Proceedings of the 8th International Conference of Control Systems, and Robotics (CDSR'21)
~~Niagara Falls, Canada~~ Virtual Conference– May 23-25, 2021
Paper No. 301
DOI: 10.11159/cdsr21.301

Optimal Load-Aware Task Offloading in Mobile Edge Computing

Odysseas Polycarpou[1], Christos Anagnostopoulos[1], Kostas Kolomvatsos[2]
[1]School of Computing Science, University of Glasgow, UK
{2210049p, christos.anagnostopoulos}@glasgow.ac.uk
[2]Department of Computer Science and Telecommunications, University of Thessaly, Greece
kostasks@uth.gr

Abstract - Mobile Edge Computing (MEC) has emerged as a new computing paradigm to provide computing resources and storing applications closer to the end-users at the operator network boundary. One of the main challenges of MEC is task offloading, i.e., the transfer of computational tasks to a remote processor or external platforms such as a grid of servers or the Cloud. Task offloading mainly faces when and where is best to offload tasks to mitigate a smart device's energy consumption and workload. This paper tackles this challenge by adopting the principles of Optimal Stopping Theory (OST) with three time-optimised sequential decision-making models. A performance evaluation is provided with upon real data-sets on which our proposed models are applied and compared to the theoretical optimal model. Our results show how close our models can be to the theoretical optimal one based on probabilistic and scaling factors. Moreover, in our performance evaluation section, we conclude that one of the applied sequential models can be extremely close to the optimal one making it suitable in single-user and competitive user scenarios.

***Keywords*:** Mobile Edge Computing, Task Offloading, Optimal Stopping Theory, Sequential decision making

1. Introduction

In recent years, Mobile Edge Computing (MEC) has been proposed as a new network concept that enables Cloud computing capabilities and information technology service environment at the edge of the network. MEC is designed to be used by machines such as mobile devices, trains, planes, private cars or even enterprise premises such as factory buildings and homes, i.e., edge nodes. Edge nodes, store and perform applications/tasks on the network closer to the end-users. The emergence of MEC has seen as the development of several applications being launched on the Internet-of-Things. Because of this, the network is heavily loaded, and devices require fast and substantial processing to handle all these applications. The basic idea behind MEC is that by bringing these tasks closer to the cellular customer, usually the Cloud or servers connected on the Cloud (edge servers), the congestion on the network is reduced and applications perform better and more efficiently.

One of the most notable mobile edge computing applications is computational offloading or task offloading in Cloud computing. One can think of the Cloud as a large data storage space for applications with many edge servers connected to it and distributed over a large area. Whenever the Cloud is busy, which is the case, applications will be transferred to one of the deployed edge servers for faster execution. Task offloading is the transfer of computational tasks from a local edge node to a separate external processor such as a server or grid of servers for execution. A central processor on a local edge node will process tasks by executing rudimentary arithmetic, control logic and operations. The efficiency of this processing is depended on the instructions per second a CPU can execute, and because of the wide range of CPU types, the processing power and efficiency varies. Moving applications to an external faster and more powerful processor such as an edge server can accelerate processing and improve application execution efficiency and latency. Another challenge that has emerged in task offloading is when a mobile node moves among many MEC servers. In the optimal scenario, the mobile node should find the best server to connect to at the best time based on the load on the MEC server at that time instance. The load on a deployed MEC server among a group of servers can broadly vary. For example, at a time instance, many users are connected on the MEC server waiting to execute their tasks. However, at the next time instance, only a few users may be connected to the same MEC server making it ideal for offloading and executing tasks. Some of the questions that arise from this scenario are: *When a decision for offloading is made, should we offload now or keep it for later if a better server may be found*? *What are the chances that we can find a better server if we keep looking*?

In this paper, we examine a scenario where a mobile node moves between multiple MEC servers. The decision for the computational offloading is made by three sequential decision-making models that can be managed and optimised by applying the principles of Optimal Stopping Theory [12]. We will monitor the behaviour of real data sets of varying CPU load values at different time points. The models applied have to pick the best server to offload the data at best possible time in a sequential manner to achieve the best execution of tasks possible on the MEC server. We also assume that in all these three decision-making models, when a server is discarded, we cannot return to pick it up. Our House Selling Optimal Stopping Time (OST) [5] method can be applied in different scenarios to be integrated with decision-making applications. Unlike other related work done on task offloading, our models explicitly focus on the CPU load value of the MEC servers. Our model can efficiently determine the best possible server to offload the tasks in most of the simulations performed in our performance evaluation.

The contributions of our work are: **(i)** We propose three OST-based models that are implemented and applied on a real data set to maximise the chance to find the optimal MEC server to offload the tasks assuming a uniform network status between the node and the server; **(ii)** We optimise our House Selling model to outperform the random and the Secretary models [5] and perform almost as efficiently as the problem's optimal solution; **(iii)** We created a time series analysis tool and a simulation executable that can allow the user to set different parameters and produce a visual representation of the results; (iv) Using those results, we provide a comparative evaluation of the models against the theoretical optimal model as well as individually.

The paper is organized as follows. Section 2 reports on the related work while Section 3 describes the necessary preliminary information. Section 4 elaborates on the proposed models and Section 5 deals with our experimental evaluation. Finally, Section 5 concludes our paper presenting future research directions.

2. Related Work

The vast majority of the work being done on task offloading focuses on whether the task should be performed locally or offloaded to the Cloud. The main goal is to be as close as possible to the optimal result by minimising the execution delay and energy consumption. The work in [1] proposes an OST model that aims to address the problem in MEC and proposes two baseline models that will improve the challenge that nodes face. That is when and where they will get connected (to an edge server) to perform computing tasks, i.e., the problem addressed is to find and offload on the best candidate among a group of edge servers and the best one to connect based on the load traffic and latency of each one. The OST-models proposed by the authors are compared to the House Selling (HS), the random and p-stochastic models. In [8], motivated by the increasing demand for computation resources of IoT mobile caused by the creation of diverse IoT applications, the authors focus on addressing the problem of the task offloading between Internet Mobile Devices and Aerial Unmanned Vehicles. The proposed model aims to minimise the overall energy consumption for accomplishing the tasks. The work presented in [3] focuses on the problem of which tasks should be offloaded and not on where the tasks will be offloaded. Therefore, the paper's main challenge is to select the appropriate tasks to be offloaded to peers or Cloud. The model proposed in this paper to solve the problem targets two characteristics, i.e., to maximise the performance and minimise the consumption of resources. The significance of the approach is the use of Machine Learning in a new proposed model upon a Long Short Term Memory (LSTM) network that will be able to indicate which tasks should be offloaded. This intelligent scheme makes this decision based on a deep learning scheme and a rewarding mechanism. In [2], given the movement of mobile nodes between MEC servers, the authors aim to propose a model that gives the connection of a mobile node to the best edge server at the best time in order to optimise the quality of service. The problem tackled is the offloading decision making by adopting the OST principles using real data in the experimental evaluation. Also, the optimal server/time is unknown and not provided meaning that the OST-based model can achieve a delay close to the optimal. The work discussed in [7] focuses on the problem of moving mobile nodes such as crewless aerial vehicles, vehicular networks, data analytics and augmented reality being able to choose the ideal time and server candidate, using principles of the OST to minimise the execution delay in a sequential decision manner. The significance of the study is that reduces the execution delay for the decision-making process and deals with how these foundations are used to reach the optimal time and server for the mobile node to connect. The work in [4] tackles with the optimal decision for computation off-loading by setting hard

task deadlines. The paper uses a Markovian OST model which is computed using dynamic programming. The proposed model is proven to be energy optimal. Unlike our approach, this study aims to optimize the model in terms of energy also guaranteeing hard task deadlines. The authors in [6] assume that the offloading decisions are given and derive the closed-form expressions of the optimal transmit power and local CPU frequencies. The proposed algorithm is based on a reduced-complexity Gibbs Sampling scheme to take the optimal offloading decisions. The aim of the authors is to minimize the complexity of the model. Almost all of the works presented above focus on reducing energy consumption, the complexity and introduce new approaches to perform the decision making. Unlike these, we focus on applying three OST-based models in a sequential manner and prove that we can perform as close as possible to the optimal solution of finding the best server at the best time to offload our tasks.

3. Preliminaries

The OST refers to the problem of choosing a time to stop and take a particular decision so that we can maximise an expected reward or minimise an expected cost. OST problems can be met in many areas such as statistics, economics and computing science. They are associated with a sequence of random variables $[X_1, X_2,\ldots X_n]$ and a sequence of reward functions y_0, $y_1(x_1)$, $y_2(x_1,x_2)$, ..., $y_\infty(x_1,x_2,...)$ that depend on the observed values in random variables. Given these, while observing the sequence of random variables at each step, we choose between stopping or continuing observing. If we stop observing at a given step, we receive the reward function sequence's corresponding reward. The aim is to stop at an observation where the expected reward will be as high as possible (or equivalently, expected cost as low as possible). OST models can be applied either sequentially or non-sequentially on a large set of variable observations. Sequential means that we divide our data into chunks and observe from the next position we choose to stop. Non-sequential means that we start to observe from the first position of each chunk of observation variables no matter where we choose to stop in the sequence. For our study, the models we implement are non-sequential as we assume the uniformity of data. Some of the most common OST problems are the Secretary problem, the random probability scheme and the house selling which are the three (3) models we apply and optimise in our evaluation [11].

The envisioned scenario is a case of a Vehicular Network or Internet of Vehicles, moving along the road and where many MEC servers are distributed around them (a similar setup like in [9]) as shown in Figure 1. MEC servers along the road provide computing resources for vehicles on the move, to offload and perform computing tasks. Such tasks involve map rendering, image recognition or data analytic tasks. Similarly, tasks can be generated by smartphones carried by the passengers in the cars. The connection between a MEC server and a vehicle on the road is established wirelessly over the cellular network. At each time instance, there is a load value L on each server, which is a random variable in our context. In simpler words, L depicts how crowded a server is in terms of connected users waiting to execute their tasks. When a MEC server is heavily packed and loaded with users, the transmission delay from the user to the server and back increases. Our approach aims to minimise the transmission delay by adopting models that can pick the best possible server with the least load (least crowded).

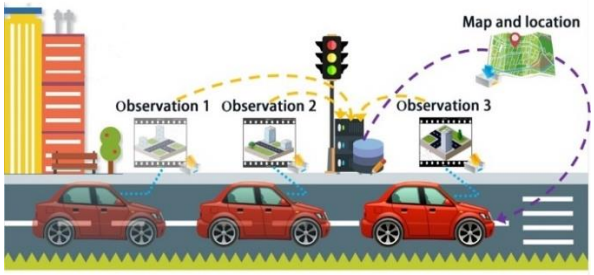


Figure 1: Car passing through MEC servers

Once a task is generated and needs to be offloaded on a MEC server, the user node passes through several servers ready to access the computation task. To do so, the current load of the server needs to be checked. When a server is picked up, then the observation stops and the rest of the servers are discarded (are not considered by the mobile node anymore). However, if the model chooses to discard a server while observing and move to the next one, it cannot return to it (e.g.,

imaging the scenario in Figure 1 where the car/vehicle cannot abruptly do the reverse while driving in one specific lane/direction). Therefore, our approach's main objective is to maximise the probability of offloading the tasks to the possible server. To formulate our problem, we made the following assumptions: **(a)** the status of network between all servers and the node users is uniformly distributed; **(b)** when a server is discarded, it cannot be re-evaluated, i.e., no recall is allowed.

4. Server Load-based & Time-optimized Task Offloading

The Random (P) Model. The random probability model or Random(P) relies on a probability P and a randomly generated float number x to decide whether to offload on a MEC server. Unlike the other two OST models, Random(P) does not consider the MEC servers' load to make the offloading decision. Whether the model will stop and offload on a MEC server or keep looking is based on how high the simulation's probability P has been set and the random variable x's value. Random(P) takes a sequence of N>0 observations which correspond to the MEC servers availability. Before applying the model, we set a fixed probability P and a function which generates a random floating number x in a given interval. In the first run, the model will look at the first entry of N, i.e., N1. If the probability P is higher than the random number x, then the model will stop and offload on that server. If the above condition is not satisfied, the model will discard the server and move to the next one, generate a new random number x and then make a decision again.

The Secretary Model. In the Secretary OST problem [14], an administrator wants to hire the best secretary out of N applicants. The order the applicants are interviewed is one at a time, and when an applicant is discarded cannot be reviewed. The administrator can evaluate and rank each seen secretary's quality and ability, but not those of the unseen applicants. The challenge here is to adapt an optimal strategy to find the best time to stop reviewing applicants to select the best. Using this context, we can apply this scenario to our case, taking candidates to be a group of N MEC servers. The algorithm needs to find the best possible server to offload the tasks to achieve optimality of task execution based on the current load of the specific server [13]. In terms of solving the Secretary problem we can scan through the first r MEC servers and then choose the first option that is better than any of the MEC servers in [1, r]. Assume that i is the greatest integer and occurs at n+1. The **following** two conditions must hold true: **(a)** the maximum integer cannot be in [1,r]. If yes, then we lose because we are missing the best option; **(b)** the equation max([1,r]) = max([1,n]) must be true. Thus, to calculate our approach's effectiveness, we need to know the probability that both will hold. Given some n, the probability is: $\frac{r}{n}\frac{1}{N}$ where: 1/N is probability that i occurs at n+1 & r/n is the probability that the aforementioned condition (b) is true. To calculate the probability for **some** r, P(R), we need to sum over n>=r:

$$P(\mathrm{R}) = \frac{1}{N}\left(\frac{r}{r} + \frac{r}{r+1} + \frac{r}{r+2} + \cdots + \frac{r}{N-1}\right) = \frac{r}{N}\ \sum_{n=r}^{N-1}\frac{1}{n}. \quad (1)$$

This is a **Riemann** approximation of an integral so we can rewrite it by letting $\lim_{N\to\infty}\frac{r}{N} = x$ and $\lim_{N\to\infty}\frac{n}{N} = t$ getting that: $P(\mathrm{R}) = \lim_{N\to\infty}\frac{r}{N}\sum_{n=r}^{N-1}\frac{N}{n}\frac{1}{N} = x\int_x^1\frac{1}{t}dt = -x\ln x$. Now, we can find the optimal r by solving for $P'(\mathrm{R}) = 0$ back plugging r_{optimal} back into $P(\mathrm{R})$ finding the probability of success which is: $P'(\mathrm{R}) = -\ln x - 1 = 0 \Rightarrow x = \frac{1}{e},\ P\left(\frac{1}{e}\right) = \frac{1}{e} \approx .37$. Therefore, this strategy selects the best server, about 37% of the time. It should be noted that we cannot select the first observed server as we have no other observation to compare with. The best approach is to choose an optimal sample size which can be done by calculating N/e. The probability that the applicant selected is also the best is given by:

$P(R) = \sum_{n=R+1}^{N} P(n\ is\ selected\ and\ is\ best) = \sum_{n=R+1}^{N} P(n\ best)\, P(next\ in\ R\ out\ of\ n-1) = \sum_{n=R+1}^{N} \frac{1}{N}\frac{R}{n-1} = \frac{R}{N}\sum_{n=R+1}^{N}\frac{1}{n-1}$.

The House Selling Model. Another famous OST problem is the House Selling problem [5]. The problem occurs when a seller has a house he/**she wants** to sell. The sequence of N observations represents the price which the seller is offered for his/her house. The approach is to maximise the amount we earn by choosing a stopping rule. If the seller sells the house on the n day, the amount earned is given by:

$$y_k = (1+b)^{N-k} X_k \quad (2)$$

where b in [0,1] is a discount factor. In our case, we can think the load values in the sequence of N available MEC servers as the *availability* values. To do this, we scale the availability values to be in the unity interval adopting the min-max normalization process. The target is to stop at a time instance 1<= k <=n, where we potentially have the highest availability. Before proceeding to the offloading decision, we need to calculate the decision values to be compared with the scaled availability values, which will lead to the decision of whether to offload or not. The decision values are calculated with backward induction which is part of a dynamic programming approach, i.e., $a_k = \left(1/(1+r)\right) * \left((1+(a_{k+1})^2)/2\right)$ for k = 0, …, n-1, α_k being the realization of the availability values threshold and initial value a_{n-1}=0.5/(b+1). When the decision-making process is started, our model will decide whether to offload on a specific server based on the α values. If the availability value at index i in the sequence of N MEC server loads, i.e., s_i is higher or equal to the corresponding decision value at the same index i then the model will stop and offload the tasks on that server. If the above condition is not satisfied, then the model will go to the next availability and decision value and compare it until the simulation is finished. Starting with the dynamic programming objective function:

$$J_n(x_n) = max\left[(1+b)^{N-k} x_n, E[J_{n+1}(w)]\right] \quad (3)$$

and $a_n = \frac{E[J_{n+1}(w)]}{(1+b)^{N-k}}$ the decision is taken as: if $s_n >= \alpha_n$ then send task; if $s_n < \alpha_n$ then continue observing the load realizations.

Let $V_k(x_n) = max(x_n, (1+b)^{-1}E[V_{k+1}(w)])$ and then $a_k = E[V_k + 1(w) * (1+b)^{-1}]$. Hence $V_k(x_n) = max(x_k, a_k)$ and $V_{k+1}(x_{n+1}) = max(x_{k+1}, a_{k+1})$ or $E[V_{k+1}(x)] = E[max(x, a_{k+1})] = E[V_{k+1}(x)] * (1+b)^{-1} = E[max(x, a_{k+1})](1+b)^{-1}$. Substituting $max(x, a_{k+1})$ into $E[V_{k+1}(x)] * (1+b)^{-1}$ gives $a_k = E[max(x, a_{k+1})] * (1+b)^{-1} \Longrightarrow a_k = (1+b)^{-1}\left[\int_{a_{k+1}}^{1} x\, df(x) + \int_0^{ak+1} a_{k+1} df(x)\right] \Longrightarrow a_k = (1+b)^{-1}\left[1 - \int_0^{a_{k+1}} x\, df(x) + \int_0^{ak+1} a_{k+1} df(x)\right] \Rightarrow a_k = (1+b)^{-1}\left[1 + \frac{{a_{k+1}}^2}{2}\right]$ with $a_{N-1} = E[V_N(w)](1+b)^{-1} = \frac{E[x]}{(1+b)}$.

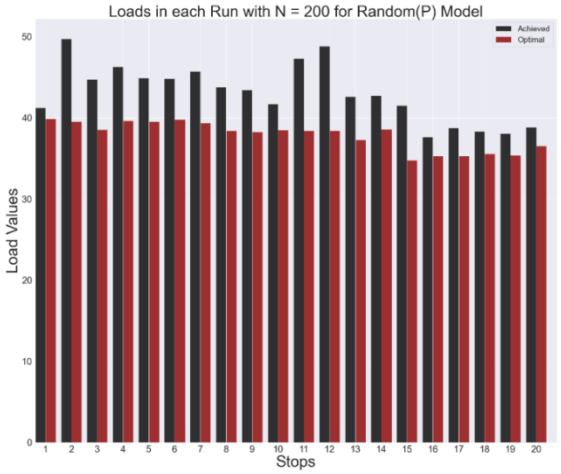
5. Experimental Evaluation

In our experimental evaluation, to simulate the MEC server candidates, we adopt the real dataset of EC2 CPU utilisation load values collected over two (2) weeks. The dataset comes from a real Amazon Web Services Cloud Watch service [10]. In this dataset, the CPU load is being recorded every five (5) minutes and contains about 4000 measurements separated in chunks of 100 values. The provided time instances are represented by scalar values instead of the real timestamps to allow us to illustrate how far we are from the optimal solution. Therefore, we have 4000 real values at 4000 points in time. To implement our OST models, we chose to work with Python 3.x, Jupyter Notebook, Numpy and Pandas libraries. In our experiments, we take chunks of N = 100 servers each. We select several probability values P to examine

the Random(P) behaviour as the load is not involved in our decision-making models setting them up to P = [0.05, 0.1, 0.2, 0.3, 0.5]. We compare the three aforementioned models, i.e., Random(P), Secretary and House Selling with the optimal solution and provide the relevant numerical results. The implemented simulations take chunks of 100 records each, and in this sequence, they need to observe each MEC server and make a decision to offload to the best possible server.

In this section, we apply the Random Probability, the Secretary Model and the House Selling Model on our set of data and pass in the chunk size we want to observe, and several parameters we want to give each model for our decision-making process. Initially, for the first set of simulations, we assigned the following parameter values: N = 200 (we try to see how the models behave with a large set of MEC servers), p = 0.1, b=0.015. In the figures depicting our results, we adopt a bar chart for the optimal values (red) compared with the load on the MEC servers achieved by our model when offloaded (black). These results are depicted at the left of each figure. Additional figures (the right part of each figure) are devoted to show the difference in time instances of the optimal points in time that we could have stopped, i.e., how far we are from the optimal stopping point in time.

As we can see in Figure 2a, the Random(P) model never achieves to offload to the optimal server in any of the runs performed. Not only that, the difference between the optimal and the achieved MEC server is high in the majority of the experimental scenarios. Besides, if we use a higher probability p than 0.1, the chance to offload the envisioned tasks too early increases, minimising the probability that we hit the best possible server. Figure 2b shows that our model's offset from the optimal stopping time is also significant, with most of the stopping times being far ahead of the optimal point.

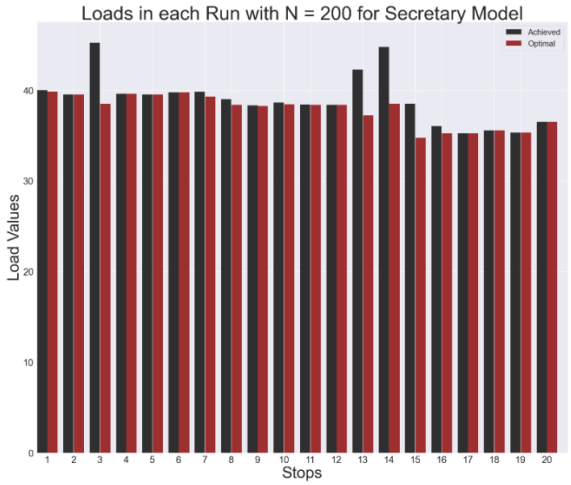


(a) Random(P) load value/stop chart

(b) Random(P) load value/stop chart

Figure 2: Performance assessment for the Random(P) model

The adoption of the Secretary model can significantly increase efficiency compared to the Random(P) model. This is because the Secretary model targets at the MEC servers' actual values and does not rely on a probabilistic approach. In Figure 3a, we can see that the Secretary model can find the optimal server to offload the tasks in the most of the runs. The model is far from the optimal in only four (4) runs. The emptiness on the time instances in the graph confirms that we have found the optimal stopping time. Additionally, in Figure 3b, it is essential to mention that, even though the times achieved show that we have significantly miss the optimal time instance, the server we choose to offload is the next best server after the optimal proving the efficiency of our Secretary model.

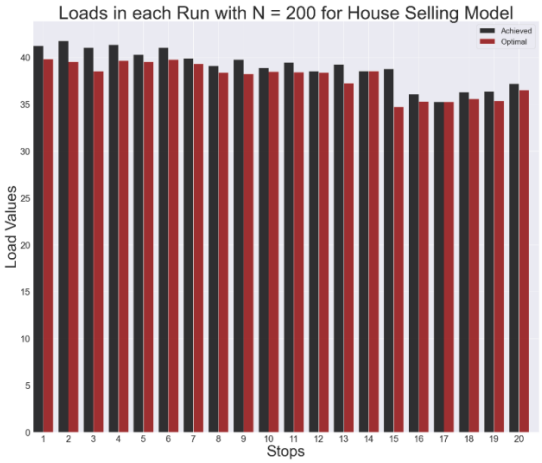


(a) The load realization when the Secretary model is adopted

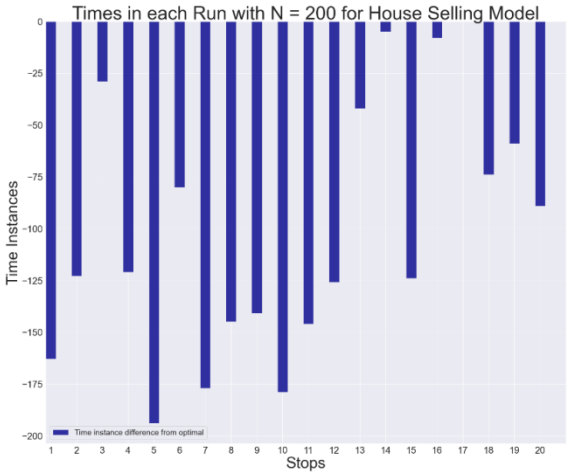
(b) Stop time instances (Secretary) chart

Figure 3: Performance assessment for the Secretary model

Recall that the House Selling model relies on a discount factor to execute the offloading decision-making process. As explained in our methodology, the closer this factor is to zero, the close to the optimal performance the model achieves. Since using a factor of zero is not realistic, we can bring it down to a very low of value equal to 0.015, which is the highest value of b at which our House Selling model performs better than the Secretary model. From the simulation results presented by Figure 4a, we can see that our House Selling model chooses the best or the second-best server to offload the tasks. The proposed model is proven to be very close to the optimal solution. Moreover, making b even smaller brings our model more close to the optimal solution than in the previous experimental scenario. As shown, the House Selling model significantly outperforms the other two schemes described in this paper.



(a) The load realization when the House Selling model is adopted



(b) Stop time instances (House Selling) chart

Figure 4: Performance assessment for the House Selling model

7. Conclusions

In this paper, we experiment with load-aware time-series analysis and propose Optimal Stopping Theory-based models capable of taking non-sequential offloading decisions for task offloading between users and MEC servers. Mobile nodes determine when and on which server to offload the tasks considering the current load on each MEC server. Our implemented models can use this information to determine when it is the best time to offload tasks. The three models are compared against the optimal solution with the House Selling Model being the best and the Secretary Model coming next.

The House Selling Model can incorporate into the decision making a discount factor. Based on this value and the chunk size of the sequence, we outperform other baseline solutions in terms of load. We examine a wide set of experimental scenarios to find these optimal schemes to detect appropriate thresholds to have an efficient model upon the House Selling approach and perform better than the remaining models.

Table 1. Nomenclature

N	Number of MEC servers in each run
p	Probability set for Random (P) problem
X	Random variable used in Random (P) problem used for comparison
R	Discount factor set for House Selling Problem
$P(R)$	Probability selecting the best candidate in Secretary model (2)
y_k	Reward from HS (11)
A	Availability values for HS
s_k	Scaled Availability Values for HS (12)
a_k	Decision Values for HS (13)

References

[1] Alghamdi, C. Anagnostopoulos and D. P. Pezaros, "On the Optimality of Task Offloading in Mobile Edge Computing Environments," 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 2019, pp. 1-6, doi: 10.1109/GLOBECOM38437.2019.9014081

[2] Alghamdi, Ibrahim; Anagnostopoulos, Christos; P. Pezaros, Dimitrios. 2019. "Delay-Tolerant Sequential Decision Making for Task Offloading in Mobile Edge Computing Environments" *Information* 10, no. 10: 312. https://doi.org/10.3390/info10100312

[3] Kolomvatsos, Kostas; Anagnostopoulos, Christos. 2020. "A Deep Learning Model for Demand-Driven, Proactive Tasks Management in Pervasive Computing" *IoT* 1, no. 2: 240-258, https://doi.org/10.3390/iot1020015

[4] A. Hekmati, P. Teymoori, T. D. Todd, D. Zhao and G. Karakostasy, "Optimal Multi-Decision Mobile Computation Offloading With Hard Task Deadlines," 2019 IEEE Symposium on Computers and Communications (ISCC), Barcelona, Spain, 2019, pp. 1-8, doi: 10.1109/ISCC47284.2019.8969696.

[5] D. Bertsekas.Dynamic Programming and OptimalControl. Athena Scientific, 2017.

[6] J. Yan, S. Bi, Y. J. Zhang and M. Tao, "Optimal Task Offloading and Resource Allocation in Mobile-Edge Computing With Inter-User Task Dependency," in IEEE Transactions on Wireless Communications, vol. 19, no. 1, pp. 235-250, Jan. 2020, doi: 10.1109/TWC.2019.2943563.

[7] I. Alghamdi, C. Anagnostopoulos and D. P. Pezaros, "Time-Optimized Task Offloading Decision Making in Mobile Edge Computing," 2019 Wireless Days (WD), Manchester, United Kingdom, 2019, pp. 1-8, doi: 10.1109/WD.2019.8734210.

[8] D. Callegaro and M. Levorato, "Optimal Computation Offloading in Edge-Assisted UAV Systems," 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 2018, pp. 1-6, doi: 10.1109/GLOCOM.2018.8648099.

[9] J. Zhang and K. B. Letaief, "Mobile Edge Intelligence and Computing for the Internet of Vehicles," in Proceedings of the IEEE, vol. 108, no. 2, pp. 246-261, Feb. 2020, doi: 10.1109/JPROC.2019.2947490.

[10] Numenta. (n.d.). Numenta/nab. Retrieved February 11, 2021, from https://github.com/numenta/NAB/tree/master/data/realAWSCloudwatch

[11] Knowing when to stop. (2018, March 05). Retrieved February 11, 2021, from https://www.americanscientist.org/article/knowing-when-to-stop

[12] Optimal Stopping and Applications, UCLA, https://www.math.ucla.edu/

[13] Rs.io. (n.d.). Retrieved February 12, 2021, from https://rs.io/the-secretary-problem-explained-dating/

[14] Duso, L. (2020, September 10). Math-based decision making: The secretary problem. Retrieved February 12, 2021, from https://medium.com/cantors-paradise/math-based-decision-making-the-secretary-problem-a30e301d8489