}[t] = E^{l}[t]$ and $\phi_{\ell}[t] = \phi^{l}[t]$
2: Compute complete D2D offloading utility functions for $\ell = 2$:
     • Consider different fractions of the MT computation resource $\widehat{\mu}^{d}[t]$.
       Let $\ell'$ represents the computation resource allocated in terms of
       equal-size chunks; i.e.,: $\ell' \in [1, 2, \ldots, \lfloor \frac{\widehat{\mu}^{d}[t] \cdot F^{d}}{S^{c}} \rfloor]$
     • Compute the utility function for every possible $\ell'$:
       - Estimate $\widehat{Y}_{\ell\ell'}[t]$ within $T_{\mathrm{s}}$
       - Set $\widehat{\mathcal{D}}_{\ell\ell'}[t] = \widehat{Y}_{\ell\ell'}[t]$
       - Estimate $E_{\ell\ell'}[t] = E^{d}_{\ell\ell'}[t]$ and $\phi_{\ell\ell'}[t] = \phi^{d}_{\ell\ell'}[t]$
3: Compute complete MEC and MCC offloading utility functions for $\ell = 3$
   and $\ell = 4$ similar to process 3.
4: Compute partial offloading including local execution and D2D offloading
   $\ell = 5$, local execution and MEC offloading $\ell = 6$, and local execution
   and MCC offloading $\ell = 7$ utility functions:
     • Consider different computation resource allocation $\ell'$ at the nodes
       (MT for $\ell = 5$, MEC for $\ell = 6$ or MCC for $\ell = 7$)
     • Compute the utility function for every $\ell'$:
       - Estimate the maximum data to be processed $\widehat{D}_{\ell'}[t]$ at the
         cooperating nodes within $T_{\mathrm{s}}$, i.e.,: $\widehat{D}_{\ell'}[t] = \widehat{Y}_{\ell\ell'}[t]$,
         $\widehat{Z}_{\ell\ell'}[t]$ or $\widehat{W}_{\ell\ell'}[t]$ for $\ell = 5, 6$ or 7, respectively
       - Set $\widehat{\mathcal{D}}_{\ell\ell'}[t] = \widehat{X}[t] + \widehat{D}_{\ell'}[t]$
       - Estimate $E_{\ell\ell'}[t] = E^{l}[t] + E^{\ell}_{\ell'}[t]$
         and $\phi_{\ell\ell'}[t] = \phi^{l}[t] + \phi^{\ell}_{\ell'}[t]$
5: Compute partial offloading including local execution, D2D and MEC
   offloading utility function with $\ell = 8$
     • Consider full utilization of the available computation resource $\widehat{\mu}^{d}[t]$
       at the MT
     • Consider different fractions of computation resource $\ell'$ available at
       the MEC less than $\widehat{\mu}^{e}[t]$
     • Compute the utility function for every possible $\ell'$:
       - Estimate $\widehat{X}[t]$ and $\widehat{Y}[t]$ within $T_{\mathrm{s}}$
       - Estimate $\widehat{Z}_{\ell\ell'}[t]$ for every $\ell'$ within $T_{\mathrm{s}}$
       - Set $\widehat{\mathcal{D}}_{\ell\ell'}[t] = \widehat{X}[t] + \widehat{Y}[t] + \widehat{Z}_{\ell\ell'}[t]$
       - Estimate $E_{\ell\ell'}[t] = E^{l}[t] + E^{d}[t] + E^{e}_{\ell\ell'}[t]$
         and $\phi_{\ell\ell'}[t] = \phi^{l}[t] + \phi^{d}[t] + \phi^{e}_{\ell\ell'}[t]$
6: Compute partial offloading including local execution, D2D, MEC and
   MCC offloading utility function with $\ell = 9$
     • Consider full utilization of the available computation resource $\widehat{\mu}^{d}[t]$
       at the MT and $\widehat{\mu}^{e}[t]$ at the MEC
     • Consider different fractions of computation resource $\ell'$ available at
       the MCC less than $\widehat{\mu}^{c}[t]$
     • Compute the utility function for every $\ell'$:
       - Estimate $\widehat{X}[t]$, $\widehat{Y}[t]$ and $\widehat{Z}[t]$ within $T_{\mathrm{s}}$
       - Estimate $\widehat{W}_{\ell\ell'}[t]$for every $\ell'$ within $T_{\mathrm{s}}$
       - Set $\widehat{\mathcal{D}}_{\ell\ell'}[t] = \widehat{X}[t] + \widehat{Y}[t] + \widehat{Z}[t] + \widehat{W}_{\ell\ell'}[t]$
       - Estimate $E_{\ell\ell'}[t] = E^{l}[t] + E^{d}[t] + E^{e}[t] + E^{c}_{\ell\ell'}[t]$ and
         $\phi_{\ell\ell'}[t] = \phi^{l}[t] + \phi^{d}[t] + \phi^{e}[t] + \phi^{c}_{\ell\ell'}[t]$
```

E. Complexity Analysis

The proposed Lyapunov-based approach is user-centric and provides simple and fast solutions based on the current observed queue state and resources availability without a-priori knowledge of the application and the channel variations [27]. At every time slot, the queued tasks are sorted which can be implemented using quick sort algorithm of complexity $O(n \log n)$. In general, the number of tasks n in the queue is very small and the overhead can be considered negligible. Then, m utility functions for the nine modes including local execution or remote execution at one MTR, MEC and MCC are computed based on Algorithm 2. Every mode ℓ has different number of utility functions that can be computed based on the resource allocation ℓ' using (21). We assume that the computation resources of the peer MT, MEC and MCC are divided into H^{d}, H^{e} and H^{c} equal-size chunks representing fractions of their computation resources, respectively. Utilities will be computed as presented in Algorithm 2 for the different modes as follows: (1) one utility for local execution $\ell = 1$, (2) H^{d} utilities for D2D offloading $\ell = 2$ and $\ell = 5$, (3) H^{e} utilities for MEC offloading $\ell = 3$ and $\ell = 6$, (4) H^{c} utilities for MCC offloading $\ell = 4$ and $\ell = 7$, (5) H^{e} utilities for $\ell = 8$, and (6) H^{c} utilities for $\ell = 9$. The worst case scenario is having all the resources available at the different nodes; accordingly, the total number of utilities will be $1 + 2H^{d} + 3H^{e} + 3H^{c}$. Every utility function requires the estimation of (1) the maximum data to be processed based on (2), (5), (8) and (11), (2) the energy consumption based on (3), (6), (9) and (12), and (3) the monetary cost based on (7), (10) and (13), for local execution, D2D, MEC and MCC offloading, respectively. Hence, a total of $4 \times (1 + 2H^{d} + 3H^{e} + 3H^{c})$ operations are needed to complete Algorithm 2. Note that in our work, we assume the total capacity of the MT and MEC resources are divided into ten equal-sized chunks, where a chunk is the minimum resource allocated in a time slot t representing 10% of their total capacity F^{d} and F^{e}, respectively. Due to the high computation capabilities of the MCC, its resources are divided into 100 chunks, where each represents 1% of the total MCC computation capacity while limiting the allocation per time slot to 20%. Accordingly, H^{d}, H^{e} and H^{c} will be equal to 10, 10, and 20, respectively; maximum of 111 utility functions are computed in that case. Accordingly, the maximization of the Lyapunov-based utility function can be easily carried out by cycling over all the possible actions and searching for the one providing the best balance between the desired objectives [27], [29]. The proposed approach then completes Algorithm 1 by selecting the mode providing the highest utility function and updates the system parameters such as queue size by removing the data processed and adding newly generated tasks' data, and the status of every queued task by checking whether it was completed within its deadline. Hence, our approach is of complexity $O(n \log n + m)$, however, the number of queued tasks and utility functions is very small and the overhead can be considered negligible.

The proposed approach is scalable since the offloading decision is made independently at the user end. It can accommodate for multi-users and multi-MTRs with multi-D2D, MEC and MCC offloading. In the case of multi-MTRs scenario, every user is responsible for their own decisions independently based on their observed system parameters. If the system is composed of N MTs, M MEC servers and a cloud, an additional complexity load for computing $N + M$ utilities is needed to select the peer MT and MEC providing the best performance for D2D and MEC offloading. The problem can then be reduced to our initial problem with one MT, one MEC and MCC regardless of the number of remote nodes in the network. Note that in real-life scenarios, for practicality, feasibility and to achieve performance gains, the number of interfaces used simultaneously by a mobile device may be limited. Accordingly, the execution time is very low and the solution might be reached within fractions of a msecond. Numerical evaluation of the time complexity of the proposed approach is presented in Section VI-C3.

V. Task Admission Using Support Vector Machines

Admitting all the tasks to the queue without considering the congestion status of the queue results in spending energy and monetary cost on processing data of tasks having high potential to be dropped. Moreover, admitting these tasks leads to a high queue size, which prevents other tasks with higher potential to be completed within their deadline from being served. Accordingly, to further enhance the performance of the LCO approach, we propose integrating an SVM-based task admission scheme to verify the availability of resources before a generated task can be admitted to the requester queue. In our work, we adopt Support Vector Machines for many reasons including: (1) the SVM-based model admission control scheme can be built offline only once, is generic for all the data sizes, and can be used in real-time task admission control while causing a negligible overhead as shown in our generated results, (2) solving a supervised binary classification problem where the classes are whether to admit a task or to drop it, (3) supporting both linear and nonlinear data which in our case are different input features and observations reflecting task characteristic, as well as, the availability of resources and queue congestion, (4) overcoming over-fitting and under-fitting problems and has greater generalization ability since it follows the Structural Risk Minimization principle, (5) effectiveness in high dimensional spaces, (6) high accuracy and ability to model complex decision boundaries, and (7) providing a compact description of the learned model so that the output can be easily predicted and classified based on the input features and observations [30], [31]. Although the size of the training set υ highly affects the performance of the SVM classifier with $O(\upsilon^3)$ time and $O(\upsilon^2)$ memory training complexities, the SVM is highly accurate and is able to model complex linear and nonlinear decision boundaries.

Therefore, we used a two-class binary SVM to decide whether to admit a task or drop it based on the following features and observations reflecting the task characteristic, as well as, the availability of resources and queue congestion:

(1) $\boldsymbol{D_i}$: is the size of computation data of task U_i
(2) $\boldsymbol{T_i^{max}}$: is the maximum delay tolerance for task U_i
(3) $\boldsymbol{Q[t]}$: is the queue size (in bits) at time slot *t*
(4) $\boldsymbol{Q'[t]}$: is the size of the tasks in the queue at time slot *t* with dynamic deadline less than $\mathcal{T}_i^d[t]$
(5) $\boldsymbol{Q_u[t]}$: is the number of tasks in the queue at time slot *t* with dynamic deadline less than $\mathcal{T}_i^d[t]$
(6) $\boldsymbol{Q_T[t]}$: is the average dynamic delay tolerance of the queued tasks with dynamic deadline less than $\mathcal{T}_i^d[t]$
(7) $\boldsymbol{\widehat{Y}[t]}$, (8) $\boldsymbol{\widehat{Z}[t]}$ and (9) $\boldsymbol{\widehat{W}[t]}$: are the estimated allowed data to be processed at the peer MT, MEC, and MCC, respectively, within T_s

To build our SVM model, we generate our training data set by extracting the features using our proposed LCO approach that outperformed all other conventional approaches and proposed system models in the literature as presented in our performance evaluation in Section VI-C2. To keep our SVM model complexity low, we consider different average data sizes $\lambda_d \in \{1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5\}$ Mbits per task. For every value of λ_d, we collected all the needed observations and tracked every task to indicate its actual class whether it was completed or dropped by the LCO approach. After extracting the features and generating our training data set, we used binary SVM with linear kernel and standardized the features using their corresponding weighted means and standard deviations to make them insensitive and invariant to the scales on which they are measured. The SVM model is then integrated with the LCO approach to form our proposed SVM-based task admission with Lyapunov computation offloading (SVM-LCO) approach presented in Figure 2. Accordingly, upon the arrival of a task, the features are evaluated at the MTR end and fed to the SVM-based task admission model, which decides whether to add the task to the queue or drop it due to its low potential to be completed within its deadline.

VI. Performance Results and Analysis

In this section, we present the performance evaluation, simulation setup, results and analysis to evaluate the proposed approaches under different system parameters and models.

A. Performance Evaluation

To assess the performance of the proposed LCO and SVM-LCO approaches, we generated results for the following different strategies and system models from the literature:

1. *Local Execution (LE):* the task is allowed to be only executed locally at the MTR without offloading.
2. *Complete MCC (CC):* the task data is allowed to be offloaded only to the cloud.
3. *Partial offloading- Local and D2D offloading (LD):* the task data is allowed to be partially executed locally and remotely using D2D offloading to one peer MT (system model adopted in [32])
4. *Partial offloading- Local, D2D and MCC offloading (LDM):* the task is allowed to be partially executed locally and remotely using D2D offloading to one peer MT and MCC offloading, simultaneously (system model adopted in [8] and [9]).
5. *Partial offloading- Local, D2D and MEC offloading mode selection (MS5):* the task is allowed to be partially executed based on 5 modes: (1) local execution, (2) complete D2D offloading, (3) partial D2D offloading with local execution, (4) complete MEC offloading, and (5) partial MEC offloading with local execution (system model adopted in [7]).
6. *Partial offloading- Local, D2D, MEC and MCC offloading mode selection (MS7):* we customized the mode selection approach (MS5) adopted in [7] to consider MCC offloading. Accordingly, MS7 considers 7 modes: offloading modes(1)-(5) presented in MS5 and two

additional modes: (6) complete MCC offloading, and (7) partial MCC offloading with local execution.

B. Simulation Setup

1) General Parameters: Our proposed approach is dynamic and performs on a time slot basis of duration T_{S}. In our work, we assume T_{S} to be 0.1 second, representing the smallest unit of time. In addition, we aimed in our objective function at equally minimizing energy consumption and monetary cost, hence, setting β to 0.5. A study on the weight of the cost function *V* is presented in Section VI-C1.

2) Computation Demands: We assume that the number of tasks requested per second and the task data size follow the Poisson distributions with average rates of λ_{N} and λ_{d}, respectively [23]. We set λ_{N} to 6 tasks/s and the computation requirement *F* of the tasks is set to 1000 CPU cycles/bit [9], [33]. We also target different data partitioned oriented applications with different computation intensity in terms of CPU cycles/bit such as Gzip compression application requiring 40 cycles/bit [25] and more computation-intensive applications such as multimedia services and video stream analysis requiring 737.5 cycles/bit [23]. Accordingly, we adopt different application profiles with different intensity *F* of 40, 500, 700 and 1000 CPU cycles/bit. The inter-arrival time between tasks follows the exponential distribution with average λ_{T^0} [33]. The maximum delay tolerance of the tasks were randomly selected from a uniform distribution, *U*(0.1, 1). Our simulations consider results for 10 minutes of tasks generation which corresponds to processing 3517 tasks with $\lambda_{\mathrm{N}} = 6$ tasks/s and different data sizes over an average of 6000 times slots.

3) Computation Resources: The local computation capacity of a requester MTR F^l and peer MTs F^d is assumed to be 2 GHz. The effective switched capacitance of a mobile terminal is assumed to be $\mathfrak{C}^l = \mathfrak{C}^d = 2 \times 10^{-26}$ reflecting the energy consumption coefficient related to the MT CPU performance [16]. The computation capacity of the MEC server F^e and cloud F^c are assumed to be 10 GHz and 1 THz, respectively [20]. We assume the MT and MEC resources are divided into ten equal-sized chunks, where a chunk is the minimum resource allocated in a time slot *t* representing 10% of their total capacity F^d and F^e, respectively. Due to the high computation capabilities of the MCC, its resources are divided into 100 chunks, where each represents 1% of the total MCC computation capacity. We limit the MCC resource allocated per time slot to 20%. We assume the available fractions of computation resources of the MT, MEC and MCC values are randomly selected between 0.1 and 1. The computation cost of the cooperating nodes may vary based on the service provider and its computation capacity. For instance, Hyve offers cloud services of 1 GB RAM and 4×3.0 GHz CPUs instance for 170 USD per month [34]. We assume the MTR pays 2 and 20 USD/month for 2 GHz and 10 GHz for a peer MT and MEC services, respectively. Accordingly, the cost of the D2D, MEC and cloud processing will be $\phi^{dc} = 0.3858 \times 10^{-7}$ USD/GHz per time slot, $\phi^{ec} = 0.7716 \times 10^{-7}$ USD/GHz per time slot and $\phi^{cc} = 0.5466 \times 10^{-6}$ USD/GHz per time slot, respectively.

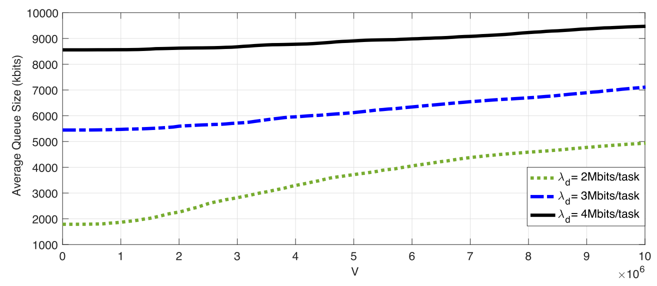

Fig. 3. Average queue size in kbits variation with respect to *V* for different data size λ_d.

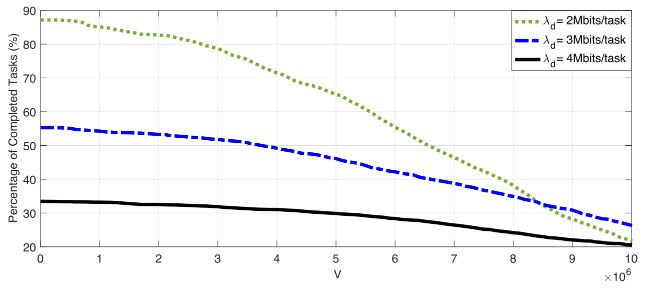

Fig. 4. Percentage of completed tasks variation with respect to *V* for different data size λ_d.

4) Communication Parameters: We assume the MTR communicates with the peer MT over Bluetooth, with the MEC over WiFi and the MCC over cellular links. The average transmission rates R^d, R^e and R^c are set to 4, 4, and 8 Mbps, with an average variation of 0.5 Mbps. We assume the power P^{dt}, P^{et} and P^{ct} consumed by the MTR to transmit over Bluetooth, WiFi and cellular networks to be 0.5, 0.5, and 0.6 Watts, and the receive power P^{dr} over Bluetooth to be 0.2 Watts [35]. We assume the transmission cost to be free over Bluetooth, 2.5685×10^{-10} and 1.3699×10^{-9} USD/bit over WiFi and cellular networks, respectively [36].

C. Simulations Results and Analysis

In this section, we first study the impact of the weight *V* and then evaluate the proposed approaches and compare the performance of the various strategies mentioned in Section VI-A under different computation demands and system load.

1) Study on the Weight V of the Energy Consumption and Monetary Cost: To study the tradeoffs between queue stability, energy consumption and monetary cost, we evaluate the performance of our proposed LCO approach for different values of *V* varying between 0 and 10^7. The values of *V* were chosen to reflect the significance of the normalized cost function $C_{\ell\ell'}[t]$ in terms of energy consumption and monetary cost while considering the expected amount of computation data to be processed $\mathbb{E}\{\mathcal{D}_{\ell\ell'}[t]|S[t]\}$ (kbits) weighted by the observed queue size *Q*[*t*] (kbits). Figures 3, 4 and 5 evaluate the average queue size, the percentage of completed tasks, the energy consumption and monetary cost per completed task, respectively, with different average task data size λ_d of 2, 3 and 4 Mbits. In Figure 6, we evaluate the percentage of effective computation

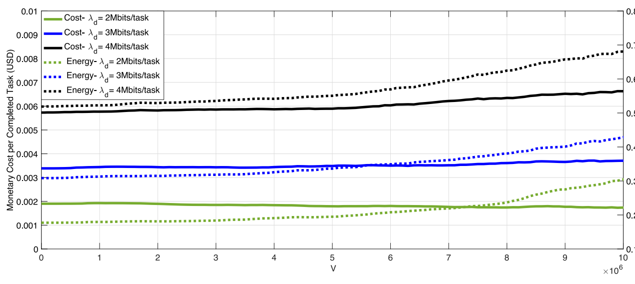

Fig. 5. Energy consumption and monetary cost per completed task variation with respect to V for different data size λ_d.

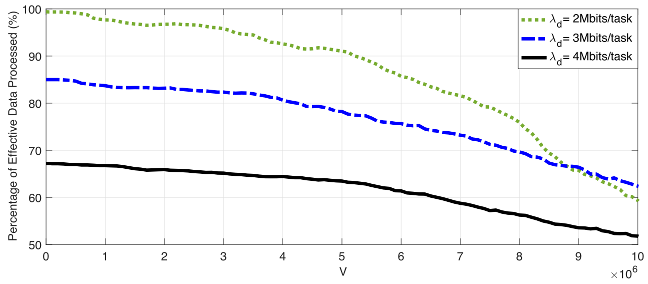

Fig. 6. Percentage of effective data processed variation with respect to V for different data size λ_d.

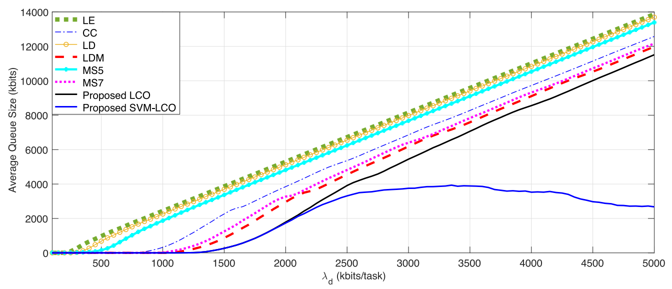

Fig. 7. Average queue size in kbits variation with respect to the average data size λ_d.

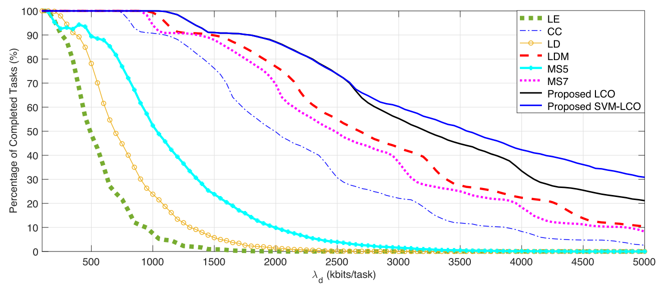

Fig. 8. Percentage of completed tasks variation with respect to the average data size λ_d.

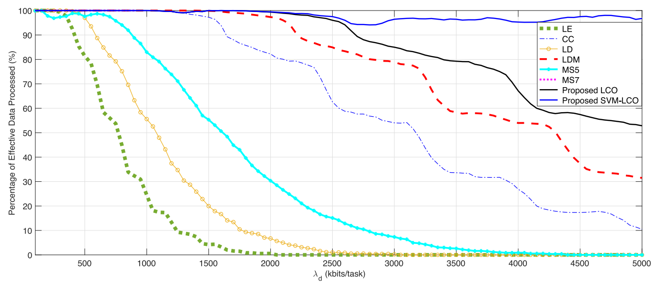

Fig. 9. Percentage of effective data processed variation with respect to V for different data size λ_d.

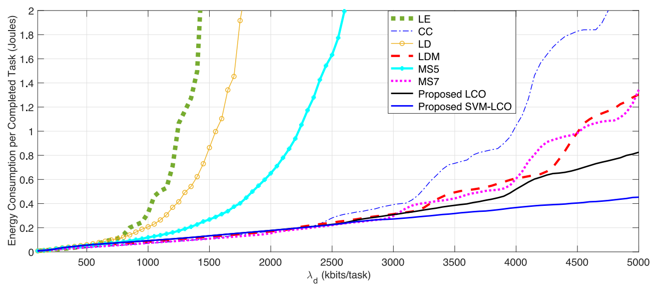

Fig. 10. Energy consumption per completed task variation with respect to the average data size λ_d.

data processed which represents the amount of computation data processed belonging only to tasks that are completed. In general, processed data can belong to tasks that can be either completed within their deadline or dropped. We consider the data processed of tasks ending up to be dropped as overhead while we aim to have more efficient processing of data of tasks with high potential to be completed. The results show that increasing V has high impact on decreasing the number of completed tasks while having low impact on reducing the energy consumption and monetary cost. This is due to the fact that the proposed approach tends to defer the offloading while considering higher values of V to save energy and cost which leads to a more congested queue and less percentage of completed tasks as presented in Figures 3 and 4. Consequently, the MTR will spend more monetary cost and energy to process data of tasks that will be eventually dropped and not counted towards the goal of maximizing the number of completed tasks, which decreases the efficiency of the data processed as shown in Figure 6 and causes an overhead in terms of energy consumption and monetary cost. This will also prevent more tasks to be completed within their deadline which decreases the number of completed tasks with the increase of V as presented in Figure 4, without achieving gains in terms of energy and monetary cost as presented in Figure 5 due to processing data of tasks ending up being dropped.

2) Performance Evaluation of the SVM-LCO Approach: In this section, we compare the performance of the proposed approaches with the different strategies and system models adopted in the state-of-the-art literature as presented in Section VI-A. We evaluate the average queue size, the percentage of completed tasks, the percentage of effective computation data processed, the energy consumption and monetary cost per completed task in Figures 7, 8, 9, 10 and 11, and respectively, while varying the average data size λ_d from 10 kbits to 5 Mbits. Based on the observed values in Section VI-C1, we adopt the weight $V = 10^5$ to capture the tradeoff between the considered performance metrics. Moreover, we further illustrate the performance enhancement provided by the SVM-LCO approach compared to the LCO approach in Figure 12 where we evaluate the percentage of completed and dropped tasks. We differentiate between two

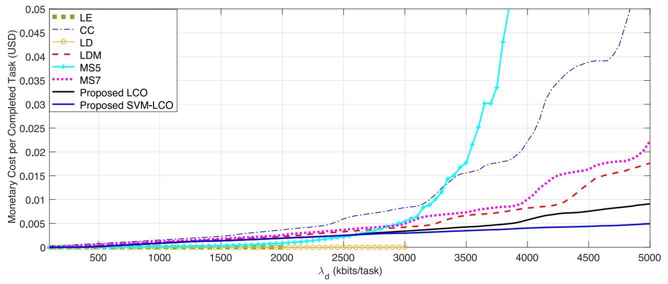

Fig. 11. Monetary cost per completed task variation with respect to the average data size λ_d.

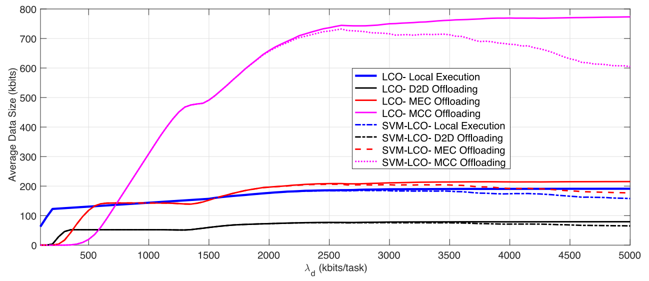

Fig. 13. Average computation data size executed locally or remotely using D2D, MEC and MCC offloading with respect to the average data size λ_d.

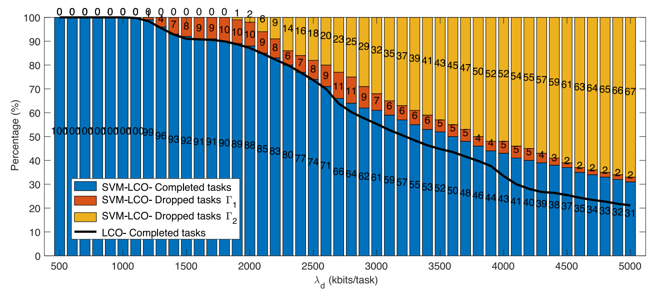

Fig. 12. Percentages of completed tasks and dropped tasks variation with respect to the average data size λ_d.

types of dropped tasks: Γ_1 the number of tasks admitted to the queue, however, they were dropped due to the lack of resources, and Γ_2 the number of tasks dropped by the SVM-based admission control.

The results show that complete local execution (LE) without using any computation task offloading is cost efficient, however, due to the limitation of the computation capabilities of the requester, the tasks with large data sizes may not be completed locally within their deadline. This drops the percentage of completed tasks from 100% to 0% when the average data size exceeds 2 Mbits and leads to a high energy consumption per task since the MTR keeps on processing tasks that will be eventually dropped. Complete MCC offloading (CC) was able to provide higher performance in terms of queue size and number of completed tasks due the high computation capacity of the cloud. However, the MTR will be using the cellular network to offload all its tasks' data to the cloud which is expensive in terms of energy consumption and monetary cost.

Using partial offloading and considering local execution with D2D offloading to a peer MT (LD) provided a higher number of completed tasks compared to local execution with higher monetary cost. The mode selection approaches (MS5 and MS7) while partitioning the data into maximum two parts, locally or remotely executed using D2D or MEC offloading provided higher number of completed tasks compared to LD while consuming more energy and monetary cost. Adding two modes to MS5 including complete and partial MCC offloading, allowed MS7 to provide higher number of completed tasks compared to MS5 and CC. Note that the energy consumption is higher when the D2D offloading is considered since we consider in our formulation the energy consumed by the MTR for transmission and by the peer MT for receiving and processing the computation data. Therefore, considering partitioning the task data into three parts to be executed locally and remotely at a peer MT and MCC, simultaneously, provided higher number of completed tasks and queue size with lower energy consumption and monetary cost per completed task.

Our proposed LCO approach adopts all the offloading modes previously presented in addition to the use of local execution, D2D, MEC and MCC offloading, simultaneously. The results show that the LCO outperforms all other conventional and proposed system models in the literature in terms of average queue size and percentage of completed tasks while reducing energy consumption and monetary cost. Even with very large data size of 5 Mbits, LCO was able to complete 21% tasks while consuming 0.8256 Joules and 0.0091 USD per completed task. Compared to LDM, LCO provides 10.66% more completed tasks, with a reduction of 58% and 93% in terms of energy consumption and monetary cost, respectively. Hence, our approach was able to smartly decide on the offloading modes that will efficiently use the computation resources and achieve performance gains. For instance, as presented in Figure 13, local execution was used without any computation offloading when the average task data size was low (e.g.,: $\lambda_d = 100$ kbits/task). However, when λ_d increases to 200 kbits/task, the computation demands become higher than the local device capabilities which requires simultaneous remote execution at a peer MT. Larger data sizes required the intervention of the edge server and cloud to be completed due to the limited computation capabilities of the MTs.

The SVM-based task admission was able to provide major enhancement in the performance of the proposed LCO approach. The proposed SVM-LCO increased the percentage of completed tasks from 21% to 31% with a high average data size of 5 Mbits as presented in Figures 8 and 12. A large number of tasks was dropped by the SVM-based task admission approach which reduces the queue size and increases the effective data processed. The number of dropped tasks by the proposed admission control scheme increases with the increase of the computation data size to provide higher efficiency for higher congested queues. Moreover, less than 12% of the tasks were admitted to the queue, and ended up being dropped, which increases the effective data processed and allows the MTR queue to accommodate for more tasks. The effective data processed was increased by 43.9% while maintaining a low average queue size with high average data size of 5 Mbits.

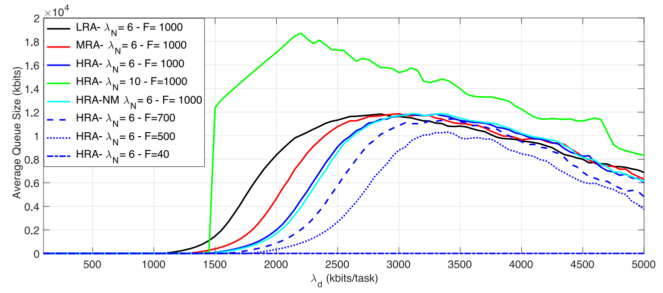

Fig. 14. Average queue size in kbits variation with respect to the average data size λ_d, number of tasks per second λ_N, computation intensity *F*, and number of nodes.

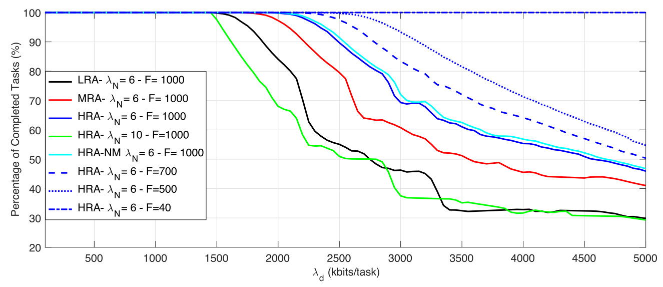

Fig. 15. Percentage of completed tasks variation with respect to the average data size λ_d, number of tasks per second λ_N, computation intensity *F*, and number of nodes.

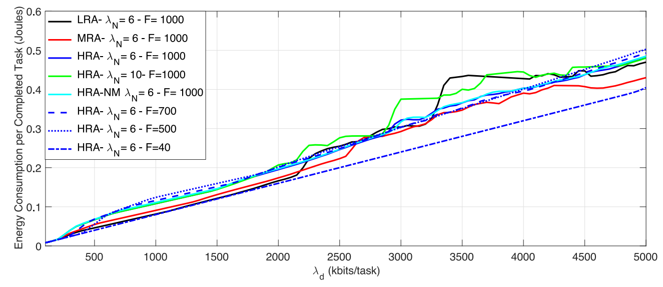

Fig. 16. Energy consumption per completed task variation with respect to the average data size λ_d, number of tasks per second λ_N, computation intensity *F*, and number of nodes.

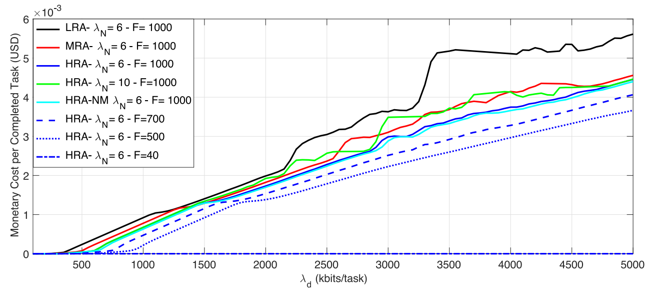

Fig. 17. Monetary cost per completed task variation with respect to the average data size λ_d, number of tasks per second λ_N, computation intensity *F*, and number of nodes.

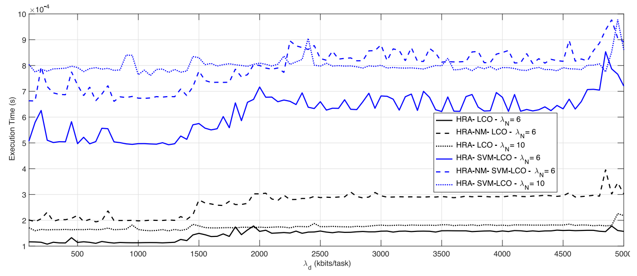

Fig. 18. Execution time per time slot with respect to the average data size λ_d, number of tasks per second λ_N, and number of nodes.

This reduces the energy consumption and monetary cost per task by more than 82% and 85%, respectively.

3) Performance Evaluation Considering Different System Parameters: We evaluate the performance of the SVM-LCO approach while varying the computation demands in terms of the average number of tasks/s λ_N, the task data size λ_d, and the computation intensity *F*, as well as, the load on multiple remote nodes in Figures 14, 15, 16, 17 and 18. We adopt three different load conditions on the remote nodes reflected by the availability of their computation resources as follows: (1) high resource availability (HRA) assuming the peer MT, edge server and cloud resources are fully available (i.e.,: $\widehat{\mu}^d = \widehat{\mu}^e = \widehat{\mu}^c = 100\%$), (2) medium resource availability (MRA) assuming $\widehat{\mu}^d = \widehat{\mu}^e = 50\%$, and $\widehat{\mu}^c = 10\%$, and (3) low resource availability (LRA) assuming $\widehat{\mu}^d = \widehat{\mu}^e = 20\%$, and $\widehat{\mu}^c = 5\%$ due to a highly congested network. We assume the task deadline is 1 s. We also target different data partitioned oriented application profiles with different computation intensity *F* of 40, 500, 700 and 1000 CPU cycles/bit.

The results show that for the same computation demands, as the load on the remote nodes increases and their resource availability decreases, the average queue size increases while decreasing the number of completed tasks and increasing the energy consumption and monetary cost. Increasing the number of tasks λ_N from 6 to 10 tasks per second increases the load on the system leading to a higher average queue size, a lower percentage of completed tasks with higher energy consumption and monetary cost. Decreasing the computation intensity demands *F* reduced the average queue size, the energy consumption and monetary cost and increased the percentage of completed tasks. For low computation demands of 40 CPU cycles/bit, the MTR was able to rely mainly on its local computation resources which leads to an empty queue and the completion of the tasks with minimum energy and monetary cost. Moreover, we have evaluated the performance of our proposed approach while considering the existence of $N = 20$ peer MTs, $M = 5$ edge servers and the cloud with high resource availability (HRA-NM). The existence of multiple cooperating nodes willing to assist the MTR in the computation processing provides slightly higher performance in terms of percentage of completed tasks with lower monetary cost, energy consumption and average queue size. This is due to the fact that at every time slot, the MTR will select the peer MT and the edge server providing the highest general utility function before proceeding with the D2D and MEC offloading.

To complement the theoretical complexity analysis presented in Section IV-E, we evaluate the average execution time per time slot in Figure 18 which increased from 1.4514×10^{-4} to 2.6677×10^{-4} s (on a 2.4 GHz Quad-Core Intel Core i5) when the number of peer MTs N increased from 1 to 20, and the number of edge servers M from 1 to 5 using the LCO approach. Similarly, it increases from 6.1647×10^{-4} to 7.9030×10^{-4} s using the SVM-LCO approach. When λ_N increases from 6 to 10 tasks per second, the execution time increases to 1.7543×10^{-4} and 7.9867×10^{-4} s for LCO and SVM-LCO, respectively. This is due to the fact that generating on average 10 tasks per second forces the proposed approach to process a much larger number of tasks and almost at every time slot, a new task is generated, and the proposed approach will predict using the SVM model whether the task should be admitted. Therefore, the simulation results showed that our proposed approaches can provide real-time offloading decisions in less than 0.8% of the time slot duration $T_s = 0.1$ s.

D. Simulations Results Outcome

The proposed approaches provide user-centric real-time computation offloading and resource allocation decisions aiming at stabilizing requester queue while minimizing the energy and monetary cost without causing additional overhead on the system. Adopting different offloading modes including local execution, partial and complete D2D, MEC and MCC offloading, simultaneously, allowed the LCO to outperform other existing system models presented in the literature. Moreover, integrating the SVM-based task admission into the LCO approach had high impact on the stability of the MTR queue and the effective data processed, and hence, reduced further the energy and monetary cost even with high computation data demands. The proposed approaches showed high efficiency under different computation demands and network load while using low-latency data partitioned oriented computation-intensive applications with very low time complexity.

VII. Conclusion

This paper provided real-time solutions for task admission, computation offloading and resource allocation aiming at stabilizing the requester queue, maximizing the number of completed tasks while minimizing energy consumption and monetary cost in D2D-enabled heterogeneous MEC network. The solution is based on a Lyapunov drift-plus-penalty formulation while adopting partial offloading where a requester offloads different parts of its computation task data simultaneously to a peer mobile terminal, mobile edge server and cloud. The admission control was based on support vector machines to evaluate the availability of resources and queue congestion and decide to admit or reject a task. The proposed approaches were evaluated under different system models presented in the literature, computation demands and network loads. The results show that the using SVM-based task admission control provided major enhancement to our proposed Lyapunov-based computation offloading approach over conventional approach and existing system models presented in the literature. As future work, the proposed approaches can be extended to consider communication resource allocation, dynamic weights for the penalty function and accommodate for different applications with heterogeneous computation demands.

References

[1] *Internet of Things Forecast: Mobility Report*, Ericsson, Stockholm, Sweden, Jun. 2019.

[2] M. Lytras, W. Al-Halabi, J. X. Zhang, R. Haraty, and M. Masud, "Enabling technologies and business infrastructures for next generation social media: Big data, cloud computing, Internet of Things and virtual reality," *J. Univ. Comput. Sci.*, vol. 21, no. 11, pp. 1379–1384, 2015.

[3] D. Wang, H. Qin, B. Song, X. Du, and M. Guizani, "Resource allocation in information-centric wireless networking with D2D-enabled MEC: A deep reinforcement learning approach," *IEEE Access*, vol. 7, pp. 114935–114944, 2019.

[4] G. Li and J. Cai, "An Online incentive mechanism for collaborative task offloading in mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 624–636, Jan. 2020.

[5] A. Mourad, H. Tout, O. A. Wahab, H. Otrok, and T. Dbouk, "*Ad hoc* vehicular fog enabling cooperative low-latency intrusion detection," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 829–843, Jan. 2021.

[6] T. Dbouk, A. Mourad, H. Otrok, H. Tout, and C. Talhi, "A novel ad-hoc mobile edge cloud offering security services through intelligent resource-aware offloading," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1665–1680, Dec. 2019.

[7] R. Chai, J. Lin, M. Chen, and Q. Chen, "Task execution cost minimization-based joint computation offloading and resource allocation for cellular D2D MEC systems," *IEEE Syst. J.*, vol. 13, no. 4, pp. 4110–4121, Dec. 2019.

[8] Y. He, J. Ren, G. Yu, and Y. Cai, "Joint computation offloading and resource allocation in D2D enabled MEC networks," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–6.

[9] Y. He, J. Ren, G. Yu, and Y. Cai, "D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 3, pp. 1750–1763, Mar. 2019.

[10] G. Baranwal and D. Vidyarthi, "Admission control policies in fog computing using extensive form game," *IEEE Trans. Cloud Comput.*, early access, Jun. 9, 2020, doi: 10.1109/TCC.2020.3000800.

[11] S. Li *et al.*, "Joint admission control and resource allocation in edge computing for Internet of Things," *IEEE Netw.*, vol. 32, no. 1, pp. 72–79, Jan./Feb. 2018.

[12] S. Pan and Y. Chen, "Energy-optimal scheduling of mobile cloud computing based on a modified Lyapunov optimization method," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 1, pp. 227–235, Mar. 2019.

[13] R. Li, Z. Zhou, X. Chen, and Q. Ling, "Resource price-aware offloading for edge-cloud collaboration: A two-timescale Online control approach," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 648–661, Jan.-Mar. 2022.

[14] J. Wang *et al.*, "Energy-efficient admission of delay-sensitive tasks for multi-mobile edge computing servers," in *Proc. 25th Int. Conf. Parallel Distrib. Syst.*, 2019, pp. 747–753.

[15] H. Chen, F. Chen, and Y. Liu, "Admission control based distributed multiuser computation offloading for edge computing," in *Proc. IEEE Globecom Workshops*, 2019, pp. 1–6.

[16] Z. Zhao *et al.*, "On the design of computation offloading in fog radio access networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 7, pp. 7136–7149, Jul. 2019.

[17] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 668–682, Mar. 2019.

[18] N. Kherraf, S. Sharafeddine, C. M. Assi, and A. Ghrayeb, "Latency and reliability-aware workload assignment in IoT networks with mobile edge clouds," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1435–1449, Dec. 2019.

[19] N. Kherraf, H. A. Alameddine, S. Sharafeddine, C. M. Assi, and A. Ghrayeb, "Optimized provisioning of edge computing resources with heterogeneous workload in IoT networks," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 2, pp. 459–474, Jun. 2019.

[20] L. Huang, X. Feng, L. Zhang, L. Qian, and Y. Wu, "Multi-server multi-user multi-task computation offloading for mobile edge computing networks," *Sensors*, vol. 19, no. 6, p. 1446, 2019.

[21] H. Wang, Z. Peng, and Y. Pei, "Offloading schemes in mobile edge computing with an assisted mechanism," *IEEE Access*, vol. 8, pp. 50721–50732, 2020.

[22] Y. Lan, X. Wang, D. Wang, Y. Zhang, and W. Wang, "Mobile-edge computation offloading and resource allocation in heterogeneous wireless networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2019, pp. 1–6.

[23] Y. Sun, T. Wei, H. Li, Y. Zhang, and W. Wu, "Energy-efficient multimedia task assignment and computing offloading for mobile edge computing networks," *IEEE Access*, vol. 8, pp. 36702–36713, 2020.

[24] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, and F. Tian, "Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3282–3299, Apr. 2020.

[25] O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Optimization of Radio and Computational Resources for Energy Efficiency in Latency-Constrained Application Offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[26] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency minimization for D2D-enabled partial computation offloading in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4472–4486, Apr. 2020.

[27] M. J. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. Williston, VT, USA: Morgan and Claypool, 2010.

[28] W. Fawaz, I. Ouaiss, K. Chen, and H. Perros, "Deadline-based connection setup in wavelength-routed WDM networks," *Comput. Netw.*, vol. 54, pp. 1792–1804, Feb. 2010.

[29] M. Assi and R. A. Haraty, "A survey of the Knapsack problem," in *Proc. Int. Arab Conf. Inf. Technol.*, 2018, pp. 1–6.

[30] M. K. J. Han and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Burlington, MA, USA: Morgan Kaufmann, 2012.

[31] J. Nalepa and M. Kawulok, "Selecting training sets for support vector machines: A review," *Springer Artif. Intell. Rev.*, vol. 52, pp. 857–900, Aug. 2019, doi: 10.1007/s10462-017-9611-1.

[32] C. You and K. Huang,, "Exploiting non-causal CPU-state information for energy-efficient mobile cooperative computing," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4104–4117, Jun. 2018.

[33] S. Pan, Z. Zhang, Z. Zhang, and D. Zeng, "Dependency-aware computation offloading in mobile edge computing: A reinforcement learning approach," *IEEE Trans. Cloud Comput.*, vol. 7, pp. 134742–134753, Sep. 2019, doi: 10.1109/ACCESS.2019.2942052.

[34] "Cloudorado-Cloud Computing Comparison Engine: Cloud Server Comparison–Price & Features." 2020. [Online]. Available: https://www.cloudorado.com/cloud_server_comparison.jsp

[35] K. Jahed, M. Younes, and S. Sharafeddine, "Energy measurements for mobile cooperative video streaming," in *Proc. IFIP Wireless Days*, 2012, pp. 1–3.

[36] N. Abbas, S. Sharafeddine, H. Hajj, and Z. Dawy, "Price-aware traffic splitting in D2D HetNets with cost-energy-QoE tradeoffs," *Comput. Netw.*, vol. 172, May 2020, Art. no. 107169.

Nadine Abbas (Member, IEEE) received the Bachelor of Engineering degree (Highest Distinction) in computer and communication engineering from Notre Dame University in 2008, and the M.E. and Ph.D. degrees in electrical and computer engineering from American University of Beirut in 2011 and 2017, respectively. She is a Visiting Assistant Professor with the Computer Science Department, Lebanese American University. Her research interests include mobile edge computing and wireless communications, mainly heterogeneous networks with intelligent multiradio networking.

Wissam Fawaz (Senior Member, IEEE) received the Ph.D. degree in network and information technology from the University of Paris XIII in 2005. He is a Professor with the Department of Electrical and Computer Engineering, Lebanese American University, Lebanon. His research interests are in the areas of delay tolerant as well as quality of service enabled vehicular ad-hoc networks and buffer-aided free space optical communication systems. He received a Fulbright Research Award in 2008. He is currently serves as an Associate Editor for the *Journal of Annals of Telecommunications* (Springer). He served as an Associate Editor for the IEEE COMMUNICATIONS LETTERS from 2013 until 2017.

Sanaa Sharafeddine (Senior Member, IEEE) received the B.E. and M.E. degrees in computer and communications engineering from the American University of Beirut in 1999 and 2001, respectively, and the Doctoral degree in communications engineering from Munich University of Technology (TUM) in 2005 in collaboration with Siemens AG Research Labs in Munich. She is a Professor of Computer Science with Lebanese American University. Her research interests are in the general area of wireless networks with focus on UAV-aided communications, edge computing, device-to-device cooperation, and multimedia services. She received the International Rising Talent Award in 2015, L'Oreal-UNESCO Pan-Arab Regional Fellowship Award in 2013. She joined the editorial boards of IEEE NETWORKING LETTERS, *Ad Hoc Networks* (Elsevier), IEEE ACCESS, and IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS–Series on Network Softwarization and Enablers.

Azzam Mourad (Senior Member, IEEE) received the M.Sc. degree in CS from Laval University, Canada, in 2003, and the Ph.D. degree in ECE from Concordia University, Canada, in 2008. He is currently a Professor of Computer Science with Lebanese American University, a Visiting Professor of Computer Science with New York University, Abu Dhabi, and an Affiliate Professor with the Software Engineering and IT Department, École de Technologie Supérieure (ETS), Montreal, Canada. He published more than 100 papers in international journal and conferences on security, network and service optimization and management targeting IoT, cloud/fog/edge computing, vehicular and mobile networks, and federated learning. He has served/serves as an Associate Editor for IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE NETWORK, IEEE OPEN JOURNAL OF THE COMPUTER SOCIETY, *IET Quantum Communication*, and IEEE COMMUNICATIONS LETTERS, the General Chair of IWCMC2020, the General Co-Chair of WiMob2016, and the track chair, a TPC member, and a reviewer for several prestigious journals and conferences.

Chadi Abou-Rjeily (Senior Member, IEEE) received the B.E. degree in electrical engineering from Lebanese University, Roumieh, Lebanon, in 2002, and the M.S. and Ph.D. degrees in electrical engineering from the École Nationale Supérieure des Télécommunications, Paris, France, in 2003 and 2006, respectively. He is a Professor with the Department of Electrical and Computer Engineering, Lebanese American University, Byblos, Lebanon. From September 2003 to February 2007, he was also a Research Fellow with the Laboratory of Electronics and Information Technology of the French Atomic Energy Commission (CEA-LETI). His research interests are in the code construction and transceiver design for wireless communication systems.