Real-Time Task Assignment in Fog/Cloud Network Environments for Profit Maximization

Jonathan Daigneault and Marc St-Hilaire
Department of Systems and Computer Engineering
Carleton University, Ottawa, Canada
JonathanDaigneault@cmail.carleton.ca, marc_st_hilaire@carleton.ca

2022 International Wireless Communications and Mobile Computing (IWCMC) | 978-1-6654-6749-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/IWCMC55113.2022.9825270

***Abstract*—This paper proposes a heuristic model for the real-time task assignment problem in fog/cloud computing networks. Its purpose is to maximize profit by assigning tasks (either to fog nodes or the cloud) while simultaneously considering green computing practices (i.e., changing the states of fog nodes). Compared to the optimal model introduced in [1], the proposed real-time task assignment heuristic performs extremely well considering the fact that it only has access to partial information as opposed to the full historical data for the optimal model. The relative profit returned by the heuristic is consistently within the 79 to 89% range with a processing time that is significantly improved thus, enabling service providers to use such a model in larger network environments and in real-time.**

***Keywords*—Fog computing, cloud computing, task assignment, assignment problem, green computing, heuristic.**

I. Introduction

There are many parallels that could be made between fog and cloud computing networks such as their ability to offload processing from the device to a remote processing entity. However, the advantage of fog computing networks lies in the low latency processing that can enable devices to be reduced to simple peripherals (audio, video and Input/Output) and offload all essential and application processing to the fog. By deploying fog nodes, network operators can offer enhanced services and therefore increase their revenue.

Documented fog computing networks are being designed initially to be efficient in their environments. This provides short term benefits to service providers, but fails to scale as demand increases or shifts within their coverage area [2]. This static development of fog networks is inefficient and lacks the ability to adapt to inherently dynamic activities such as allocating tasks to fog node resources. As the demand frequency and patterns shift over time, the fog computing network becomes less efficient at supporting these requests and will relegate additional tasks to the cloud when it can't support the low latency resources demanded by the applications.

To address this problem, network operators must use resource assignment models to assign fog network assets to task requests while maximizing profit. The problem is further complicated when combining multiple objectives such as service coverage (assigning as many incoming tasks to a fog node) and reducing operating cost by adopting green computing practices (e.g., shutting down unused fog nodes and only starting them up when required).

As can be seen, an efficient management of available resources is paramount to operating a fog computing network. In a previous work [1], we proposed a mathematical model to determine the optimal profit for task assignment within a 2-tier fog/cloud network environment. Although the model can be extremely useful for benchmarking, it cannot be used in a real-time environment due to the following two main limitations.

- Knowledge needed: The model uses complete knowledge of all inputs to find optimal solutions. In a real environment, some of this information (such as start time, end time, task duration, etc.) is not available at the time to make a decision. Therefore, real-time models should be able to make good decisions with partial information.
- Execution time: The optimal model takes an extensive amount of computation time to find optimal solutions. As reported in [1], the solver (CPLEX) takes several seconds to solve small problems to several minutes/hours for larger problems. Clearly, this is not appropriate if real-time decisions are needed.

To address the above limitations, this paper proposes a heuristic algorithm for the real-time task assignment problem in fog/cloud network environments. The algorithm assigns tasks to fog nodes as they arrive and simultaneously manages the status of all fog nodes. This algorithm can be useful for network operators to improve profit generation and user experience.

The rest of this paper is organized as follows. In Section II, we present relevant background information along with the related work including a small overview of the mathematical model presented in [1]. Then, we introduce the proposed heuristic in Section III and analyze its performance in Section IV. Finally, the paper is concluded in Section V.

II. Background and Related Work

Figure 1 shows a typical fog computing network. As shown, tasks request resources to the fog computing network and a network controller (for example) determines how they are assigned: either by assigning them to a fog node or the cloud. Assignment to a fog node leads to an improved experience to the user (i.e., significant improvement in latency). Tasks can also be relegated to the cloud for processing, taking no fog computing resources, but eliminating any performance benefits of operating in the fog. Each fog assignment is also not equivalent. Notably, an allocation to a distant fog may not

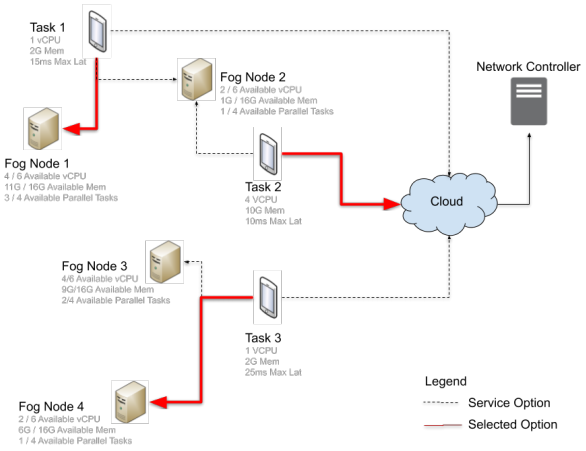

Fig. 1. Fog computing network overview.

provide any performance benefits to the end user while still taking away resources in the fog for other tasks that could benefit from a fog assignment. Moreover, the operational status of the fog nodes must also be considered before assigning tasks. In this paper, we assume that a network controller, located in the cloud, is the decision making authority for task assignment and fog node status change decisions. It is not a latency sensitive function since it relies on internal processing for its decision making and sends short messages to the various network elements (clients and fog nodes) to communicate its decisions.

Previous models, such as [3] and [4], were made to optimize the initial location (i.e., deployment) of fog nodes. They largely rely on predictions and historical data related to tasks requested [5] or use a population density coverage model [6]. Regardless of the performance of a fog network at a given start point, its performance in time varies since the static allocation of the network must serve a dynamic set of tasks in real-time. For this reason, real-time task assignment algorithms are required to optimize the performance of a given fog/cloud infrastructure. The task scheduling problem assumes that the fog network is established and functional and can not directly impact the location of the fog network hardware.

The task assignment problem is well known in operation research and several models have been proposed from different angles. In the context of cloud computing, Xiao et al. suggest a model to address the problem of over-provisioning of physical servers in cloud computing applications [7],. The model is based on two goals: avoiding server overload and green computing. Both goals are somewhat contradictory, but they claim to develop an automated system for a balanced resource provisioning. Similarly, in [8], Wei et al. propose a cost-time optimized algorithm for client to cloud resource allocation.

As mentioned previously, a real-time task assignment algorithm only has partial knowledge about the incoming tasks (for example, the task duration may be unknown upon arrival). Therefore, several authors tried to schedule tasks with uncertainty in a cloud environment. For example, Kianpisheh et al. [9] apply various machine learning algorithms to attempt to improve predictions. Similarly, Smith et al. [10] and Ramirez et al. [11] use historical processing statistics to generate an estimate.

Some other papers looked at the task assignment problem specifically in the context of fog networks. It is important to note that adding an extra layer of computing (i.e., the fog layer) adds to the complexity of the problem. In a recent comprehensive survey [12], Shakarami et al. note that there is severe competition among end users to gain access to limited fog/edge computation and communication resources. They also mention that resource allocation schemes are extremely important and to that end, present a review of various resource provisioning approaches in edge/fog computing.

Researchers in [13] looked at the optimal task assignment in a fog-cloud computing environment. From their formulation, an optimal assignment is a solution that can minimize cost and makespan (i.e., the time taken to process all requests). In our current paper, we are mainly interested in processing tasks within the requested delay without necessarily trying to minimize the overall makespan.

Other researchers, such as Do et al. [14], introduce a proximal algorithm to optimize resource allocation in a fog computing network. Geo-distributed cloud networks are often energy inefficient and have a large carbon footprint. Traditional optimization resolution methods such as the steepest gradient do not always provide an optimal solution to the problem. The paper proposes using proximal functions to transform the optimization algorithms to eliminate the possibility of dealing with convex functions, thus improving resource allocation in fog computing networks overall. Similarly, Lai et al. [15] propose a model that allocates users to a fog node using a dynamic quality of service. They showed that maximizing allocated users while diminishing QoS can significantly improve user experience across the network.

More recently, Daigneault et al. [1] proposed a mathematical model to optimally assign resources and ensure that profit is maximized while considering individual task revenue and operating costs. This was the first model to simultaneously consider resource allocation and green computing in fog computing networks. Unfortunately, as mentioned in the paper, the model relies on perfect knowledge of incoming tasks which is unrealistic. Moreover, the model takes a significant amount of time to find solutions. However, the model can be extremely useful as a benchmark to evaluate real-time resource allocation algorithms. Since the heuristic proposed in this paper is based on [1] and that we also use it for comparison, we present a short (due to space limitation) summary of the model in Figure 2. We can see that the objective function tries to maximize of profit. This is essentially achieved by maximizing the number of time windows where tasks are assigned to fog nodes (thus generating an extra relative profit) while simultaneously minimizing the operating cost. The operating cost can be minimized by implementing green computing practices such as properly managing the status of the various

Objective function:

maximize Profit = {Revenue − Operating cost}

Subject to:

- **Fog nodes status loop prevention constraints**: make sure that each fog node can only be in one given state (on, off, starting up, shutting down) at a time.
- **Fog node status verification constraints**: make sure that tasks are only assigned to fog nodes that are powered on.
- **Capacity constraints**: make sure that the fog node has enough capacity (vCPU, memory, and parallel task) to handle all assigned tasks.
- **Task single assignment constraints**: make sure that tasks are assigned to at most one fog node.
- **Latency constraints**: make sure that the selected fog node meets the latency constraints (in terms of distance).
- **Tasks start/stop time constraints**: make sure that we meet the requested start time and end time (duration) of tasks.
- **Fog nodes status transition constraints**: make sure that we follow the proper transition sequence for the fog node status.
- **Variable bound constraints**: define the domain of the variables.

Fig. 2. Summary of the mathematical model proposed in [1].

fog nodes.

To the best of our knowledge, no related work was found that proposes a real-time resource allocation model for profit maximization while considering the green computing requirements of a fog computing network. In the next section, we introduce the proposed heuristic to tackle this problem.

III. Proposed Heuristic

In order to be consistent with the benchmark proposed in [1], this paper uses identical assumptions, notations and optimization function to propose a solution for real-time task allocation in fog computing networks. For example, we use the same set I for the set of tasks, J for the set of fog nodes and T for the set of time windows. For more information about the heuristic algorithm, please refer to [16].

A. Input Data

The input parameters are manipulated between time windows to simulate a real-time task demand schedule. A start time constraint is used to ensure that each new task i is revealed to the model as a new request based on its start time. A task duration time constraint ensures that the duration parameter of each task i is revealed once a task makes a stop request to the model. These changes modify the information available to a fog computing network controller to make task allocation and fog node status change decisions without knowledge of future time windows.

The heuristic model must make a determination on active task assignment at every time window $t \in T$. Once the model

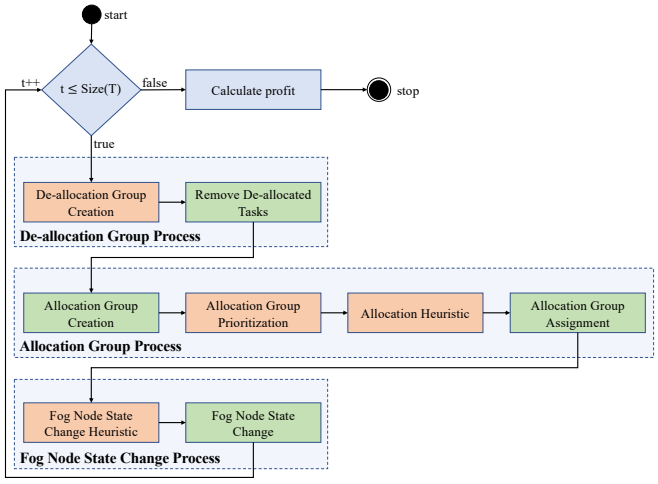

Fig. 3. Heuristic Model General Framework Flow.

makes an assignment decision, it can not be modified at a later time window. This is of significant importance to achieving accurate results since the fog node start-up and shut-down decisions are only made from the state of the fog computing network at time t and its past trends. This creates a larger duality between the green computing and service provision requirements.

B. Heuristic Model Framework

A general overview of the heuristic model framework is shown in Figure 3. Boxes coloured in green are fixed functions (i.e., they do not contain configurable values) while boxes coloured in orange can be tailored as they contain values that can be customized to improve performance. The framework is designed to perform task allocation, de-allocation and fog node state change determinations at every time window as required in a real-time allocation application. During operation, the framework accounts for it by looping the processes over the total number of time windows in its operating environment. These groups include all tasks that must be processed within a time window t.

1) De-allocation Group Process: The de-allocation group process lists the tasks that are tagged for de-allocation in the current time window. It adds any completed tasks to the de-allocation data set. The heuristic model will only populate the task duration parameter if a completed request is made to the fog computing network.

Tasks that have completed previously are obviously included in the de-allocation group since they offer no benefit to being assigned to the fog computing network. However, task completion can also be predicted and used for advance de-allocation. This can be notably beneficial in scenarios where operating costs could be reduced to outweigh the revenue generation of having it mapped to the fog node. The performance of advance de-allocation is dependent on the predictability of task duration and its benefits are also based on available resources

in the fog computing network. In this paper, the advance deallocation procedure is based on the predicted duration which is simply the average of all completed tasks in the simulation. Using more advanced prediction mechanisms such as machine learning is part of future work.

2) Allocation Group Process: The creation of the allocation group is a trivial process. Any tasks $i \in I$ that have a start time less than t are added to the allocation group. Then, the allocation group prioritization is used to establish the processing order within the allocation group. This is an important preliminary step to the heuristic since task processing order can impact allocation performance. A complementary weight data set W is added with values for every w_i for all $i \in I$. The allocation group is ordered based on a weight parameter included to each of its member. The task prioritization weight algorithm adds additional verification mechanics to better prioritize the allocation group. It uses a latency verification mechanism to determine how many active fog nodes can service each task and are weighted in ascending order with the tasks with fewer options prioritized higher using the $FnWeight$ parameter. The algorithm also weighs tasks based on their age using the $AgeWeight$ parameter, prioritizing tasks that are newer for assignment to the network.

The allocation heuristic is used to assign tasks to the fog computing network. Each task in the allocation group is selected in its prioritized order and allocated to the fog computing network using a best fit heuristic where the fog node in range with the least available resources is selected. This algorithm was selected because it requires very little processing time and returned very good solutions when compared to the optimal model during the testing phase. Due to the modularity of the proposed framework, different heuristics can be used in this module without changing the overall process. Finally, the allocation group assignment will use the allocation heuristic output and perform the task assignment on the fog computing network.

3) Fog Node State Change Process: The fog node state change heuristic is the set of functions used to modify the state of fog nodes in the heuristic model. It is executed as the final process in each time window to allow the model to include the current time window in its decision process. As described in [1], fog nodes that are starting up in time window t will only be available to the fog network in time window $t + 1$. Any status changes that could benefit the current time window are not possible based on that constraint.

The fog node shutdown algorithm provides an overview of the operations needed to initiate the shutdown of a fog node. It aims to determine situations where the operating cost is higher than the revenue of its critical tasks and predicted demand for future tasks that could only be serviced by the candidate fog node. Tasks are generated from the active and completed task sets to simulate future tasks. The $P(Stop)$ threshold is a configured value to determine the minimum probability of a task being assigned to fog node j before it is believed to be valuable to power it off.

The fog node startup algorithm is the opposing process and is designed in the same architecture. Similar to the previous algorithm, it seeks to determine the probability that a task is requested in future time window that could not be serviced by the current fog node coverage. The resulting probability is compared with a configured $P(Start)$ threshold to determine whether a fog node should be started. However, the algorithm also tracks cases where the network should have turned on fog node j at time window t, but failed to do so with the predictive algorithm. In these cases, the heuristic model starts j at the end of time window t so it is available at time window $t + 1$ given the average duration of historic tasks is longer than 1 time window.

4) Calculate Profit Process: The calculate profit process implements the same optimization function as the mathematical model described in [1] and shown in Figure 2. At each time window, the process simply tracks the task assignment to calculate revenue and the fog node status operations to calculate the operating cost.

IV. Results

A. Simulation Environment

The simulation environment used to generate the results uses a Unix operating system with a 2.66 GHz Intel Core i7 processor, 8 GB (1067 MHz DDR3) of memory, and a storage of 256 GB (SSD).

The heuristic algorithm proposed in this paper was implemented with Matlab 2019b. To evaluate the quality of the proposed heuristic, we also implemented the theoretical model from [1] with CPLEX and used the IBM iLog CPLEX Optimization Studio version 12.10 (OPL) command line interface (CLI) to run it.

The input parameters used for both methods are listed in Table I. As can be seen, five different task profiles are available. For each task, the actual duration is generated using a normal distribution where the mean is equal to the average duration and the standard deviation is equal to $1/6$ of the duration. Similarly, we also have 5 different types of fog nodes with different hardware specifications.

B. Selecting the Parameters

In order to evaluate the performance of the heuristic model, the allocation group prioritization and the fog node state change parameters must be set. To achieve this, a series of tests were conducted to determine the optimal values for these parameters.

The value of the $FnWeight$ and $AgeWeight$ parameters were determined by setting all maintenance costs in the simulations to $null$ in order to eliminate all fog node status changes. Preliminary results demonstrated that $FnWeight$ had the highest impact to improve profit and was optimal when set at a $2:1$ ratio to $AgeWeight$.

The fog node status manipulation parameters $P(start)$ and $P(stop)$ were tested for various values in the $[0-1]$ range. We compared the relative profit $(Profit(H)/Profit(M)$ where H is the heuristic model and M is the mathematical model proposed in [1]) based on the values set to parameters. Our

TABLE I
INPUT PARAMETERS

Parameter	Value
Area (units)	100 x 100
Tasks & fog node locations	Generated randomly within the area
Task profile - requested from each task (max distance, vCPU, memory, avg duration)	Randomly selected among the following: {(10, 1, 1, 8), (15, 2, 1, 6), (30, 2, 2, 12), (20, 1, 2, 10), (40, 1, 1, 4)}
Fog types - available resource at each fog nodes (# of parallel tasks, # of vCPU, memory)	Randomly selected among the following: {(3, 6, 6), (2, 2, 2), (4, 5, 5), (1, 2, 2), (2, 3, 4)}
Initial state of fog nodes	Powered off
Maintenance, startup, and shutdown costs (in currency units)	0.1, 0.5, 0.2
Revenue (in currency units)	1

testing based on 10 networks with random task locations showed that the $P(start)$ parameter has little impact on the overall performance of the heuristic. On the other side, the $P(stop)$ parameter benefited from values of 0.2 and under. In higher values, it was too optimistic in shutting down fog nodes, only to have them needed in future states.

We also tested these parameters in networks with more predictable task locations where tasks are concentrated around x areas. As the location of tasks is more predictable, we believe the heuristic could perform better as we decrease the number of areas. To test this hypothesis, tasks are simulated around infinity (random), 10, 5 and 3 concentration areas within the network service area. This was used to simulate concentration areas in a typical urban scenario. For example, each centralized area represents densely populated areas with a majority of tasks, while other areas represent rural or sparsely populated areas.

Under this scenario, profit was consistently higher in tests where $P(start) = 0.4$ and $P(stop) = 0.2$. These parameters showed an increase correct predictive status changes. Tests with higher values for the parameters became too optimistic and led to an increase in incorrect status changes. Tests were processed with 200 tasks with varying numbers of concentration location coordinate to simulate task predictability and 20 fog nodes. The heuristic model performance over 30 tests showed consistent improved results as task location became more predictable. Figures 4 and 5 show the histogram and a normally generated distribution of tested events respectively. This illustrates an overall improvement in profit for cases where there were fewer task location concentration areas.

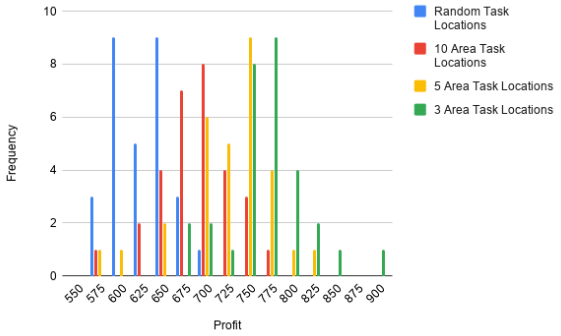

Fig. 4. Histogram profit distribution of 200 tasks and 20 fog nodes.

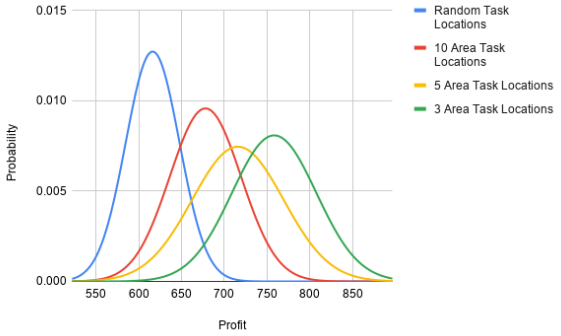

Fig. 5. Normal profit distribution of 200 tasks and 20 fog nodes.

C. Heuristic Model Performance Analysis

The heuristic model was processed and compared with the mathematical model for a variety of network sizes. The networks generated included 5 concentration areas within the service coverage grid.

The results, shown in Table II, show an average value for 5 tests at each network size with a 95% confidence interval. The first 2 columns represent the number of task ($|I|$) and the number of fog nodes ($|J|$) in the network area. The third column shows the average relative profit of the heuristic compared to the optimal model proposed in [1]. In the next three columns, the processing time of the theoretic model is presented with its minimum (min), median (med), and maximum (max) values to better illustrate the large variance in processing time. It is important to note that a time limit of 8,000 seconds was set to prevent some simulations to run over several days without improvement to the profit value. In some cases, the theoretical model was not able to find a solution within the time limit or ran out of memory. This is indicated with NS (No Solution) and OoM (Out of Memory) respectively. Finally, the last two columns show the average processing time of the heuristic and its average processing time per time window.

1) Processing Time: As shown and explained in [1], we can clearly see that the processing time of the theoretical model increases very fast. The quantity of variables and constraints to solve are directly related to the number of tasks and fog nodes which increases the complexity of the optimization problem. CPLEX was consistently unable to find a solution in

TABLE II
PERFORMANCE COMPARISON BETWEEN THEORETIC AND HEURISTIC MODELS

$\|I\|$	$\|J\|$	Average Relative Profit	Theoretic Model			Heuristic Model	
			Processing Time (s)			Average Processing Time (s)	Average Processing Time per Window (s)
			Min	Med	Max		
15	10	0.843 ± 0.030	32.5	50.1	102.7	6.7 ± 0.8	0.6 ± 0.0
	20	0.761 ± 0.045	229.8	8000.2	8012.9	11.6 ± 1.2	0.8 ± 0.1
	30	0.679 ± 0.045	8000.1	8000.1	8012.1	15.3 ± 1.6	1.1 ± 0.1
25	10	0.834 ± 0.067	115.5	196.6	2249.8	11.4 ± 0.5	0.7 ± 0.1
	20	0.809 ± 0.087	8000.1	8000.1	8008.6	19.5 ± 0.6	1.1 ± 0.1
	30	0.784 ± 0.106	8000.1	8000.2	8009.6	22.7 ± 3.4	1.4 ± 0.1
35	10	0.832 ± 0.049	290.6	564.0	8000.3	17.2 ± 2.2	0.6 ± 0.1
	20	0.807 ± 0.115	8000.2	8000.2	8000.2	24.9 ± 1.8	1.1 ± 0.1
	30	0.795 ± 0.131	8000.1	8000.3	8000.4	38.4 ± 4.2	1.5 ± 0.1
45	10	0.820 ± 0.062	689.6	8000.1	8000.6	21.1 ± 2.3	0.8 ± 0.1
	20	0.795 ± 0.100	8000.1	8000.2	8000.3	36.3 ± 4.1	1.1 ± 0.1
	30	0.811 ± 0.162	8000.2	8000.4	8000.4	47.2 ± 4.0	1.7 ± 0.2
55	10	0.840 ± 0.064	38.0	8000.1	8000.2	27.3 ± 2.6	0.8 ± 0.1
	20	0.808 ± 0.096	8000.3	8000.3	8010.1	42.3 ± 3.3	1.2 ± 0.1
	30	NS	OoM	OoM	OoM	60.4 ± 4.0	1.7 ± 0.2
65	10	0.896 ± 0.124	1477.3	8000.2	8010.4	57.0 ± 8.2	1.5 ± 0.1
	20	NS	OoM	OoM	OoM	98.6 ± 7.3	2.6 ± 0.2
	30	NS	OoM	OoM	OoM	132.1 ± 17.8	3.5 ± 0.3

larger simulations. In fact, several tests were made for larger networks, each resulting in an out of memory error.

The heuristic model processing time was divided in the full processing time and the processing time per time window. The most important value to consider for the heuristic model is the processing time per time window since it must remain under the time window size (2 seconds in this case) for the model to complete its decisions before the next time window begins.

The processing time of the heuristic model increases primarily based on the number of fog nodes for the network to manage and does not increase significantly based on the task set size. The task set size is never used by the heuristic model since it only considers active tasks in its assignment.

2) Relative Profit: The values for relative profit fluctuations were uncorrelated to the network size with the exception of networks with small number of tasks and large number of fog nodes (ex: 15 tasks, 30 fog nodes). The relative profit for these networks was consistently worse than other networks. The reasons for this behaviour is based on the difficulty of the heuristic model to predict the correct fog node state when an abundance of choices are offered. The values for the relative profit are mostly contained within the 79% to 89% range when removing simulations where the number of fog nodes exceeds the number of tasks. After evaluating the difference in profit between the heuristic and the optimal model, we were able to categorize them with the following categories.

- Lack of information in the heuristic model: due to its intent on running in real-time rather than on historical data as with the theoretical model, the heuristic only has partial information from the incoming tasks. Therefore, it has to guess several decisions. This is the largest contributor in the heuristic model making sub-optimal decisions but this is how a real-time algorithm should operate.
- Incorrect fog node status changes in early time windows: The early time windows inherently have less historical knowledge of tasks than later time windows. As such, the list of simulated tasks will be less representative than later time windows when the model has been exposed to more task elements. Therefore, there are often unnecessary fog nodes that are turned on due to this issue.
- Differences for fog node status change decisions at the end of the simulation: The heuristic model does not consider the number of remaining tasks within a simulation. The primary reason is that a practical utilization of this application wouldn't technically have an end. Tasks are expected to be constantly requested from the network although their frequency are likely to change over time. For this reason, the heuristic model was not designed to consider the end of a simulation and makes decisions accordingly although the theoretic model does so.
- Challenges to shut down fog nodes: Shutting down fog nodes is more challenging to replicate between the theoretic and heuristic models since they heavily rely on knowledge of future events to configure optimally. The heuristic model relies on a given fog node to not have tasks assigned to it (or tasks that can be transferred to another fog node) in order to turn it off. It also uses a series of simulated tasks to predict future tasks and proceed to a quicker shutdown. However, there are differences between both of these routines and the theoretic solution.

V. CONCLUSION AND FUTURE WORK

The heuristic model offers service providers an algorithm to allocate tasks in a fog computing network in real-time. Manipulating the task order at each time window proved an essential step to simplifying the assignment heuristic and provided optimal results in nearly all cases when removing other factors from the simulations. Modifying the status of fog nodes proved a more complex operation since it can not rely on knowledge of the future to make perfect decisions. Instead, it used a probabilistic approach using the current and historic demand on the network to predict which fog nodes required

a status change to improve the state of the fog computing network.

The heuristic model parameters were tested and ideal values were found for the type of networks tested. However, a service provider utilizing this technique is recommended to evaluate its historic data and determine the values that are ideal for a given practical application. The heuristic model was able to achieve results between 78 to 88% of the theoretical model for various network sizes. There was also a direct correlation between predictability of the input data set and the relative proximity to the upper bound. This relationship is logical since the status change operations of fog nodes rely on the relevance of historic to future input data sets.

In terms of future work, here are some ideas we believe are worth pursuing.

- Task-based billing: The heuristic model design presented in this paper assigns revenue to a service provider at each time window. This simulates a volume-based billing model that would be implemented for charging customers. However, task-based billing process provides an additional dimension of decisions to the problem. The implementation of service levels agreements to the dynamic allocation of tasks increases the importance of the de-allocation group process, notably with the addition of predictive task de-allocation.
- Routing between fog nodes: This paper presents a model where fog nodes have no interactions between themselves. A task being presented to the network is in range of n fog nodes based on the maximum latency demanded. Once a fog node is at capacity, it can't service a new task. However, a modification to the model could include the routing to another fog node using a low latency back-haul connection which would greatly increase connectivity options for end users.
- Advanced prediction mechanisms: In the current model, task duration is simply predicted by calculating the average of all completed tasks. We believe that the advance de-allocation procedure could benefit from more advanced prediction mechanisms based on machine learning to enhance performance.
- Detecting areas of poor coverage: The dynamic task allocation processes could assist in detecting recurring areas with poor service coverage. This information would be fed to network planners as they increase fog node capacity and physical fog resources. The dynamic task allocation algorithm presented in this paper optimizes the current state of the network, but service providers also seek areas to improve network layout.

REFERENCES

[1] J. Daigneault and M. St-Hilaire, "Profit maximization model for the task assignment problem in 2-tier fog/cloud network environments," *IEEE Networking letters*, vol. 3, no. 1, pp. 19–22, 2021.

[2] L. F. Bittencourt, M. M. Lopes, I. Petri, and O. F. Rana, "Towards virtual machine migration in fog computing," in *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pp. 1–8, IEEE, 2015.

[3] F. Haider, D. Zhang, M. St-Hilaire, and C. Makaya, "On the planning and design problem of fog computing networks," *IEEE Transactions on Cloud Computing*, 2018.

[4] D. Zhang, F. Haider, M. St-Hilaire, and C. Makaya, "Model and algorithms for the planning of fog computing networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3873–3884, 2019.

[5] J. Ejarque, A. Micsik, R. Sirvent, P. Pallinger, L. Kovacs, and R. M. Badia, "Semantic resource allocation with historical data based predictions," in *The First International Conference on Cloud Computing, GRIDs, and Virtualization*, pp. 104–109, IARIA, 2010.

[6] S. Schneider, F. Seifert, and A. Sunyaev, "Market potential analysis and branch network planning: application in a german retail bank," in *2014 47th Hawaii International Conference on System Sciences*, pp. 1122–1131, IEEE, 2014.

[7] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE transactions on parallel and distributed systems*, vol. 24, no. 6, pp. 1107–1117, 2012.

[8] W. Wei, X. Fan, H. Song, X. Fan, and J. Yang, "Imperfect information dynamic stackelberg game based resource allocation using hidden markov for cloud computing," *IEEE transactions on services computing*, vol. 11, no. 1, pp. 78–89, 2016.

[9] S. Kianpisheh, S. Jalili, and N. Charkari, "Predicting job wait time in grid environment by applying machine learning methods on historical information," *Int. J. Grid Distrib. Comput*, vol. 5, no. 3, pp. 11–22, 2012.

[10] W. Smith, I. Foster, and V. Taylor, "Predicting application run times using historical information," in *Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 122–142, Springer, 1998.

[11] J. M. Ramírez-Alcaraz, A. Tchernykh, R. Yahyapour, U. Schwiegelshohn, A. Quezada-Pina, J. L. González-García, and A. Hirales-Carbajal, "Job allocation strategies with user run time estimates for online scheduling in hierarchical grids," *Journal of Grid Computing*, vol. 9, no. 1, pp. 95–116, 2011.

[12] A. Shakarami, H. Shakarami, M. Ghobaei-Arani, E. Nikougoftar, and M. Faraji-Mehmandar, "Resource provisioning in edge/fog computing: A comprehensive and systematic review," *Journal of Systems Architecture*, vol. 122, p. 102362, 2022.

[13] J.-F. Tsai, C.-H. Huang, and M.-H. Lin, "An optimal task assignment strategy in cloud-fog computing environment," *Applied Sciences*, vol. 11, no. 4, 2021.

[14] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, and C. S. Hong, "A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing," in *2015 International Conference on Information Networking (ICOIN)*, pp. 324–329, IEEE, 2015.

[15] P. Lai, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Edge user allocation with dynamic quality of service," in *International Conference on Service-Oriented Computing*, pp. 86–101, Springer, 2019.

[16] J. Daigneault, "Model and algorithm for real-time resource allocation in fog computing networks," Master's thesis, Carleton University, 2020.