Heuristic-Based Proactive Service Migration Induced by Dynamic Computation Load in Edge Computing

Amr M. Zaki, Sara A. Elsayed, Khalid Elgazzar, Hossam S. Hassanein
School of Computing, Queen's University, Kingston, ON, Canada
Email: amr.zaki@queensu.ca, selsayed@cs.queensu.ca, khalid.elgazzar@ontariotechu.ca, hossam@cs.queensu.ca

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10000913

***Abstract*—Edge Computing (EC) has paved the way toward the realization of the Internet of Things (IoT). This can be attributed to the ability of EC to bring the computational resources within close proximity to end-users, which significantly improves the response time. However, performance gain in EC can be compromised by service interruptions triggered by various dynamic changes. Consequently, reliable service migration is crucial in EC. However, most service migration schemes either fail to consider the profound impact of the dynamic computation load on service continuity or provide impractical and time-inefficient solutions based on optimization techniques. This paper proposes the Heuristic-based Load-induced Proactive Migration (HLPM) scheme. HLPM incorporates a Finite State Machine (FSM) to model the dynamic computation load. It then makes proactive migration decisions based on the underlying transition probabilities. The proactive migration problem is solved using the MTHG heuristic algorithm. Performance evaluation shows that HLPM produces a significant decrease of up to 97% in migration decision latency compared to conventional optimization techniques. Furthermore, the performance gap of HLPM with respect to the optimal migration solution is just 1.44% latency and 3.89% number of migrations.**

***Index Terms*—Edge computing, migration decision latency, MTHG heuristic algorithm, proactive migration,**

I. Introduction

With the substantial growth of the Internet of Things (IoT), it is expected that by 2023 IoT devices will account for 50% (14.7 billion) of all globally networked devices [1]. IoT devices are expected to foster a broad range of time-sensitive services, including healthcare, industrial IoT (IIoT) and infotainment. [2].

The aforementioned time-sensitive applications cannot be adequately served by conventional cloud computing paradigms [3] as they often rely on geographically distant data centers, which make them susceptible to service interruptions, causing significant delay [4] [5]. Edge computing has emerged as a ubiquitous computing paradigm to alleviate such challenges. This can be attributed to the close proximity of edge computing servers to IoT devices, thus significantly reducing the delay [4].

The performance gain achieved by edge computing tends to be compromised by dynamic changes, typically attributed to users' mobility and the restricted coverage of edge servers [6]. Such changes cause service disruptions and drastic network performance deterioration, profoundly affecting the delay [7]. Migration policies must be implemented to manage the migration of the affected service to the closest node (i.e., edge server) capable of accommodating the request. Migration policies should avoid performing too many or too few migrations. This is because frequent migrations can lead to service downtime. In contrast, infrequent migrations can trigger increased delay because of continuing to perform the service too far away from the user [8]. Accordingly, efficient service migration schemes need to thoroughly determine when to migrate and to which edge node.

Service migration schemes can be categorized into reactive and proactive schemes. In reactive schemes, migration occurs in response to a triggering event [7]. In contrast, in proactive schemes, the need for migration is estimated in advance, and migration decisions are made beforehand [8]. Accordingly, proactive migration can significantly reduce delays compared to reactive migration [6]. Most existing proactive migration schemes focus on addressing dynamic changes resulting from the mobility of users. In particular, such schemes tend to predict the future location of users and incorporate mobility predictions into the migration decision. Unlike users' mobility, the dynamic computation load tends to be overlooked in most service migration schemes.

Heavy computation loads can dramatically reduce the compute power of an edge node, which can profoundly affect its ability to serve newly migrated services. Because such computation load is highly dynamic, it is imperative for service migration policies to account for this dynamicity. Recently, optimization techniques have been utilized to consider the dynamic computation load [9]. Such optimization techniques are mostly based on branch-and-bound algorithms to find the exact optimal assignments [10]. However, these optimization techniques are both time-inefficient and impractical, as they require a considerable amount of time to reach an optimal solution. This renders them unsuitable for the time-sensitive nature of modern IoT applications. This problem is exacerbated even further when extending the edge computing environment to work with a multitude of edge nodes and IoT devices. As the number of edge nodes and devices increases, the complexity of the optimization problem increases, which reflects in the time needed to solve it optimally [11]. Consequently, the need to study heuristic methods, capable of reaching near optimal migration decisions under dynamic

computation load while maintaining time efficiency arises. This aspect has been overlooked in existing schemes.

In this paper, we propose the Heuristic-based Load-induced Proactive Migration (HLPM) scheme. HLPM employs the use of a Finite State Machine (FSM) to model the dynamic computation load. HLPM relies on the underlying dynamic changes in the computation load of edge nodes to make proactive migration decisions to the nearest edge node that has the highest probability of having the lowest load. HLPM solves the service migration problem under dynamic computation load using the MTHG heuristic approach. A greedy heuristic has also been proposed. The proposed heuristic schemes are compared to conventional optimization techniques [9]. Performance evaluations show that HLPM significantly reduces up to 97% migration decision latency compared to conventional optimization techniques. HLPM has also outperformed the greedy heuristic while maintaining comparable migration decision latency.

The remainder of the paper is organized as follows. Section II highlights some of the related work. Section III provides a detailed description of HLPM and presents the underlying greedy and MTHG heuristics. Section IV discusses the performance evaluation and simulation results. In section VI, we draw our conclusions and outline future directions.

II. Related Work

Multiple recent works have thoroughly studied service migration in edge computing [6]. Service migration is associated with various challenging issues that can decrease the quality of service (QoS) provided to end users. These issues include user mobility and the limited coverage of edge servers [7].

Most existing migration schemes focus on the mobility of users and consider it the predominant triggering cause of migration. Such schemes can be categorized into reactive and proactive schemes. Migration is initiated in reactive schemes after a handoff [7], thus keeping the service close to the user. However, reactive schemes can cause prolonged delays. This delay occurs after the handoff process is concluded; the user utilizes a new access point to access the corresponding source edge node, which leads to more than one communication hop to access the service [12]. In contrast, proactive schemes enable migration to be initiated before the handoff process, which allows the service to be at the destination node once the handoff process is concluded [12].

Most proactive migration schemes incorporate mobility predictions into the migration decision-making process. In this context, numerous prediction models have been proposed. In [13], Markovian-based approaches are considered, whereas deep learning models, such as Long-Short Term Memory (LSTM), are considered in [14]. Despite the demonstrated leverage of designing proactive migration schemes based on mobility predictions, such schemes tend to overlook the impact of the dynamic computation load of edge nodes [9].

Recent work by authors in [9] introduced a proactive service migration scheme for a dynamic-load edge computing environment. Their scheme optimizes the service migration placement strategy only on nodes with a higher probability of being more computationally capable than other nodes. They formulate the problem as a generalized assignment problem (GAP), which is solved periodically to decide whether migration is needed. If the migration is needed, decide which node to migrate to is the best one. The formulated model optimized the total average delay throughout all mobile devices. They then used an IBM CPLEX solver [15] to solve the problem and find the optimal solution. However, optimization techniques such as branch-and-bound tend to be both time-inefficient and impractical for industry use. Furthermore, authors in [11] found optimization solvers were incapable of solving problems for large scenarios with a multitude of users and edge nodes.

Encouraged by the findings of [11], we extend on the problem presented in [9] to investigate the workings of different heuristic approaches in the context of service migration in a dynamic-load edge computing environment to provide practical time-efficient resource allocation algorithms. First, we introduce two heuristic approaches: a greedy approach and a scheme that uses a heuristic method called MTHG [10] known to solve the GAP problem. We then compare the results obtained by MTHG with that of the optimization solver and the greedy heuristic.

III. Heuristic-based Load-induced Proactive Migration (HLPM)

This section provides a detailed description of the system model, the migration problem under dynamic computation load, and the proposed heuristic schemes HLPM-Greedy and HLPM-MTHG.

A. System Model and Overview

Consider a set of services for the mobile devices D= $\{d_1$, d_2,..., $D\}$ to be proactively migrated to a set of edge nodes N= $\{n_1, n_2, \ldots, N\}$ at any given time. Each service requires a certain computation load, which is the number of instructions that must be executed (measured in the number of instructions executed batch size β). Each service should be allowed to run on the closest most computationally capable edge node. The dynamic load is modeled using a FSM. The computational capability of each edge node n_i can be in one of three possible states (expressed in terms of million instructions per second (MIPS) and denoted ν_{si}). These three states are high, medium, and low. Each edge node is assumed to build its local model that captures its transition probabilities (ψ_{si}) from moving from one computational capability state to another. This model is then sent to a centralized entity that runs the migration decision. These probabilities are used to proactively migrate services to edge nodes that are best capable of serving the migrated service. The migration decision is performed periodically (each t_r seconds) by the centralized entity that tries to optimize the total average delay of all the participating users in the environment. In this work, we consider vehicle-to-network services (V2N), which 3GPP mandates to have very short deadlines, far less than (t_r), and shorter than one second.

In addition, V2N services have small instance data sizes (θ); besides this, the VM migration strategy typically prevents the migration process from taking too long [16]. Thus, this work does not focus on optimizing the migration time of services between edge nodes.

We aim to maximize the difference between the expected delays of both the end user and the edge node currently serving the mobile device and between the end user and a potential next edge node to which the service can migrate. If the latter's expected delay is lower than that of the former, migration should be triggered. The problem is solved periodically (each $t_r = 1$sec) at a central entity, enabling the system to identify when and where to migrate.

The total delay between the end user and any edge node, denoted t_{dn}, is given by Eq. 1, where c_{dn} is the computation delay, and z_{dn} is the propagation delay. The computation delay is the time taken by edge node n to compute the task of mobile d, whereas the propagation delay is the time taken for the service to reach the user. Transmission delay is ignored, as the data rate is assumed to be the same for all devices across all nodes.

$$t_{dn} = c_{dn} + z_{dn} \tag{1}$$

The transition probabilities ψ_{si} are used to compute the expected computational delay of the service of mobile device d on the edge node n, which is given by Eq. 2.

$$c_{dn} = (\beta/\nu_{s1}) \times \psi_{s1} + (\beta/\nu_{s2}) \times \psi_{s2} \tag{2}$$

The main objective is to optimize the expected differential delay x_{dn} between the total delay t_{d1} of the mobile device d and the current occupied edge node $(n = 1)$, and the total delay t_{d2} of the mobile device d and a new node $(n = 2)$, as given by Eq. 3.

$$x_{dn} = t_{d1} - t_{d2} \tag{3}$$

Thus, the objective is given by Eq. 4.

$$\max \sum_{d=0}^{D} \sum_{n=0}^{N} x_{dn} \tag{4}$$

Assignment restrictions are considered to ensure that each edge node can have at most one migrated service, and that each service should be allocated to only one edge node. The assigned edge node is either a new node to which the service migrates or the current node where the service resides (if no migration occurs).

Memory and energy constraints ensure that the memory requirements (m_d) and the migrated service's energy consumption (e_d) do not exceed certain predefined thresholds, denoted M_n and E_N, respectively. This energy constraint is only enforced when migration occurs ($X_{dn} \neq 0$), as in the case of no migration, the current node will be selected regardless of the energy constraint.

A threshold constraint is needed to avoid migration to a new node when the value of x_{dn} is smaller than a predefined threshold (τ). This case was found to occur when two nodes of similar computation load are close to one another; this would cause an oscillation effect, making the model oscillate in the assignment between these two edge nodes. This constraint is only enforced for the case when $x_{dn} \neq 0$ (i.e., a migration is needed); otherwise the current edge node is selected regardless of the threshold.

This optimization problem is a modification of the GAP problem [17]. This problem is known to be NP-hard [17], requiring optimization techniques to reach an optimum solution. In [9], the authors used IBM CPLEX [15] to periodically solve the problem of knowing when and where to migrate. However, optimization solvers are an impractical approach due to their time inefficiency. Encouraged by this, we propose multiple heuristics capable of reaching a near-optimum assignment in a time-efficient practical manner.

B. HLPM-Greedy

The first heuristic presented in this work is a greedy approach, detailed in Algorithm 1. This heuristic is used as a baseline, to compare other heuristics in terms of the optimality of the found solution and the time efficiency of the proposed heuristic.

Algorithm 1 : HLPM-Greedy

Input: Devices (D), Nodes (N), total delay (X), energy constraints (E), memory constraints (M), oscillation threshold (τ)
Output: nodeAssignments

1: $nodeAssignments = \{\}$
2: **for** $d \in \mathcal{D}evices$ **do**
3: $maxGain$ = 0
4: $chosenNode$ = -1
5: **for** $n \in \mathcal{N}odes$ **do**
6: **if** $x_{dn} > maxGain$ AND $x_{dn} >= \tau$ AND
7: not used by other devices AND
8: not assigned in previous loops to other devices AND
9: within energy and memory constraints **then**
10: $maxGain \leftarrow x_{dn}$
11: $chosenNode \leftarrow n$
12: **end if**
13: **end for**
14: $nodeAssignments[d] \leftarrow chosenNode$
15: **end for**

In HLPM-Greedy, for each mobile device d, all edge nodes N are searched to find the node n that has the highest expected differential delay. This search is done while ensuring the following: 1. the expected differential delay x_{dn} is greater than a predefined threshold τ (line 6), 2. each edge node is assigned to only one device (Lines 7 and 8), and 3. abidance to the required energy and memory constraints on the selected edge node (Line 9).

Given the simplicity of the greedy approach, it can provide a more time efficient solution than conventional optimization techniques, such as branch and bound. However, greedy heuristics can easily be trapped in local minima, increasing the gap to the optimal solution. To alleviate this risk, another heuristic, namely HLPM-MTHG, is introduced.

C. HLPM-MTHG

HLPM-MTHG uses the MTHG heuristic [10] [18] , an effective polynomial-time heuristic capable of solving the GAP optimization problems. It is a combination of both constructive and local search heuristics.

Algorithm 2 : HLPM-MTHG

Input: Devices (D), Nodes (N), total delay (X), energy constraints (E), memory constraints (M), oscillation threshold (τ)

Output: nodeAssignments

1: $nodeAssignments = \{\}$
2: $cost = \{\}\{\}$
3: // Enforce constraints using masks
4: **for** $n \in \mathcal{N}odes$ **do**
5: **for** $d \in \mathcal{D}evices$ **do**
6: **if** ($x_{dn} >= \tau$ AND
7: within energy and memory constraints) OR
8: $x_{dn} == 0$ **then**
9: $cost[n][d] \leftarrow x_{dn} * -1$
10: **else**
11: $cost[n][d] \leftarrow \infty$
12: **end if**
13: **end for**
14: **end for**
15: // MTHG algorithm
16: // Step1: Constructive search
17: Compute the desirability f_{dn} metric of all the assignments
18: **while** node not assigned **do**
19: Find the node n ($chosenNode$) with the maximum difference between the largest and second largest assignment desirability f_{dn} (greatest regret) that can be feasibly assigned
20: $nodeAssignments[d] \leftarrow chosenNode$
21: **end while**
22: // Step2: Local search
23: **for** $n \in \mathcal{N}odes$ **do**
24: $C_{dn} \leftarrow min(f_{dn})$
25: **if**
26: ($C_{dn} <$ regret of assigning this node) **then**
27: $nodeAssignments[d] \leftarrow newChosenNode$
28: **end if**
29: **end for**

As detailed in Algorithm 2, HLPM-MTHG enforces the previously mentioned memory, energy, and threshold constraints. This is done by masking the x_{dn} values that lie outside these constraints by setting their values to infinity (Lines 4–14), to not be considered by the MTHG heuristic. This masking step only occurs for the case when migration is considered (i.e., $x_{dn} \neq 0$), since otherwise, the current edge node would be selected regardless of these constraints.

In order to ensure that each edge node has at most one migrated service, and that each service is allocated to only one edge node, we set the resource consumption of each device to 1 so that each device can use at most one node. We also set the resource capacity of each edge node to 1 so that it can host at most one device.

After setting the constraints of the optimization problem, the MTHG heuristic is applied. In its first phase (Lines 15–21), MTHG calculates f_{dn}, which measures the desirability of assigning the device d to a certain edge node n (benefit). By iteratively considering all unassigned devices, the scheme assigns the device to the edge node n that has the maximum difference between the largest and second largest f_{dn} (regret). In the second phase (Lines 22–29), MTHG improves on the solution found in the first phase through local search by iteratively analyzing a subset of the search space close to the found solution (within its neighborhood).

IV. Performance Evaluation

In this section, the performance of our heuristic proactive schemes HLPM-MTHG and HLPM-Greedy are compared with the Dynamic Load-based Proactive Migration (DLPM) scheme [9], as well as the Reactive Greedy Lowest Latency (RGLL) scheme [12]. Note that DLPM uses conventional optimization techniques to solve the proactive migration problem, and RGLL uses a common baseline reactive migration approach.

We use the following performance metrics: 1. the average delay experienced by the selected mobile devices starting from the time a request is sent until a response is received, 2. the average number of migrations triggered by the system, and 3. the migration decision latency, which is the run time that the scheme takes to make the migration decisions. Evaluating the migration decision latency of the proactive schemes is important for assessing their time efficiency and practicality for use in the industrial sector.

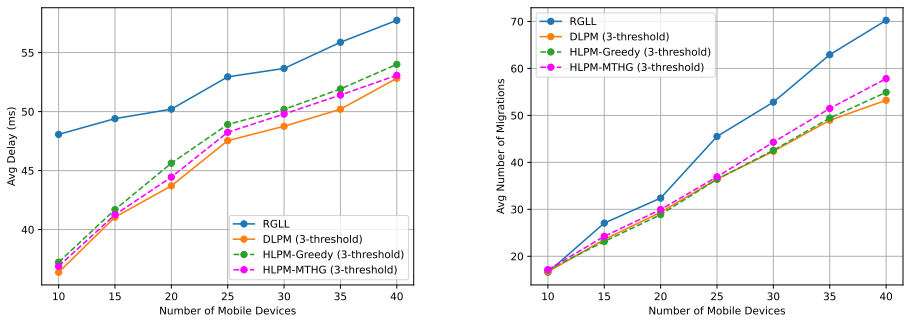
A. Simulation Setup

Simulations are conducted using the MobFogSim simulator [12], a Java-based simulator used to simulate service migration between mobile devices and edge nodes. Realistic mobility traces are employed by incorporating the Luxembourg SUMO Traffic dataset (LuST) [19]. These traces constitute the mobility patterns of buses in Luxembourg City, traveling an average speed of 22.3 km/h on routes averaging 26.44 min. Simulations are conducted over an area of 16kmx16km, which includes evenly distributed 144 edge nodes each with a communication range of 1km. The number of mobile devices varies from 10 to 40. The batch size of services is set to be 20k million instructions. The data rate of all the services of the participating devices is fixed to the same value. The simulation period is set to 10 minutes (i.e., t_s = 600 seconds). Each node changes its computation load every 30 seconds (i.e., $t_f = 30$ seconds) according to the FSM. The oscillating threshold τ is set to 3. Table I summarizes the simulation parameters.

B. Simulation Results and Analysis

In our experiments, we evaluate the performance of HLPM-MTHG, HLPM-Greedy, DLPM, and RGLL over a varying number of users. Simulation results are presented at a confidence level = 95%.

Figure 1a depicts the performance of HLPM-MTHG, HLPM-Greedy, DLPM, and RGLL in terms of the average delay. Note that as the number of requesting devices increases, the system becomes more congested, which causes the average delay of mobile devices to increase. This delay is due to the



(a) Average delay

(b) Average number of migrations

Fig. 1: Performance results of HLPM-MTHG, HLPM-Greedy, DLPM, and RGLL over varying numbers of users.

TABLE I: List Of Simulation Parameters

Parameters	**Value**
Simulation Time (t_s)	10 mins
Frequency of MIPS State Change (t_f)	30 secs
Frequency of Running the Decision Making Algorithm (t_r)	1 sec
Number of Edge Nodes	144
Batch Size (β)	20k million instructions
Number of Users	10,15,20,30,35,40
Oscillating Threshold (τ)	3
State A (normal MIPS)	3234 MIPS
State B (low MIPS)	646 MIPS
State C (high MIPS)	4851 MIPS
Service Instance Data Size (θ)	100 KB

contention on system resources as the number of requests increases. Thus, making it harder to find a better edge node to migrate to since it might already be serving other requests. The reactive approach RGLL yields the highest delay than the proactive schemes HLPM-MTHG, HLPM-Greedy, and DLPM.

Also as depicted in the Figure. 1a, all the proactive decision making schemes that take the probabilities of the CPU capabilities change for the edge node into consideration, yields much better average delay than that of the RGLL scheme. This is because in contrast to RGLL, the proactive schemes HLPM-MTHG, HLPM-Greedy, and DLPM account for the dynamic computational capability of edge servers to make proactive migration decisions beforehand. This capability facilitates selecting the edge nodes with a higher probability of having higher compute power, resulting in faster processing of the migrated services. DLPM renders the lowest delay among all schemes, since it provides the optimal solution. Both HLPM-Greedy and HLPM-MTHG approach the optimal solution. However, HLPM-MTHG is closer to the optimal solution (i.e., DLPM) than HLPM-Greedy. In particular, the performance gap of HLPM-Greedy compared to DLPM is 2.82%, while that of HLPM-MTHG is 1.44%. This behavior can be observed when varying the number of users because, HLPM-Greedy has a higher chance of being trapped in a local minima, since it loops through the devices sequentially to optimize the average delay of each device on its own before considering the others. In contrast, HLPM-MTHG strives to optimize the delay for all the devices at once without considering each one sequentially.

The same experiment is conducted in terms of the average number of migrations. As depicted in Figure 1b, the number of migrations increases in all schemes as the number of users increases. RGLL yields the highest number of migrations among all the schemes because, in RGLL, migrations are triggered by the mobility of users. In particular, when a mobile user moves away from an edge node, service migration is triggered. Note that users' mobility change at a faster rate than the computational capability, increasing the rate at which migrations are triggered in RGLL compared to the proactive schemes that consider the dynamic computational capability. DLPM exhibits the lowest number of migrations, whereas both HLPM-MTHG and HLPM-Greedy closely approach the optimal solution (i.e., DLPM). The performance gap between HLPM-Greedy and HLPM-MTHG compared to DLPM is 0.49% and 3.89%, respectively. Thus, HLPM-Greedy is closer to achieving the optimal number of migrations. Unlike HLPM-Greedy, HLPM-MTHG triggers better assignments to prompt a lower delay by solving the migration problem for mobile devices in parallel rather than in sequence, causing more migrations in HLPM-MTHG.

We evaluate the proactive schemes HLPM-MTHG, HLPM-Greedy, and DLPM in terms of the migration decision latency over a varying number of users. As depicted in Figure 2, the decision latency increases as the number of users increases. This increase in decision latency is due to the increase in the complexity of the migration problem. Note that the heuristic proactive schemes HLPM-Greedy and HLMP-MTHG yield significant reductions, of up to 99.6% and 97.1%, respectively, in the time taken to solve the migration problem compared to the conventional optimization technique adopted by DLPM. Furthermore, HLPM-Greedy demonstrates the lowest time due to its simplicity, whereas HLPM-MTHG takes slightly more time to solve the migration problem.

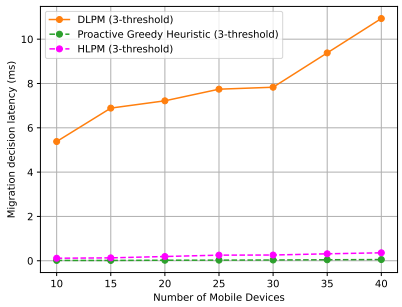


Fig. 2: Average migration latency of HLPM-MTHG, HLPM-Greedy, and DLPM over varying number of users.

V. Conclusion and Future Work

In this work, we study a typically overlooked aspect of service migration, accounting for the dynamic computation load of edge nodes. We have developed multiple heuristics to efficiently solve the assignment problem of the service migration for a dynamic-load edge computing. We propose both a greedy heuristic and a heuristic method that uses the MTHG scheme. Quantitative analysis demonstrates the ability of the proposed heuristics to provide time-efficient and accurate decision-making that adapts to large scenarios of multiple mobile devices with edge nodes having dynamic computation capability. In contrast to defying a custom heuristic to solve our problem, we focus on studying generalized heuristic approaches that can easily be integrated to solve similar resource allocation tasks. Performance evaluation has shown that using the HLPM-MTHG heuristic achieves significant reduction in the decision making time by 97% compared to using an optimization solver, without compromising the optimality of the assignments. We have shown that HLPM-MTHG achieves better results in terms of lower average delay than a simple greedy heuristic, with a non-significant increase in the decision time and the average number of migrations. Our work proves the impact of integrating generic heuristic methods to solve assignment problems in edge computing.

In the future, we plan to propose various prediction models to estimate the dynamic computation load and incorporate such predictions in the proactive migration decision.

Acknowledgment

This research is supported by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant number ALLRP 549919-20.

References

[1] "Cisco annual internet report (2018–2023) white paper," https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html, 2020.

[2] J. Lee, D. Kim, and J. Lee, "Zone-based multi-access edge computing scheme for user device mobility management," *Applied Sciences*, vol. 9, no. 11, p. 2308, 2019.

[3] G. Fortino, C. Savaglio, C. E. Palau, J. S. de Puga, M. Ganzha, M. Paprzycki, M. Montesinos, A. Liotta, and M. Llop, "Towards multi-layer interoperability of heterogeneous iot platforms: The inter-iot approach," in *Integration, interconnection, and interoperability of IoT systems*. Springer, 2018, pp. 199–232.

[4] 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)*, 2015, pp. 1–8.

[5] C. Stergiou, K. E. Psannis, B.-G. Kim, and B. Gupta, "Secure integration of iot and cloud computing," *Future Generation Computer Systems*, vol. 78, pp. 964–975, 2018.

[6] Z. Rejiba, X. Masip-Bruin, and E. Marín-Tordera, "A survey on mobility-induced service migration in the fog, edge, and related computing paradigms," *ACM Comput. Surv.*, vol. 52, no. 5, Sep. 2019. [Online]. Available: https://doi.org/10.1145/3326540

[7] P. Bellavista, A. Zanni, and M. Solimando, "A migration-enhanced edge computing support for mobile devices in hostile environments," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2017, pp. 957–962.

[8] D. Gonçalves, K. Velasquez, M. Curado, L. Bittencourt, and E. Madeira, "Proactive virtual machine migration in fog environments," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, 2018, pp. 00 742–00 745.

[9] A. M. Zaki and S. Sorour, *Proactive Migration for Dynamic Computation Load in Edge Computing*. ieee international conference on communications, 2022.

[10] S. MARTELLO, "An algorithm for the generalized assignment problem," *Operational Research*, pp. 589–603, 1981.

[11] N. Godinho, H. Silva, M. Curado, and L. Paquete, "Energy and latency-aware resource reconfiguration in fog environments," in *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, 2020, pp. 1–8.

[12] C. Puliafito, D. M. Gonçalves, M. M. Lopes, L. L. Martins, E. Madeira, E. Mingozzi, O. Rana, and L. F. Bittencourt, "Mobfogsim: Simulation of mobility and migration for fog computing," *Simulation Modelling Practice and Theory*, vol. 101, p. 102062, 2020, modeling and Simulation of Fog Computing. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X19301935

[13] N. Abani, T. Braun, and M. Gerla, "Proactive caching with mobility prediction under uncertainty in information-centric networks," pp. 88–97, 09 2017.

[14] H. Khelifi, S. Luo, B. Nour, A. Sellami, H. Moungla, and F. Naït-Abdesselam, "An optimized proactive caching scheme based on mobility prediction for vehicular networks," pp. 1–6, 2018.

[15] I. I. Cplex, "V12. 1: User's manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[16] X. Yu, M. Guan, M. Liao, and X. Fan, "Pre-migration of vehicle to network services based on priority in mobile edge computing," *IEEE Access*, vol. 7, pp. 3722–3730, 2019.

[17] D. G. Cattrysse and L. N. Van Wassenhove, "A survey of algorithms for the generalized assignment problem," *European Journal of Operational Research*, vol. 60, no. 3, pp. 260–272, 1992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/037722179290077M

[18] "Generalized assignment solver," https://github.com/fontanf/generalizedassignmentsolver, Dec. 2021.

[19] L. Codecá, R. Frank, S. Faye, and T. Engel, "Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 2, pp. 52–63, 2017.