Singh, R., Armour, S., Khan, A., Sooriyabandara, M., & Oikonomou, G. (2020). Heuristic Approaches for Computational Offloading in Multi-Access Edge Computing Networks. In *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications* (Vol. 31). IEEE Computer Society. https://doi.org/10.1109/PIMRC48278.2020.9217181

Heuristic Approaches for Computational Offloading in Multi-Access Edge Computing Networks

Raghubir Singh†, Student Member, IEEE, Simon Armour†,
Aftab Khan‡, Mahesh Sooriyabandara‡, and George Oikonomou†
Communication Systems & Networks Research Group, University of Bristol, Bristol, UK†
Department of Electrical and Electronic Engineering, University of Bristol, UK†
Telecommunications Research Laboratory, Toshiba Research Europe Limited, Bristol, UK‡
E-mail: raghubir.singh@bristol.ac.uk

***Abstract*—Computational offloading is a strategy by which mobile device (MD) users can access the superior processing power of a Multi-Access Edge Computing (MEC) server network. In this paper, we contribute a model of a system that consists of multiple MEC servers and multiple MD users. Each MD has multiple computational tasks to perform, and each task can either be computed locally on the MD, or it can be offloaded to one of the MEC servers. For this system and having global knowledge, we compute the theoretical optimal allocation that minimises the time required to complete the computation of all tasks. Subsequently, we contribute a distributed heuristic algorithm that allows each MD to independently, and using local knowledge only, decide how to handle each individual job. Furthermore, we propose three approaches to decide whether to offload each individual job, and three mechanisms to determine which MEC server each task should be offloaded to. We use simulations to evaluate those approaches in terms of how well they can approximate the theoretical optimum. The proposed heuristic algorithm is tested on a range of experiments, and the results demonstrate that the heuristic algorithm can produce reasonable quality solutions.**

***Keywords*—Multi-Access Edge Computing, computation offloading, heuristic algorithm, theoretical optimal.**

I. Introduction

As IT developed after 2000, the reliance on mobile devices (MDs) are increased as they can provide convenient services such as the ability to communicate with people almost anytime anywhere, greater access to modern apps and services and the ability to accept payments wirelessly. However, due to limitations of MDs in terms of battery life, limited computing power and requirement for executing complex application is continuously increasing. Therefore, recent research advances in computational sciences are looking into ways to optimize the mobile users' Quality of Experience. Computational offloading is a problem-solving technique that addresses the limitation of available resources in MDs. The key advantages of using computational offloading are efficient power management, fewer storage requirements and enhanced user experience [1].

Several computational offloading approaches exist in the literature, and some of these have been demonstrated in real-life applications. One of the well-known strategies is Cloud Computing [2]. This approach has its merits and limitations. For example, a fundamental flaw of the Cloud paradigm is the remote location, which resulted in excessive latency and low bandwidth issues. As a result, latency-sensitive mobile applications, such as augmented reality, facial recognition, interactive gaming and natural language processing, so on offload to the far-off cloud may not be the ideal solution [3].

Recent approaches to computational offloading are focusing on bringing computational services closer to the mobile user in the form of MEC. MEC networks offer improved solutions to real-time and delay-sensitive mobile applications as the proximity of cellular networks [4]. A key challenge in making use of the MEC paradigm is determining an optimal schedule of offloading the computational jobs to MEC servers. The schedule needs to strike a balance between available MD resources, MEC computational capabilities as well as the capacity of the network. Published work on offloading between MDs and Edge Computing or Mobile Cloud Computing servers has not fully considered optimal scheduling of multiple jobs from a single MD or multiple MDs offloaded or processed locally [5]–[17]. Previous work from the authors on MEC computation offloading focused on how the advantages of offloading from MDs were determined by investigation of relationships between several important parameters: the computing power of MDs and MECs, communication links, size and complexity of jobs to be offloaded [18].

In this paper, we propose a computational offloading algorithm wherein we seek to minimise the completion time of all jobs in a multi-user multi-MEC set up. The contributions of this paper are twofold:

- A heuristic algorithm is proposed, which is suitable for distributed deployment at the MDs, uses local knowledge to handle each individual job and is able to approach a good quality solution.
- Different approaches to determining whether to of-

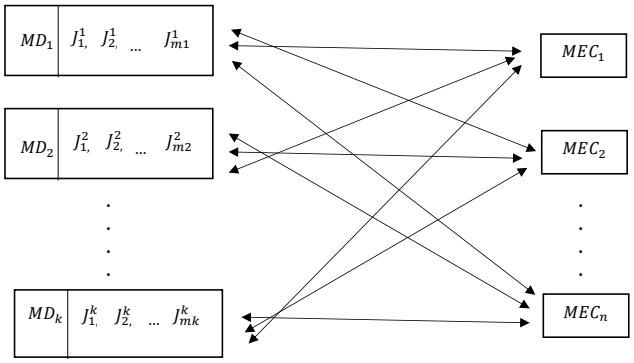

Figure 1. The model of computation offloading in a MEC network.

fload each job and selection mechanisms for the MEC server to which a job is offloaded are proposed and evaluated.

The remainder of the paper is organised as follows. Section II presents the problem formulation that has been developed to minimizing the global completion time for multiple offloaded tasks. Section III shows the outline scheme for the heuristic algorithm for computation offloading and all the different approaches that were developed with three strategies to allocating offloading probabilities. Section IV shows numerical results simulations on job completion times with single or multiple MDs offloading to multiple MEC servers. Section V discusses the performance of the algorithms. Finally, Section VI concludes the work and explores possible further developments of MEC networks offloading computational tasks from MDs.

II. Problem formulation

This section presents a mathematical model for computational offloading. A typical multi-user MEC network with a set of MDs, each with a given number of job requests, is shown in Fig. 1. The connections between mobile devices and the MEC servers represent the transmission channels that are used for communicating requests. The symbols used in Sections II and III are defined in Table I. Let MD denote the set of mobile devices. Let J_i^k denote the i^{th} job on the mobile device k, respectively. Also, let MEC represent the set of MEC devices available for the data requests from the given set of mobile users. Note that every MD is independent of the other and computation tasks offloaded on the MEC servers are processed on first in, first out basis.

Let $u_{j,c}$ be the binary variable that models the offloading of a job j on MEC c, respectively. The binary variable is defined as follows:

$$u_{j,c} = \begin{cases} 1 & \text{if job } j \text{ is offloaded to } c, \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Table I. Notations used in the paper.

Symbol	Definition
$i \in M$	$\{i = 1, \ldots, k\}$ set of mobile devices
$j \in J_s^i$	$\{s = 1, \ldots, m_i\}$ set of jobs on mobile device s
$c \in C$	$\{c = 1, \ldots, n\}$ set of MECs
m_i	Number of jobs submitted by MD i (bits)
X_j^i	Data size of job j on MD i in bits
X_j^c	Data size of job j on MEC c in bits
λ_j^i	An application on the MD i in bits/instructions
λ_j^c	An application on the MEC c in bits/instructions
α_i	The computing capability of MD i in instructions/sec
β_c	The computing capability of MEC c in instructions/sec
T_i^{MD}	Total computational time to execute job j on MD i in sec
$T_{i,c,j}$	Transmission time of offloading job j from MD i to MEC c in sec
T_c^{MEC}	Total computational time to execute jobs on MEC c in sec
π	Proportion of data size reduction after processing on MEC c
$T_{c,i,j}$	Receiving time of offloading job j from MEC c to MD i in sec
T_c^{Total}	Total offloading time of job j in sec
$\Gamma_{i,c}$	The link speeds between MD i and MEC c in bits/second
L_{c}	Loading on MEC c in bits
T_{c}	Computational time on MEC c in sec
P_{j}^i	Probability of job j on MD i
O^{DD}	Offloading decision based on data size
O^{DFP}	Offloading decision based on fixed probability
O^{DPD}	Offloading decision based on probability distribution
M^{R}	Random allocation of MECs
M^{J}	MECs allocation based on job size
M^{T}	MECs allocation based on computational time

where $J_i \cap J_{i'} = \phi \ \forall \ \ i, \quad i' \in M \quad i \ \neq \ i'$.

From Eqs. 2a, 2b, 2c, 2d, we can get the absolute values for all the sets J , M, C and $\Gamma_{i,c}$ for the MD i and the MEC c are denoted as follows:

$$m = \mid J \mid = \sum_{i=1}^{k} m_i \tag{2a}$$

$$\mid M \mid = k \tag{2b}$$

$$\mid C \mid = n \tag{2c}$$

$$\mid L \mid = 2(i \times c) \tag{2d}$$

Let X_j^i denote the computational data (in bits) as the size of input data that needs to be processed from an application that is running on MD i at λ_j^i (bits per instruction). In the following, we provide equations for calculating the local and remote computation time of a job.

1) Computational processing time on a mobile device: Let α_i be the on-board processor speed of MD i in instructions per second (IPS). The time to compute the job on MD i is given as follows:

$$T_i^{\text{MD}} = \frac{\sum_{j \in Ji} X_j^i (1 - \sum_{c \in C} u_{j,c})}{\alpha_i \lambda_j^i} \tag{3}$$

2) Computational processing time on a MEC: Let X_j^c denote the computational data (in bits) as the size of input data that needs to be processed from an application that is running on MEC c at λ_j^c (bits per instruction). Let β_c be the on-board processor speed of MEC c in instructions per second (IPS). The computational time to process a job on MEC c server is given as follows:

$$T_c^{\text{MEC}} = \frac{\sum_{j \in J} X_j^c U_{j,c}}{\beta_c \lambda_j^c} \tag{4}$$

Let γ^{UL} be the up-link speed in bits/second. The following equation gives the time to send the job to the MEC server:

$$T_{i,c,j} = \frac{\sum_{j \in J_i} U_{j,c} X_j^i}{\gamma^{\text{UL}}} \tag{5}$$

Let π be the proportion of data size reduction after processing on MEC c. Let γ^{DL} be the downlink speed in bits/second. The links between MD i and MEC c are symmetric, which means device i can send and receive data to and from MEC c at the same rate. The receiving time of the processed data can be calculated as follows:

$$T_{c,i,j} = \frac{\sum_{j \in J_i} \pi X_j^c}{\gamma^{\text{DL}}} \tag{6}$$

$$\gamma^{\text{UL}} = \gamma^{\text{DL}} = \Gamma_{i,c} \tag{7}$$

Eqs. 4, 5 and 6 can be represented as:

$$T_c^{\text{Total}} = \frac{\sum_{j \in J} X_j^c U_{j,c}}{\beta_c \lambda_j^c} + \frac{\sum_{j \in J_i} U_{j,c} X_j^c}{\gamma^{\text{UL}}} + \frac{\sum_{j \in J_i} \pi X_j^c}{\gamma^{\text{DL}}} \tag{8}$$

A binary variable should respect the following constraint:

$$\sum_{j \in J_i} U_{j,c} \leq 1 \ \ \forall \ i \in M, c \in C \tag{9}$$

When the job is processed locally, the right-hand side of the equation is equal to zero. When the task is offloaded on a server, the constraint ensures that it is not offloaded on more than one MEC server.

III. Heuristic algorithm for computation offloading

This section presents a heuristic algorithm that is aimed at finding a near-optimal scheduling solution for a given number of job requests from MDs to a given set of MECs. It is assumed that the MDs communicate key statistics for their requests, including the number of jobs, job sizes and their computing power. The scheduling algorithm then proposes a solution for solving these jobs while minimizing the overall computational time.

Fig. 2 presents a flowchart of the proposed heuristic method. J represents the set of all jobs. Following are the two key decision-making steps in the algorithm:

- Whether to offload a job to a MEC or compute locally on the MD?
- If the decision from the above step is to offload, then determine each job to offload on which MEC.

In this paper, a total of nine policies (3 each for the above two decision questions) are tested. These versions differed for the allocation of an offloading probability for an individual job on each MD as shown by ("$j \ \leq \mid J \mid$") in the flowchart) and the allocation of a MEC server in the network to process an individual job as (indicated by "Determine the MEC to offload" in the flowchart). The detail regarding these policies is provided in the following subsections.

A. The offloading decision

Three different strategies were adopted for allocating the offloading probability of individual jobs on a MD.

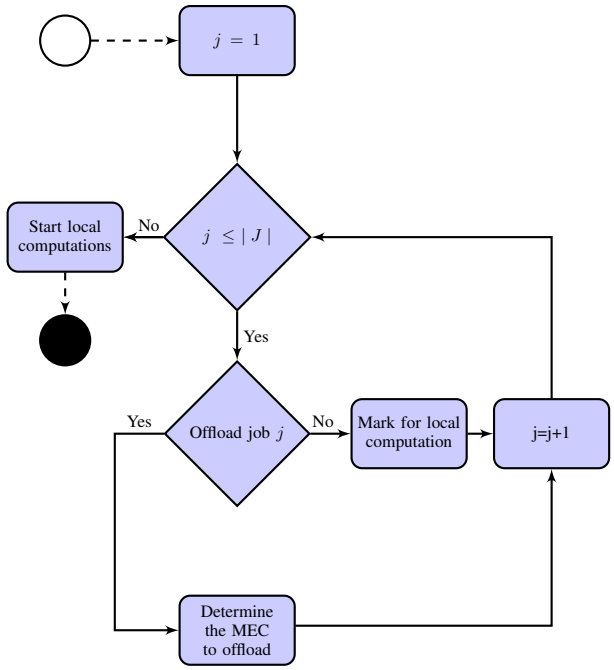

Figure 2. Flowchart of the heuristic algorithm for computation offloading.

1) Probability calculations for job offloading based on data file size ("O^{DD}"): The algorithm follows the formula of job probability. The algorithm uses the following method to determine the offloading probability of individual jobs:

$$P_j^i = \frac{X_j^i}{\sum_{j \in J_i} X_j^i} \tag{10}$$

where J_i is a set of all jobs on MD i and X_j^i is the data size of job j. The motivation here was to bias the algorithm to preferentially offload the larger job sizes, which would be most advantageous to reduce completion time on the faster server processors.

2) Offloading based on a fixed probability ("O^{DFP}"): Each job has the same probability of being offloaded regardless of data size; in our experiments, we used a fixed probability of 0.5. The motivation here was to test how giving each job the same possibility would impact on the total tasks offloaded by the algorithm.

3) Offloading based on a known probability distribution ("O^{DPD}"): In this policy, the probability of offloading is kept fixed for each job on the individually MD. The probability may be obtained by pre-solving job allocation as a local sub-problem on an or each MD. A further weighting added that includes the number of MDs and MECs a system to influence the probability. The main aim of this policy is that a mobile device may have a preference for offloading a certain number of jobs to the MEC server. The resulting probability can be derived as follows:

$$P_j^i = \frac{O_i^J}{T_i^J} \times \frac{T^{MEC}}{T^{MDs}} \tag{11}$$

where O_i^J is the total number of jobs offloaded, T_i^J is the total number of jobs on the MD i, respectively. T^{MEC} denotes the total number of MECs and T^{MDs} is the total number of MDs.

In the case when the number of MDs is less than the number of MEC servers, the formula in Equation (11) is simplified as follows:

$$P_j^i = \frac{O_i^J}{T_i^J} \tag{12}$$

B. MEC allocation of offloaded jobs

1) Random allocation of MEC ("M^R"): After a job offload decision is made, the job is allocated to a random MEC server $\{c = 1, \ldots, n\}$ in the network without knowledge of how busy the MEC is.

2) Offloading based on the job size ("M^J"): In this offloading policy, the jobs are assigned to MECs based on their current loading. This policy ensures that the next offloading job is assigned to the MEC that has the least loading. Mathematically, the offloading is achieved using the following equation:

$$MEC^{\mathrm{L,Offload}} = \min\{L_1, L_2, \ldots, L_{\mathrm{n}}\} \tag{13}$$

where L_c is the loading on MEC c, and is defined as follows:

$$L_c = \sum_{j \in J} X_{j,c} U_{j,c}$$

3) Minimum MEC computational time ("M^T"): After a job offload decision is made, the job is allocated to the MEC server which is calculated to complete job processing quickest. This policy requires knowledge of a MEC processing capability to calculate the minimum computational time.

$$MEC^{\mathrm{T,Offload}} = \min\{T_1, T_2, \ldots, T_{\mathrm{n}}\} \tag{14}$$

where T_c is the computational time on MEC c, and is defined as follows:

$$T_c = X_j^c \times (\frac{1}{\beta_c \lambda_j^c})$$

C. Computation of the benchmark theoretical optimum

To validate and to provide a comparison benchmark for results generated by the heuristic algorithms, linear programming optimization was performed using CPLEX[1]. Using this

[1]IBM CPLEX Linear programming problem solver: https://www.ibm.com/pt-en/products/ilog-cplex-optimization-studio

Table II. PARAMETERS SELECTED FOR THE SIMULATIONS IN (CASE 1,2 & 3)

Entity	Parameter	Value	Unit
Jobs	X_j^i	2-9 (1), 1-7 (2), 1-6 (3)	MB
MDs	α_i	3.60×10^9	IPS
MEC1	β_1	1.40×10^{11}	IPS
MEC2	β_2	1.40×10^{11} (1), 3.68×10^{10} (2 & 3)	IPS
MEC3	β_3	1.40×10^{11} (2) 3.68×10^{10} (1 & 3)	IPS
Network	MD - MEC 1	15	Mbps
	MD - MEC 2	28	Mbps
	MD - MEC 3	25	Mbps

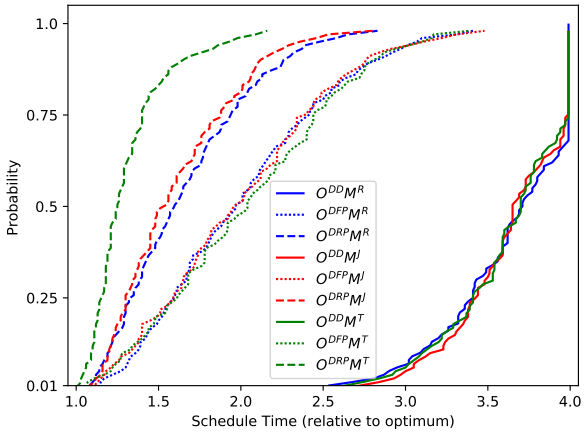

Figure 3. Cumulative Distribution Function graphs for heuristic algorithm outputs of schedule times for Case 1 (1 MD, 3 MEC servers, 20 jobs).

mathematical model, we compute the theoretical optimum job allocation. Sub-optimal job allocations generated by the heuristic algorithms were then compared to this theoretical optimal allocation to assess the performance of the heuristic under different combinations of MDs and MEC servers.

IV. NUMERICAL RESULTS

In this section, we investigate the performance of the heuristics under systems consisting of various numbers of MDs, jobs and MECs. This explored the scalability of the algorithms and how the methods for assigning the probabilities of offloading individual jobs varied with increasing job numbers and total data sizes. Using the estimates of bits per instruction for nine scientific applications quoted in [19] a value of 2.27×10^{-3} instructions per bit was chosen for the simulations. Numerical simulations were performed in this study to explore the performance of different algorithms before direct testbed experimentation is performed, as has been demonstrated for facial recognition and mobile gaming [20], [21]. The model presented in Section III assumes low memory and CPU usages because job sizes are small and job numbers per MD are low.

A. *Offloading from a single MD (Case 1)*

The parameters for this simulation are included in Table II; the processor speeds were taken from [19]. In the simulations, fixed link speeds are assumed to be valid for the short times in which the linear optimization or heuristic algorithms gather information and identify optimal or sub-optimal schedules for immediate offloading. Fig. 3 shows combined results from 20 jobs on the single MD. The algorithm was assumed to be run on the MD or (on a one-to-one basis) server-side.

Offloading based on a fixed probability (O^{DFP}) and offloading based on known probability distribution (O^{DPD}) with three different mechanisms for server allocation (random (M^{R}), job size (M^{J}) or minimum remaining computational time (M^{T}) performed far better than the calculating the probability for job offloading based on data file size (O^{DD}). The best version of the algorithm with offloading based on probability distribution with minimum remaining MEC computation time reached least-time schedules within 1% of the optimum time as identified by linear programming.

B. *Offloading from multiple MDs (Case 2)*

The parameters for this network simulation are included in Table II. In this simulation, 10 MDs attempted to offload a total of 72 jobs to 3 MEC servers. Fig. 4 shows that the three different approaches in the algorithms yielded different best schedule times but that offloading based on known probability distribution (O^{DPD}) with minimum remaining MEC computation time (M^{T}) could reach a schedule time only 13% greater than the optimum value as identified by linear programming. The variation in performance between the nine versions of the algorithm was less pronounced than in the scenario of case 1 and all schedules were within 55% longer times than the optimum as deduced by linear programming.

C. *Offloading from multiple MDs (Case 3)*

The parameters for this larger network simulation are included in Table II. In this simulation, 13 MDs attempted to offload a total of 115 jobs to 3 MEC servers. Fig. 5 shows that calculating the probability for job offloading based on data file size (O^{DD}) was much inferior to offloading based on a fixed probability (O^{DFP}) and offloading based on known probability distribution (O^{DPD}). Offloading based on known probability distribution (O^{DPD}) with minimum remaining MEC computation time (M^{T}) could reach a schedule time within 20% of the optimum value as identified by linear programming.

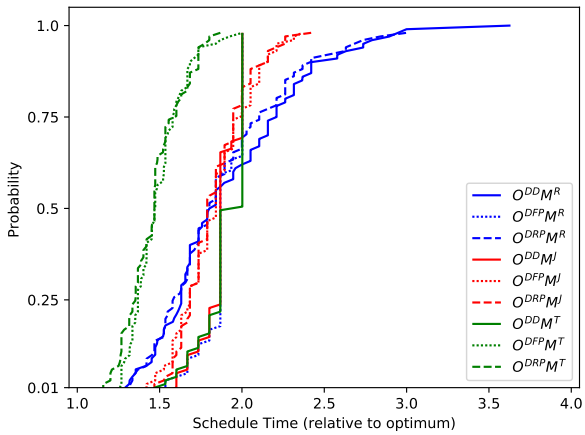

Figure 4. Cumulative Distribution Function graphs for heuristic algorithm outputs of schedule times for Case 2 (10 MDs, 3 MEC servers, 72 jobs).

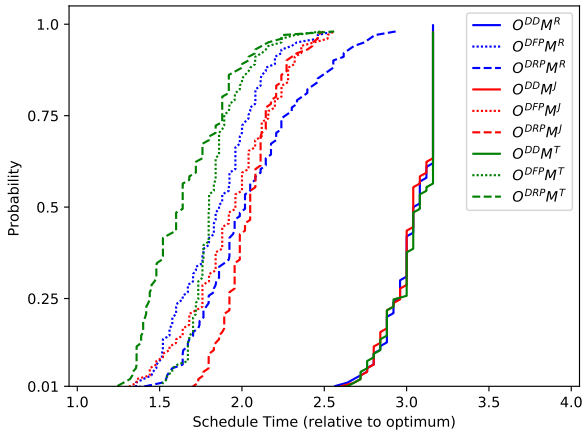

Figure 5. Cumulative Distribution Function graphs for heuristic algorithm outputs of schedule times for Case 3 (13 MDs, 3 MEC servers, 115 jobs).

V. Discussion

With different approaches to determining whether to offload each job combined with selection mechanisms for the MEC server to which a job is offloaded, a heuristic approach has been constructed which can closely rival optimum schedule times deduced by linear programming. In numerical simulations, the best performing algorithm used random probabilities for job offloading with MEC server allocation according to minimum-time server availability.

This is the first successful combining of schedule identification with heuristic approaches. We believe this represents an advance on previous approaches because it focuses on how best to partition computation capacity between MEC server and MD processors. With total task completion time as the criterion for success, the heuristics now developed can be flexibly deployed to optimize offloading from a single MD or optimally use resources in a heterogeneous MEC network. The heuristic algorithms can be run on individual MDs to allocate resources efficiently when multiple MDs attempt to offload simultaneously. It is anticipated that MEC networks will rely on multiple servers and have back-up processing power available in case of network overloads. In general, users of MDs with faster on-board processors will expect to offload fewer jobs if total completion time is the sole criterion. If link speeds decrease because of network congestion, an intuitive deduction is that fewer jobs will be successfully offloaded. As link speeds increase with the further development of 5G networks, the advantages of offloading to users of MDs will significantly increase because data transfer times in 3G and 4G networks greatly exceed server processing times.

The model presented here has simplifications built into the mathematics so that the performance of different algorithms was clearly testable. Conclusions drawn from work with single offloaded jobs can be readily transferred to multiple-job schedules when variable link speeds, computational complexity, network congestion and latency are considered [6]. In general, faster MD processor speeds will not favour offloading to shorten task completion times but offloading can still be beneficial from the MD with reduced energy consumption and increased battery life [6].

Our model "quantizes" the offloading process to either local or offloaded processing. This is a viable implementation of a Service-level Agreement (SLA) on a contractual basis between a user and a service provider. An individual user would run the heuristic to define an best-option schedule for that user communicating with a network. A centralized allocation of tasks could limit job offloading differentially between MD users but would be compatible with SLA clauses relating to fair-usage and network congestion avoidance.

VI. Conclusions and Future Work

A heuristic framework is presented in this paper for computational offloading of tasks from one or more MDs to MECs. In total, nine approaches were devised, which differed in how the offloading probability for an individual job was calculated and in which MEC server was then selected for offloading. Numerical testing of the algorithms explored scenarios with up to 115 multiple jobs from different numbers of MDs offloading to MEC servers. Our numerical simulations suggest that the solution obtained by our heuristic approach is between 1% and 20% of the global optimal solution obtained by a linear programming model. It was observed that the best solutions are obtained by using a known probability distribution for the offloading decision (O^{DPD}) and choosing a MEC with minimum solution time (M^{T}).

The heuristic approach presented in this paper only considered minimisation of the overall computational time. Our future work is looking into ways of incorporating energy use

and the price cost of offloading to the user to explore multi-objective optimization, including the impact of running the heuristic itself algorithms on time and MD energy factors.

VII. Acknowledgement

This work was supported by the Engineering and Physical Sciences Research Council grant number EP/P510427/1, Toshiba Research Europe Limited, and the University of Bristol.

References

[1] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1584–1607, Aug 2019.

[2] P. Corcoran and S. K. Datta, "Mobile-edge computing and the internet of things for consumers: Extending cloud computing and services to the edge of the network," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 73–74, 2016.

[3] S. Clinch, J. Harkes, A. Friday, N. Davies, and M. Satyanarayanan, "How close is close enough? understanding the role of cloudlets in supporting display appropriation by mobile users," in *2012 IEEE International Conference on Pervasive Computing and Communications*. IEEE, 2012, pp. 122–127.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[5] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[6] H. Guo, J. Liu, and J. Zhang, "Efficient computation offloading for multi-access edge computing in 5G hetnets," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[7] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks," *IEEE access*, vol. 4, pp. 5896–5907, 2016.

[8] N. I. M. Enzai and M. Tang, "A heuristic algorithm for multi-site computation offloading in mobile cloud computing," *Procedia Computer Science*, vol. 80, pp. 1232–1241, 2016.

[9] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3887–3901, 2016.

[10] J. Plachy, Z. Becvar, and P. Mach, "Path selection enabling user mobility and efficient distribution of data for computation at the edge of mobile network," *Computer Networks*, vol. 108, pp. 357–370, 2016.

[11] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE transactions on parallel and distributed systems*, vol. 26, no. 12, pp. 3317–3329, 2014.

[12] S. Paris, F. Martignon, I. Filippini, and L. Chen, "An efficient auction-based mechanism for mobile data offloading," *IEEE Transactions on Mobile Computing*, vol. 14, no. 8, pp. 1573–1586, 2014.

[13] M. Deng, H. Tian, and B. Fan, "Fine-granularity based application offloading policy in cloud-enhanced small cell networks," in *2016 IEEE International Conference on Communications Workshops (ICC)*. IEEE, 2016, pp. 638–643.

[14] P.-Q. Huang, Y. Wang, K. Wang, and Z.-Z. Liu, "A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing," *IEEE transactions on cybernetics*, 2019.

[15] N. Kaushi and J. Kumar, "A computation offloading framework to optimize energy utilisation in mobile cloud computing environment," *Int. J. Comput. Appl. Inf. Technol.*, vol. 5, no. 2, pp. 61–69, 2014.

[16] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[17] D. Satria, D. Park, and M. Jo, "Recovery for overloaded mobile edge computing," *Future Generation Computer Systems*, vol. 70, pp. 138–147, 2017.

[18] R. Singh, S. Armour, A. Khan, M. Sooriyabandara, and G. Oikonomou, "The advantage of computation offloading in multi-access edge computing," in *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 2019, pp. 289–294.

[19] S. Melendez and M. P. McGarry, "Computation offloading decisions for reducing completion time," in *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan 2017, pp. 160–164.

[20] F. Messaoudi, A. Ksentini, and P. Bertin, "On using edge computing for computation offloading in mobile network," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–7.

[21] F. Messaoudi, A. Ksentini, and P. Bertin, "Toward a mobile gaming based-computation offloading," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–7.