METODE PEMILIHAN KEPUTUSAN OFFLOADING UNTUK EFISIENSI ENERGI DAN RENDAH LATENSI PADA LINGKUNGAN SIMUASI YANG HETEROGEN

Achmadaniar Anindya Rhosady[1)], Radityo Anggoro [2)]
[1, 2)]Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember
Jalan Teknik Kimia, ITS, Sukolilo, Surabaya, 60111
e-mail: achmadaniar.19051@mhs.its.ac.id[1)], onggo@if.its.ac.id[2)]

ABSTRAK

Mobile Cloud Computing (MCC) adalah salah satu teknologi yang dapat mengatasi permasalahan komputasi tinggi dan keterbatasan sumber daya yang dimiliki oleh mobile device. Tetapi pada praktiknya MCC memiliki jarak transmisi yang sangat jauh dengan mobile device sehingga timbul latensi yang besar pula. Mobile Edge Computing (MEC) merupakan teknologi yang hadir untuk mengatasi permasalahan ini. Namun demikian timbul permasalahan baru dari kehadiran MEC ini.

Salah satu permasalahan yang timbul adalah pemilihan keputusan offloading dari mobile device. Beberapa penelitian mempertimbangkan efisiensi energi / besar latensi ataupun keduanya dalam menentukan keputusan offloading. Namun demikian belum banyak penelitian yang mempertimbangkan pergerakan dari mobile device dalam menentukan keputusan offloading. Padahal pergerakan mobile device ini juga sangat berpengaruh besar terhadap latensi karena task perlu dimigrasi ke edge server lain saat suatu mobile device telah bergerak. Beberapa penelitian yang telah mengatasi hal ini menerapkan solusinya kepada lingkungan simulasi yang kecil dan kurang heterogen.

Pada penelitian ini digunakan sebuah metode baru dalam pengambilan keputusan offloading yang memperhatikan pergerakan dari mobile device pada lingkungan yang heterogen. Metode yang diusulkan ini menggunakan Black Widow Optimization dalam menyelesaikan permasalahan pemilihan keputusan saat oflloading. Dari hasil simulasi yang dilakukan performa dari metode yang diusulkan lebih baik daripada metode pembandingnya dari segi besarnya konsumsi energy dan delay latency

***Kata Kunci**: Offloading, Mobile Edge Computing, Black Widow Optimization*

OFFLOADING DECISION SELECTION METHOD FOR ENERGY EFFICIENCY AND LOW LATENCY IN HETEROGENE SIMUATION ENVIRONMENTS

Achmadaniar Anindya Rhosady[1)], Radityo Anggoro [2)]
[1, 2)]Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember
Jalan Teknik Kimia, ITS, Sukolilo, Surabaya, 60111
e-mail: achmadaniar.19051@mhs.its.ac.id[1)], onggo@if.its.ac.id[2)]

ABSTRACT

Mobile Cloud Computing (MCC) is a technology that can overcome the problems of high computing and limited resources owned by mobile devices. However, in practice, MCC has a very long transmission distance from the mobile device, resulting in a large latency. Mobile Edge Computing (MEC) is a technology that exists to overcome this problem. However, new problems arise from the presence of this MEC.

One of the problems that arise is the selection of offloading decisions from mobile devices. Several studies consider energy efficiency / large latency or both in determining offloading decisions. However, there are not many studies that consider the movement of mobile devices in determining offloading decisions. Even though the movement of mobile devices is also very influential on latency because tasks need to be migrated to another edge server when a mobile device has moved. Several studies that have addressed this issue apply the solution to smaller, less heterogeneous simulation environments.

This study used a new method of offloading decision-making that pays attention to the movement of mobile devices in a heterogeneous environment. This proposed method uses Black Widow Optimization in solving the problem of decision selection when offloading. From the simulation results, the performance of the proposed method is better than the comparison method in terms of the amount of energy consumption and delay latency.

***Keywords** Offloading, Mobile Edge Computing, Black Widow Optimization*

I. PENDAHULUAN

Pada era perkembangan digital saat ini, *mobile* device telah berkembang menjadi perangkat yang diperlukan untuk memenuhi kebutuhan komputasi yang tinggi. Kendati demikian perangkat mobile device memiliki keterbatasan sumber daya untuk melakukan komputasi yang tinggi dan terus menerus. *Mobile Cloud Computing (MCC)* hadir sebagai solusi permasalahan keterbatasan sumber daya yang dimiliki *mobile device* tersebut [1]. Dengan teknologi *MCC,* task yang membutuhkan komputasi yang tinggi dapat di *offload* untuk dikerjakan di *cloud server* yang notabenenya memiliki daya komputasi yang jauh lebih besar dari *mobile device.* Namun demikian task yang dioffload ke *cloud server* akan memiliki latensi yang tinggi karena berada sangat jauh dari *mobile device.*

Mobile Edge Computing (MEC) hadir untuk mengatasi permasalahan latensi dari *Mobile Cloud Computing (MEC)* [2]. Teknologi *Mobile Edge Computing (MEC)* memungkinkan kapabilitas dari *cloud server* dibawa lebih dekat ke pengguna dalam hal ini mobile device. Dengan adanya *MEC,* mobile device yang perlu untuk mengoffload tasknya tidak perlu jauh – jauh untuk mengoffload task di *cloud server* tapi bisa lebih dekat di *network edge.* Namun demikian dengan kehadiran *MEC* juga muncul tantangan baru dalam dunia *cloud computing* yang perlu diselesaikan.

Salah satu tantangan dari *MEC* adalah pemilihan keputusan offloading dari mobile device. Pemilihan keputusan offloading menjadi permasalahan saat mempertimbangkan faktor dari *Quality* of Service *(QoS)* seperti besarnya konsumsi energi [3], besar latensi [4], atau gabungan dari keduanya [5]. Kebutuhan energy maupun besar latensi saat *offloading* juga dipengaruhi oleh beberapa hal seperti jarak *mobile device* dengan *edge server/cloud server*, kemampuan komputasi dari *cloud server/edge server*, dan *mobile device*, serta *bandwidth* dari jaringan yang membangunnya.

Permasalahan pemilihan keputusan yang mempertimbangkan banyak faktor ini dikenal juga dengan *NP-Hard-Problem*. Beberapa peneliti telah menyelesaikan permasalahan ini dengan algoritma meta-heuristic seperti *genetic algorithm* [6] ataupun *particle swarm optimization* [7] . Namun demikian belum banyak penelitian yang mempertimbangkan faktor perpindahan (mobility) dari *mobile device*. Padahal menurut Zhan (2020) [8] faktor perpindahan berpengaruh besar terhadap konsumsi energi ataupun besar latensi dari *mobile device offloading.*

Ada beberapa penelitian yang berusaha mengatasi permasalahan *offloading* pada *mobile device* yang bergerak. Salah satunya adalah Puliafito (2020)[9] yang mengusulkan sebuah strategi migrasi task saat *mobile device* sudah terlalu jauh dari *edge server* originalnya. Dia menyebutkan migrasi task dapat dikategorikan menjadi 3 jenis yaitu proactive migrasi, reactive migrasi, dan concurrent migrasi. Namun demikian metode ini dan beberapa metode penelitian lainnya mensimulasikan solusi mereka pada lingkungan simulasi yang kecil dan kurang heterogen. Padahal faktor heterogenisasi seperti spesifikasi *mobile device*, spesifikasi *base station*, spesifikasi *edge server*, dll juga berpengaruh dalam pemilihan keputusan saat *offloading*

Oleh karena itulah dalam makalah ini diusulkan metode baru berbasis *Black Widow Optimization (BWO)* dalam pemilihan keputusan offloading. Metode ini dipilih karena keputusan *offloading* yang dilakukan sangat dipengaruhi oleh faktor heteorigenisasi dari lingkungan simulasi. Dengan adanya faktor heterogenisasi keputusan *offloading* yang optimal bisa tersebar pada lingkungan simulasi. Selain itu dengan adanya faktor lain yaitu pergerakan dari *mobile device* pemilihan keputusan *offloading* juga akan sering berubah seiring dengan pergerakan *mobile device* tersebut. Metode BWO memiliki beberapa fase seperti *cannibalism* dan *mutation* yang membuat pemilihan keputusan bisa cepat menyebar dan tidak mengelompok pada satu kelompok keputusan.

Dalam makalah ini terdiri dari Pendahuluan yang berisikan tentang pengenalan tentang *Cloud/Edge Computing* dan latar belakang ditulisnya makalah ini. Selanjutnya pada Studi Literatur berisikan tentang beberapa penelitian sebelumnya yang berhubungan dengan makalah ini serta studi tentang beberapa simulator yang digunakan dalam *Edge Computing,* Kemudian pada bagian metode yang diusulkan berisikan tentang *modelling* dari simulator dan metode yang diusulkan dalam penelitian ini. Hasil dan kesimpulan menjelaskan evaluasi dari simuilai yang telah dilakukan dalam penelitian ini dan penjelasan secara ringkas tentang hasil evaluasi yang didapatkan.

II. STUDI LITERATUR

Penelitian tentang pemilihan keputusan offloading telah banyak dilakukan. Selain itu penelitian – penelitian ini lebih banyak menggunakan simulator saat lingkungan *edge computing* yang dilakukan lebih luas dan membutuhkan banyak data. Pada bab ini kami mengkompilasi beberapa penelitian sebelumnya serta beberapa simulator yang digunakan dalam *task offloading* pada *cloud/edge computing* .

A. Offloading Decision Method

Pemilihan keputusan offloading menjadi krusial saat memperhatikan *quality of service (qos)* seperti konsumsi

energy atau besar latensi. Beberapa penelitian berusaha untuk menyelesaikan konsumsi energy dan latensi tersebut [6] [10] [11]. Namun demikian penelitian tersebut masih kurang memperhatikan pergerakan (*mobility*) dari mobile device.

P. Mach and Z. Becvar 2015 [12] mengusulkan sebuah algoritma untuk meningkatkan ratio keberhasilan pengiriman task pada lingkungan *mobile device* yang bergerak. Mereka mengusulkan metode yang bernama *cloud-aware power control (CaPc)* dengan memperhatikan task yang memiliki *delay sensitive*. Mereka mengeklaim bahwa metode yang diusulkan memiliki ratio keberhasil pengiriman yang lebih tinggi daripada metode -metode sebelumnya.

Sun (2016) [13] mengusulkan sebuah strategi migrasi avatar untuk mengatasi pergerakan dari *mobile device*. Setiap *mobile device* akan berlangganan terhadap satu buah avatar. Avatar ini merupakan *virtual machine (vm*) yang berada pada suatu *cloudlet* dan selalu tersedia untuk *mobile device* yang berlangganan terhadapnya. Dia mengusulkan sebuah strategi migrasi bernama PRofIt Maximization Avatar pLacement (PRIMAL). Strategi migrasi ini adalah strategi migrasi avatar yang mempertimbangkan *migration cost* dan *migration gain*. Migrasi avatar ini dilakukan saat suatu *mobile device* sudah bergerak cukup jauh dari *cloudlet* serta koneksi antara *original cloudlet* dan *mobile device* sudah tidak tersedia. Dia mengklaim bahwa strategi yang dia usulkan memiliki *E2E delay* lebih kecil daripada 2 strategi pembandingnya yang lain.

Plachy et all (2016) [14] mengusulkan sebuah algoritma baru dalam menentukan jalur *offloading mobile device* yang bergerak. Dalam algoritmanya ini mereka mengestimasi transmission delay dan energy yang dikonsumsi untuk mendapatkan jalur pengiriman data yang tepat. Algoritma yang mereka usulkan ini diformulasikan menggunakan *Markov Decision Process (MDP).* Mereka mengeklaim algoritma yang mereka usulkan dapat mengurangi delay sebanyak 54.3% dan konsumsi energy sebanyak 7.5% dibandingkan dengan metode konvensional.

Sun (2019) [15] mengusulkan sebuah strategi lain dalam mengatasi *E2E delay* pada *offloading mobile device* yang bergerak. Dalam penelitiannya ini dia mengusulkan sebuah algoritma replika avatar yang *bernama LatEncy Aware Replica placemeNt (LEARN)*. Dengan menempatkan beberapa replica dari suatu avatar di cloudlet yang sesuai, dia mengeklaim bahwa metode yang diusulkannya ini memiliki *E2E delay* yang lebih minimal daripada metode metode sebelumnya.

Zhan et al. (2019) [16] mengusulkan sebuah metode *meta-heurisitc* untuk mengurangi latensi dan energi konsumsi dari *mobile device* yang bergerak, Metode mereka ini berbasis *Genetic Algorithm (GA*) untuk menyelesaikan permasalahan pemilihan keputusan offloading yang terjadi. Metode yang mereka usulkan ini diklaim memiliki performa lebih baik terhadap 4 metode pembandingnya.

Zhan et al. (2020) [8] mengusulkan sebuah metode lain dalam mengatasi pemilihan keputusan *offloading* dari *mobile device* yang bergerak. Mereka menyebutkan pemilihan keputusan *offloading* merupakan permasalahan *NP-hard* dan membutuhkan metode *meta-heuristic* untuk menyelesaikannya. Metode yang *bernama heuristic mobility- aware offloading algorithm (HMAOA)* ini dikkalim memiliki performa yang lebih baik terhadap 6 metode pembandingnya.

Al-Tarawneh (2020) [17] mengusulkan sebuah algoritma migrasi container dalam mengatasi *offloading* pada *mobile device* yang bergerak. Algoritma yang diusulkan dia ini berbasis *multicriteria decision making (MCDM*). Metode *MCDMS* yang diusulkan ini menggunakan metode *Entropy-TOPSIS* sebagai strategi dalam melakukan migrasi container saat *mobile device* sudah terlalu jauh dari original cloudletnya. Dia mengeklaim algoritma yang dia usulkan ini meraih 48%, 48%, 20% and 36% peningkatan pada *migration time, service downtime, migration reliability* dan *service lost rate.*

Puliafito et al. (2020) [9] mengusulkan sebuah strategi migrasi dalam mengatasi *offloading* pada *mobile device* yang bergerak. Dalam penelitiannya ini dia menyebutkan ada beberapa scenario migrasi task yang dapat terjadi. Dia mengkategorikan scenario migrasi ini menjadi 3 jenis yaitu proactive migrasi, reactive migrasi, dan concurrent migrasi.

B. Task Offloading Simulator

Beberapa task offloading simulator telah diusulkan sebagai alat untuk mensimulasikan scenario *offloading*. Simulator – simulator ini melingkupi simulator untuk *cloud computing, fog computing, dan edge computing*.

Salah satu simulator yang banyak digunakan dan diteliti adalah *Cloudsim* [18]. *Cloudsim* merupakan simulator untuk melakukan simulasi *cloud computing*. Simulator ini dapat memodelkan beberapa hal seperti *Cloud computing data center, application container, energy-aware computanional resources, datacenter topology, provisioning host resources to virtual machine*, dll. Simulator ini juga sebagai pondasi terciptanya simulator – simulator lainnya.

IFogSim [19] merupakan salah satu simulator yang merupakan pengembangan dari *cloudsim*. Dalam pengembangannya ini *IFogSim* menambahkan beberapa modul sebagai simulasi dari *fog computing*. Dari modul – modul yang ditambahkan ini *IFogSim* dapat melakukan *resource management technique* pada lingkungan *fog computing*.

MobFogSim [9] adalah simulator yang merupakan pengembangan dari *IFogSim*. Pengembangan ini berasal dari adanya kemampuan mobilitas yang ada pada *mobile device module* di dalamnya. Selain itu simulator ini juga memiliki kapabilitas dalam melakukan *migration task. Migration task* yang dilakukan simulator ini sebagai cara *cloudlet* mengirimkan *output task* kepada *mobile device* yang telah bergerak jauh dari original *cloudlet*. Namun kekurangan dari simulator ini adalah karena fokus simulatornya adalah tentang migrasi task sehingga tidak adanya kemampuan komputasi pada *mobile device* yang menyebabkan semua task akan di offload *ke edge/cloud server*

Simulator offloading lainnya adalah *EdgeCloudSim* [20]. Sama halnya dengan *IFogSim*, simulator ini juga merupakan pengembangan dari *cloudsim*. Namun dalam simulator ini fokus yang diberikan adalah pada *Edge Computing*. Simulator ini mengkategorikan beberapa modul untuk melakukan modelling menjadi 3 yaitu *Network modeling, Edge Specific Modeling, dan Computanional modeling*. Selain itu simulator ini juga banyak digunakan karena mudahnya penambahan module serta penerapan algoritma pada simulator.

III. METODE YANG DIUSULKAN

Dalam penelitian ini simulator yang akan digunakan adalah *EdgeCloudSim* [20]. Tahap – tahap dari proses simulasi yang digunakan dalam penelitian ini dapat dilihat pada gambar 1.

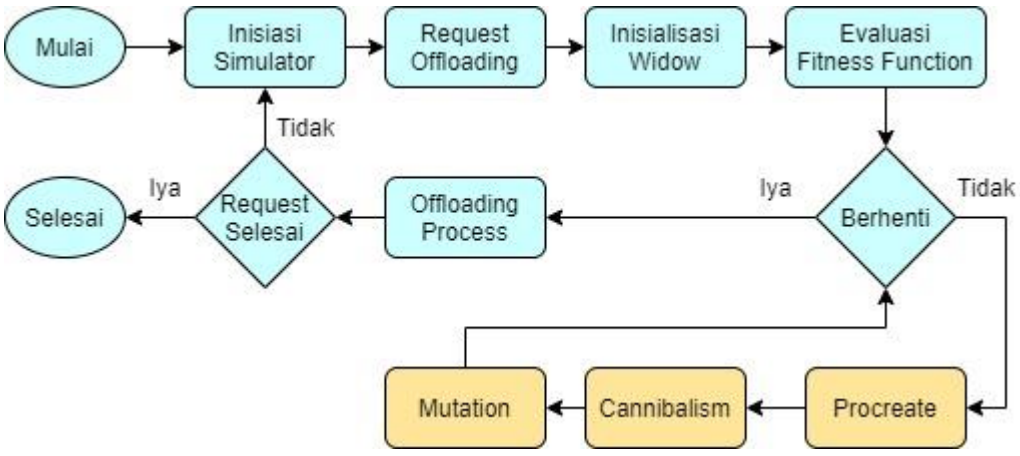
A. Inisiasi Simulator

Tahap awal dari simulator dimulai dengan menginiasiasi beberapa komponen yang ada pada simulator. Komponen – komponen ini antara lain adalah *mobile device, base station, edge server, dan cloud server*. Gambar arsitektur *Mobile Edge Computing(MEC)* yang dibangun oleh komponen ini dapat dilihat pada gambar 2. Mobile device adalah suatu perangkat bergerak yang memiliki beberapa task yang harus diselesaikan. Dalam satu arsitektur akan terdapat beberapa mobile device yang memiliki spesifikasi *(CPU, Memory, Storage)* yang berbeda beda. Task yang harus diselesaikan oleh mobile device juga berbeda – beda. Task ini dapat diselesaikan secara lokal atau dilakukan secara *offloading* ke *edge/cloud server*.

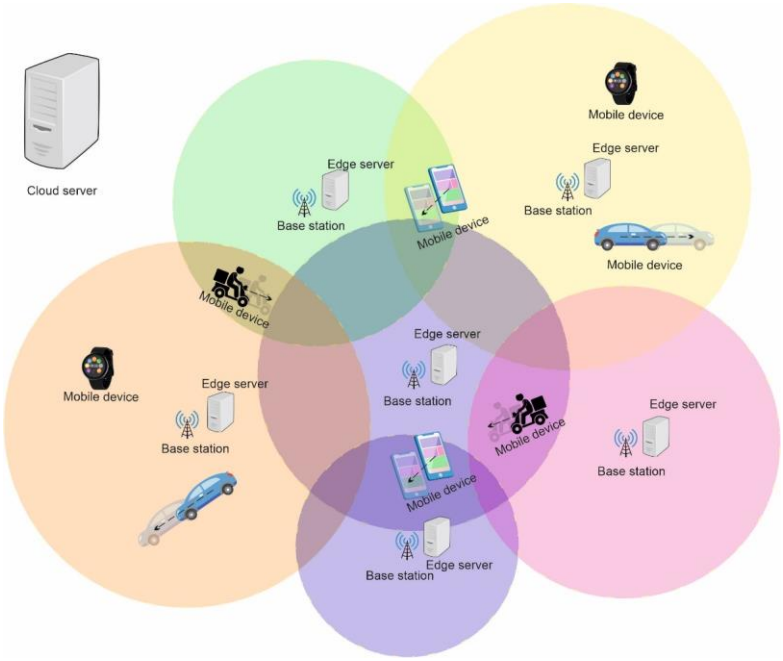
Base station adalah suatu perangkat yang menghubungkan *mobile device* dengan server. *Base station* ini melakukan koneksi dengan *mobile device* melalui wireless channel. *Base station* ini juga memiliki *coverage area* jadi hanya *mobile device* yang berada pada *coverage area* tersebut yang dapat terkoneksi dengan *base station*. Dalam satu arsitektur akan terdapat beberapa base station dan setiap base station ini memiliki spesifikasi (*coverage area*) yang berbeda beda

Cloud server adalah sebuah virtual machine yang memiliki kemampuan komputasi (*CPU, Memory, Storage*) yang tinggi. Meskipun memiliki kemampuan komputasi yang tinggi, *cloud server* ini berada sangat jauh dari *mobile device*, sehingga jarak transmisi antar keduanya juga sangat jauh. Sedangkan *edge server* merupakan *virtual machine* yang lebih dekat dengan *mobile device*. Meskipun kemampuan komputasi dari *edge server* tidak sebesar *cloud server*, namun jarak transmisi data antara *mobile device* lebih dekat.

Dalam arsitektur *Mobile Edge Computing* ini dapat dimodelkan beberapa hal yang terkait proses Offloading yaitu:



Gambar 1. Tahap Proses Simulasi



Gambar 2. Arsitektur Simulator

1) Energy Model

Energy Model didefinisikan sebagai besarnya konsumsi energy yang dimiliki oleh mobile device selama simulasi berlangsung. Konsumsi energy ini dilakukan mobile device saat melakukan dua hal yaitu saat memproses task secara lokal (locally) atau mengoffload task ke edge server/cloud server. Beloglazov, 2012 [21] menyebutkan bahwa konsumsi energy saat pemrosesan pada lokal bergantung pada konsumsi energy statis saat device idle, konsumsi energy maximal dari device saat memproses task, dan besarnya CPU Utilization dari device. Persamaan dari konsumsi energy lokal dapat dilihat pada persamaan (1)

$$E_{local} = P_{max} + (P_{max} - P_{min}) * U_{CPU} \quad (1)$$

2) Delay Latency Model

Sheng (2020) *[22]* mendefinisikan *delay latency* sebagai waktu delay execution task yang dimiliki oleh mobile device. Delay ini bisa terjadi dari dua hal yaitu delay dari local execution atau delay dari task offloading. *Delay latency* dari local execution dipengaruhi oleh kemampuan CPU dan besar task yang harus dikerjakan *[23]*. Delay ini dapat dirumuskan pada *persamaan (2)*. Dimana C_n adalah computation resource yang diperlukan untuk menjalankan task n dan V_L adalah execution rate dari CPU mobile device.

$$DL_{LOCAL} = \frac{C_n}{V_L} \quad (2)$$

Delay latency untuk offloading execution dipengaruhi oleh 3 hal yaitu waktu transmission saat task di upload ke server (T_{up}), dan waktu eksekusi task yang berjalan pada server (T_{exe})[23]. Rumus dari *delay latency* dapat dilihat pada persamaan (3)

$$DL_{OFFLOADING} = T_{up} + T_{exe} \quad (3)$$

Waktu transmission saat task di upload (T_{up}) dapat dihitung meenggunakan rumus persamaan (4). Dimana D_n adalah besar data yang harus diupload, N merupakan Gause Noisse Power yang berada di wireless channel, W adalah bandwidth dari base station, P_{up} adalah kemampuan upload dari mobile device, dan Los adalah channel gain. Los pada lingkungan wireless mengacu pada fungsi jarak yaitu $Los = d^{-\propto}$

$$T_{up} = \frac{D_n}{W\ log_2\ (1 + \frac{P_{up} + Los}{N})} \quad (4)$$

Sedangkan waktu eksekusi task oleh server (T_{exe}) dapat dirumuskan pada persamaan (5), dimana V_{cloud} adalah execution rate yang dimiliki oleh cpu server

$$T_{exe} = \frac{C_n}{V_{cloud}} \quad (5)$$

B. *Black Widow Optimization* dan *Offloading decision*

Black widow optimization (BWO) merupakan algoritma *meta-heuristic* untuk menyelesaikan permasalahan *NP-Hard*. Algoritma ini diperkenalkan oleh Hayyolalam (2020) [24]. Algoritma ini mengambil contoh siklus hidup

dari Laba -laba *Black Widow*. Dari penelitian yang dilakukannya dia mengeklaim bahwa algoritma *black widow* ini memiliki performa yang bersaing dengan metode *meta-heuristic* lainnya. Dia juga mengatakan performa yang baik ini dikarenakan *early convergences* yang ada pada algoritma tersebut sehingga solusi optimal lebih mudah ditemukan.

Offloading decision atau pemilihan keputusan *offloading* dimulai setelah simulator telah selesai di inisiasi. *Offloading decision* ini akan menggunakan algoritma *Black Widow Optimization (BWO)*. Tabel notasi untuk algoritma tersebut dapat dilihat pada tabel 1. *BWO* dimulai dengan inisiasi widow. Panjang array dari widow ini sebanyak task yang harus diselesaikan per simulasi dapat dilihat pada persamaan (6). Setiap solusi task ($ans_{i,j}$) yang didapatkan merepresentasikan offloading decision yang harus dilakukan task tersebut. Fungsi objektif dari tiap widow merupakan penjumlahan dari setiap solusi task yang dimiliki dan dapat dilihat pada persamaan (7).

$$widow = [ans_{i,j}, ans_{i,j+1}, \quad \ldots, ans_{M,N}\,] \tag{6}$$

$$f\_widow_{max}(widow) = ans_{i,j} + ans_{i,j+1} \ldots + ans_{M,N} \tag{7}$$

TABEL I
NOTASI ALGORITMA OFFLOADING DECISION.

Notasi	Pengertian Notasi
$\boldsymbol{i}$	Index mobile device
$\boldsymbol{md_i}$	Mobile device ke-$\boldsymbol{i}$
$\boldsymbol{M}$	Jumlah mobile device
$\boldsymbol{j}$	Index task dari mobile device
$\boldsymbol{task_{i,j}}$	Task ke-j dari mobile device ke-$\boldsymbol{i}$
$\boldsymbol{N_i}$	Jumlah task dari mobile device ke-$\boldsymbol{i}$
$\boldsymbol{x_i}$, $\boldsymbol{y_i}$	Posisi (x, y) dari mobile device ke-$\boldsymbol{i}$
$\boldsymbol{v_{x,i}}$, $\boldsymbol{v_{y,i}}$	Kecepatan dan arah (v_x, v_y) dari mobile device ke-$\boldsymbol{i}$
$\boldsymbol{C_i}$	Kemampuan komputasi dari mobile device ke-$\boldsymbol{i}$
$\boldsymbol{E_i}$	Energi sisa mobile device ke-$\boldsymbol{i}$
$\boldsymbol{E_{local,i,j}}$	Energi yang dibutuhkan mobile device ke-$\boldsymbol{i}$ melakukan task ke- j di lokal
$\boldsymbol{k}$	Index base station
$\boldsymbol{bs_k}$	Base station ke-k
$\boldsymbol{P}$	Jumlah base station
$\boldsymbol{cvg_k}$	Coverage area dari base station ke-k
$\boldsymbol{x_k}$, $\boldsymbol{y_k}$	Posisi (x, y) dari base station ke-k
$\boldsymbol{l_{i,k}}$	Jarak antara mobile device ke-i dengan base station ke-k
$\boldsymbol{E_{offloading,i,j,k}}$	Energi yang dibutuhkan mobile device ke-$\boldsymbol{i}$ melakukan task ke- j secara offloading melewati base station ke- $\boldsymbol{k}$
$\boldsymbol{u}$	Index Edge server
$\boldsymbol{es_u}$	Edge Server ke-u
$\boldsymbol{Q}$	Jumlah Edge Server
$\boldsymbol{x_u}$, $\boldsymbol{y_u}$	Posisi (x, y) dari edge server ke-u
$\boldsymbol{C_u}$	Kemampuan komputasi dari edge server ke-u
$\boldsymbol{band_u}$	Bandwidth dari edge server ke-u
$\boldsymbol{cs}$	Cloud server
$\boldsymbol{C_{cs}}$	Kemampuan komputasi dari cloud server
$\boldsymbol{band_{cs}}$	Bandwidth dari cloud server
$\boldsymbol{x_{cs}}$, $\boldsymbol{y_{cs}}$	Posisi (x, y) dari cloud server
$\boldsymbol{ans_{i,j}}$	Solusi offloading decision untuk task ke-j dari mobile device ke-$\boldsymbol{i}$
$\boldsymbol{widow}$	Widow dari black widow optimization
$\boldsymbol{f\_widow_{max}(widow)}$	Fungsi objektif untuk widow
$\boldsymbol{f\_task}([\boldsymbol{offload_{i,j}}, \boldsymbol{edge_{i,j}}, \boldsymbol{cloud_{i,j}}])$	Fungsi task untuk task ke-j dari mobile device ke-$\boldsymbol{i}$
$\boldsymbol{f\_local}(\boldsymbol{offload_{i,j}}, \boldsymbol{i}, \boldsymbol{j})$	Fungsi objektif menjalankan task ke-j dari mobile device ke-$\boldsymbol{i}$ secara lokal
$\boldsymbol{f\_e\_local}(\boldsymbol{i}, \boldsymbol{j})$	Fungsi konsumsi energi untuk menjalankan task ke-j dari mobile device ke-$\boldsymbol{i}$ secara lokal
$\boldsymbol{f\_c\_local}(\boldsymbol{i}, \boldsymbol{j})$	Fungsi komputasi untuk menjalankan task ke-j dari mobile device ke-$\boldsymbol{i}$ secara lokal
$\boldsymbol{f\_e\_edge}(\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k})$	Fungsi konsumsi energi untuk mengofflading task ke-j dari mobile device ke-$\boldsymbol{i}$ melewati base station ke- k
$\boldsymbol{f\_c\_edge}(\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{v})$	Fungsi komputasi untuk menjalankan task ke-j dari mobile device ke-$\boldsymbol{i}$ secara offloading pada edge server ke-v
$\boldsymbol{f\_d\_edge}(\boldsymbol{i}, \boldsymbol{k})$	Fungsi jarak mobile device ke-i dengan base station ke-k
$\boldsymbol{f\_c\_cloud}(\boldsymbol{i}, \boldsymbol{j})$	Fungsi komputasi untuk menjalankan task ke-j dari mobile device ke-$\boldsymbol{i}$ secara offloading pada cloud server
$\boldsymbol{f\_edge}(\boldsymbol{edge_{i,j}}, \boldsymbol{i}, \boldsymbol{j})$	Fungsi objektif menjalankan task ke-j dari mobile device ke-$\boldsymbol{i}$ secara offloading edge server
$\boldsymbol{f\_cloud}(\boldsymbol{cloud_{i,j}}, \boldsymbol{i}, \boldsymbol{j})$	Fungsi objektif menjalankan task ke-j dari mobile device ke-$\boldsymbol{i}$ secara offloading cloud server

[1,2,0]	[0,0,0]	[1,3,0]	[0,0,1]	[1,5,0]	…..
[0,0,0]	[1,5,0]	[1,2,0]	[1,1,0]	[1,5,0]	…..
[0,0,1]	[0,0,0]	[0,0,1]	[0,0,0]	[0,0,0]	…..
[1,3,0]	[0,0,1]	[1,2,0]	[1,1,0]	[0,0,0]	…..
…					
…					

Gambar 3. Inisiasi Populasi dari Widow

Solusi task ($ans_{i,j}$) sendiri didapatkan dari fungsi task yang dimiliki oleh task tersebut yang dapat dilihat pada persamaan (8). Input dari fungsi ini adalah array yang berukuran 1×3 dimana tiap indexnya merepresentasikan :

1) Index pertama ($offload_{i,j}$), berisi $0/1$ yang merepresentasikan task dikerjakan secara lokal (0) atau dikerjakan secara offloading (1).
2) Index kedua ($edge_{i,j}$) berisi 1,2, …, Q yang merepresentasikan index dari edge server yang mengerjakan task. Index ini akan bernilai 0 saat task dikerjakan secara lokal atau offloading di cloud server
3) Index ketiga ($cloud_{i,j}$), berisi $0/1$ yang merepresentasikan task dikerjakan secara tidak pada cloud server (0) atau dikerjakan pada cloud server (1).

$$f\_task\left(\left[offload_{i,j}, edge_{i,j}, cloud_{i,j}\right]\right) = f\_local\left(offload_{i,j}, i, j\right) + f\_edge\left(edge_{i,j}, i, j\right) + f\_cloud(cloud_{i,j}, i, j) \quad (8)$$

Fungsi task merupakan hasil penjumlahan dari fungsi local, fungsi edge, dan fungsi cloud. Fungsi local pada persamaan (9) sendiri merupakan rata-rata dari fungsi energy local pada persamaan (10) dengan fungsi komputasi local pada persamaan (11).

$$f\_local\left(offload_{i,j}, i, j\right) = \boldsymbol{mean}(f\_e\_local(i, j),\ f\_c\_local(i, j)) \quad (9)$$

$$f\_e\_local(i, j) = \frac{\boldsymbol{E_{local,i,j}}}{\boldsymbol{E_i}} \quad (10)$$

$$f\_c\_local(i, j) = \frac{\boldsymbol{C_i}}{(\boldsymbol{task_{i,j}})^2} \quad (11)$$

Sedangkan fungsi edge pada persamaan (12) merupakan rata-rata dari fungsi energy edge pada persamaan (13), fungsi consumption edge pada persamaan (14), dan fungsi jarak pada persamaan (15). Karena $edge_{i,j}$ merupakan index dari base station. Maka nilai $edge_{i,j}$ juga merepresentasikan nilai k. Dikarenakan juga, satu base station hanya memiliki satu edge server maka nilai k juga merepresentasikan nilai u (16).

$$f\_edge(k, i, j) = \begin{cases} if\ edge_{i,j} = 0 \rightarrow\ 0, \\ else\ \rightarrow \boldsymbol{mean}(f\_e\_edge(i, j, k), f\_c\_edge(i, j, k), f\_c\_dist(i \end{cases} \quad (12)$$

$$f\_e\_edge(i, j, k) = \frac{\boldsymbol{E_{offloading,i,j,k}}}{\boldsymbol{E_i}} \quad (13)$$

$$f\_c\_edge(i, j, v) = \frac{\boldsymbol{C_v}}{(\boldsymbol{task_{i,j}})^2} \quad (14)$$

$$f\_d\_edge(i, k) = \frac{\sqrt{(x_i + v_{x,i} - x_k)^2 + (y_i + v_{y,i} - y_k)^2} + \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}}{cvg_k} \quad (15)$$

$$edge_{i,j} = k = u \quad (16)$$

Sedangkan fungsi cloud pada persamaan (17) sendiri merupakan rata-rata dari fungsi energy edge pada persamaan (13) dengan fungsi komputasi cloud pada persamaan (18). Dalam fungsi cloud perhitungan fungsi energy menggunakan fungsi energy edge dimana base station k yang dipilih adalah base station terdekat dengan mobile device i.

$$f\_cloud\left(cloud_{i,j}, i, j\right) = cloud_{i,j} \times \boldsymbol{mean}(f\_e\_edge(i, j, k),\ f\_c\_cloud(i, j)) \quad (17)$$

$$f\_c\_cloud(i, j) = \frac{\boldsymbol{C_{cs}}}{(\boldsymbol{task_{i,j}})^2} \quad (18)$$

Dari beberapa fungsi tersebut iniasi populasi dari widow dapat digambarkan pada gambar 3 dimana per baris kotak merepresentasikan widow dan jumlah baris merepresantasikan populasi dari widow.

Tahap berikutnya dari BWO mengevalusi fitness function yang didapatkan dari populasi ini. Dalam evaluasi fitness function ini juga akan dilakukan eliminasi terhadap solusi widow yang tidak mungkin dilakukan terhadap keputusan offloading. Eliminasi ini didasarkan atas kemungkian sumber daya tidak mencukupi untuk melakukan keputusan offloading tersebut. Setelah evaluasi fitness function fase berikutnya adalah procreate. Dalam fase procreate populasi akan direproduce sesuai dengan persamaan (19). Setelah melakukan fase procreate fase cannibalism dan mutation dilakukan dengan mempertimbangkan ketersedian resource sesuai dengan keputusan offloading dari widow.fase – fase dari BWO ini akan diulang hingga solusi widow optimal diperoleh.

$$\begin{cases} y_1 = \propto x_1 + (1- \propto) x_2 \\ y_2 = \propto x_2 + (1- \propto) x_1 \end{cases} \quad (19)$$

IV. HASIL DAN KESIMPULAN

Simulasi dilakukan dengan menggunakan parameter simulasi yang dapat dilihat pada Tabel 2. Untuk parameter resources yang ada pada mobile device, base station, dan edge server dihasilkan dari pengacakan agar didapatkan lingkungan simulasi yang heterogen [25]. Jumlah mobile device yang digunakan pada percobaan juga bervariasi untuk melihat efisiensi dan efektivitas metode yang diusulkan.

Untuk pergerakan (mobility) dari mobile device sendiri didapatkan dari hasil traffic simulation framework yaitu Simulation of Urban Mobility (SUMO). Dari hasil traffic simulation framework ini akan didapatkan posisi mobile device (x dan y), kecepatan mobile device (m/s), arah mobile device (angle), serta waktu (s).

Aplikasi yang berjalan pada simulator dibagi menjadi 4 jenis yaitu Augmented Reality (AR), Health Application (HA), Infotainment Application (IA), dan Heavy Computer Application (HC). Parameter dari aplikasi tersebut terdiri dari resources yang dibutuhkan oleh aplikasi. Setiap mobile device akan menghasilkan aplikasi secara acak tiap waktu tergantung dari *usage percentage* yang dimiliki oleh aplikasi. Parameter – parameter yang dimiliki oleh aplikasi ini dapat dilihat pada tabel 3.

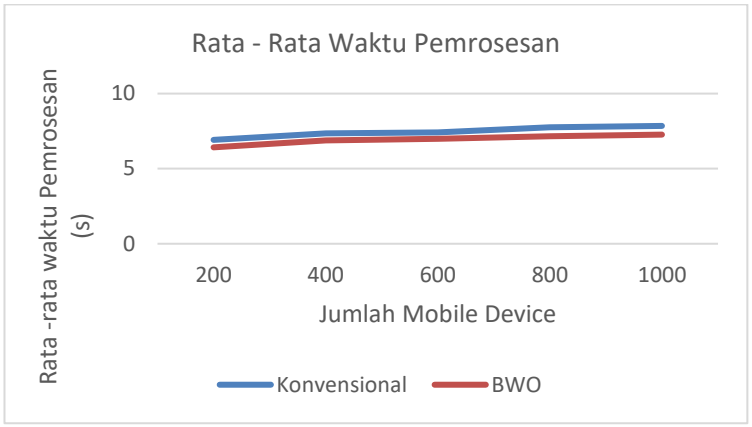
Dari hasil simulasi yang dijalankan terdapat peningkatan efisiensi dari metode yang diusulkan dibandingkan dengan metode konvensional. Jumlah task yang gagal dieksekusi juga lebih sedikit daripada metode konvensional terlihat pada grafik (Gambar 4). Ini dikarenakan pada fitness function metode yang diusulkan memperhatikan pergerakan (mobilitas) dari mobile device. Dalam grafik ini jumlah task yang gagal dieksekusi bertambah secara linear dengan jumlah mobile device yang berada pada simulasi.

TABEL 2
PARAMETER SIMULASI

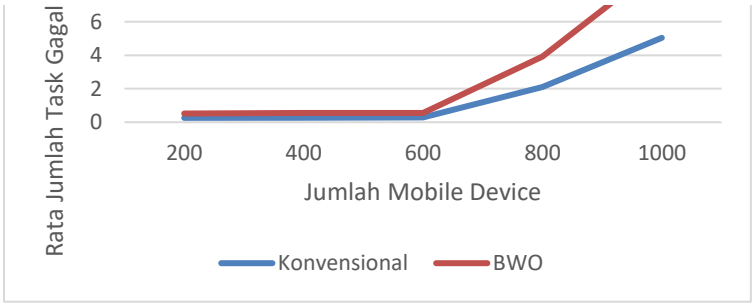
Parameter	Nilai
Waktu Simulasi (menit)	33
Warm-up Period (menit)	3
Luas Lingkungan Simulasi (meter)	2000 x 2000
Jumlah Mobile Device	200/400/600/800/1000
Jumlah VM Cloud & Cloud Server	1
Jumlah Edge Server	11
Jumlah VM Edge Server	4/8
Jumlah Core Edge Server	4/8/16
CPU Speed Cloud VM (GIPS)	200
CPU Speed Edge Server VM (GIPS)	5/10
Base Station Coverage Area (meter)	125/180
Kapasitas Storage Edge Server (GB)	100/200/400
Random Access Memory (GB)	2/8/16

TABEL 3
PARAMETER APLIKASI

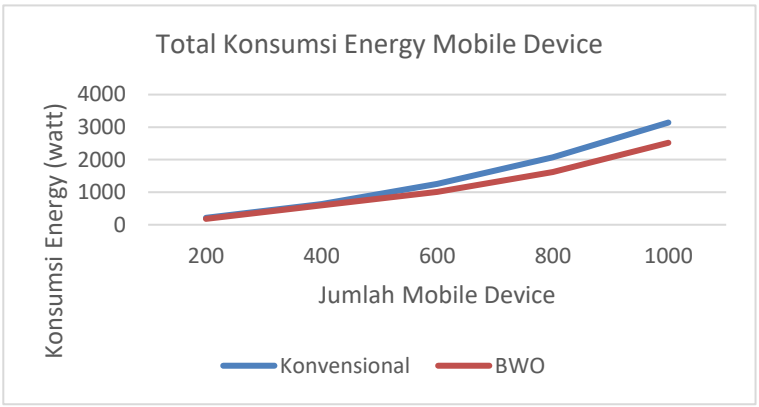
	AR	HA	IA	HC
Usage Percentage (%)	30	20	30	20
Task interarrival Time (s)	2	10	5	10
Idle period Duration (s)	20	90	25	120
Active period duration 9s)	40	15	45	60
Upload data size (kB)	1500	1250	25	2500
Download data size (kB)	25	250	750	200
Task Length (giga instruction)	20	2.5	7.5	30000
Virtual Machine utilization of tasks (%)	10	2	5	100



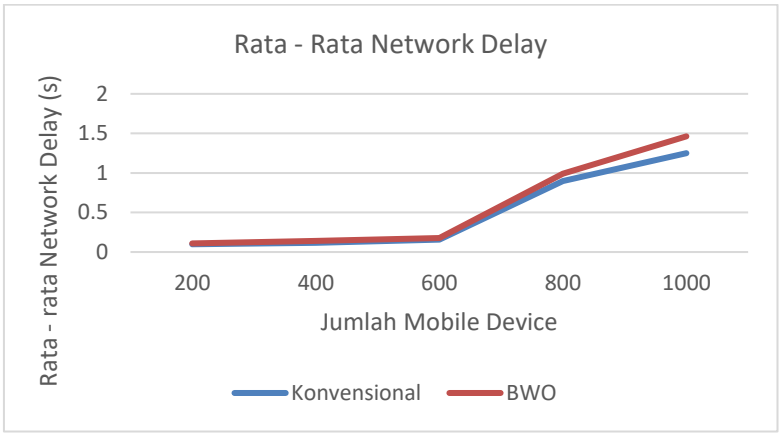
Gambar 7 Rata Rata Waktu Pemrosesan



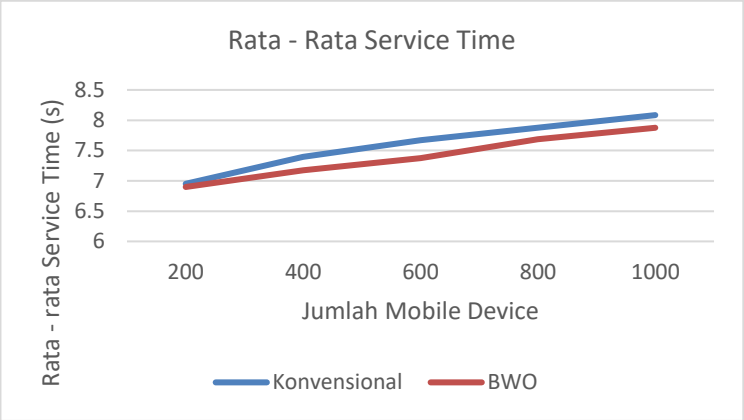
Gambar 4 Jumlah persentase task yang gagal



Gambar 5 Total konsumsi energy mobile device



Gambar 6 Rata Rata Network Delay



Gambar 8 Rata Rata Service Time

Dalam grafik ini juga terlihat bahwa pada mobile device yang berjumlah sedikit jumlah task dari metode yang diusulkan dan metode konvensional cenderung sama karena banyaknya task yang dieksekusi dalam satu simulasi masih berjumlah sedikit sehingga availability dari edge server maupun cloud server cenderung lebih sering tersedia. Selain itu metode yang diusulkan juga memiliki jumlah task gagal yang lebih sedikit daripada metode konvensional. Ini dikarenakan pada metode konvensional task cenderung akan di eksekusi di edge server daripada cloud server saat edge server tersedia. Namun metode konvensional tidak memperhatikan saat mobile device sudah berada pada ujung coverage area dari base station. Hingga menyebabkan saat task telah selesai dieksekusi oleh edge server, task gagak dikembalikan ke mobile device karena mobile device sudah tidak berada pada coverage area dari base station. Berbeda dengan metode yang diusulkan yang juga memperhatikan jarak mobile device dengan coverage area base station dari mobile device tersebut saat mensubmit task.

(Gambar 5) merupakan gambar grafik konsumsi energy pada mobile device saat simulasi dilakukan. Sumbu x merepresentasikan jumlah mobile device sedangkan sumbu y merepresentasikan besarnya energy yang dikonsumsi dalam watt. Dalam grafik ini terlihat bahwa jumlah konsumsi energy pada mobile device bertambah seiring dengan bertambahnya pula jumlah mobile device. Ini dikarenakan semakin banyak mobile device tentu saja setiap mobile device akan menyumbangkan konsumsi energynya terhadap total konsumsi energy. Konsumsi energy dari metode yang diusulkan juga jauh lebih sedikit daripada metode konvensional. Ini dikarenakan pada metode konvensional task cendereng akan dieksekusi di mobile device tersebut daripada di offload ke edge server. Ini menyebabkan mobile device lebih sering melakukan komputasi secara lokal daripada mengoffloadnya ke edge server terdekat.. Berbeda dengan metode yang diusulkan yang mempertimbangkan jarak juga dalam penentuan eksekusi dari task pada mobile device. Dari metode yang diusulkan task mobile device justru akan cenderung lebih dieksekusi pada edge server saat mobile device tersebut memiliki jarak yang sangat dekat atau hampir sama lokasinya dengan edge server.

(Gambar 6) merupakan gambar grafik rata – rata dari network delay. Disini memang terlihat bahwa metode yang diusulkan memiliki network delay yang jauh lebih besar daripada metode konvensional. Ini dikarenakan pada metode konvensional task cenderung lebih sering akan dilakukan secara lokal sehingga tidak ada network delay yang terjadi karena mobile device tidak mengoffloading task tersebut. Berbeda dengan metode yang diusulkan task akan tetap dioffloading meskipun mobile device sebenarnya dapat menjalankan task tersebut

(Gambar 7) merupakan gambar grafik rata waktu pemrosesan data. Pada grafik ini terlihat bahwa waktu pemrosesan data dari metode yang diusulkan jauh lebih baik daripada metode konvensional. Hal ini dikarenakan saat mobile device dekat sekali dengan base station, mobile device akan dioffload ke edge server terdekat. Kemampuan komputasi dari edge server yang lebih tinggi daripada mobile device yang menyebabkan task lebih cepat diesekusi di edge server daripada daripada di mobile device

(Gambar 8) merupakan gambar grafik rata – rata waktu service dari sebuah task. Rata – rata waktu service ini adalah kombinasi dari network delay dan waktu pemrosesan dari task. Dari grafik ini terlihat bahwa metode yang diusulkan memiliki service time yang lebih baik. Meskipun service time dari metode yang diusulkan lebih baik namun perbedaannya tidak terlalu besar. Ini dikarenakan pada metode yang diusulkan memiliki network delay yang lebih besar daripada metode konvensional

Dari hasil simulasi yang dilakukan dapat disimpulkan bahwa metode yang diusulkan memiliki performa yang lebih baik daripada metode pembandingnya. Performa ini disimpulkan berdasarkan lebih rendahnya konsumsi energy dari mobile device serta lebih rendahnya latensi delay yang ditandai dari lebih cepatnya service time dari metode yang diusulkan. Metode yang diusulkan juga berhasil untuk mengatasi permasalahan offloading pada mobile device yang bergerak

DAFTAR PUSTAKA

[1] H. Elazhary, "Internet of Things (IoT), mobile cloud, cloudlet, mobile IoT, IoT cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions," *J. Netw. Comput. Appl.*, vol. 128, no. June 2018, pp. 105–140, 2019.

[2] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile Edge Computing - A key technology towards 5G," *ETSI White Pap. No. 11 Mob.*, vol. 11, no. 11, pp. 1–16, 2015.

[3] H. Guo, J. Zhang, J. Liu, and H. Zhang, "Energy-aware computation offloading and transmit power allocation in ultradense IoT networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4317–4329, 2019.

[4] V. Scoca, A. Aral, I. Brandic, R. De Nicola, and R. B. Uriarte, "Scheduling latency-sensitive applications in edge computing," *CLOSER 2018 - Proc. 8th Int. Conf. Cloud Comput. Serv. Sci.*, vol. 2018-Janua, pp. 158–168, 2018.

[5] X. Cheng *et al.*, "Space/Aerial-Assisted Computing Offloading for IoT Applications: A Learning-Based Approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, 2019.

[6] M. Goudarzi, M. Zamani, and A. Toroghi Haghighat, "A genetic-based decision algorithm for multisite computation offloading in mobile cloud computing," *Int. J. Commun. Syst.*, vol. 30, no. 10, pp. 1–13, 2017.

[7] T. Djemai, P. Stolf, T. Monteil, and J. M. Pierson, "A discrete particle swarm optimization approach for energy-efficient IoT services placement over fog infrastructures," *Proc. - 2019 18th Int. Symp. Parallel Distrib. Comput. ISPDC 2019*, pp. 32–40, 2019.

[8] W. Zhan, C. Luo, G. Min, C. Wang, Q. Zhu, and H. Duan, "Mobility-Aware Multi-User Offloading Optimization for Mobile Edge Computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3341–3356, 2020.

[9] C. Puliafito *et al.*, "MobFogSim: Simulation of mobility and migration for fog computing," *Simul. Model. Pract. Theory*, vol. 101, no. December 2019, 2020.

[10] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, S. U. Khan, and P. Li, "A Double Deep Q-Learning Model for Energy-Efficient Edge Scheduling," *IEEE Trans. Serv. Comput.*, vol. 12, no. 5, pp. 739–749, 2019.

[11] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2651–2664, 2018.

[12] P. M. and Z. Becvar, "Cloud-aware power control for real-time application offloading in mobile edge computing," *Trans. Emerg. Telecommun. Technol.*, vol. 25, no. 3, pp. 294–307, 2015.

[13] X. Sun and N. Ansari, "PRIMAL: PRofIt Maximization Avatar pLacement for mobile edge computing," *2016 IEEE Int. Conf. Commun. ICC 2016*, pp. 6–11, 2016.

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

[15] X. Sun, "Adaptive Avatar Handoff in the Cloudlet Network Systems," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 7, no. 3, pp. 987–990, 2019.

[16] W. Zhan, H. Duan, and Q. Zhu, "Multi-user offloading and resource allocation for vehicular multi-access edge computing," *Proc. - 2019 IEEE Int. Conf. Ubiquitous Comput. Commun. Data Sci. Comput. Intell. Smart Comput. Netw. Serv. IUCC/DSCI/SmartCNS 2019*, pp. 50–57, 2019.

[17] M. A. B. Al-Tarawneh, "Mobility-aware container migration in cloudlet-enabled IoT systems using integrated muticriteria decision making," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 9, pp. 694–701, 2020.

[18] C. A. F. D. R. and R. B. Rodrigo N. Calheiros1, Rajiv Ranjan2, Anton Beloglazov1, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. - Pract. Exp.*, vol. 39, no. 7, pp. 701–736, 2010.

[19] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Softw. - Pract. Exp.*, vol. 47, no. 9, pp. 1275–1296, 2017.

[20] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of edge computing systems," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 11, pp. 1–17, 2018.

[21] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers," *Concurr. Comput. Pract. Exp.*, vol. 24, no. 13, pp. 1397–1420, 2012.

[22] M. Sheng, Y. Dai, J. Liu, N. Cheng, X. Shen, and Q. Yang, "Delay-Aware Computation Offloading in NOMA MEC under Differentiated Uploading Delay," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 4, pp. 2813–2826, 2020.

[23] J. Sheng, J. Hu, X. Teng, B. Wang, and X. Pan, "Computation offloading strategy in mobile edge computing," *Inf.*, vol. 10, no. 6, pp. 1–20, 2019.

[24] V. Hayyolalam and A. A. Pourhaji Kazem, "Black Widow Optimization Algorithm: A novel meta-heuristic approach for solving engineering optimization problems," *Eng. Appl. Artif. Intell.*, vol. 87, no. November 2018, p. 103249, 2020.

[25] R.Rachmat, S. Djanali and R. Anggoro, "[2] Modifikasi Protokol AODV-BR Menggunakan Link Expiration Time (LET) untuk Meningkatkan Stabilitas Link di Lingkungan Mobile Ad-Hoc Network (MANET)," Jurnal Ilmiah Teknologi Informasi (JUTI), Vol 15, No. 2, p. 129-139, Juli 2017.