ENERGY AWARE SCHEDULING FOR HETEROGENEOUS MOBILE TASK COMPUTING

Mr. Vittayasak Rujivorakul

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Computer Science and
Information Technology
Department of Mathematics and Computer Science
Faculty of Science
Chulalongkorn University
Academic Year 2017

การจัดตารางงานโดยคำนึงถึงพลังงานสำหรับการคำนวณภารกิจเคลื่อนที่แบบวิวิธพันธุ์

นายวิทยาศักดิ์ รุจิวรกุล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2560

Thesis Title	ENERGY AWARE SCHEDULING FOR HETEROGENEOUS MOBILE TASK COMPUTING
By	Mr. Vittayasak Rujivorakul
Field of Study	Computer Science and Information Technology
Thesis Advisor	Professor Chidchanok Lursinsap, Ph.D.
Thesis Co-Advisor	Associate Professor Peraphon Sophatsathit, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

_______________________________Dean of the Faculty of Science

(Associate Professor Polkit Sangvanich, Ph.D.)

THESIS COMMITTEE

_______________________________Chairman

(Atchara Mahaweerawat, Ph.D.)

_______________________________Thesis Advisor

(Professor Chidchanok Lursinsap, Ph.D.)

_______________________________Thesis Co-Advisor

(Associate Professor Peraphon Sophatsathit, Ph.D.)

_______________________________Examiner

(Assistant Professor Saranya Maneeroj, Ph.D.)

_______________________________Examiner

(Assistant Professor Suphakant Phimoltares, Ph.D.)

_______________________________External Examiner

(Assistant Professor Saichon Jaiyen, Ph.D.)

วิทยาศักดิ์ รุจิวรกุล : การจัดตารางงานโดยคำนึงถึงพลังงานสำหรับการคำนวณภารกิจเคลื่อนที่แบบวิวิธพันธุ์ (ENERGY AWARE SCHEDULING FOR HETEROGENEOUS MOBILE TASK COMPUTING) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ศ. ดร.ชิดชนก เหลือสินทรัพย์, อ.ที่ปรึกษาวิทยานิพนธ์ร่วม: รศ. ดร.พีระพนธ์ โสพัศสถิตย์, 66 หน้า.

ปัญหาของการจัดลำดับการส่งชุดของงานที่มีความสัมพันธ์กันจากโทรศัพท์มือถือของผู้ใช้ไปยังเครื่องที่ให้บริการหลายตัวที่เชื่อมต่อกับเซลล์เครือข่ายขณะที่ผู้ใช้กำลังเคลื่อนที่ไปตามเซลล์เหล่านี้ด้วยความเร็วต่าง ๆ ได้รับการศึกษาในวิทยานิพนธ์ฉบับนี้ ปัญหาที่ท้าทายคือความเร็วในการทำงานของแต่ละเครื่องที่ให้บริการและความเร็วในการเคลื่อนที่ของผู้ใช้ทำให้ไม่สามารถทำงานร่วมกันได้อย่างลงตัวซึ่งจะนำไปสู่ความยากในการกำหนดกำหนดกลุ่มงานที่มีความสัมพันธ์กันให้กับแต่ละเครื่องที่ให้บริการ โดยต้องใช้เวลาในการประมวลผลกลุ่มงานพร้อมกันกับการใช้พลังงานในการสื่อสารระหว่างเครื่องที่ให้บริการภายในเซลล์เดียวกันและพลังงานที่ใช้ในอุปกรณ์ของผู้ใช้ให้น้อยที่สุด การศึกษาครั้งนี้นำเสนอขั้นตอนวิธีใหม่ในการจัดตารางงานที่มีความสัมพันธ์กันภายใต้ข้อ จำกัด จากปัญหาเหล่านี้ สามแนวคิดใหม่เกี่ยวกับ (1) การเลือกเซลล์เพื่อดำเนินการตามขั้นตอนที่กำหนดให้กับอัลกอริทึม (2) การแบ่งและกำหนดลำดับงานให้กับเครื่องที่ให้บริการในเซลล์ที่เลือกและ (3) สลับลำดับงานที่กำหนดไว้เบื้องต้นเพื่อให้ความยาวของการประมวลผลสั้นที่สุดและใช้พลังงานน้อยที่สุดได้ถูกนำเสนอในการศึกษานี้ จากผลการทดลองเปรียบเทียบกับขั้นตอนวิธีที่ใช้ในปัจจุบันเช่น HEFT, PEFT, HETS จากกราฟการทำงานของงานสังเคราะห์ที่ซับซ้อนหลายแบบ ผลลัพธ์ที่ได้แสดงให้เห็นว่าส่วนใหญ่ของความยาวของการประมวลผลที่พบโดยขั้นตอนวิธีที่นำเสนอจะสั้นกว่าที่พบในขั้นตอนวิธีอื่น ๆ แต่ในแง่ของการใช้พลังงานผลลัพธ์ทั้งหมดที่กำหนดโดยขั้นตอนวิธีที่นำเสนอจะใช้พลังงานน้อยกว่าที่ได้จากขั้นตอนวิธีอื่น ๆ อย่างมีนัยสำคัญ

ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	ลายมือชื่อนิสิต	
		ลายมือชื่อ อ.ที่ปรึกษาหลัก	
สาขาวิชา	วิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ	ลายมือชื่อ อ.ที่ปรึกษาร่วม	
ปีการศึกษา	2560		

5473106723 : MAJOR COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
KEYWORDS: ENERGY AWARE SCHEDULING / MOBILE EDGE COMPUTING / MOBILE SCHEDULING

VITTAYASAK RUJIVORAKUL: ENERGY AWARE SCHEDULING FOR HETEROGENEOUS MOBILE TASK COMPUTING. ADVISOR: PROF. CHIDCHANOK LURSINSAP, Ph.D., CO-ADVISOR: ASSOC. PROF. PERAPHON SOPHATSATHIT, Ph.D., 66 pp.

The problem of scheduling a set of dependent tasks from a user mobile device to several servers in communication cells while the user is moving along these cell in various speed is studied in this thesis. The challenging issue is the execution speed of each server and the speed of user's movement are not compatible. This leads to the difficulty of assigning and finishing the subset of scheduled dependent tasks to each server within the limitation of execution time during passing a cell. Another concern involved this study is the constraints on the length of makespan in terms of minimum communication time among servers in the same cell and the energy consumed by the servers as well as the energy spent by user's mobile device. This study proposed a new algorithm to schedule a set of dependent tasks under the constraints from these issues. Three new concepts of (1) selecting cells for executing scheduled tasks proposed algorithm, (2) partitioning and scheduling tasks to be assigned to the servers in the selected cell, and (3) shuffling the tentatively assigned tasks of all servers to minimize the makespan and energy consumption were proposed in this study. The experimental results were compared with the current practically used algorithms, i.e. HEFT, PEFT, HETS based on several complex synthetic task flow graphs. The obtained results showed that the most of makespan lengths found by our algorithm are shorter than those found by the other algorithms. But in terms of energy consumption, all results scheduled by our algorithm significantly consume less energy than those from the other algorithms.

Department:	Mathematics and Computer Science	Student's Signature	
		Advisor's Signature	
Field of Study:	Computer Science and Information Technology	Co-Advisor's Signature	
Academic Year:	2017		

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor Professor Chidchanok Lursinsap for the continuous support and guidance though out the course of my study. I would also like to thank my co-advisor Associate Professor Peraphon Sophatsathit for the helpful suggestion on the research. Also, the dissertation committee: Associate Professor Soranya Maneeroj, Associate Professor Suphakant Phimoltares, Dr Atchara Mahaweerawat, and Associate Professor Saichon Jaiyen whose useful suggestions and penetrating analysis help shape this study.

Finally, I would like to thank the Office of the Higher Education Commission for financial support of this research under the Higher Education Research Promotion Scholarship

CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In Mobile Edge Computing (MEC)[1], Mobile Cloud Computing (MCC)[2], Small Cell Cloud (SCC)[3] Wireless network architecture, and Vehicular Network, non-stationary mobile users can send the task flow from their devices to processing on the nearby heterogeneous servers provided by the service provider, within the SCC wireless network. The task flow is described by a directed acyclic graph (DAG) of application tasks. The question is how user task flow can operate over a small cells wireless network with limited service boundaries and can guarantee all assigned task is completed in time before leaving the network with minimal power required. This requires a robust algorithm to handle the tasks. In addition, research issue on the heterogeneous environment scheduling is still being studied extensively.

Most available scheduling algorithms such as HEFT[4], PEFT[5], CPOP[4] often choose the task from the queue that is optimized for server processing, this increases server utilization and minimizes the makespan. However, they do not take into account some of the critical constraints described below.

Firstly, with the advantage of mobile cloud computing technology and small cell wireless network, the environment has changed from a processor node with no time limited to a processed cell through a wireless network having limited time to stay in the network cell. Moreover, the server in the network may not be able to connect to the servers in another network. To prepared the suitable environment, network cells selection must be performed. Unfortunately, most of the existing network cell selection mechanisms [6] consider only the signal strength and data rate of the network without taking the computation capability into account.

Secondly, most of the existing heterogeneous scheduling algorithms focus on makespan. PEFT[5], PHTS[7], MOHEFT[8], and HVS[9] focus on execution time. CEAS[10] focuses on execution time, energy, and deadline constraints. Researchers on Genetic Algorithm (GA)[11], Particle Swarm Optimization (PSO)[12], and Ant Colony Optimization (ACO)[13] are applied to scheduling on the cloud system with energy consumption constraint. However, the time complexity is too high. Other energy-aware mobile scheduling techniques are offloading algorithm [14] that focus on offload by decision policy to save energy on mobile devices. However, they do not take care of task dependency.

Bearing the uncertainty of the processing and transmission performance of the server, along with modified by the congestion of the application in mind, it may happen that the task submitted may not get processed before the user leaves the network. New algorithms that can adjust the time to stay on the network longer without affecting the user experience must be developed to help ensure successful processing of tasks.

Motivated by the above aspects, this dissertation addresses the problem of non-stationary mobile task flow scheduling constrained using energy-aware offloading tasks with dependency onto the limited-time access stationary servers that are connected to the small-cell wireless network. The energy-aware offload scheduling algorithms will be proposed while the focusing on the overall energy consumption and the time to finish the task before the deadline.

1.1 Objectives

The objectives of this study are as follows:

- To derive a scheduling algorithm for offloading tasks from non-stationary user to a stationary service provider under low energy stipulation.
- To derive an objective function for task scheduling on the mobile edge computing environment.

1.2 Scope of Work

In this dissertation, the scope of work is constrained as follows:

- The wireless network architecture requires that the user stay connected to the server inside the network, having a level of service confined to the strength of the signal.
- The performance of the network depends on the volume of congestion of the communication channel at that time.
- Each server can send data to another server within the same network but cannot send across the network. Before leaving the network, the server must send the data back to the user's device.
- Each server is different in terms of execution speed, and speed of data transfer. It can process all type of tasks sent by a user. They also consume different levels of energy.
- A user's mobile device can be connected to one wireless network at a time. The device can receive information about the network from the route planning and communication protocols. System performance, and energy consumption rate, and data transfer efficiency are known.
- The tasks of user's mobile device to be offloaded are organized in the form of a task flow graph. The device can use this graph to evaluate the time spent on the processing and transmission of data.

1.3 Contribution

This dissertation proposes an objective function for reducing the energy consumption of the whole system under the conditions that the user travels through a network having limited service boundaries in order to minimize the overall energy consumption. The amount of energy consumption encompasses user device, network equipment, and the server. The processor is diverse in processing power and data transmission capabilities. The time spent on processing and transmitting data is converted into energy as a condition of the decision. This work also proposes three relating algorithms as follows.

The first algorithm is to select a network cell including server within a given travel route. To provide the service range consistent with the type of work to be sent and makes it is possible to utilize the network efficiently with low energy consumption.

The second algorithm is a grouping of interrelated tasks and manages the time to gather and send data to reduce the amount of traffic and overhead, which directly affects the energy consumption for users and system devices.

The third algorithm is to adjust the user's movement speed to correspond to the workload exported to the server. It still maintains the minimum travel time by using minimal energy and can guarantee the completion of the work, thus increasing the satisfaction of the system.

1.4 Dissertation Organization

This dissertation is organized as follows. Chapter 2 describes the related backgrounds. Problems formulation and scope are described in Chapter 3. Chapter 4 provides the detail of the proposed method. Experiments and results are presented in Chapter 5, Discussion and future work concluded in Chapter 6.

CHAPTER 2

LITERATURE

This chapter presents the background of mobile scheduling with energy awareness on mobile edge computing environment. The existing algorithms and architecture are reviewed and discussed.

2.1 Task Scheduling Algorithm

Scheduling problem has been proved to be NP-complete in the general case as well as several limitations. Heterogeneous-Earlies-Finish-Time(HEFT)[4] algorithm and the Critical-Path-on-a-Processor (CPOP) algorithm [4]are two well know algorithms for scheduling on heterogeneous computing environment. HEFT algorithm selects the highest ranking jobs on each level and determines the options to be processed. It reduces the earliest finish time using the insertion-base method. On the other hand, CPOP algorithm uses up and down priority to prioritize tasks. They differ in processor selection which schedules essential tasks to the processor so as to reduce the overall running time of critical tasks. Path-Based-Heuristic-Task-Scheduling (PHTS)[7] algorithm focuses on reducing the production process. PHTS consists of three steps: (1) The path priority to select all possible paths from the given graph and to sort by descending order; (2) Task selection select the tasks from the sorted paths; (3) Processor selection to assign tasks to the processor by reducing the completion time. All algorithms run on processors with unlimited service times and do not focus on energy saving.

The enhancing of HEFT and CPOP algorithm called, EHEFT and ECPOP [15] are proposed to address the time and energy efficient task flow scheduling. Both algorithms use RE performance metric to identify inefficient processors and shut them down to reduce energy consumption.

There are several algorithms to solve multi-objective task flow scheduling by metaheuristic optimization techniques, for example Particle Swarm Optimization (PSO) [16], Ant Colony Optimization (ACO) [13], and Genetic Algorithms (GA) [11], while maintaining user QoS[17] requirements. However, one disadvantage of these methods is long computational time due to slow convergence.

2.2 Mobile Cloud Computing and Network Architecture

Satyanarayanan[18] showed the VM-based cloudlet that could improve usability on the user interaction between the mobile device and cloudlet with the low latency. Simanta et al.[19] to prototyped a reference architecture for the mobile device code offloading to cloudlet VM within the single-hop proximity. They applied to face recognition to revises to offload the resource-intensive execution and provided the rapid delivery and rapid application active time.

Fesehaye et al.[20] studied the impact of cloudlet in interactive mobile cloud application such as file editing, video streaming, and collaborative chatting. The result found that cloudlet could reduce the data transfer delay and increased the throughput of content delivery. Suggestions from the experimental were the maximum cloudlet number should not over than two hops.

Lui et al.[21] introduced mobile cloud architecture and the state of the art of mobile cloud computing with offloading technology. The applications are mobile computing on augmented reality (AR), Remote HealthCare, and web. That showed opportunities for comprehensive energy-saving interaction, the overhead of virtual machine migration, privacy, and security.

Duo and Heinzelman[22] demonstrated the utilization of edge-server (cloudlet) in the mobile cloud hybrid architecture (MOCHA) and set up network latency measurements. The result showed that dynamic profiling and random server selection approach could utilize cloudlet and provide acceptable latency with high redundancy.

Di Valerio and Lo Presti [23] used Markov Decision Process (MDP) to improve the mobile user experience with the optimization of virtual machines (VMs) allocation across the femtocell mobile cloud computing. The system overhead, network delay, and migration cost were taking into account and benchmarked with the efficient heuristics.

Vondra and Becvar [17] proposed the computing cells selection algorithm to increase user's satisfaction with the transmission and computation delay experienced in small cells could network (SCeNBs). The simulation results of the proposed algorithm could provide higher satisfaction compared to the competitive approaches for all type of backhauls (ADSL, GPON).

Lobillo et al.[24] introduced a Small Cell Manager (SCM) to optimizes the cloud-enabled small cells operation. The concepts could be deployed in an LTE environment at the centralized entity (Gateway) to decrease complexity and provided greater control.

Barbarossa et al.[25] showed the distributed cloud scenario on the 5G network to integrated Cloud, Femtocloud, Microcell, and Small cell to provide the Quality of Experience (QoE) and Quality of Service (QoS) with minimum transmit energy under computational constraint and minimum transmit power under delay constraint.

Liu et al.[26] proposed the converged edge infrastructure for future called CONCERT based on control/data plane decoupling various physical resources. The radio interface equipment, computational resources are controlled and presented as virtual resources which software-defined services.

Luan et al.[27] showed the concepts and main features of Fog computing, the use case scenario to be deployed at the shopping center, parkland, inter-state bus, and vehicular fog computing networks. The main features were wireless, local services, and distributed management. The concepts showed the environment of computing shifted to the streets or shopping malls. Its opened the research issues on network management and service delivery.

Kaur et al.[28] presented the architecture for task selection and scheduling using container-as-a-service (CoaaS). The cooperative game theory and multi-objective function were developed to reduce an energy consumption and makespan by memory, CPU, and user budget, lightweight containers on virtual machines were used to reduce the overhead and response time and overall energy consumption on fog computing device.

2.3 Code Offloading Techniques

Kaur[29] compared the transmission technique to offloading code and the method of application partition on mobile cloud computing. The route based techniques were selecting the best cloud-path for offloading work. The resource-based techniques selected the best resource based on following criteria: energy cost, bandwidth, reliability, service quality, and dependency level. Techniques such as depth-first search (DFS), game theory, min-cut maximum flow. Cloudlet based techniques showed that multi-threaded applications could send messages to a nearby server. An application partitioning method showed that graph-based partition algorithm could be applied offloading technique according to CPU load, network condition, and user input. The adaptive bandwidth partitioning used weighted graph object relation to avoid the overhead of dynamic partitioning. Moreover, combining static analysis and dynamic profiling required bandwidth to minimize the time saving and energy consumption.

Flores et al.[30] proposed the solution to migrate the limitation of code offloading, code profiling, integration complexity, dynamic configuration of the system, offloading scalability. The result of evaluation by offloading an NQueens algorithm from smartphone to cloud calculated how to place *n* queens on and *n x n* chessboard. That showed offloading as a service was a primary challenge and limitation. Zhang et al.[31] developed the optimal offloading algorithm to make an offloading decision by using a Markov Decision Process (MDP) with intermittent connectivity to minimize the computation and offloading the cost of the task flow as a job queue with the threshold policy.

2.4 Comparative Algorithms Analysis

Many existing heterogenous scheduling algorithms have different tasks priority and processor assignments. In this work, HEFT[4], PEFT[5], HETS[32], CEAS[10], and DGES[33] are selected to compare with one cell scenario benchmark. For the multiple cells scenario, this dissertation has enhanced the feature of the candidate to supported the intermitted processing architecture.

2.4.1 Selected Algorithms

Heterogeneous Earlies-Finish-Time (HEFT): This state of the art scheduling has two-step of the process. First is sorting the task with the priority of upward rank value base on computation and communication cost. Second is selected processor to assign the task that has lower earliest available time by using the insertion method.

Predict Earliest Finish Time (PEFT): The algorithm uses optimistic cost table (OCT) to indicate the maximum optimistic processing time of the child task. The task priorities use the average of OCT upward rank and the optimistic EFT to forecast the finish time of the next steps.

Heterogeneous Edge and Task Scheduling (HETS): This algorithm focuses on the minimization of communication overhead, which calculates the edge priority as well as node priority.

Cost and Energy Aware Scheduling (CEAS): The algorithm uses the method to sequence and parallel tasks merging to reduce the execution cost and reduce the energy consumption while meeting the deadline constraint. The VM reuse policy is proposed to utilize the slack time to save energy of leased VM instance.

Global DVFS-enable Energy-efficient Scheduling (GDES): This energy awareness algorithm generates minimum dynamic energy consumption by reassigning tasks to processor slack.

All of the above algorithms cannot directly support multi-cell operation. In comparing cases with multiple cell travel, the policy needs to be modified to stop processing and return the result to user devices, then sending data to the next cell. Time and energy are taken into account from the first cell until the finish. The network cell is viewed as a processing environment, and the processor is represented by a server capability inside the cell.

CHAPTER 3

PROBLEM FORMULATION

The main problem is scheduling a task flow from a mobile user device to process on a server that connects to a wireless network with limited coverage service time. However, low-efficiency selection of network cells and servers are affected by the power consumption to complete the task flow. An algorithm for Energy Aware Mobile Scheduling for Heterogeneous Edge Computing (EAMS-HEC) selecting the appropriate network cells for continuous processing is proposed. Details are described below.

3.1 User Task Flows

The task flow of on the user's device is represented by a Directed Acyclic Graph (DAG) that can estimate the processing time and data transmission time. The selected tasks are assigned to the stationary server under the wireless network that the user's device can connect to when they reach the service coverage area while moving at a fixed average speed.

Definition 1: *A user task flow graph* $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{D}, \boldsymbol{E})$ is a DAG representing the relationship between tasks to be processed by several servers. $\boldsymbol{V} = \{v_1, \dots, v_n\}$ is a set of tasks; $\boldsymbol{D} = \{d_1, \dots, d_n\}$ is a set of executed data and instructions corresponding for each v_i; and $\boldsymbol{E} = \{(v_i, v_j) | v_i, v_j \in \boldsymbol{V}\}$ is a set of directed edges connecting dependent tasks. An edge (v_i, v_j) implies that task v_i must be processed before v_j.

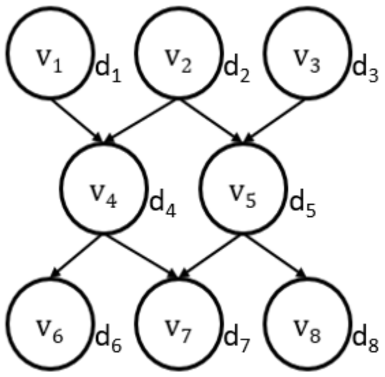


Figure 3.1: An example of the user task flow graph.

Figure 3.1 shows an example of a task graph. There are eight tasks, v_1 to v_8, and three sets of independent tasks which are $\{v_1, v_2, v_3\}$, $\{v_4, v_5\}$, and $\{v_6, v_7, v_8\}$.

Definition 2: *The estimated processing time* of each v_i at user's device, denoted as $\tilde{p}(d_i)$, is the estimated time to execute v_i based on the processing speed of user's device.

Definition 3: *The sending time* $\tau^{(send)}(d_i)$ is the time to send the data of size d_i from user's device to a server.

Definition 4: *Result data* $d_o$is the amount of data generated by a server as the result of processing the received input data d_i from user's device.

Definition 5: *The receiving time* $\tau^{(receive)}(d_o)$ is the waiting time of user's device or other server for receiving the result of size d_o from a processing server.

3.2 Studied Wireless Network Cell Architecture

The studied wireless network cell architecture consists of a set of server clusters. In, each server cluster is called a cell. The number of servers in each cell is not equal. Let $\boldsymbol{C} = \{c_1, \ldots, c_k\}$ denote a set of cells. Some of them are connected as shown in Figure 3.2. For example, $c_1$is connected to c_2 and c_2 is connected to c_3.But, $c_1$is not connected to c_4 and c_5. We assume that a user's vehicle must drive along a path passing through some of connected cell. One possible path is driving through $c_1 \to c_8 \to c_3 \to c_2 \to c_4 \to c_7 \to c_6$. The shape of each cell is assume to be circular. The radius of each cell is not equal. However, the region of cell c_i may overlap with the connected cell c_j due to the signal strength.

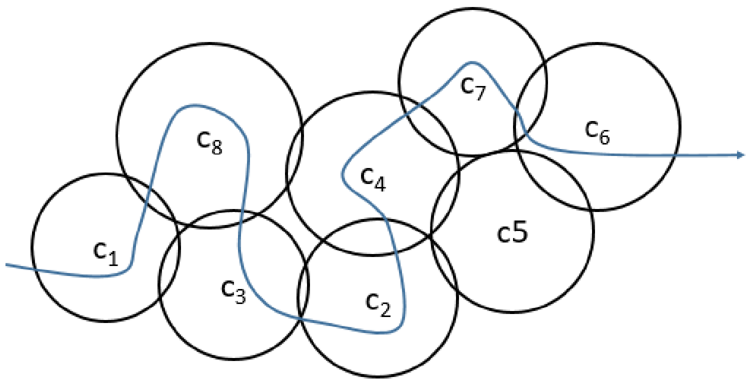


Figure 3.2: An example of cell architecture. There are eight cells.

In each cell region, there is a set of servers located at different positions inside the region. Furthermore, the distribution of servers within the region is already pre-determined by the service provider. The scheduling algorithm obtains the information of server location from the service provider. The servers in each cell region can communicate with each other, but they cannot communicate with the servers in other cells. At cell c_i, let $\boldsymbol{S}_i = \{s_{i,1}, \ldots, s_{i,m}\}$ be a set of servers in cell c_i.

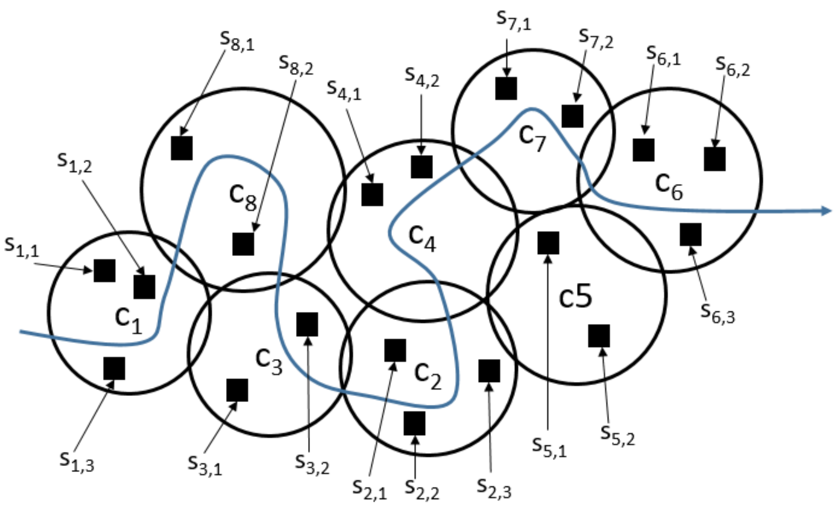


Figure 3.3: An example server cluster in each cell.

Figure 3.3 shown an example of the server cluster in each cell. The black square is representing to the server. In cell c_1, there are three servers, $s_{1,1}$, $s_{1,2}$, and $s_{1,3}$.

Definition 6: *User's device processing speed* u is the amount of data of size δ processed in one unit time.

Definition 7: *Server processing speed* $p_{i,j}$ of server $s_{i,j}$ is the amount of data of size $k_{i,j}\delta$ process in one unit time. $k_{i,j}$ is a constant defined for server $s_{i,j}$.

Definition 8: *Processing time* $t_{i,j}$ to process data d_a from task v_a on server $s_{i,j}$ is computed by the following equation.

$$t_{i,j} = \frac{d_a}{k_{i,j}\delta} \tag{3.1}$$

Definition 9: *Processing energy constant* $\alpha_{i,j}$ is the amount of energy consumed by the server $s_{i,j}$ for processing data in one unit time.

Definition 10: *Transmission energy constant* $\lambda_{i,j}$ is the amount of energy consumed by the server $s_{i,j}$ for transmitting the processed data back to user or other server in one unit time.

Definition 11: *Transmission energy constant* γ is the amount of energy consumed by user's device for transmitting data to the server in one unit time.

Definition 12: *Transmission energy constant* $\beta_{i,j}$ is the amount of energy consumed by cell c_i for transmitting the data between user's device to server and server to other server in one unit time.

Definition 13: *Cell capability* ρ_i of cell c_i is the summation of server processing speed measured in terms of user's device speed as follows

$$\rho_i = \sum_{j=1}^{n_i} p_{i,j} = \sum_{j=1}^{n_i} k_{i,j}\delta \tag{3.2}$$

where n_i is a number of servers in cell c_i.

For example in cell c_1 if server $s_{1,1}$ has processing speed $p_{1,1} = 2\delta$,server $s_{1,2}$has processing speed $p_{1,2} = 3\delta$, and server $s_{1,3}$has processing speed $p_{1,3} = 1\delta$. From equation 3.2 the cell capability ρ_1 of cell c_1 is 6δ.

3.3 Energy Model

The amount of energy consumed in this study determined by converting the processing time and transmission time into the energy form.

Definition 14: *Server processing energy consumption* $e_{i,j}$ is the summation of time to processing assigned tasks from task flow G multiply by processing energy constant $\alpha_{i,j}$ show as follow equation.

$$e_{i,j} = \sum_{k=1}^{n_k} \alpha_{i,j}\, t_{i,j}\,(d_k) \tag{3.3}$$

Where $n_k$is number of vertices from task flow G assigned to server $s_{i,j}$.

Definition 15: *The server sending time* $\pi_{i,j}^{(send)}(b)$ is the time to send the data of size b from processing server $s_{i,j}$ to other servers.

Definition 16: *The server receiving time* $\pi_{i,j}^{(receive)}(b)$ is the waiting time of server $s_{i,j}$ for receiving the result of size b from a processing server.

Definition 17: *Server sending energy consumption* $r_{i,j}^{(send)}(b)$ is the energy to send the data of size b from processing server $s_{i,j}$ to other servers show as follow equation.

$$r_{i,j}^{(send)}(b) = \lambda_{i,j} \cdot \pi_{i,j}^{(\mathrm{send})}(\mathrm{b}) \tag{3.4}$$

Definition 18: *Server receiving energy consumption* $r_{i,j}^{(receive)}(b)$ is the energy of server $s_{i,j}$ for receiving the result of size b from a processing server. show as follow equation.

$$r_{i,j}^{(receive)}(b) = \lambda_{i,j} \cdot \pi_{i,j}^{(receive)}(\mathrm{b}) \tag{3.5}$$

Definition 19: *Total server energy consumption* Γ_S is the summation of $e_{i,j}$ and $r_{i,j}$ from all selected server to processing task flow G. Can find by follow equation.

$$\Gamma_S = \sum_{i=1}^{m_i} \sum_{j=1}^{n_j} \left(e_{i,j} + r_{i,j}^{(receive)}(b) + r_{i,j}^{(send)}(b) \right) \tag{3.6}$$

where $m_i$is number of selected cell from the user path, $n_j$is the number of selected server in each cell c_i.

Definition 20: *The cell sending time* $\Pi_i^{(send)}(b)$ is the time to send the data of size b from network cell equipment to the processing server and user's device.

Definition 21: *The cell receiving time* $\Pi_i^{(receive)}(b)$ is the waiting time of network cell equipment for receiving the result of size b from a processing server.

Definition 22: *Total cells energy consumption* Γ_C is the summation of communication time between user's device and server multiply by β_i .

$$\Gamma_C = \sum_{i=1}^{m_i} \beta_i \left(\Pi_{\mathrm{i}}^{(receive)}(\mathrm{b}) + \tau \Pi_{\mathrm{i}}^{(\mathrm{send})}(\mathrm{b}) \right) \tag{3.7}$$

Where $m_i$is number of selected cell from the user path.

Definition 23: *Total user's device energy consumption Γ_U* is the summation of communication time between user's device to the server multiply by γ.

$$\Gamma_U = \sum_{i=1}^{n_i} \gamma \cdot \tau^{(send)}(d_i) + \sum_{o=1}^{n_o} \gamma \left(\tau^{(\mathrm{receive})}(d_o) \right) \qquad (3.8)$$

where $n_i$is number of tasks in task flow graph G sent to the servers, n_o is number of tasks returned form the servers to user's device after processing.

Definition 24: *Total energy consumption in the system Γ_G* is the summation of the total user's device energy consumption Γ_U, total cells energy consumption Γ_C , and total server energy consumption Γ_S show as follow equation.

$$\Gamma_G = \Gamma_\mathrm{U} + \Gamma_C + \Gamma_S \qquad (3.9)$$

3.4 Studied of User Speed Control and Condition

The minimum speed limit determines the speed of the user's movement. The definition of variable A_1 and the maximum speed are defined in variable A_2. If the user moves slower than the minimum speed, they will not be able to reach their destination on time. And if the user moves faster than the maximum speed, they will not be able to connect and access the network cell. The variable υ is the average speed of the user's movement. The equation can be written as follows.

$$\upsilon \in [A_1, A_2] \qquad (3.10)$$

Definition 22: *Cell service length* ϕ_i is the distance of user path through cell c_i.

Definition 23: *Duration time* T_i is the time to stay on cell c_i, depends on user moving speed υ and distance of each cell ϕ_i. As shown by the following equation:

$$T_i = \frac{\phi_i}{\upsilon} \tag{3.11}$$

The speed of the user device is inversely proportional to the time it is connected to the wireless network, which affects the amount of workload that can be sent out to the servers.It was done by starting from the time spent processing on the network shown by the variable T_i'' , the new speed can be obtained by the following equation.

$$\upsilon' = \frac{\phi_i}{T_i'} \tag{3.12}$$

CHAPTER 4

PROPOSED METHOD

This dissertation proposes an algorithm for scheduling task flows from non-stationary user’s device to processed on stationary servers connected to wireless cells network. This work is divided into three main steps, which are related to the algorithm presented. The first step is to select the appropriate cell for processing as shown in Algorithm 1. The second step is to group and assigning the task flow into each server by considering the task dependency and transmission overhead that affects to energy consumption as shown in Algorithm 2. Finally, the user's speed adjustment algorithm is proposed to guarantee the success of the processing before disconnecting from the network as shown in Algorithm 3. This work aims to reduce the overall system energy consumption and maintain user experience that allows users to reach their destination in time.

4.1 Cell Selection Algorithm

The decision to select the appropriate cell user path that defines in Section 3.2 for this dissertation is focused on the cell capability ρ_i that define in Definition 13, which is related to the server processing speed $p_{i,j}$ in definition 7 . Cell information is provided by the service provider. The example of cell information from Figure 3.3 are shown by the following format $[start\ time, T_i\ , \rho_i, set\ of\ p_{i,j}]$. For example of cell c_1 , The information is $[\ 0, 130, 6\delta, \{\ 3\delta, 1\delta, 2\delta\}]$. c_1 has $c_i^{(start)} = 0$; duration to stay on the cell $T_i = 130$; Cell performance $\rho_1 = 6\delta$ that is calculated from the summation of processing speed of each server. $s_{1,1}$ has $p_{1,1} = 3\delta$, $s_{1,2}$ has $p_{1,2} =\ 1\delta$, and $s_{1,3}$ has $p_{1,3} = 2\delta$.

When the user's start the trip passes through the cells service region, they will give the information from the service provider for all possible connected cells.

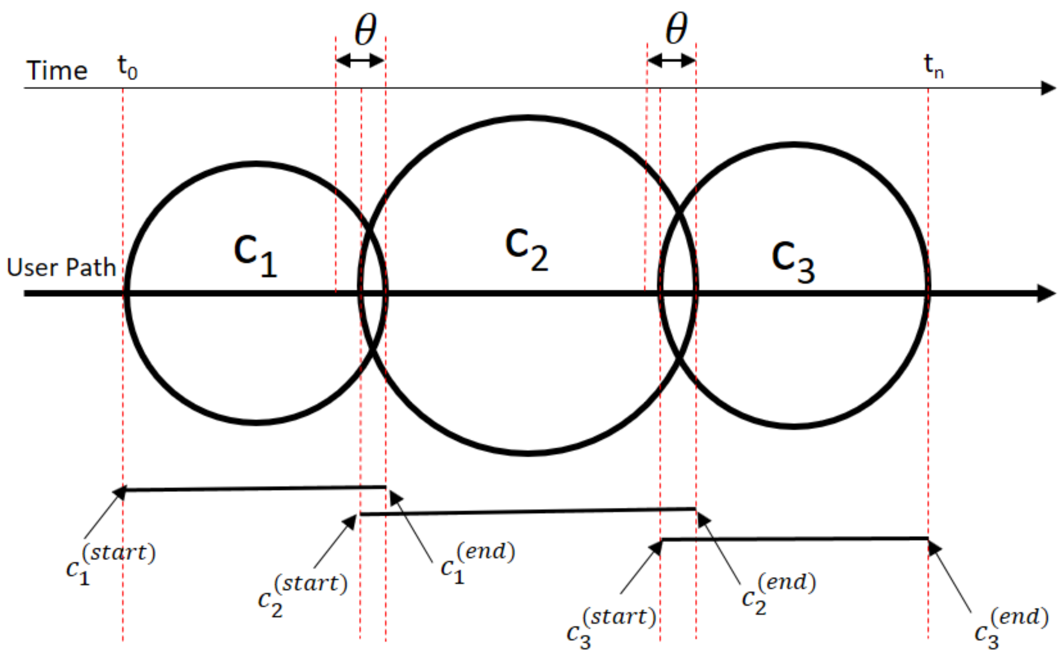


Figure 4.1: Cell structure and user path moving time line.

Figure 4.1 shows cell structure when user start to moving pass through cell c_1, c_2, and c_3. $c_1^{(start)}$ is the time point that user first enters the region of cell c_1. Cell end service time $c_1^{(end)}$ is the time point that user leave the region of cell c_1. And θ is the constant to reserve transition time between any neighboring cells. From the cell structure the cell selection algorithm is show as follow:

Algorithm 1 Cell Selection.

Input: A set of all cell $\boldsymbol{C}'$ on the user path.

Output: A selected cell in set $\boldsymbol{C}$.

```
1:  Sorting $\boldsymbol{C}'$ by the start time to connecting.
2:  Let $c_x$ be the first cell to be connecting.
3:  Let $\boldsymbol{C}$ be an empty set.
4:  $\boldsymbol{C} = \{c_x\} \cup \boldsymbol{C}$
5:  For $c_i \in C'$ do
6:     If $c_i^{(start)}$ equal to $c_x^{(start)}$ then
7:        If $(\rho_i \times T_i) > (\rho_x \times T_x)$ then
8:           Replace $c_x$ in $\boldsymbol{C}$ with $c_i$ .
9:           Let $c_x = c_i$.
10:       EndIf
11:       Else
12:          If $c_i^{(start)} \geq (c_x{}^{(end)} - \theta)$ and $c_i^{(start)} < c_x^{(end)}$ then
13:             $C = \{c_i\} \cup \boldsymbol{C}$ .
14:             Let $c_x = c_i$.
15:       EndIf
16:    EndIf
17: End For
18: return $\boldsymbol{C}$
```

The following is an example of how the algorithm works by simulating a cell through 10 cells from the following information table.

Table 4.1: An example cell information in a set of $\boldsymbol{C}'$.

c_i	Start time	T_i	ρ_i	$T_i \times \rho_i$	Set of $p_{i,j}$
c_1	0	130	3δ	390	$\{1\delta, 2\delta\}$
c_2	0	120	6δ	720	$\{3\delta, 3\delta\}$
c_3	100	160	4δ	640	$\{1\delta, 2\delta, 1\delta\}$
c_4	100	190	4δ	760	$\{2\delta, 2\delta\}$
c_5	120	150	5δ	750	$\{2\delta, 2\delta, 1\delta\}$
c_6	290	130	6δ	780	$\{2\delta, 2\delta, 1\delta, 1\delta\}$
c_7	320	100	5δ	500	$\{3\delta, 2\delta\}$
c_8	350	150	2δ	300	$\{1\delta, 1\delta\}$
c_9	400	100	4δ	400	$\{2\delta, 2\delta\}$
c_{10}	400	120	6δ	720	$\{2\delta, 2\delta, 2\delta\}$

The following steps show how to select a cell from Table 4.1 by setting $\theta = 20$.

Table 4.2: Steps of the cell selection algorithm to process the provider information C*from Table 4.1.*

Cell	$c_i^{(start)}$	$c_i^{(end)}$	$c_x^{(start)}$	$c_x^{(end)}$	$c_x^{(end)} - \theta$	$\boldsymbol{C}$
Initial step set c_x = c_1 and put c_1 in $\boldsymbol{C}$						
c_1	0	130	0	130	110	$\{c_1\}$
Start to get each cell c_i from $\boldsymbol{C}'$						
$c_2^{(start)}$ *equal to* $c_x^{(start)}$, but c_2 has $T_i \times \rho_i$ greater than c_x, replace c_1 by c_2 in $\boldsymbol{C}$, set $c_x = c_2$.						
c_2	**0**	**120**	**0**	130	**110**	$\{c_2\}$
$c_3^{(start)}$ *not equal to* $c_x^{(start)}$ and $c_3^{(start)} \geq c_x^{(end)} - \theta$, add c_3 to $\boldsymbol{C}$, set $c_x = c_3$.						
c_3	**100**	260	0	120	**100**	$\{c_2, c_3\}$

$c_4^{(start)} equal\ to\ c_x^{(start)}$, but c_4 has $T_i \times \rho_i$ greater than c_x, replace c_3 by c_4 in $\boldsymbol{C}$, set $c_x = c_4$.						
c_4	100	290	100	260	240	$\{c_2, c_4\}$
$c_5^{(start)} not\ equal\ to\ c_x^{(start)}$ and $c_5^{(start)} \leq c_x^{(end)} - \theta$, skip this cell.						
c_5	**120**	270	100	290	**270**	$\{c_2, c_4\}$
$c_6^{(start)} not\ equal\ to\ c_x^{(start)}$ and$c_6^{(start)} \geq c_x^{(end)} - \theta$, add c_6 to $\boldsymbol{C}$, set $c_x = c_6$.						
c_6	**290**	420	100	290	**270**	$\{c_2, c_4, c_6\}$
$c_7^{(start)} not\ equal\ to\ c_x^{(start)}$ and $c_7^{(start)} \leq c_x^{(end)} - \theta$, skip this cell.						
c_7	**320**	420	290	420	**400**	$\{c_2, c_4, c_6\}$
$c_8^{(start)} not\ equal\ to\ c_x^{(start)}$ and $c_8^{(start)} \leq c_x^{(end)} - \theta$, skip this cell.						
c_8	**350**	500	290	420	**400**	$\{c_2, c_4, c_6\}$
$c_9^{(start)} not\ equal\ to\ c_x^{(start)}$ and$c_9^{(start)} \geq c_x^{(end)} - \theta$, add c_9 to $\boldsymbol{C}$, set $c_x = c_9$.						
c_9	**400**	500	290	420	480	$\{c_2, c_4, c_6, c_9\}$
$c_{10}^{(start)} equal\ to\ c_x^{(start)}$, but c_{10} has $T_i \times \rho_i$ greater than c_x, replace c_9 by c_{10} in $\boldsymbol{C}$, and set $c_x = c_{10}$.						
c_{10}	**400**	520	**400**	420	480	$\{c_2, c_4, c_6, c_{10}\}$

The result of an algorithm are ready in set $C = \{c_2, c_4, c_6, c_{10}\}$, with capabilities $\{6\delta, 4\delta, 6\delta, 6\delta\}$,And detail information of each cell is:

$c_2 = \{ s_{2,1}, s_{2,2} \}$, with processing speed $\{3\delta, 3\delta\}$.

$c_4 = \{ s_{4,1}, s_{4,2} \}$, with processing speed $\{2\delta, 2\delta\}$.

$c_6 = \{ s_{6,1}, s_{6,2}, s_{6,3}, s_{6,4} \}$, with processing speed $\{2\delta, 2\delta, 1\delta, 1\delta\}$.

$c_{10} = \{ s_{10,1}, s_{10,2}, s_{10,3} \}$, with processing speed $\{2\delta, 2\delta, 2\delta\}$.

4.2 Task Assignment Algorithm

User's device can be sending task flow which in a group to processing on the server cluster that connected to the cell. The primary energy consumes in the wireless network system is the communication power, which this assumption the objective function is proposed to reduce overall energy consumption in the system and guarantee the task flow processing is complete on time. From the difference processing speed and communication capability of each server, an algorithm is developed to assign task flow into the cluster server in the cell within the path.

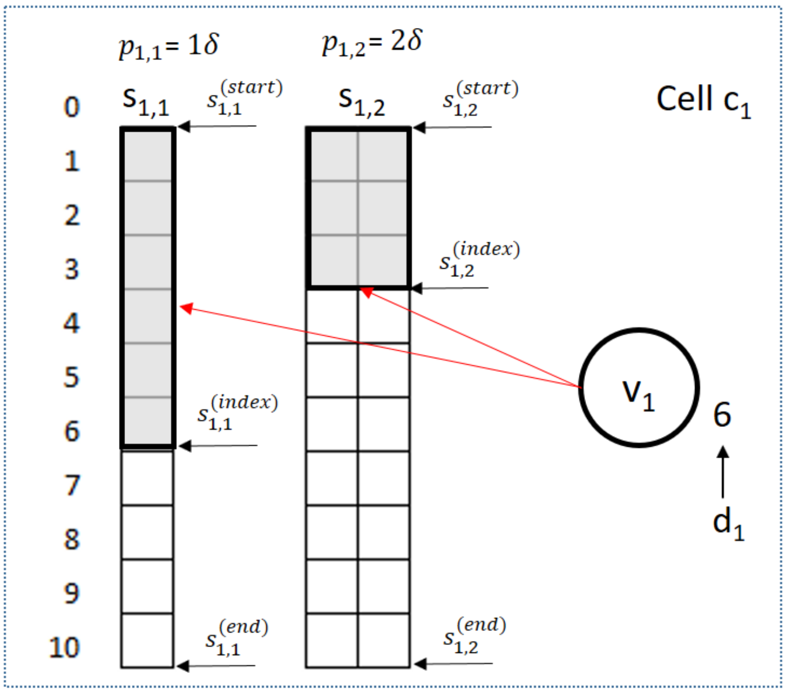


Figure 4.2: An example of task assignment in difference processing speed servers and algorithm parameters.

Figure 4.2 shows an example of assigned task $v_1$with data size $d_1 = 6$ from user task flow $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{D}, \boldsymbol{E})$ to the different processing speed servers $s_{i,j}$. The server $s_{1,1}$ has processing speed $p_{1,1} = 1\delta$ and server $s_{1,2}$ has $p_{1,2} = 2\delta$ that is the server cluster connected to cell $c_1$that have the duration time $T_1 = 10$. When the task d_1 assign to the difference processing speed server the time point of $s_{i,j}^{(index)}$ is also different.

From an example, If d_1 assign to $s_{1,1}$ the $s_{1,1}^{(index)}$is point to $6/1 = 6$, On the other hand, if d_1 assign to $s_{1,2}$ the $s_{1,2}^{(index)}$is point to $6/2 = 3$ because of the server has difference processing speed. $s_{i,j}^{(start)}$ is the time that server $s_{i,j}$ starts to process the receive data from user's device. Server end service time $s_{i,j}^{(end)}$ is the time that server $s_{i,j}$ stops processing the user's data.

Algorithm 2 Task Flow Assignment Algorithm.

Input: Given User's device task flow in set $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{D}, \boldsymbol{E})$.

Set of cell $\boldsymbol{C}$ from Algorithm 1.

Output: Scheduling of $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{D}, \boldsymbol{E})$ on cluster servers $s_{i,j}$.

```
1:  Leveling and sorting tasks in $\boldsymbol{G}$ by the size of $\tilde{p}(d_i)$ in each level into set
    $\boldsymbol{W}$.
2:  For $c_i \in \boldsymbol{C}$ do
3:      Set $s_{i,j}^{(index)} = c_i^{(start)}$ for all $j$.
4:      Set $s_{i,j}^{(end)} = c_i^{(end)}$ for all $j$.
5:      Set $time = c_i^{(start)}$.
6:      While $time < c_i^{(end)}$ do
7:          Get the first element $v_k$ from $\boldsymbol{W}$
8:          $\boldsymbol{W} = \boldsymbol{W} - \{v_k\}$
9:          Let $s_{i,j} = \arg\min_{s_{i,x} \in c_i} (s_{i,x}^{(index)} + t_{i,x}(d_k))$
10:         Assign $v_k$to $s_{i,j}$
11:         Update $s_{i,j}^{(index)} = s_{i,j}^{(index)} + t_{i,j}(d_k)$
12:         If $v_l$ is a child of $v_k$ such that $v_l$ does not have data dependency
            from other $v_a$ . then
13:             Assign $v_l$ to $s_{i,j}$ server.
14:         EndIf
15:     EndWhile
16: EndFor
```

An example of how algorithm two works. The user task flow is a large workload that cannot be processed in a single cell. By simulating a small circle, each task is a task that needs to be processed, and the link is the relationship between tasks.

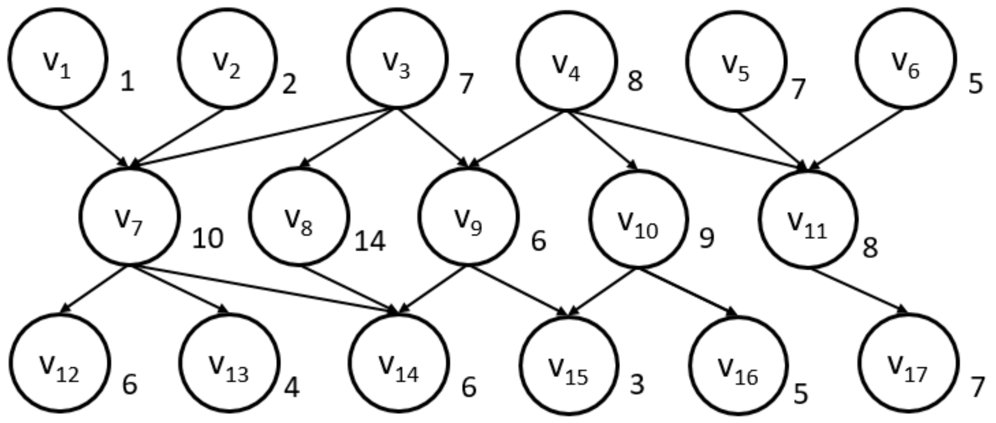


Figure 4.3: An example of user task flow for demonstrate algorithm.

Figure 4.1 shows the task flow from user's device $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{D}, \boldsymbol{E})$ which have 17 tasks from $v_1$to v_{17}. v_1 have data d_1= 1, which mean there has three level of independent tasks. The edge E represents the relation between tasks. For example v_7 is depends on $\{v_1, v_2, v_3\}$ it can't start to processing if the dependent tasks not complete processed.

Level 1 has tasks $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ with corresponding data sizes of $\{1,2,7,8,7,5\}$.

Level 2 has tasks $\{v_7,\ v_8, v_9, v_{10}, v_{11}\}$ with corresponding data sizes of $\{10, 14, 6, 9, 8\}$.

Level 3 has tasks $\{v_{12}, v_{13}, v_{14}, v_{15}, v_{16}, v_{17}\}$ with corresponding data sizes of $\{6, 4, 6, 3, 5, 7\}$.

Assume that the result of Algorithm 1 is shown as follows.

$c_1 = (0, 10, 3\delta, \{1\delta, 2\delta\})$.

$c_2 = (10, 10, 6\delta, \{2\delta, 2\delta, 2\delta\})$.

$c_3 = (25, 10, 4\delta, \{2\delta, 2\delta\})$.

The information from the cell is explained in detail as follows:

- c_1 has start service at time 0, service length $T_i = 10$, cell capability $\rho_1 = 3\delta$, and connects to servers $s_{1,1}$ and $s_{1,2}$ with processing speeds of 1δ and 2δ , respectively.
- c_2 has start service at time 10, service length $T_i = 10$, cell capability $\rho_2 = 6\delta$, and connects to servers $s_{2,1}$, $s_{2,2}$, and $s_{2,3}$, with processing speeds $2\delta, 2\delta$, and 2δ, respectively.
- c_3 has start service at time 25, service length $T_i = 10$, cell capability $\rho_3 = 4\delta$, and connects to servers $s_{3,1}$ and $s_{3,2}$, with processing speeds 2δ and 2δ, respectively.

After leveling and sorting level 1 task set by task size, the sorted tasks are $\{v_4, v_3, v_5, v_6, v_2, v_1\}$ with corresponding data sizes of $\{8, 7, 7, 5, 2, 1\}$. Table 4.2 shows the steps to assign the first level of user task flow $\boldsymbol{G}$ into servers in cell c_1.

Table 4.3: Step to assign task set level 1 from user's task flow into server in cell c_1, which duration time T_i = 10, $c_1^{(start)} = 0,$ $c_1^{(end)} = 10$, $p_{1,1} = 1\delta$, and $p_{1,2} = 2\delta$.

Tasks v_k	d_k	$s_{1,1}^{(index)}$	$s_{1,2}^{(index)}$	Assign to
Initial step	-	0	0	-
v_4	8	0	8/2 = 4	$s_{1,2}$
v_3	7	0	4	$s_{1,1}$
v_5	7	7/1 = 7	4	$s_{1,2}$
v_6	5	7	7.5 + (5/2) = 10	$s_{1,2}$
v_2	2	7	10	$s_{1,1}$
v_1	1	(7+2) = 9	10	$s_{1,1}$
Final step	-	10	10	All server are fully assign

The result of all step is shown in the following figure:

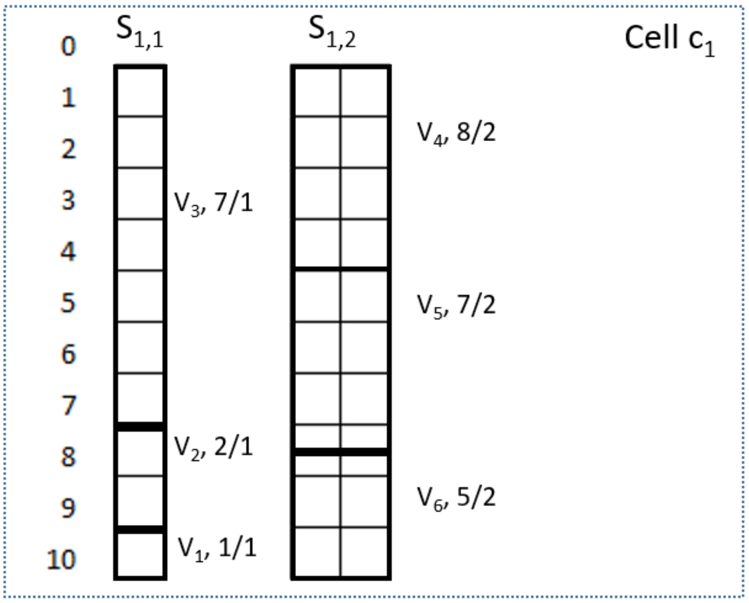


Figure 4.4: The result of task assignment to cell c_1.

All tasks is processed on servers $s_{1,1}$ and $s_{1,2}$ have returned the result to user before user exit the cell and disconnect. Next step is provided example of assign task set level 2 and 3 to next cell.

The new sorting of task set in level 2 is $\{ v_8, v_7, v_{10}, v_{11}, v_9\}$ with corresponding data sizes of $\{14, 10, 9, 8{,}6\}$. Task set Level 3 is $\{v_{17}, v_{12}, v_{14}, v_{13}, v_{16}, v_{15}\}$ with corresponding data sizes of $\{7, 6, 6, 5, 4, 3\}$.

Table 4.4: Step to assign task set level 1 from user's task flow into server in cell c_2, which service time = 10, and $p_{2,1} = 1\delta$, $p_{2,2} = 2\delta$. and $p_{2,3} = 2\delta$.

Tasks v_k	d_k	$s_{2,1}^{(index)}$	$s_{2,2}^{(index)}$	$s_{2,2}^{(index)}$	Assign to
Initial step	-	0	0	0	-
v_8	14	14/2=7	0	0	$s_{2,1}$
v_7	10	7	0	0	$s_{2,2}$
Found v_{12} that is a dependent task of v_7. v_{12} and does not have data dependency from other tasks Assign v_{12} to the same server $s_{2,2}$.					
v_{12}	6	7	(10/2)=5	10	$s_{2,2}$
v_{10}	9	7	5 + (6/2)=8	0	$s_{2,3}$
v_{11}	8	7	8	9/2 = 4.5	$s_{2,3}$
Found v_{17} that is a dependent task of v_{11} and does not have data dependency from other tasks .Assign v_{17} to the same server $s_{2,3}$.					
v_{17}	7	7	8	4.5+(4/2)=6.5	$s_{2,3}$
Server $s_{2,3}$ are full processing capability					
v_9	6	7	8	6.5+(7/2)=10	$s_{2,2}$
All tasks in task set level 2 already setting the remaining task is $\{ v_{14}, v_{16}, v_{13}, v_{15}\}$ which data $\{ 6, 5, 4, 3\}$.					
v_{14}	6	7 + (6/2) =10	7	10	$s_{2,2}$

Final step	-	10	7+(6/2)=10	10	-
The remaining tasks are $\{v_{13}, v_{16}, v_{15}\}$ the scheduler can alarm to user or notify to the automatic smart car to reduce speed to extends time to stay on cell c_2 rather than go to process on next cell.					

The benefit of the proposed algorithm is to reduce the transmission cost by process layer by layer and enhance merging the child that without dependent parent processing task. That provides low transmission cost between server.

4.3 Speed Adjustment

The speed adjustment algorithm was developed to guarantee the success of the process, using the principle of stretching or shrinking time from the speed at which it changes into the more extended service area of the network. The speed adjustment will be in the limited speed range by considering the speed of the following case.

Case 1: When it cannot be processed at the scheduled time, in order not to have to throw a job and re-do it in the next cell, it must be slowed down. That allows all jobs to be processed and sent back to the user's device.

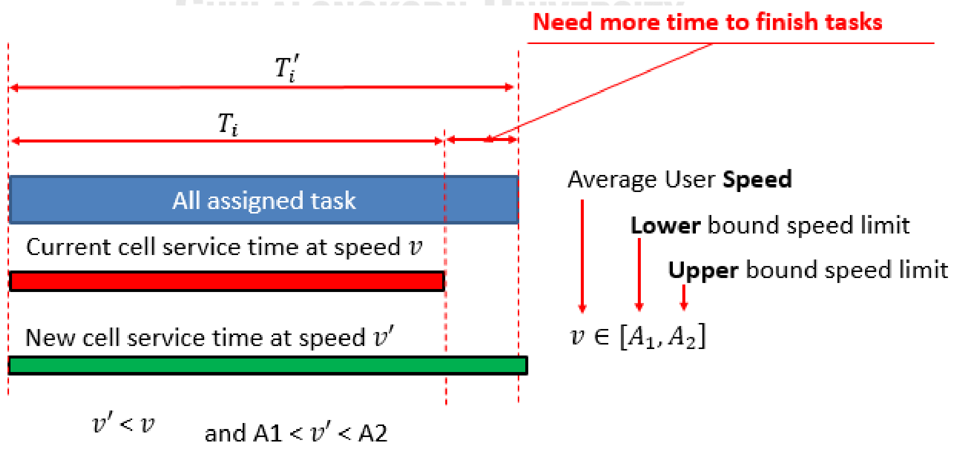


Figure 4.5: Simulation case of slow down speed adjustment

Case 2: When performance results are shorter than service times or due to slowdowns. During processing, there are two options: allocating new workgroups by considering the cost of re-scheduling or speeding up user movement to compensate for lost time.

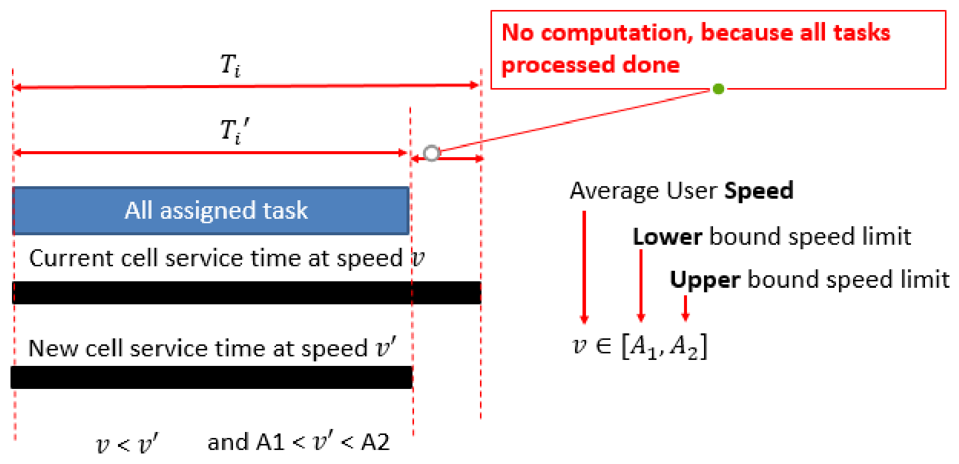

No computation, because all tasks processed done
T_i
T_i'
All assigned task
Current cell service time at speed v
New cell service time at speed v'
$v < v'$ and A1 < v' < A2
Average User **Speed**
Lower bound speed limit
Upper bound speed limit
$v \in [A_1, A_2]$

Figure 4.6: Simulation case of speed up adjustment to exit cell before the deadline.

Case 3: In case of passing through the cell while in progress, it is advised that performance is down due to user congestion. Accelerate this range to maintain overall efficiency of processing. Rush into cells with higher processing power (note may also be considered in the cell selection process).

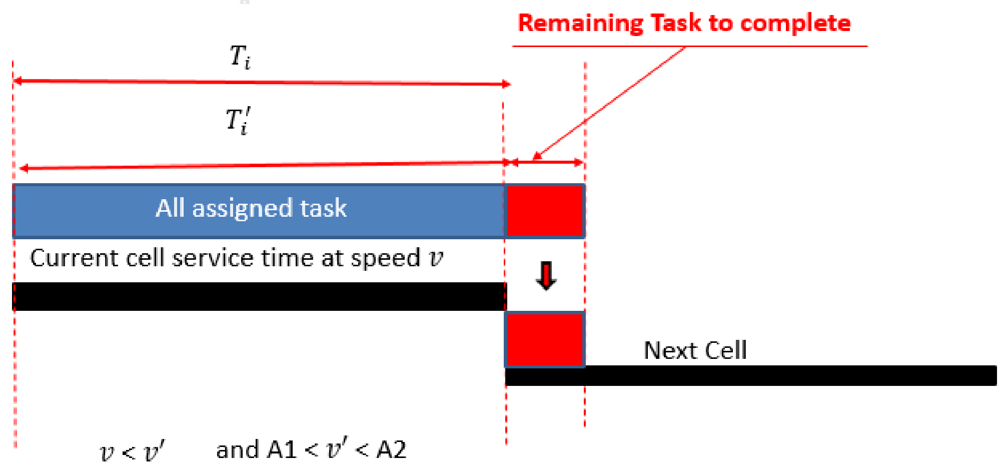

Remaining Task to complete
T_i
T_i'
All assigned task
Current cell service time at speed v
Next Cell
$v < v'$ and A1 < v' < A2

Figure 4.7: Simulation case of no changing speed.

From all of three cases, we can create the speed adjustment algorithm as follow.

```
Algorithm 3 Algorithm for Speed adjust.
Input: Information on cell $c_i$.
Output: The new recommended speed for $c_i$.
1:  Let $A_1$ and $A_2$ be the limit speed form eq. (3.10).
2:  Let $T_i$ be the estimated service time of $c_i$ from eq. (3.11).
3:  Let $T_i^{'}$ be the new service time for cell $c_i$.
4:  Let $v$ be the the average speed of the user's movement
5:  Let $\phi_i$ be the distance  of user path through cell $c_i$.
6:  If $T_i^{''} > T_i$ then
7:      Calculate new speed $v'$ from eq. (3.11).
8:          If new $v^{'} > A_1$ and $v^{'} < A_2$ then
9:              Decrease speed by set $v$ to $v'$.
10:         Else
11:             Don't change speed and notify the user.
12:         EndIf
12:     EndIf
13:     Return new speed $v$
```

The example to use the benefit of algorithm 3 can be explained continuously from the output of algorithm 2. We have the remaining task $\{v_{13}, v_{16}, v_{15}\}$ need more time to process all of them in c_2. The remaining time to process is $\{4, 5, 3\}$ and current T_i = 10, the new time we need to process the whole task set including the remaining task is 10 + (5/2) = 12.5 . Assume that the distance that user moving in cell $c_2$is $\phi_2 = 200m$, speed limit $A_1 = 50\ Km/h$, and $A_2 = 60\ Km/h$. The new speed is calculating from equation (3.12).

$$v' = \frac{200}{12.5} = 16\ \mathrm{m/s} = 57.6\ \mathrm{Km/h}$$

The new speed v' are in the range $[50, 60]$ that makes possible to process the whole task flow finished in c_2.

CHAPTER 5

EXPERIMENTS AND RESULTS

This section is describing the experimental setup, the evaluation measure, and the result, comparison of the proposed EAMS-HEC algorithm with other scheduling methods. The simulation is set by the control parameters, and the result shows the performance of makespan, transmission cost, and overall energy consumption.

5.1 Experimental Setup

This dissertation controls an experimental environment by defining the variable for the user task flow and the structure of the network cell as follow.

5.1.1 Random User Task Flow Generator

To evaluate the relative performance of the proposed algorithm, compare with others candidate. The DAG generation program was developed for the simulation which follows parameters.

- ***n:*** number of tasks (v_i) in user task flow ($\boldsymbol{G}$).
- **shape:** this parameter affects the high (number of levels) and the width (number of tasks in each level), can find by the number of levels divided by the average number of tasks in all level.
- **dep:** this is a number of child dependent task that relate to set $\boldsymbol{E}$ of task flow $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{D}, \boldsymbol{E})$.
- **CCR:** Communication to computation ratio. That affect to size of data d_i ,It is the ratio of the average communication cost to the average computation cost. If CCR value is too high, its mean that DAG is communication intensive.

The setting of parameters of user task flows for this experiment is show as follow:

- ***n*** = [20, 40, 60, 100, 200, 400];
- ***shape*** = [0.5, 1.0, 2.0];
- ***dep*** = [1, 2, 3, 4, 5];
- ***CCR*** = [0.1, 0.5, 1.0, 5.0, 10.0];

The example of generated task flow with the different parameters condition as shown in this section. From Figure 5.1,each task shows in rectangle and assume that $v_1 = T01$, , and assuming the direction of task flow is only top to bottom relation. The number on the right hand side of each rectangle is the estimated processing time $\tilde{p}(d_i)$ from definition 2.

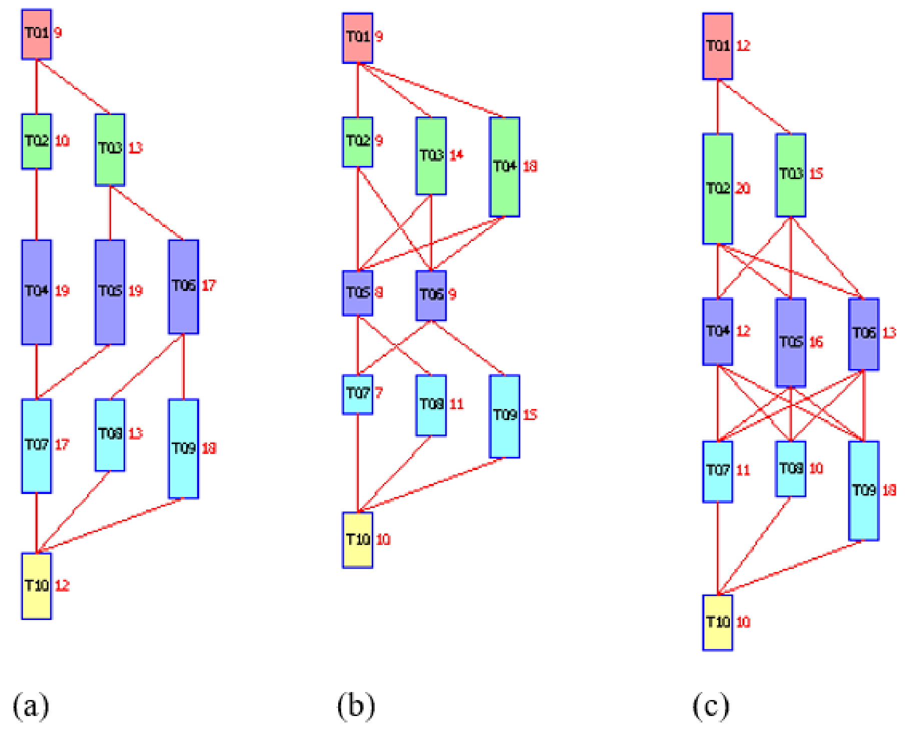


Figure 5.1: An example of task flow random generator with size n=10, shape=1.0: (a) low dependency dep=1, *(b) medium dependency* dep=2, *and (c) high dependency* dep=3.

Figure 5.1 shows the three examples of different random user task flow with various of *dep* parameter that use to evaluate the performance of loosely and tight dependency.

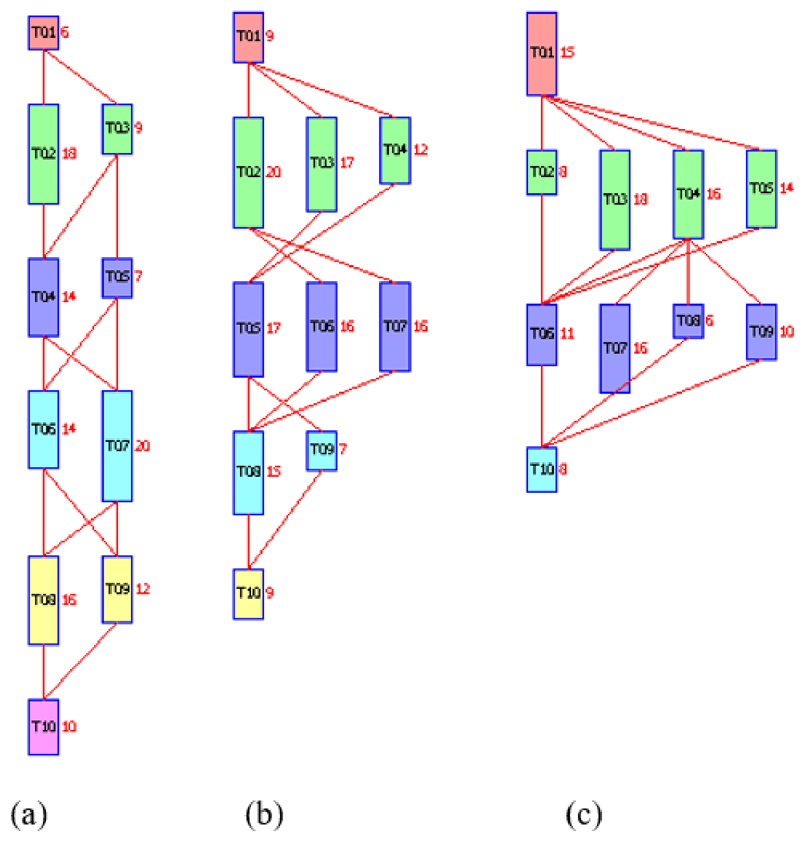
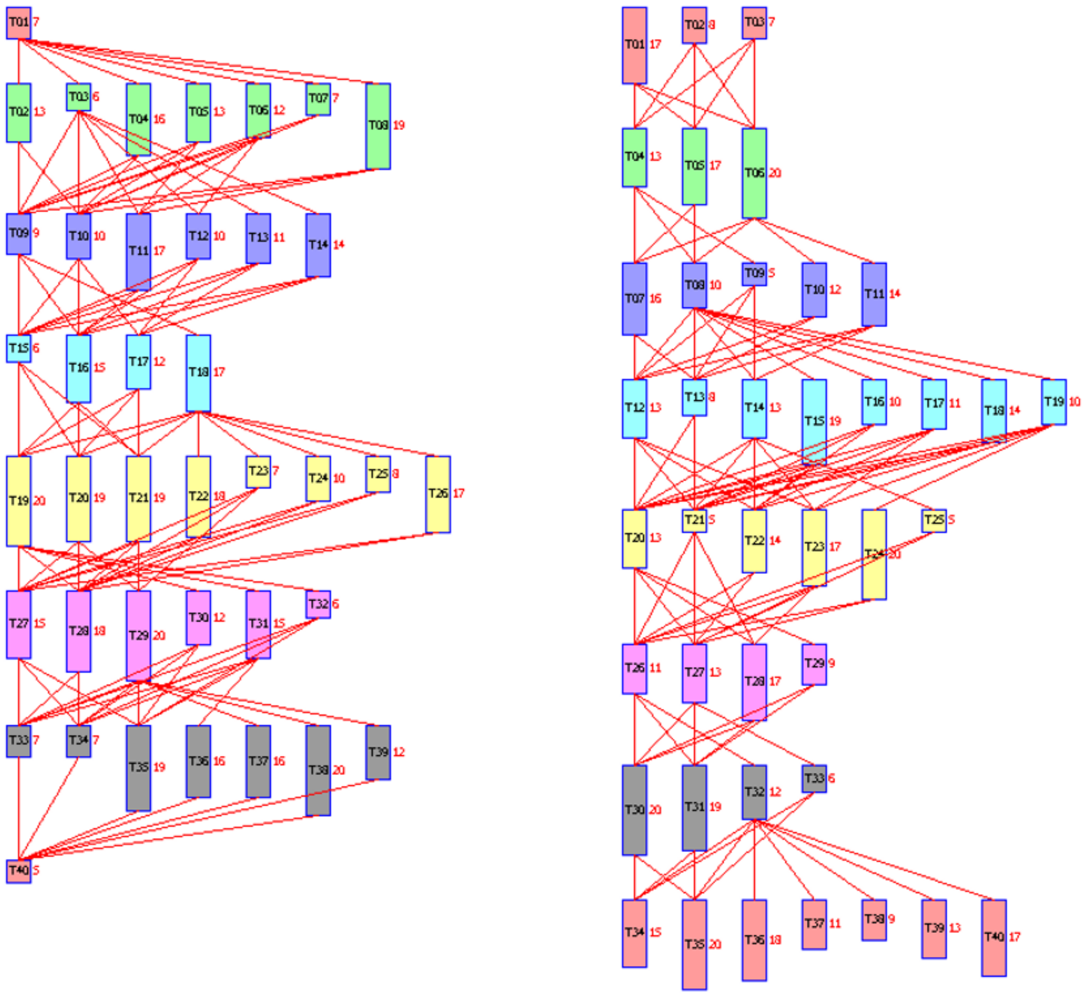


Figure 5.2: An example of task flow random generator with size n=10: (a) thin task flow shape=0.5, (b) symmetry task flow shape=1, and (c) fat task flow shape=2.

Figure 5.2 shows the difference values for the shape parameter that affect the task flow processing. From Figure 5.2a shows an average number of tasks in all level is 2, and the number of levels without start and end task is 4. Thus, *shape* is obtained from the ratio between 2 and 4 so *shape* is equal to 0.5. Figure 5.2b and Figure 5.2b obtain *shape* as 1.0 and 2.0 in respectively.

In addition, this experiment is evaluated on an application task flow that had multiple starts and multiple ends task which presented by the following figure.



(a) (b)

Figure 5.3: An example of generated task flow size n = 40, (a) task flow with one start and one end, (b) task flow with multiple starts and multiple end tasks.

5.1.2 Random Cells Generator

This dissertation separated the environment into two parts; the first part is described above in the section 5.1.1, the user task flow random generator. The computation environment is provided by the multiple hop small-cell networks, and each cell is connected by the different capability servers. The control parameters are defined as follows:

- n_i: number of cells c_i that user passes through the path.
- m_j: number of servers $s_{i,j}$ in each cell.
- T_i : service length of each cell, to simulate the coverage area for the mobile user which upon the signal strength of the network device.
- $p_{i,j}$: computation ratio that compares the server performance with the user device.
- *comm_ratio*: communication bandwidth for each server that affects the time to transmit data between user device and servers.

The setting parameters for network cell structure for this environment are shown as follow:

- n_i= [2, 3, 4, 5, 6, 8, 10];
- m_i = [2, 3, 4, 5, 6];
- T_i = [50, 100, 150, 200, 250, 300];
- $p_{i,j}$ = $[1\delta, 2\delta, 3\delta]$

Follow example shows the result of generated network cells architecture with the different setting parameters.

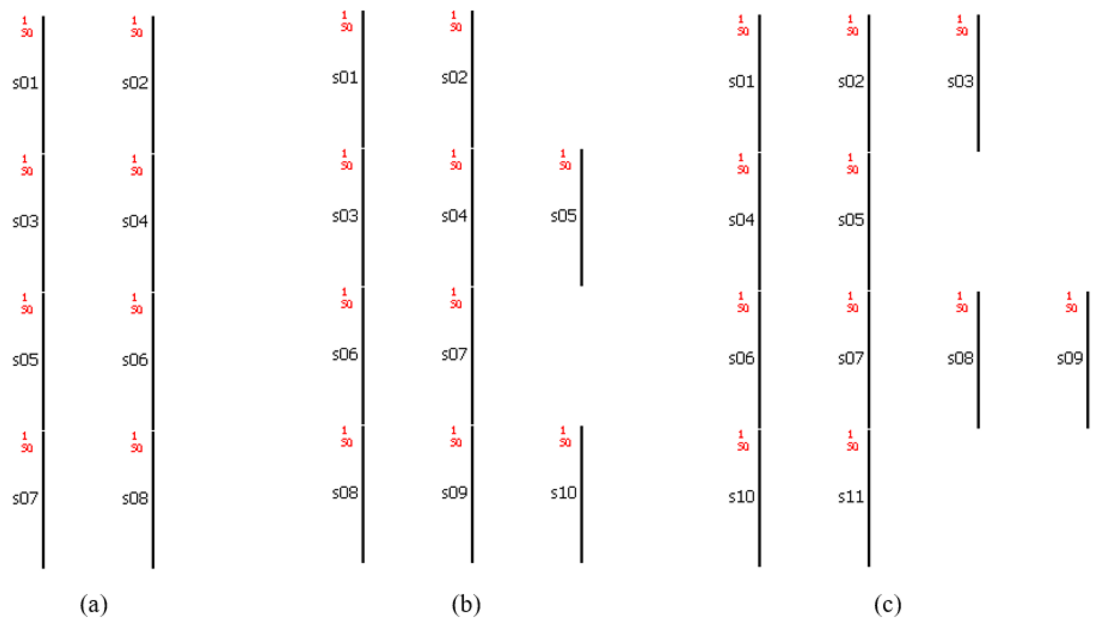


Figure 5.4: An Example of cells and servers random generator with T_i *= 50,* n_i*=4,* $p_{i,j} = 1\delta$*, number of servers (a)* m_j*=2, (b)* m_j *= 2 to 3, and (c)* m_j *= 2 to 4.*

The random generator tolls are created three different cells architecture for the environment as shown in Figure 5.4. All of them has four cells in the trips.

(a) has c_1= [1, 1] , c_2=[1, 1], c_3=[1, 1], and c_4=[1, 1].

(b) has c_1= [1, 1] , c_2=[1, 1, 1], c_3=[1, 1], and c_4=[1, 1].

(c) has c_1= [1, 1, 1] , c_2=[1, 1], c_3=[1, 1, 1, 1], and c_4=[1, 1].

Denote that in cell architecture (b) cell c_1 has two servers with performance $s_{1,1}$ has processing speed $p_{1,1} = 1\delta$ and $s_{1,2}$ has processing speed $p_{1,2} = 1\delta$ then the total performance of c_1 is 2δ. While cells c_2, c_3, and c_4 get 3δ, 2δ, and 3δ, respectively.

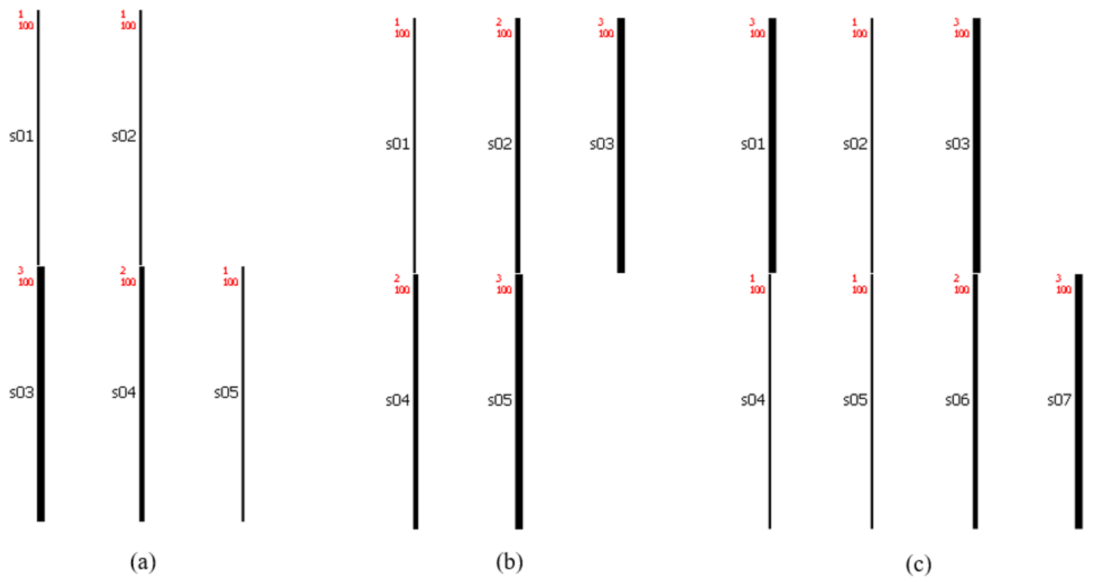


Figure 5.5: An example of cells and servers random generator with T_i = 100, n_i=2, number of servers (a) m_j=2, (b) m_j = 2 to 3, and (c) m_j = 2 to 4.

From figure 5.5 the random generator tolls are created three different cells architecture for the environment. All of them has four cells in the trips.

(a) has C_1 = [1, 1] , C_2=[3, 2, 1].

(b) has C_1 = [1, 2, 3] , C_2=[2, 3].

(c) has C_1 = [3, 1, 3] , C_2=[1, 1, 2, 3].

Denote that in cell architecture (c) cell c_1 has three servers with different performance $s_{1,1} = 3\delta$, $s_{1,2} = 1\delta$,and $s_{1,3} = 3\delta$ then the total performance of c_1 is 7δ. c_2 has four servers with different performance $s_{2,1} = 1\delta$, $s_{2,2} = 1\delta$, $s_{2,3} = 2\delta$, and $s_{2,4} = 3\delta$ then total performance of c_2 is 7δ.

5.2 Comparison Metrics

The performance of the proposed algorithm is evaluated by comparing with the other candidate algorithms are defined as follows.

1. **Makespan:** Makespan is the completion time of the whole user task flow from the start of the first tasks to the end of the last task in the task flow. The shortest makespan is considered as the best result.
2. **Communication link:** the number of communication between server to server in the same cell, and between server to the mobile user device that affects the transmission overhead. The lower number of the link is the best result.
3. **Communication cost:** the total cost of transmission in the time domain defined as a summation of all time use to send and receive data between server to server, and between servers to the mobile user device.
4. **Total energy:** The total conversion of energy usage for the whole system in one trip can calculate by using the equation in section 3.3.

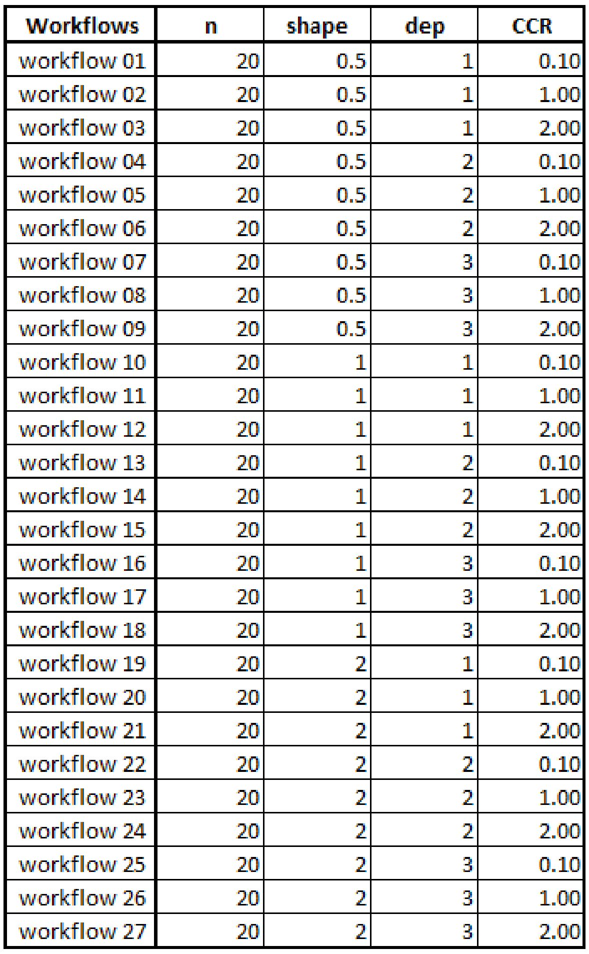
5.3 Performance Evaluation Results

The performance evaluation on the environment is separated into two scenarios; one cell and multiple cells scenario. The performance of the one cell scenario is measured on the environment of data without the deadline constraint. On the other hand, in the situation of multiple cells used to determine the time given in each cell by allowing the user to move at constant speed.

5.3.1 One cell scenario

This experiment is conducted with very long service length cells with difference server number and combination of user task flow size n = 20, shape= [0.5, 1, 2], dep= [1, 2, 3], and CCR = [0.1, 1.0, 10], the total is 27 type of random task flows. The result compares with HEFT, PEFT, and HETS is shown as follow.

Table 5.1: The parameters for randomly generated task flow on one cell scenario performance evaluation.



Workflows	n	shape	dep	CCR
workflow 01	20	0.5	1	0.10
workflow 02	20	0.5	1	1.00
workflow 03	20	0.5	1	2.00
workflow 04	20	0.5	2	0.10
workflow 05	20	0.5	2	1.00
workflow 06	20	0.5	2	2.00
workflow 07	20	0.5	3	0.10
workflow 08	20	0.5	3	1.00
workflow 09	20	0.5	3	2.00
workflow 10	20	1	1	0.10
workflow 11	20	1	1	1.00
workflow 12	20	1	1	2.00
workflow 13	20	1	2	0.10
workflow 14	20	1	2	1.00
workflow 15	20	1	2	2.00
workflow 16	20	1	3	0.10
workflow 17	20	1	3	1.00
workflow 18	20	1	3	2.00
workflow 19	20	2	1	0.10
workflow 20	20	2	1	1.00
workflow 21	20	2	1	2.00
workflow 22	20	2	2	0.10
workflow 23	20	2	2	1.00
workflow 24	20	2	2	2.00
workflow 25	20	2	3	0.10
workflow 26	20	2	3	1.00
workflow 27	20	2	3	2.00

Table 5.1 shows the task flows 1-9 is the thin shape task flow that has many chain tasks, task flow 10-18 is the symmetry shape task flow, and task flow 19-27 is the large task flow that has too many parallel tasks.

From the setting, environment follows this is the results of the proposed algorithm HAMS-HEC compare to HEFT and PEFT, HETS. Table 5.2 show the part of result of comparison between HAMS-HEC and HEFT, Other result for PEFT and HETS are compare in the same metrics.

Table 5.2: Comparison of makespan, links, communication cost and energy for HAMS-HEC and HEFT with 27 random task flows.

	HAMS-HEC				HEFT			
Workflows	makespan	links	comm	total energy	makespan	links	comm	total energy
workflow 01	**121**	**7**	**24**	**231**	**132**	**12**	**39**	**246**
workflow 02	120	9	119	330	120	9	112	323
workflow 03	124	**8**	**206**	**406**	117	**10**	**278**	**478**
workflow 04	147	**14**	**83**	**321**	146	**15**	**69**	**307**
workflow 05	**133**	**16**	**188**	**395**	**135**	**17**	**202**	**409**
workflow 06	**151**	15	424	687	**152**	15	409	672
workflow 07	**130**	17	97	311	**131**	17	97	311
workflow 08	**118**	18	204	402	**120**	17	199	397
workflow 09	**131**	17	341	547	**132**	17	341	547
workflow 10	138	**9**	**41**	**274**	138	**11**	**49**	**282**
workflow 11	116	**10**	**121**	**322**	116	**11**	**136**	**337**
workflow 12	138	**11**	**253**	**500**	135	**12**	**267**	**514**
workflow 13	139	16	88	338	139	13	67	317
workflow 14	132	**15**	**154**	**383**	132	**16**	**163**	**392**
workflow 15	127	**11**	**165**	**376**	123	**13**	**212**	**423**
workflow 16	135	**13**	**80**	**295**	127	**17**	**94**	**309**
workflow 17	142	17	191	424	140	17	191	424
workflow 18	145	**15**	**318**	**563**	145	**16**	341	**586**
workflow 19	142	**8**	**43**	**291**	138	**11**	**69**	**317**
workflow 20	147	11	159	423	146	11	148	412
workflow 21	218	24	24	429	211	24	24	429
workflow 22	120	14	90	291	119	14	75	276
workflow 23	126	13	137	365	125	13	134	362
workflow 24	**134**	**11**	**246**	**486**	**135**	**14**	**368**	**608**
workflow 25	140	16	87	329	135	16	87	329
workflow 26	**113**	16	114	318	**115**	16	114	318
workflow 27	132	17	87	330	128	13	76	319

Table 5.2 shows the performance comparison between the proposed algorithm and HEFT in the term of makespan, links, communication cost and energy. The result of performance evaluations is described with Better, Equal, and Worse, respectively. By which means the lowest value is the better

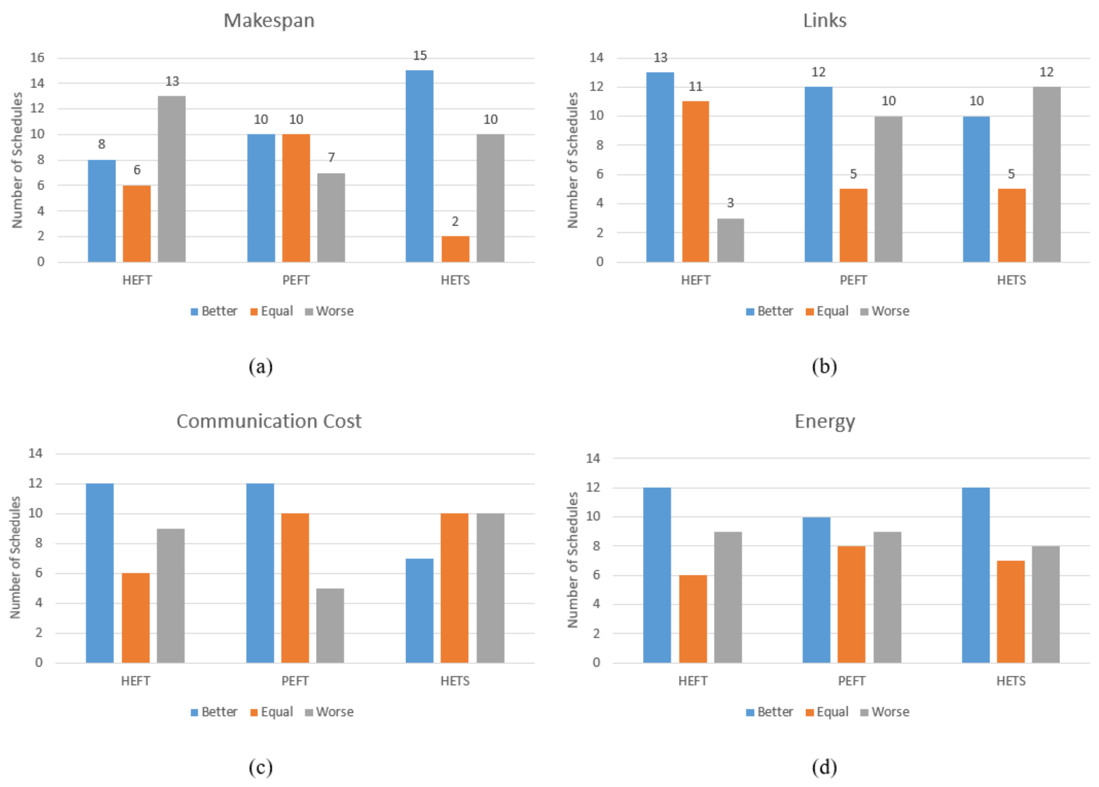


Figure 5.6: Comparison of quality of (a) makespan, (b) number of links, (c) communication cost, and (d) total energy

Figure 5.6 shows the performance of the proposed algorithm compare to three well-known heterogeneous algorithms HEFT, PEFT, and HETS. The result shows that the proposed algorithm is a kindly low performance in term of makespan compare with HEFT and higher performance than two other algorithms. From the result of a number of a communication link between server to server the proposed algorithm is a high performance to reduce communication link than HEFT and PEFT, But it nearly performance compare with HETS, because of its focus on the edges between tasks. The result of communication cost comparison the propose is higher performance than HEFT and PEFT related to the number of the communication link. The last chart is the compared to overall energy consumption; the proposed algorithm is a higher performance in the average on one cell scenario.

5.3.2 Multiple cells scenario

The next experiment is multiple cells scenario. This research generated eight types of task flow dataset which the random in 100 user task flow for each type. The cells architecture is also randomly generated into eight types of cells structure with the 100 randomly structure for each type. The number of simulation of one task flow type to multiple cells architecture is 100 x 800 = 8,000 scheduling. All of simulation of 8x8 is 8 x 8,000 = 64,000 case.

The structure of generated of user task flow is shown as follow tables.

Table 5.3: Setting parameter for the random generated of user task flow.

Symbol	$\tilde{p}(d_i)$	Number of edges E	d_i	Amount of dataset
SLL	Low	Low	Low	100
SLH	Low	Low	**High**	100
STL	Low	**High**	Low	100
STH	Low	**High**	**High**	100
LLL	**High**	Low	Low	100
LLH	**High**	Low	**High**	100
LTL	**High**	**High**	Low	100
LTH	**High**	**High**	**High**	100

Table 5.3 shows the parameter to set the generated task flow $G = (V, D, E)$ .The low $\tilde{p}(d_i)$ is mean each task is not the processing intensive task. And the low number of edge E is mean this task flow is low dependency task. When if the dependency to high it will affect the communication intensive task set due to the overhead of transmission and will giving high energy consume on the transmission. The low d_i is claimed that the task v_i is not the communication intensive. Can imply

that we can control the CCR rate by control the ratio of $\boldsymbol{\tilde{p}(d_i)}$ and $\boldsymbol{d_i}$. For example, the task set SLH, STH, LLH, LTH are generated by the high CCR rate and all of them is communication intensive task flow, Otherwise, is the computation-intensive task flow.

Table 5.4: Setting parameter for the random generated of cell architecture.

Symbol	$\boldsymbol{T_i}$	Variety of $\boldsymbol{p_{i,j}}$	Variety of $\boldsymbol{T_i}$	Amount of dataset
SSS	Low	Low	Low	100
SSV	Low	Low	**High**	100
SMS	Low	**High**	Low	100
SMV	Low	**High**	**High**	100
LSS	**High**	Low	Low	100
LSV	**High**	Low	**High**	100
LMS	**High**	**High**	Low	100
LMV	**High**	**High**	**High**	100

Table 5.4 shows the parameter for the randomly generated cell architecture including the server cluster. The control parameter T_i provides to control the service length of each cell that user's device pass through, The notation Low is mean that cell are very short time to service the processing for user's device. The variety of $p_{i,j}$ is setting the difference processing speed on each server $s_{i,j}$ in each cell the High value mean that the high different in term of processing speed. The variety of T_i provides the control of different service length of cell in the user path, High value means some cell may be had too short service length, and some cell are too long service length.

The simulation is running on 64,000 cases to compare the performance in term of makespan, overall energy consumption, communication number, communication cost, number of a gap (the remaining gap on each server), slack, and the average performance, respectively. The result is scheduling base on each cell structure by eight type of user's task flow structure.

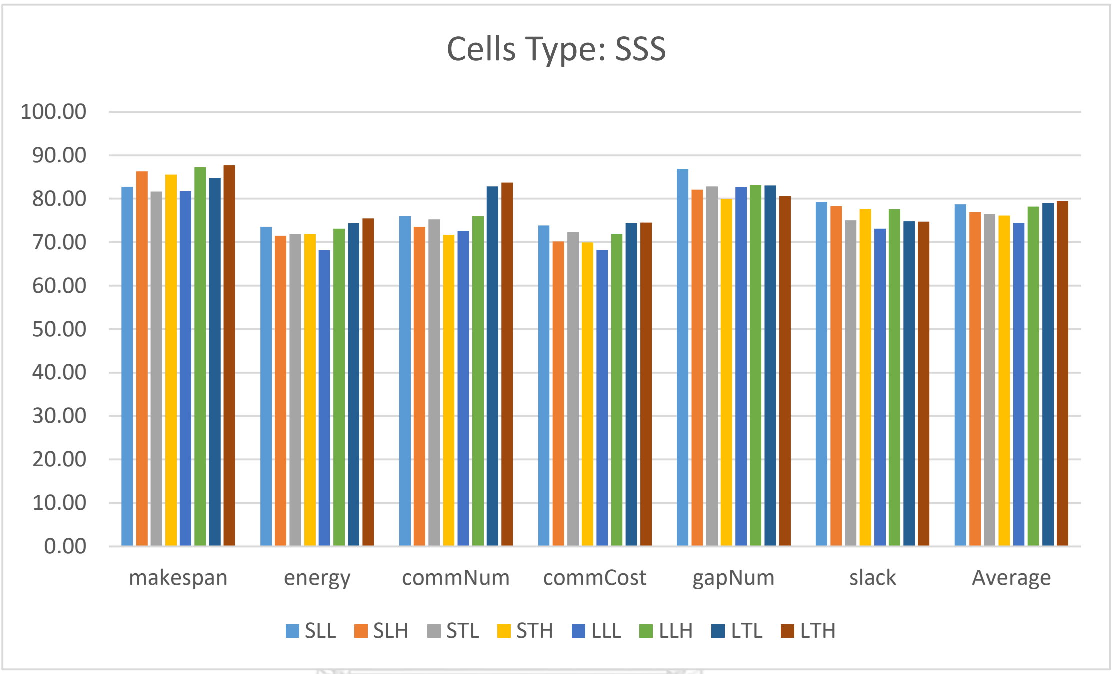


Figure 5.7: The comparison of 100 cell structure type SSS to 8x100 task flow types.

Figure 5.7 shows the result of the evaluation algorithm performance in comparison with candidate algorithms while scheduling the random generated eight types x 100 data set into the arbitrary cell structure type SSS. The SSS type is defined for short service length, low variety of processing speed and all cell is quite same service length. For this kind of cell structure, it affects to the energy consumption that related to the communication number and communication cost. Because the service time is too short, that can't assign that much task to the server in each cell that is generated many transmission overheads. The task flow type LTL also affect this type of structure because this task structure is high computation consume on each task and to many dependency tasks to wait before process the child task.

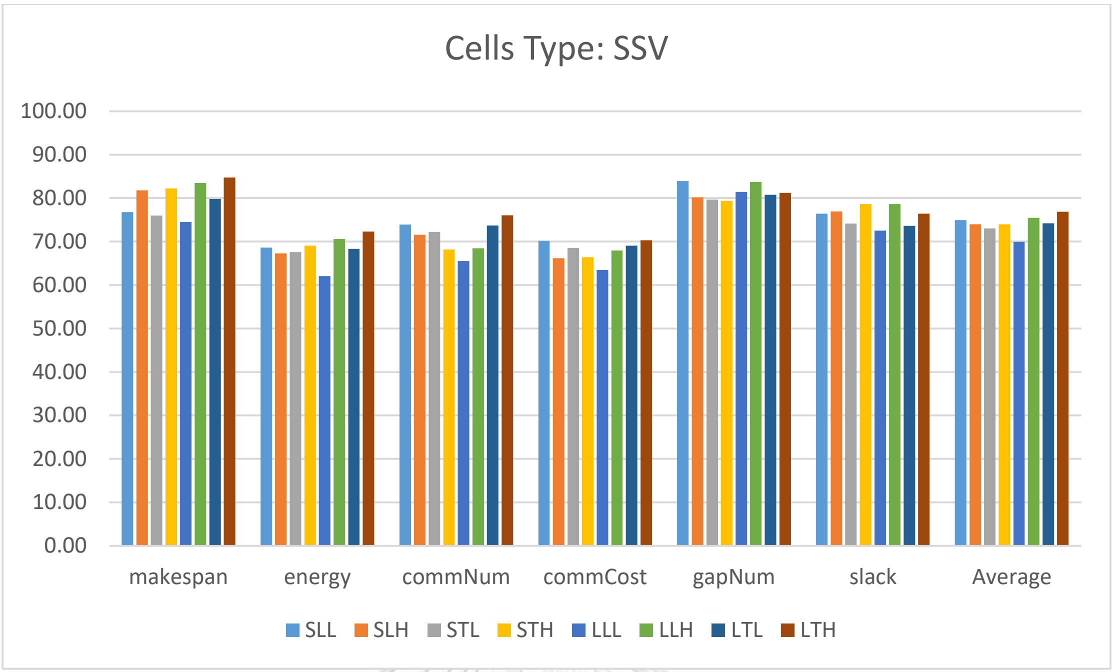


Figure 5.8: The comparison of 100 cell structure type SSV to 8x100 task flow types.

Figure 5.8 shows the result of the algorithm performance compares with candidate algorithms while scheduling the random generated eight types x 100 data set into the arbitrary cell structure type SSV. The SSS type defined for the short service length, low variety of processing speed but all cell is different the service length. For this kind of cell structure, it has affected the overall energy consumption due to the variety of service length. The service time is hard to schedule that also affects to the high computing intensive and high dependency task flow LTL.

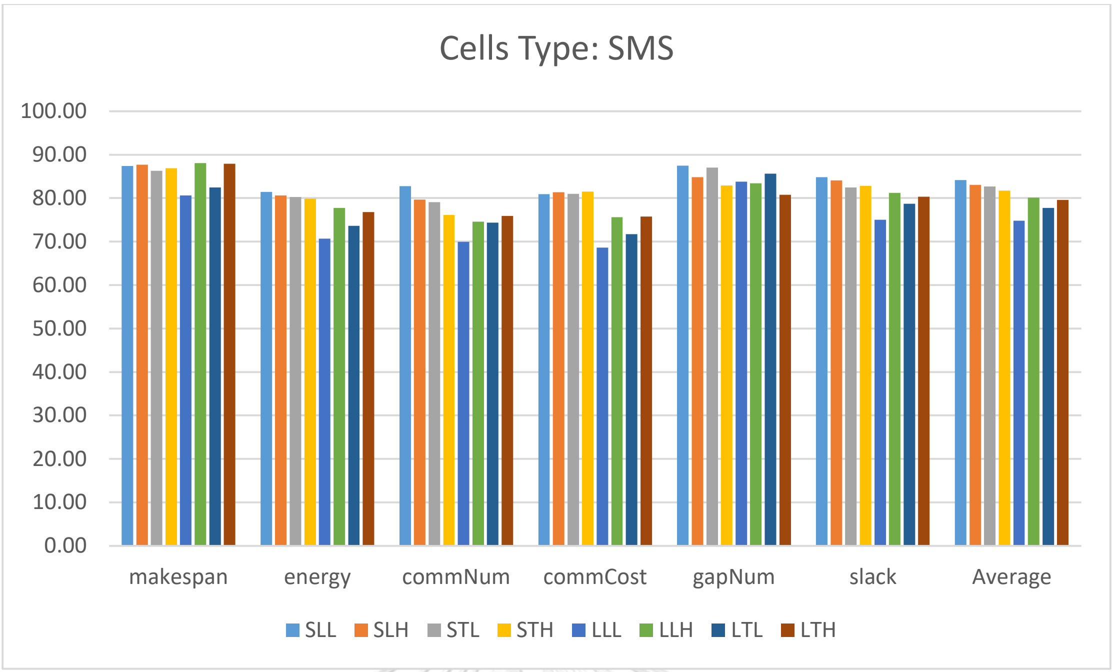


Figure 5.9: The comparison of 100 cell structure type SMS to 8x100 task flow types.

Figure 5.9 shows the result of the algorithm performance compares with candidate algorithms while scheduling the random generated eight types x 100 data set into the arbitrary cell structure type SMS. The SMS type define for the short service length, too high different of processing speed but all cell has the same service length. For this kind of cell structure, it affects the overall energy consumption due to the variety of service length and processing speed of the server cluster. The user's task flow that affects to this cell structure is LLL because it is a computation intensive task flow type, to scheduling on the cell structure that short service length and variety of processing speed is not getting the good result.

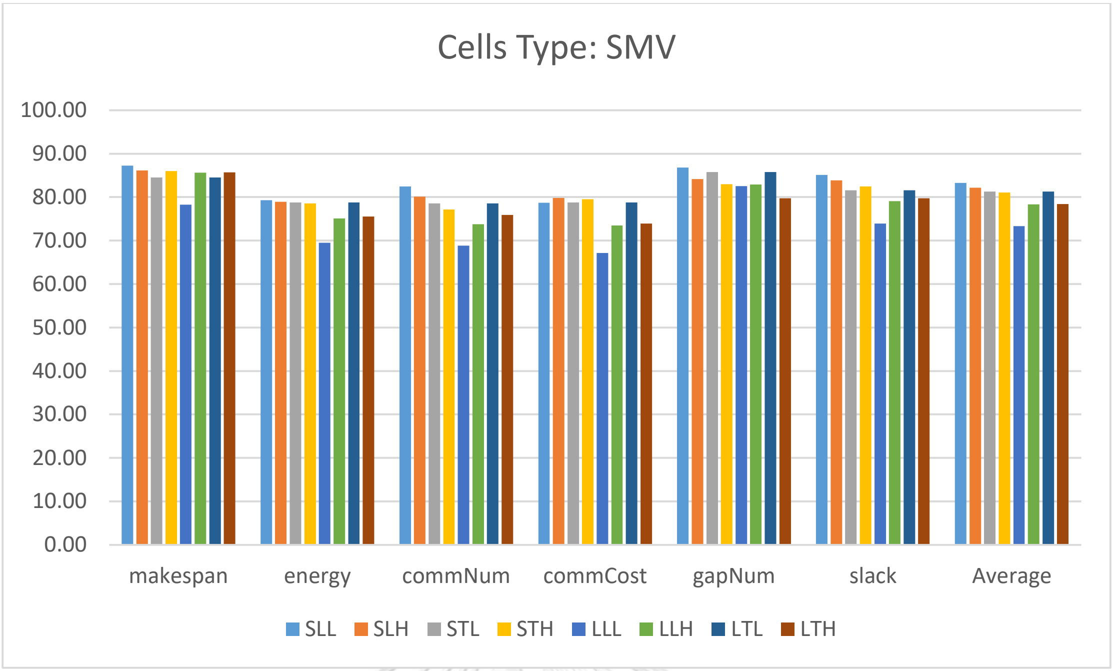


Figure 5.10: The comparison of 100 cell structure type SMV to 8x100 task flow types.

Figure 5.10 shows the result of the algorithm performance compares with candidate algorithms while scheduling the random generated eight types x 100 data set into the arbitrary cell structure type SMV. The SMV type define for the short service length, too high different of processing speed and diffident of service length. For this kind of cell structure, it has affected the overall energy consumption due to the variety of service length and processing speed of the server cluster. The user's task flow that affects to this cell structure is LLL because it is a computation intensive task flow type, to scheduling on the cell structure that short service length and variety of processing speed is not getting the good result.

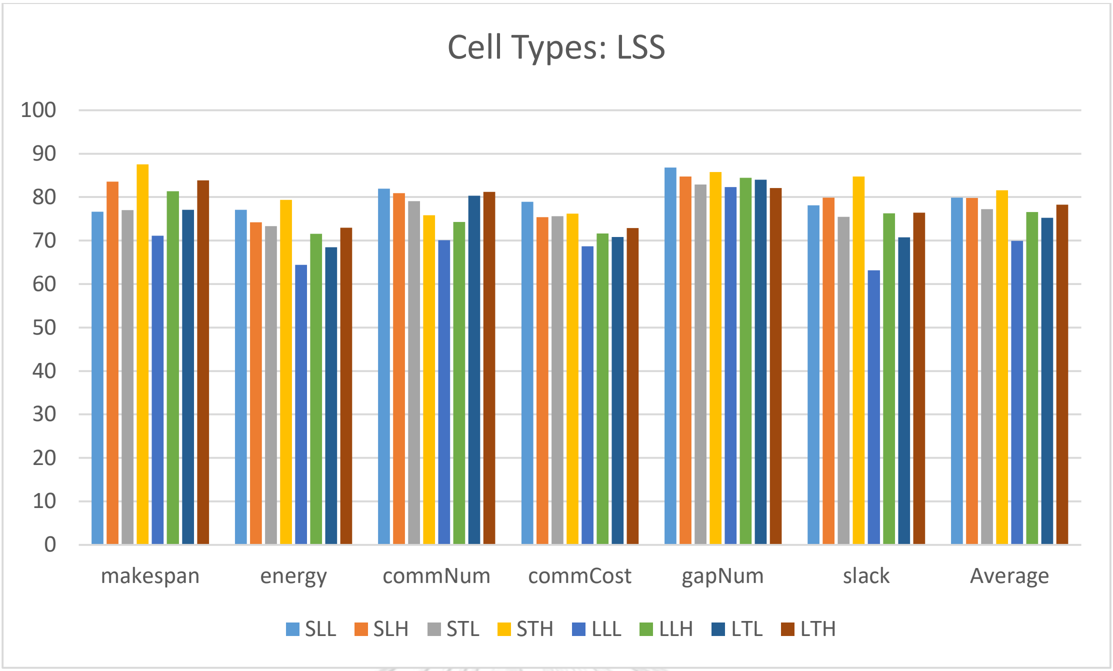


Figure 5.11: The comparison of 100 cell structure type LSS to 8x100 task flow types.

Figure 5.11 shows the result of the algorithm performance compares with candidate algorithms while scheduling the random generated eight types x 100 data set into the arbitrary cell structure type LSS. The LSS type define for the long service length, with the same processing speed and service length. For this kind of cell structure, it has long duration time to stay in the cell. The result show that the scheduling on average are the get the result nearly to the candidate algorithm. Except for the task flow structure type LLL gives lower than 70 percent win in term of energy. The LLL task flow is the high computation intensive task type and hard to use the balance of server index to scheduling. It also affects other candidate algorithms too.

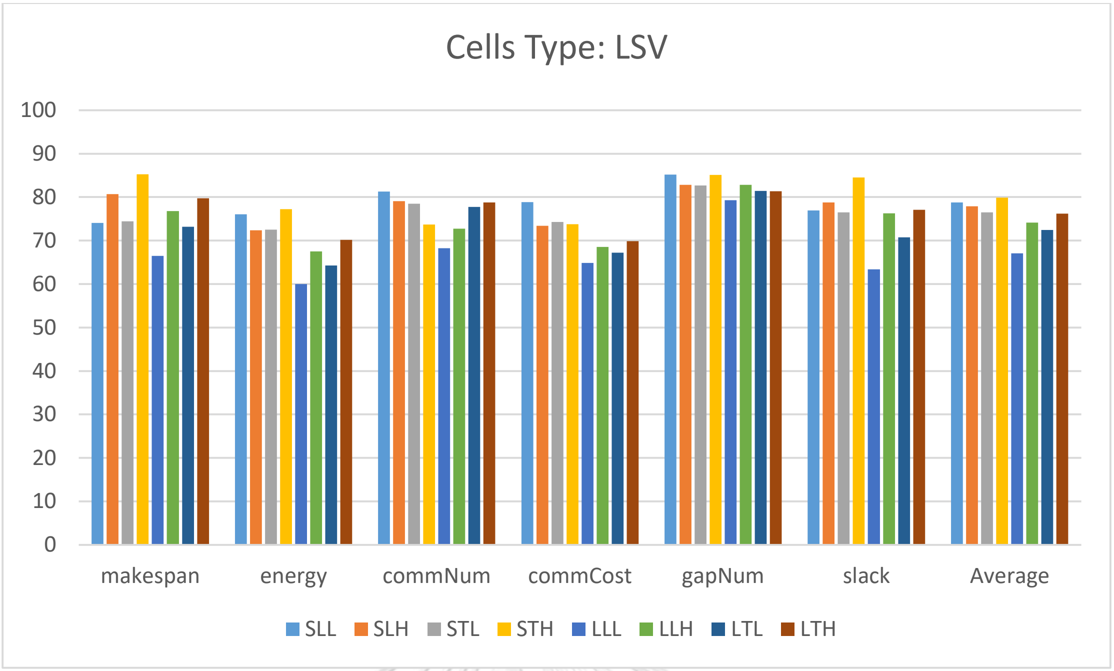


Figure 5.12: The comparison of 100 cell structure type LSV to 8x100 task flow types.

Figure 5.12 shows the result of the algorithm performance compares with candidate algorithms while scheduling the random generated eight types x 100 data set into the arbitrary cell structure type LSV. The LSV type is the long service length, with the same processing speed but to a variety of service length. For this kind of cell structure, it has long duration time to stay in the cell. The result is the lowest overall performance, especially to the energy consumption. The task flow type LLL is getting impact too. From the chart, the LLL type win in percentage is lower than 60 percent. However, another candidate also gets this effect too.

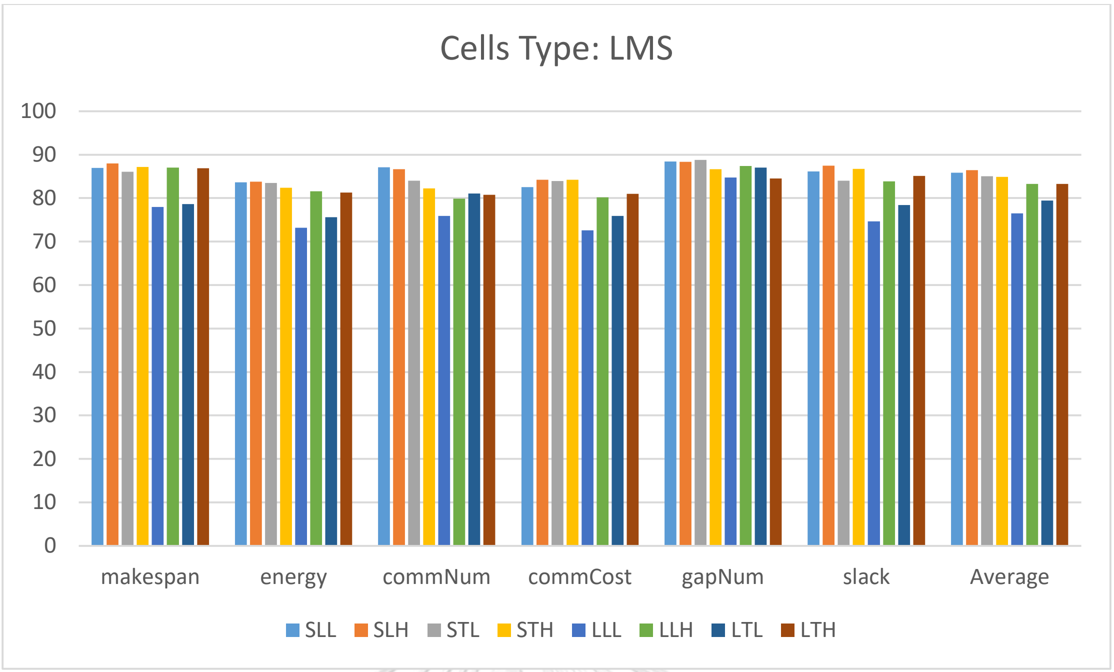


Figure 5.13: The comparison of 100 cell structure type LMS to 8x100 task flow types.

Figure 5.13 shows the result of the algorithm performance compares with candidate algorithms while scheduling the random generated eight types x 100 data set into the arbitrary cell structure type LMS. The LMS type is the long service length, with the different processing speed but has same service length. For this kind of cell structure, it has long duration time to stay in the cell. The average result is too high due to we have the time to balance the server load with the server index, and the task can complete immediately. The task flow structure type LLL is also getting the impact, but the result is better than other cell structures.

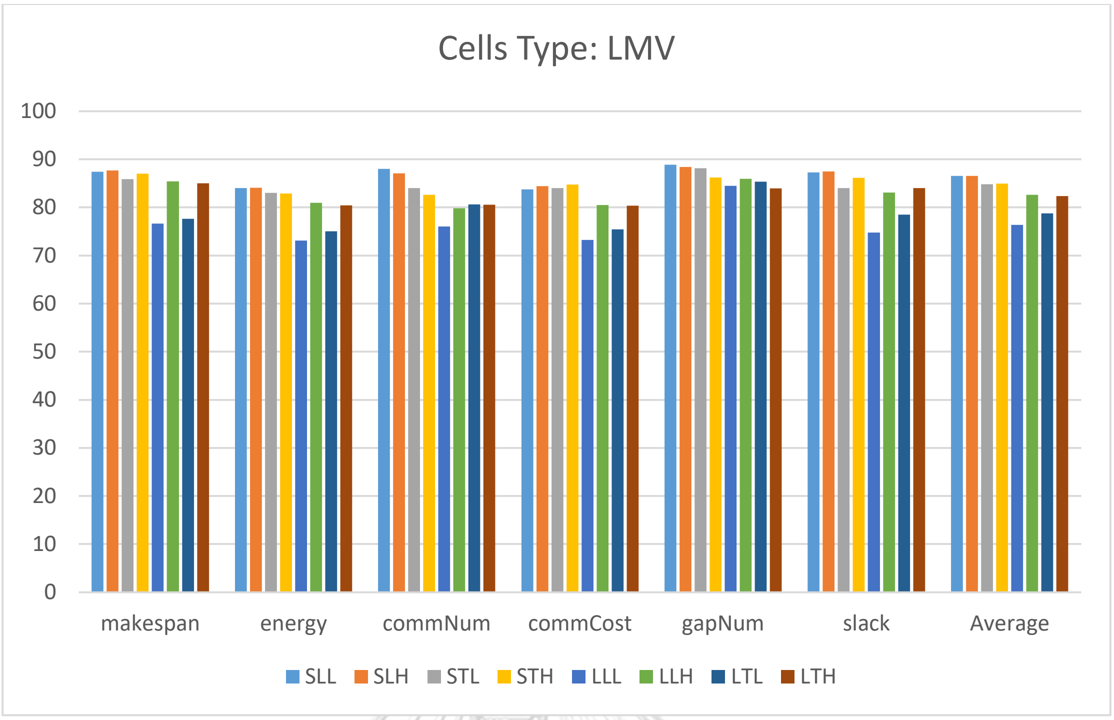


Figure 5.14: The comparison of 100 cell structure type LMV to 8x100 task flow types.

Figure 5.14 shows the result of the algorithm performance compares with candidate algorithms while scheduling the random generated eight types x 100 data set into the arbitrary cell structure type LMV. The LMS type is the long service length, with the different processing speed and variety of service length. For this kind of cell structure, it gets the best result compared to other seven cell structure. The task flow type LLL and LTL.

it has long duration time to stay in the cell. The average result is too high due to we have the time to balance the server load with the server index, and the task can complete immediately. The task flow structure type LLL is also getting the impact, but the result is better than other cell structures.

In summary concludes, Short service duration affects overall performance, the different of processing speed effect to the overall energy consumption, and the variety of length affect to makespan and the overall energy consumption.

5.3.3 Energy saving comparison

The performance value in term of energy consumption can explain by downscale of the randomly generated task flow and cell structures. The simulation is setting the control parameter from section 5.11, and 5.12, to point the benefit of the proposed algorithm in the energy consumption reduction. The setting is set as follow:

1) Compare one random generated user's task flow to 20 random generated cell structure. The example that, scheduling $\boldsymbol{G}$ to $\boldsymbol{C_1}$, $\boldsymbol{G}$ to $\boldsymbol{C_2}$ until $\boldsymbol{G}$ to $\boldsymbol{C_{20}}$.
2) Compare 50 random generated user's task flow to one random cell structure. Explain that assign $\boldsymbol{G_1}$ to $\boldsymbol{C}$, $\boldsymbol{G_2}$ to $\boldsymbol{C}$, until $\boldsymbol{G_{50}}$ to $\boldsymbol{C}$.
3) Compare 50 random generated user's task flow to 50 random cell structure, which schedules one to one. It means that assign $\boldsymbol{G_1}$ to $\boldsymbol{C_1}$, and $\boldsymbol{G_2}$ to $\boldsymbol{C_2}$ until $\boldsymbol{G_{50}}$ to $\boldsymbol{C_{50}}$.

The result of the simulation is shown as follow:

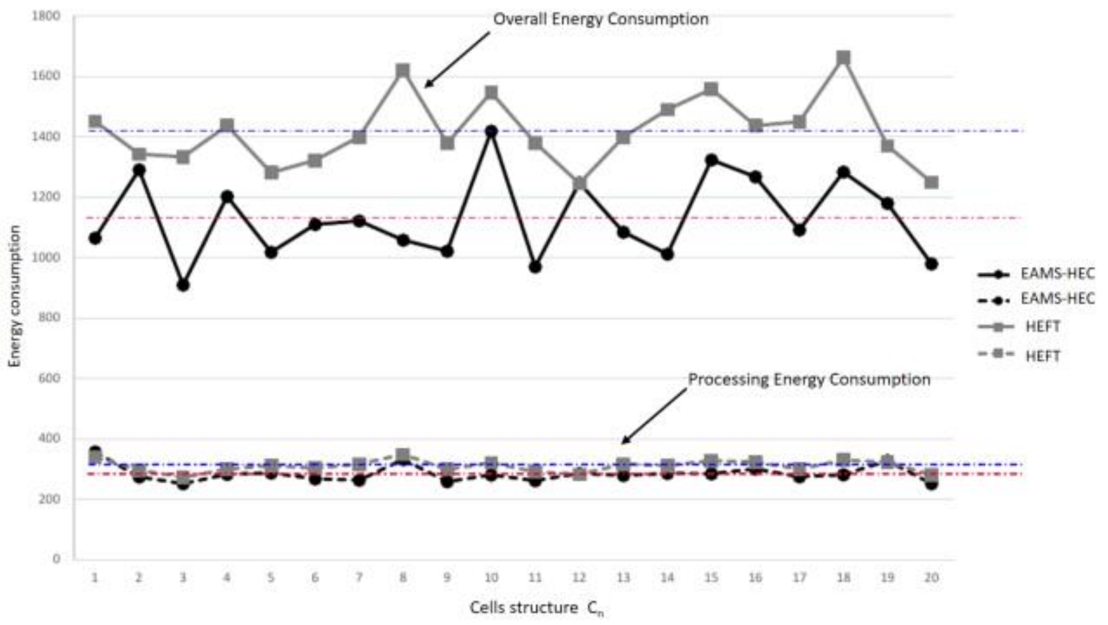


Figure 5.15: The energy consumption results from scheduling by assign one task flow to 20 random cell structure.

Figure 5.15 shows an average energy consumes on the server Γ_S of the proposed algorithm is 284.65 compared to the heuristic algorithm HEFT is 310.1. That means an algorithm EAMS-HEC can save 8.2% of processing energy consumption. In the cell structure 12, the result is very closet value, its mean that the generated structure provides the best effect for heuristic scheduling at the same situation of the proposed algorithm, explain that it cannot be grouping, swapping and manage the communication channel too much. Apparently, the average total energy consumption Γ_G of the propose algorithm is 1133.05 and significantly less than heuristic algorithm 1417.95. Imply that, we can save 20% of the overall energy consumption.

The result of compared 50 random generated user's task flow to one random cell structure is shown as follow.

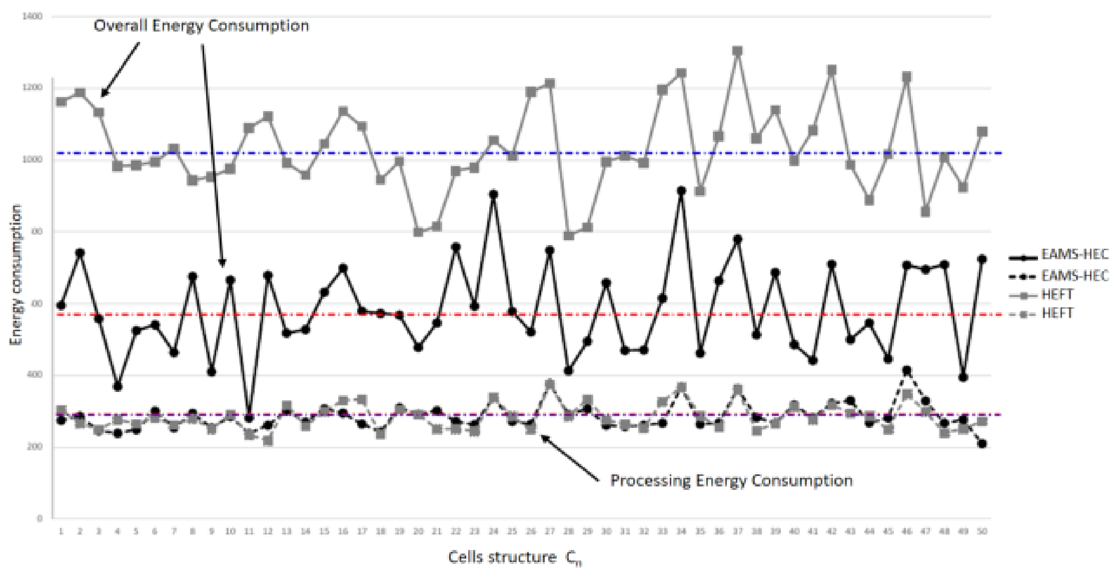


Figure 5.16: The energy consumption results of scheduling by assign 50 random generated user's task flow to one random cell structure.

Figure 5.16 shows the benefit of the proposed algorithm in case of the multi-structure of user's task flow on the repeat usage structure that implies to the end user run the different application which generated the different shape of the task flow and scheduling on the cell structure that they pass through every day. The result shows that an average of total energy consumption can save more than 40% compared to the heuristic algorithm. The proposed algorithm is mainly saved energy from the communication overhead. In the other word, this algorithm can reduce the transmission latency between user's device and servers that can improve the quality of services, and provide the best quality of experience to the end users.

The last comparison is to random generated 50 user's task flow to 50 random cell structure; the result is shown as follow.

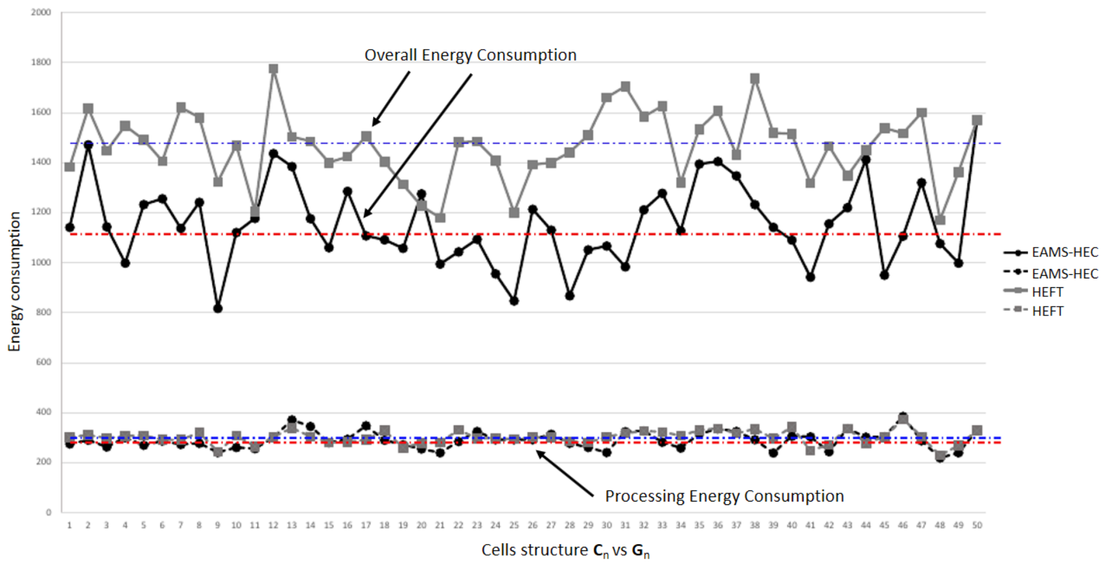


Figure 5.17: The energy consumption results from scheduling by assign 50 random generated user's task flow to 50 random cell structure.

Figure 5.17 shows the result the proposed algorithm in the situation of the scheduling a random user's task flow to the arbitrary cell structure. The simulates scenario of the user use the different application that generated different task flow and schedule to the cell structure that provides on the different path. The result also claims that the proposed algorithm can save the overall energy for the multiple random situations more than 20% compared to the heuristic algorithm.

CONCLUSION

The result of simulation presented the proposed algorithm with the scenario of a smart car, and smart city environment is outperformed in perspective of energy consumption and communication latency. The direct effect is it can help to increase the quality of service and quality of experience. It can be applying to the next future application development to utilize the edge computing to handle high energy consume work and make the more powerful application and service without interrupting the user experience.

- Choosing a wireless network for processing by taking into account the overall performance and cutting off a low-efficient server can help reduce overall power consumption by enabling interoperability. There are no gaps and waiting times.
- Grouping by transverse cutting can help reduce waiting times in parallel processing. Continuous processing of workloads and consideration of workload relationships in processing schedules can reduce the number of data transfers resulting in reduced power consumption.
- Speed adjust method can be used to compensate for the processing time and is suitable for use with intelligent car systems, automatic robots, or unmanned aerial vehicles.
- The combination of all three algorithms: Cell selection, Task Assignment, and Speed adjustment is more powerful and can guarantee all work can be complete.
- All datasets of the user task flow and cell structure constructed from the devices used in this study can be used to simulate the mobile edge computing research environment. The research is planned to be open source to develop this field further.

- Based on the EAMS-HEC algorithm, all data sets have an average winning rate of 75% for both the makespan, energy consumption, communication overhead, and all work guaranteed to complete processing.
- To compare with the state-of-the-art scheduling EAMS-HEC are outperform on energy saving to 20%.

Discussion and Future Works

The mobile edge computing and energy-aware scheduling are many topics that so interesting and practical for the mobile network and IoT industry shows as follow.

- scheduling for multiple user's devices that connect to the same cell.
- scheduling for a non-stationary server with the different service length.
- The offloading framework for micro-service on mobile edge computing architecture.

REFERENCES

1. Salman, O., et al., *Edge computing enabling the Internet of Things.* IEEE World Forum on Internet of Things, WF-IoT 2015 - Proceedings, 2015: p. 603-608.
2. Tawalbeh, L., Y. Jararweh, and F. Dosari, *Large Scale Cloudlets Deployment for Efficient Mobile Cloud Computing.* 2015. **10**: p. 70-76.
3. Oueis, J., et al., *On the impact of backhaul network on distributed cloud computing.* 2014 IEEE Wireless Communications and Networking Conference Workshops, WCNCW 2014, 2014: p. 12-17.
4. Topcuoglu, H., S. Hariri, and I.C. Society, *Performance-Effective and Low-Complexity.* 2002. **13**: p. 260-274.
5. Arabnejad, H. and J.G. Barbosa, *List scheduling algorithm for heterogeneous systems by an optimistic cost table.* IEEE Transactions on Parallel and Distributed Systems, 2014. **25**: p. 682-694.
6. Becvar, Z., J. Plachy, and P. Mach, *Path selection using handover in mobile networks with cloud-enabled small cells.* IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC, 2014. **2014-June**: p. 1480-1485.
7. Eswari, R. and S. Nickolas, *Path-Based Heuristic Task Scheduling Algorithm for Heterogeneous Distributed Computing Systems.* Advances in Recent Technologies in Communication and Computing (ARTCom), 2010 International Conference on, 2010.
8. Durillo, J.J., H.M. Fard, and R. Prodan, *MOHEFT: A multi-objective list-based method for workflow scheduling.* CloudCom 2012 - Proceedings: 2012 4th IEEE International Conference on Cloud Computing Technology and Science, 2012: p. 185-192.
9. Xie, G., et al., *A High-performance DAG Task Scheduling Algorithm for Heterogeneous Networked Embedded Systems.* 2014.
10. Xie, G., et al., *Energy-efficient Scheduling Algorithms for Real-time Parallel Applications on Heterogeneous Distributed Embedded Systems.* IEEE Transactions on Parallel and Distributed Systems, 2017. **28**: p. 3426-3442.

11. Yu, J., *A Budget Constraint Scheduling of Workflow Applications on Utility Grids using Genetic Algorithms.* Workshop on Workflows in Support of Large-Scale Science (WORKS06), 2006: p. 1-10.

12. Al-Maamari, A. and F.a. Omara, *Task Scheduling using Hybrid Algorithm in Cloud Computing Environments.* IOSR Journal of Computer Engineering, 2015. **17**: p. 2278-661.

13. Chen, W.-N. and J.Z.J. Zhang, *An Ant Colony Optimization Approach to a Grid Workflow Scheduling Problem With Various QoS Requirements.* IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2009. **39**: p. 29-43.

14. Chun, B., et al., *Clonecloud: Elastic Execution Between Mobile Device and Cloud.* Proceedings of the sixth conference on Computer systems, 2011: p. 301-314.

15. Thanavanich, T. and P. Uthayopas, *Efficient energy aware task scheduling for parallel workflow tasks on hybrids cloud environment.* 2013 International Computer Science and Engineering Conference, ICSEC 2013, 2013: p. 37-42.

16. Rodriguez, M.A. and R. Buyya, *and Scheduling Algorithm for Scientific Workflows on Clouds.* 2014. **2**: p. 222-235.

17. Vondra, M. and Z. Becvar, *QoS-ensuring distribution of computation load among cloud-enabled small cells.* 2014 IEEE 3rd International Conference on Cloud Networking, CloudNet 2014, 2014: p. 197-203.

18. Satyanarayanan, M., et al., *The Case for VM-Base Cloudlets in Mobile Computing.* Pervasive Computing, 2009. **8**(4): p. 14--23.

19. Simanta, S., et al., A Reference Architecture for Mobile Code Offload in Hostile Environments. 2012 IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture. 2012: p. 282-286.

20. Fesehaye, D., et al., *Impact of cloudlets on interactive mobile cloud applications.* Proceedings of the 2012 IEEE 16th International Enterprise Distributed Object Computing Conference, EDOC 2012, 2012: p. 123-132.

21. Of, N., N. Of, and N. Of, *M OBILE C LOUD C OMPUTING G EARING R ESOURCE -P OOR M OBILE D EVICES WITH P OWERFUL C LOUDS : A RCHITECTURES , S CIENCE AND T ECHNOLOGY D I N IU , U NIVERSITY OF A LBERTA.* 2013: p. 14-22.

22. Dou, Z. and W.B. Heinzelman, *Benefits of Utilizing an Edge Server (Cloudlet) in the MOCHA Architecture.* 2013: p. 1- 49.

23. Di Valerio, V. and F. Lo Presti, *Optimal Virtual Machines allocation in mobile femto- cloud computing: An MDP approach.* 2014 IEEE Wireless Communications and Networking Conference Workshops, WCNCW 2014, 2014: p. 7-11.

24. Lobillo, F., et al., *An architecture for mobile computation offloading on cloud-enabled LTE small cells.* 2014 IEEE Wireless Communications and Networking Conference Workshops, WCNCW 2014, 2014: p. 1-6.

25. Barbarossa, S. and S.a. Sardellitti, *Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks.* IEEE Signal Processing Magazine, 2014. **31**(6): p. 45--55.

26. Liu, J., et al., *CONCERT: A cloud-based architecture for next-generation cellular systems.* IEEE Wireless Communications, 2014. **21**(6): p. 14--22.

27. Luan, T.H., et al., *Fog Computing: Focusing on Mobile Users at the Edge.* eprint arXiv:1502.01815, 2015.

28. Kaur, K., et al., *Container-as-a-Service at the Edge: Trade-off between Energy Efficiency and Service Availability at Fog Nano Data Centers.* IEEE Wireless Communications, 2017. **24**: p. 48-56.

29. Kaur, A., *A Comparative Study of Code Offloading Techniques and Application Partitioning Methods in Mobile Cloud Computing.* 2016. **143**: p. 1-8.

30. Flores, H., et al., *Mobile code offloading: From concept to practice and beyond.* IEEE Communications Magazine, 2015. **53**(3): p. 80--88.

31. Zhang, W., Y. Wen, and D.O. Wu, *Energy-efficient scheduling policy for collaborative execution in mobile cloud computing.* Proceedings - IEEE INFOCOM, 2013: p. 190-194.

32. Masood, A., et al., *HETS: Heterogeneous Edge and Task Scheduling Algorithm for Heterogeneous Computing Systems.* 2015 IEEE 17th International

Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems, 2015: p. 1865-1870.

33. You, C. , et al. , *Energy- Efficient Resource Allocation for Mobile- Edge Computation Offloading.* IEEE Transactions on Wireless Communications, 2017. **16**: p. 1397-1411.

APPENDIX

VITA

Rujivorakul Vittayasak was born in Bangkok. He received B.Eng from Faculty of Engineering,Rajamankala Institute of Technology and M. SIT from Kasetsart University in 2000 and 2005. He received a scholarship from the Higher Education Research Promotion through the Office of the Higher Education Commission Ph.D. program in 2011