CHAPTER III

PROLBEM FORMULATION

Energy consumption and execution time are closely related but they cannot be directly inferred from one another. If a given task spends time t_1 on CPU a but it spends time $t_2 < t_1$ on CPU b, then the consumed energy by CPU a may not be more the energy consumed by CPU b. However, suppose there are two tasks running on the same CPU. The first task spends less time than the second task. In this situation, the first task obviously does consume less energy than the second task. To estimate the amount of energy consumption by any CPU, the frequency of CPU clock cycle must be involved besides the CPU execution time. This study did not concern the problem of how to estimate the energy consumption as a function of CPU clock frequency and computational time. The following assumptions are established in this study.

- There exists a mathematical function in terms of CPU clock frequency and the execution time of a task used for estimating the energy consumption of the task.
- There exists an algorithm for estimating the execution time of a task.
- For any processing unit, the energy consumed by the operating system and the methods for alleviating the energy consumption by the operating system such as lowering the clock frequency or dimming the brightness of screen of the processing unit are assumed to be considered by the energy estimating algorithm.
- The amount of energy and energy consumption rate used in this study, such as those reported in [35], [36], [37], [38], are assumed to be provided by the energy estimating algorithm.

Three sets of information were involved in this study. The first set is a set of dependent tasks captured in forms of a dependency task graph with the estimated execution time of each task. The second set is a set of energy consumption rate of each processing unit. This energy consumption rate is based on the above assumption of the existence of energy estimating algorithm. The last set is a set of types of task executable on each processing unit. Each processing unit is capable of

executing at least one type of task. This study consists of two relevant scheduling aspects. The first aspect concerns the problem of how schedule a given dependent task graph to achieve minimum total energy consumption of all involved processing units. The second aspect is how to achieve the minimum scheduling length with minimum energy consumption not exceeding the amount of energy given to each processing unit prior to the execution. This aspect also involves the computational time and energy consumption for performing the scheduling algorithm and executing each scheduled task.

Define a dependent task graph G to be a function of tasks V and their dependencies E, that is $G=(V,E)$. V consists of a set of tasks v_i, $V=\{v_1,\ldots,v_n\}$ and E represents a set dependency edges among the tasks, $E=\{(v_i,v_k)\,|\,v_i,v_k\in V\}$. The notation (v_i,v_k) represents the sequence of execution of v_i then v_k. A set of tasks occurs on client sites which can be a mobile phone or a PC and executed either on client or server sites. Because processing units on client or server machines can be used to execute tasks, they are indistinguishable and referred to as processing unit. Each processing unit is capable of processing more than one task and there can be more than one processing unit which can process the same task. The amount of energy consumed by a processing unit a to process task v_i is denoted by $e_a(v_i)$.

3.1 Scheduling to minimize total energy consumption

Assume that the number of tasks is greater than the number of processing units, the problem in this scenario case is formulated to minimize $\sum_{1\le i\le m, v_j\in \mathbf{g}_i} e_i(v_j)$. The implemented method will partition tasks $v_i\in V$ into m groups of tasks $\{\mathbf{g}_1,\ldots,\mathbf{g}_m\}$ such that group $\mathbf{g}_j$ can be performed on processing unit j to attain the minimum total energy consumed by all processing units.

3.1.1 Constraints on energy-aware during processed a tasks

The total amount of energy consumed by a processing unit is considered to be a combination of execution energy, transmission energy, and idle energy. Execution energy is calculated using server competency and task characteristics. Let $t_a(v_i)$ be the amount of time used by processing unit a to execute task v_i and α_a

be the energy per unit time consumed by processing unit a to execute a task. The execution energy for processing unit a to execute task v_i is thus $\alpha_a t_a(v_i)$.

Transmission energy occurs while data are being transferred between processing units as dictated by their data dependencies. It is computed by $\lambda_a d_a(v_i)$, where λ_a is the transmission energy characteristic of processing unit a and $d_a(v_i)$ the amount of data in task v_i which is transferred from processing unit a to other processing units in the same dependency path of G.

Assume that when a processing unit is not processing any tasks, it is in idle state. When task v_i is transferred to processing unit a, the processing unit is activated but remains in idle state for $w_a(v_i)$ unit time until all related data of v_i have been received. Therefore, the idle energy, β_a, depends on energy waste during data transfer period before the processing unit is activated and energy consumed during the processing unit is idle. That is $\beta_a(\psi_a(v_i)+w_a(v_i))$, where $\psi_a(v_i)$ is the time that processing unit a is in idle state before it is activated.

However, if a task is the last task of G, so called a leaf vertex, processing this task will incur no idle or transmission energy. Two constants, namely, $\kappa_a \in \{0,1\}$ and $\mu_a \in \{0,1\}$, are thus introduced to identify whether a task v_i executed by processing unit a is a leaf or internal vertex in G. When the number of out-degrees of v_i is zero, $\kappa_a = 0$, and when the number of in-degrees of v_i is zero, $\mu_a = 0$. Otherwise, both constants are equal to 1. Hence, the total energy of G, Γ_G, is calculated from

$$\begin{aligned}\Gamma_G &= \sum_{1\le a\le m, v_j\in \mathbf{g}_a} e_a(v_j) + \sum_{1\le a\le m, v_i\in \mathbf{g}_a} \beta_a \psi_a(v_i) \\ &= \sum_{1\le a\le m, v_j\in \mathbf{g}_a} \alpha_a t_a(v_j) + \kappa_a \lambda_a d_a(v_j) + \mu_a \beta_a w_a(v_j) \\ &+ \sum_{1\le a\le m, v_i\in \mathbf{g}_a} \beta_a \psi_a(v_i)\end{aligned} \quad (1)$$

3.1.2 Energy-efficient process clustering assignment (EPC) algorithm

In the scope of this study, it is assumed that the values of $t_a(v_i)$, α_a, $d_a(v_i)$, λ_a, $w_a(v_i)$, and β_a for any processing unit a can be determined prior to the task assignment. Each processing unit is capable of processing more than one kind of tasks unless it is specified otherwise. Let P_{v_i} represent a set of processing units which is capable of processing task v_i. The steps in EPC algorithm proceed as follows.

1) Identify primary, secondary and tertiary candidate processing units for each task v_i. A primary candidate processing unit for task v_i is the processing unit which consumes the least energy to process task v_i compared to all processing units in P_{v_i}. The remaining processing unit a in P_{v_i} is termed a secondary candidate processing units if $\alpha_a t_a(v_i) \leq \tau$, otherwise it is termed a tertiary candidate processing unit. The algorithm for this step is given in EPC Algorithm 1.
2) Re-evaluate the energy consumption for primary and secondary candidate processing units defined in the previous step to identify actual candidate processing units. When a task v_i is assigned to a candidate processing unit a, other dependent tasks of v_i are virtually assigned to a, if they are executable by a. Taking this into account, energy consumption of the primary and secondary candidate processing units is re-evaluated and compared. The candidate processing unit which consumes the least energy is defined as an actual candidate processing unit. The algorithm for this step is given in EPC Algorithm 2.
3) Perform task scheduling for each actual candidate processing unit. This step is divided into two phases. In the first phase, a set of dependent tasks are arranged in the order according to their dependency paths in G. Each task is represented by its processing time at each server. The scheduling is arbitrary for independent tasks. As for every task processing, there is an interval of idle time, the schedule sequence at any processing units will therefore consists of alternating time slots between the task processing duration and idle time duration. The algorithm for this phase is given in EPC Algorithm 3. In the

second phase, tasks are re-assigned to minimize idle energy of the whole system. As one task is executable by more than one processing unit, a task is re-assigned to other possible processing unit if the new total energy is not increased. The algorithm for this phase is given in EPC Algorithm 4.

Algorithm 1 Identifying Preliminary Candidate Processing Units

Require: v_i, P_{v_i}, and G.

1: **for** $v_i \in G$ **do**
2: Let o be the number of out-degrees of v_i.
3: $\tau = \min_{a \in P_{v_i}}(\alpha_a t_a(v_i)) + \max_{a \in P_{v_i}}(\ o\kappa_a \lambda_a d_a(v_i) + \mu_a \beta_a w_a(v_i))$
4: **for** each processing unit $a \in P_{v_i}$ **do**
5: **if** $\alpha_a t_a(v_i) \leq \tau$ **then**
6: mark a as a secondary candidate processing unit.
7: **else**
8: mark a as a tertiary candidate processing unit.
9: **end if**
10: **end for**
11: **end for**

Figure 1 EPC algorithm 1: Identifying preliminary candidate processing units.

Algorithm 2 Identifying Actual Candidate Processing Units

Require: v_i, P_{v_i}, $\mathbf{g}_a$, and G.

1: **for** $v_i \in G$ **do**
2: **for** each secondary candidate processing unit $a \in P_{v_i}$ **do**
3: Let $\mathbf{g}_a = \phi$ for all a.
4: **for** all ancestors v_j's of v_i **do**
5: **if** v_j is executable by a **then**
6: Let $\mathbf{g}_a = \mathbf{g}_a \cup \{v_j\}$.
7: **end if**
8: **end for**
9: **for** all descendants v_k's of v_i **do**
10: **if** v_k is executable by a **then**
11: Let $\mathbf{g}_a = \mathbf{g}_a \cup \{v_k\}$.
12: **end if**
13: **end for**
14: **end for**
15: **end for**
16: Descendingly sort $|\mathbf{g}_a|$.
17: Mark any processing unit $a \in P_{t_i}$ having maximum $|\mathbf{g}_a|$ and lowest execution energy as a primary candidate processing unit.
18: Mark the rest of processing units in P_{v_i} as a secondary candidate processing units.
19: **for** $v_i \in G$ **do**
20: **for** each secondary candidate processing unit $b \in P_{v_i}$ **do**
21: Compute energy $e_b(v_i)$.
22: **if** $\exists$ candidate processing unit $a \in P_{v_i}$ such that $e_b(v_i) < e_a(v_i)$ **then**
23: Mark processing unit a as a tertiary candidate processing unit.
24: Mark processing unit b as a primary candidate processing unit.
25: **end if**
26: **end for**
27: **end for**

Figure 2 EPC algorithm 2: Identifying actual candidate processing units

Algorithm 3 Phase-1 Preliminary Task Scheduling in Processing Units

Require: v_i, P_{v_i}, and G.

1: Let v_f be the latest task in its primary processing unit in P_{v_f}.
2: Let ζ_{v_f} be the finishing time of task v_f.
3: Descendingly sorted task v_i at first level of G by execution time on its primary processing unit.
4: **for** each sorted task v_i at first level of G **do**
5: **if** primary processing unit P_{v_i} is an empty slot **then**
6: Assign v_i to its primary processing unit in P_{v_i}.
7: **else**
8: Assign v_i to its primary processing unit in P_{v_i} at time $\zeta_{v_f} + 1$.
9: **end if**
10: **end for**
11: **while** $\exists$ unassigned v_i **do**
12: Let δ be a set of v_k having all ancestor tasks already assigned to their primary processing units.
13: **for** each task $v_k \in \delta$ **do**
14: Let A_{v_k} be a set of ancestor tasks of v_k already assigned to their primary processing units.
15: Find a task $v_j \in A_{v_k}$ having the latest finishing time at time ζ_{v_j}.
16: **end for**
17: Find a task $v_k \in \delta$ having earlier finishing time ζ_{v_j} and the shortest execution time on its primary processing unit.
18: **if** time slot $\zeta_{v_j} + wc_a(v_k)$ of primary processing unit in P_{v_k} is empty **then**
19: Assign v_k to its primary processing unit in P_{v_k} at time slot $\zeta_{v_j} + wc_a(v_k)$.
20: **else**
21: Assign v_k to its primary processing unit in P_{v_k} at time slot $\zeta_{v_f} + 1$.
22: **end if**
23: **end while**

Figure 3 EPC algorithm 3: Phase-1 preliminary task scheduling in precessing units.

Algorithm 4 Phase-2 Minimizing Idle Energy in Task Scheduling

Require: a list of scheduled tasks in each $\mathbf{g}_a$ and G.

1: Let S be a set of $|\mathbf{g}_a|$ sorted in descending order.
2: **for** each corresponding $\mathbf{g}_a \in S$ **do**
3: Let v_k be the last task in $\mathbf{g}_a$.
4: **while** task v_k is not the first task in $\mathbf{g}_a$ **do**
5: Traverse the task list upward starting at v_k until an empty slot E is found and let v_j be the task next to this idle slot.
6: Let A_{v_j} be a set of ancestors of v_j.
7: **for** all tasks $v_i \in A_{v_j}$ **do**
8: **if** there exists a processing unit b with an empty slot **and** the data dependency of v_i and its ancestors in any processing units is not violated **then**
9: Temporarily remove v_i from the present slot.
10: Insert v_i to the beginning of this new empty slot.
11: Reschedule all tasks whose starting times are after the ending time of v_i by using Algorithm 3.
12: Let G' be the new dependency task graph after the temporary assignment of v_i.
13: Compute the total energy $\Gamma_{G'}$.
14: **if** $\Gamma_{G'} > \Gamma_G$ **then**
15: Remove v_i from processing unit b and assign it back to its original processing unit.
16: **else**
17: Permanently assign v_i to this new empty slot on processing unit b.
18: **end if**
19: **end if**
20: **end for**
21: **if** v_k is not the first task in $\mathbf{g}_a$ **then**
22: Let v_k be a new task found after traversing list $\mathbf{g}_a$ upward starting from the empty slot E.
23: **end if**
24: **end while**
25: **end for**

Figure 4 EPC algorithm 4: Phase-2 minimizing idle energy in task scheduling.

Task graphs

The four algorithms will be described through their implementations on a dependent task graph G shown in Figure 5. From the figure, there are 17 tasks represented by v_1 to v_{17} and 4 processing units a, b, c, and d. Tasks are represented by graph vertices with the name of the tasks written in the circles and the processing units capable of processing the tasks written in the parentheses placed to the right of the vertices. For example, it can be taken from this graph that task v_1 is executable by processing units a and c.

Table 1 Energy consumption constants for each processing units $i \in \{a,b,c,d\}$. α_i is the execution energy at peak state per unit time. β_i is the idle energy at idle state per unit time. And λ_i is the transmission energy per unit data.

Energy constants	Processing Units			
	a	b	c	d
$\alpha_{i\in\{a,b,c,d\}}$	0.067	0.065	0.064	0.062
$\beta_{i\in\{a,b,c,d\}}$	0.029	0.031	0.029	0.029
$\lambda_{i\in\{a,b,c,d\}}$	0.045	0.047	0.050	0.048

Hypothetical values for all variables in Equation (1) for the task graph are given in Table 1, Table 2, and Table 3. The amount of processing energy per unit time (α_a), waiting energy per unit time (β_a), and transmission energy per unit amount of data (λ_a) of each processing unit a are presented in Table 1. Transmission speed of the link between each processing unit pair in unit amount of data per unit time are presented in Table 2. The data transmission time among processing units calculated from the transmission speed presented in Table 2 along with the amount of transmitted data of each task v_i are presented in Table 3. The method used to estimate energy consumption is shown through the activities of task v_1.

Table 2 Estimated data transmission rate $r_{a,b}$ between any processing unit pairs in unit amount of data per unit time.

Processing Unit	a	b	c	d
a	-	1	0.5	1
b	1	-	0.5	1.25
c	0.5	0.5	-	1.25
d	1	1.25	1.25	-

Table 3 Initial values of output data size and estimated execution time of each task by each processing unit in the first study case.

Task	Amount of transmitted data	Amount of execution time			
v_i	$d_a(v_i)$	$t_a(v_i)$	$t_b(v_i)$	$t_c(v_i)$	$t_d(v_i)$
1	60	260	-	760	-
2	80	350	160	-	-
3	10	-	-	890	820
4	60	420	150	-	-
5	90	-	-	940	430
6	30	-	400	-	560
7	50	670	-	850	-
8	40	300	-	-	630
9	50	-	470	-	-
10	30	920	-	-	950
11	80	-	550	120	-
12	10	-	200	-	130
13	90	-	100	790	-
14	40	940	-	580	-
15	50	-	-	350	310
16	40	930	-	-	750
17	70	-	370	-	-

Table 4 Estimated execution energy for each task on each processing unit in the first study case.

Task	Execution energy			
v_i	$\alpha_a t_a(v_i)$	$\alpha_b t_b(v_i)$	$\alpha_c t_c(v_i)$	$\alpha_d t_d(v_i)$
1	17.42	-	48.64	-
2	23.45	10.40	-	-
3	-	-	56.96	50.84
4	28.14	9.75	-	-
5	-	-	60.16	26.66
6	-	26.00	-	34.72
7	44.89	-	54.40	-
8	20.10	-	-	39.06
9	-	30.55	-	-
10	61.64	-	-	58.90
11	-	35.75	7.68	-
12	-	13.00	-	8.06
13	-	6.50	50.56	-
14	62.98	-	37.12	-
15	-	-	22.40	19.22
16	62.31	-	-	46.50
17	-	24.05	-	-

From Table 1 and Table 3, the processing energy, α_a, and the amount of time to process task v_1 on a processing unit a, $t_a(v_1)$, are 0.067 and 260, respectively. Thus, if task v_1 is executed on processing unit a, the execution energy $\alpha_a t_a(v_1) = 0.067 \times 260 = 17.42$. The execution energy for other tasks are calculated in a similarly manner and the results are shown in Table 4.

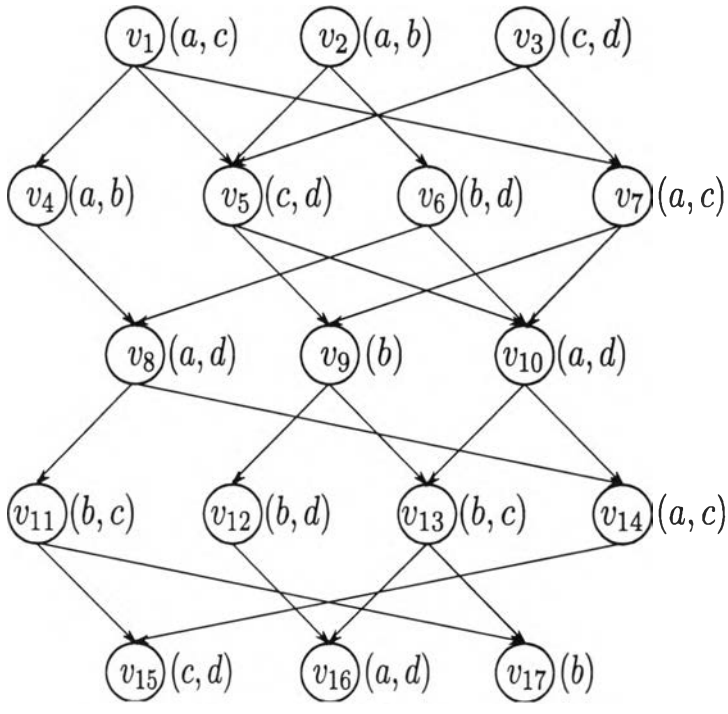
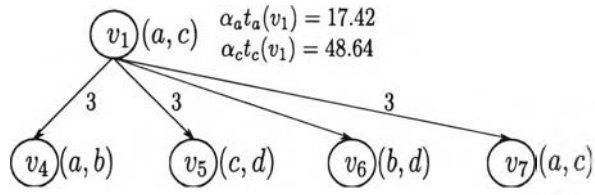


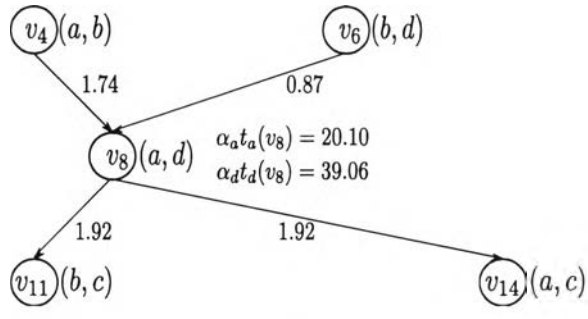
Figure 5 A dependent task graph G. Each graph vertex represents a task. The names of the tasks, v_1 to v_{17}, are written in the circles and the candidate processing units which are capable of executing each task are written in the parentheses placed to the right of each vertex.

Once task v_1 has been processed, the results will be sent to its descendant tasks which are tasks v_4, v_5, v_6, and v_7. As descendent tasks can be executed by processing units other than a, data transfer between processing unit pairs will incur transmission energy. For example, if data are sent from v_1 to v_7 and suppose that processing unit c has been assigned to v_7, data will be transferred from processing units a to processing unit c. The transmission energy for this transfer is calculated from $\lambda_a d_a(v_1) = 0.045 \times 60 = 2.7$, where the values for λ_a and $d_a(v_1)$ are taken from Table 1 and Table 3. The values of transmission energy for other processing unit pairs are calculated in the same manner.

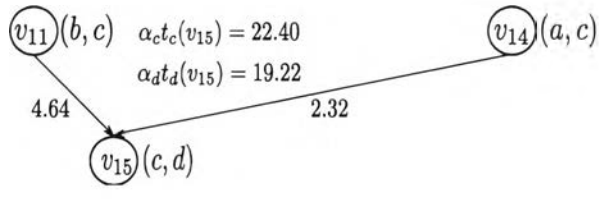
Idle energy can be determined from data transmission rate between the sending and receiving processing unit. Consider data transfer between tasks v_6 and v_8. Assume that processing unit d is assigned to process task v_6 and processing unit a to process task v_8. From Table 3, the amount of data of task v_6 to be transmitted from processing unit d is $d_d(v_6) = 30$ and the data transmission rate from d to a is $r_{d,a} = 1$. The data transmission time $w_a(v_8)$ is calculated from $d_d(v_6) / r_{d,a} = 30 / 1 = 30$. Thus, the idle energy at processing unit a for v_8 is equal to $\beta_a w_a(v_8) = 0.029 \times 30 = 0.87$.



(a)



(b)



(c)

Figure 6 Diagrams show 2(a) task v_1, 2(b) task v_8, and 2(c) task v_{15}, along with their ancestor and descendent tasks. Also shown in the diagrams are the values for maximum idle energy, maximum transmission energy, and execution energy required to process the tasks. Idle energy appears when data are passed from a processing unit of an ancestor task to that of the current task. The maximum values determined from all candidate processing units in each transmission are denoted next to graph incoming edges. Transmission energy appears when data are passed from a processing unit of a current task to that of a descendent task. The maximum values determined from all candidate processing units in each transmission are denoted next to graph outgoing edges. The values for execution energy for both possible processing units are denoted next to the graph vertices.

Algorithms implementation

The results produced from EPC Algorithms 1 to 4 are explained. In EPC Algorithm 1, tasks are categorized into three types. The first type includes tasks with no incoming degree such as task v_1. The second type includes tasks which have both incoming and outgoing degrees such as task v_8. And the third type includes task with no outgoing degree such as task v_{15}. Minimum energy τ is estimated from every processing units in P_{v_1}. For tasks v_1, v_8 and v_{15}, the values for τ are

$$\begin{aligned}\tau &= \min_{k\in P_{v_1}}(\alpha_k t_k(v_1)) + \max_{k\in P_{v_1}}(o\kappa_k\lambda_k d_k(v_1) + \mu_k\beta_k w_k(v_1)) \\ &= 17.42 + 3\times 3 + 0 \\ &= 26.42\end{aligned}$$

$$\begin{aligned}\tau &= \min_{k\in P_{v_8}}(\alpha_k t_k(v_8)) + \max_{k\in P_{v_8}}(o\kappa_k\lambda_k d_k(v_8) + \mu_k\beta_k w_k(v_8)) \\ &= 20.10 + 2\times 1.92 + 1.74 \\ &= 25.68\end{aligned}$$

$$\begin{aligned}\tau &= \min_{k\in P_{v_{15}}}(\alpha_k t_k(v_{15})) + \max_{k\in P_{v_{15}}}(o\kappa_k\lambda_k d_k(v_{15}) + \mu_k\beta_k w_k(v_{1i5})) \\ &= 19.22 + 0 + 4.64 \\ &= 23.86\end{aligned}$$

Note that because v_1 has no incoming degree, the term $\mu_k\beta_k w_k(v_1)$ becomes zero.

By calculating τ, secondary and tertiary candidate processing units for each task can be identified. From Figure 6, as $\alpha_a t_a(v_1) \le \tau$ and $\alpha_c t_c(v_1) \ge \tau$, processing units a and c are selected as the secondary and tertiary candidate processing units, respectively. Figure 7 shows all secondary candidate processing units for each task after applying EPC Algorithm 1.

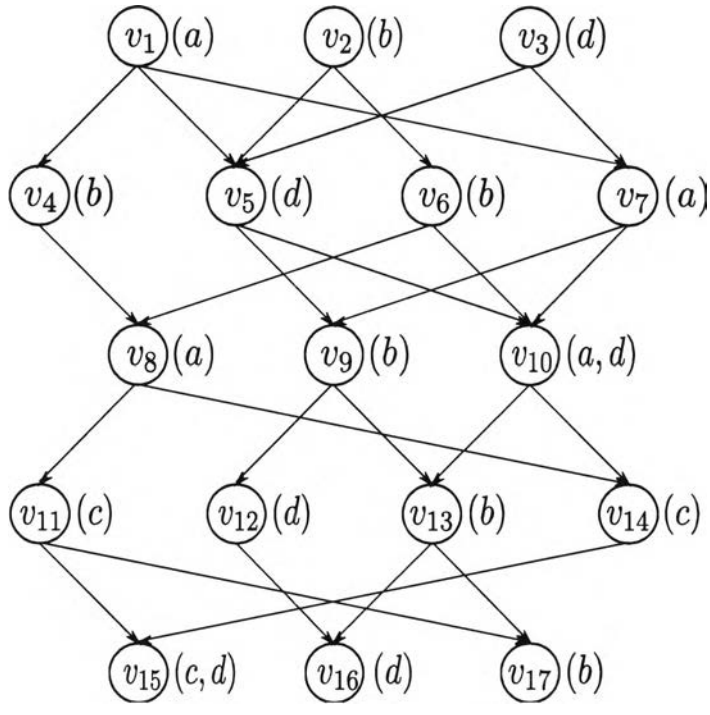


Figure 7 Identifies secondary candidate processing units for tasks v_1 to v_{17}. The processing units are identified using EPC Algorithm 1 and are shown in the parentheses next to the tasks. Note that for tasks v_{10} and v_{15}, both of their executable processing units are identified as secondary candidate processing units.

The result from EPC Algorithm 1 is used to identify the actual candidate processing units by EPC Algorithm 2. In this step, if a task is only assigned to one secondary candidate processing unit, the secondary candidate processing unit becomes primary candidate processing unit. If more than one secondary candidate processing unit is assigned to the task, the algorithm counts the number of ancestor and descendent tasks which are assigned to the same processing units. The secondary candidate processing unit which has the greatest number of ancestor and descendent tasks assigned to it is set as primary candidate processing unit. The energy consumptions for each primary and secondary candidate processing unit are then computed.

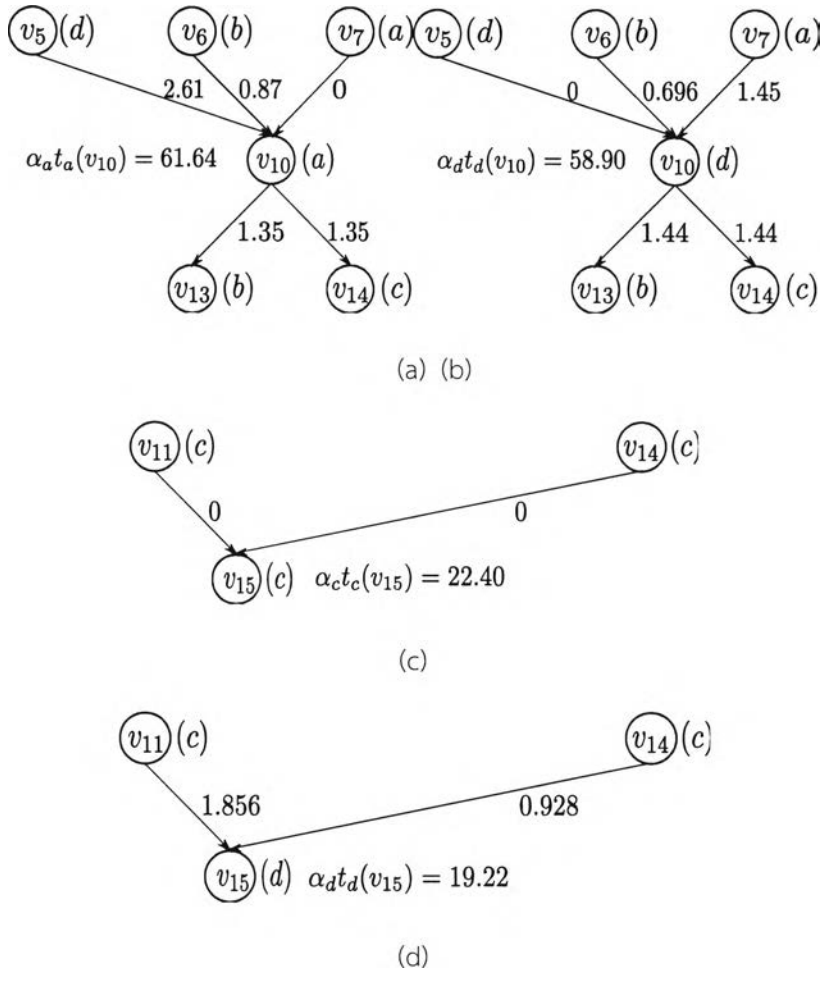


Figure 8 Diagrams show tasks v_{10} and v_{15} along with their ancestor and descendent tasks. In 4(a) task v_{10} is executed by processing unit a, 4(b) task v_{10} is executed by processing unit d, 4(c) task v_{15} is executed by processing unit c, and 4(d) task v_{15} is executed by processing unit d. Also shown in the diagrams are the values for idle energy, transmission energy and execution energy. The values are denoted next to the graph incoming edges, outgoing edges, and vertices, respectively.

Figure 8 shows diagrams of data dependencies for tasks v_{10} and v_{15} which have two secondary candidate processing unit after applying EPC Algorithm 2. For task v_{10}, both processing units a and d have the same number of ancestor and descendent tasks assigned to them. In this case, processing unit d is set as a primary candidate processing unit because its execution energy on task v_{10} is lower. For task v_{15}, the number of ancestor and descendent tasks which can be assigned to processing unit c is 3 (including task v_{15}) and that of processing unit d is zero. Thus, processing unit c is set as the primary candidate processing unit.

After candidate primary candidate processing units for all tasks have been identified, energy consumptions based on both primary and secondary candidate processing units of each task are calculated. The results are used in the next step to identify actual candidate processing units. Consider task v_{10}, the energy consumptions of processing units a and d to execute task v_{10} are shown in Figure 8(a) and Figure 8(b), respectively. They are calculated from

$$\begin{aligned} e_a(v_{10}) &= \alpha_a t_a(v_{10}) + \kappa_a \lambda_a d_a(v_{10}) + \mu_a \beta_a w_a(v_{10}) \\ &= 61.64 + 2 \times 1.35 + 2.61 \\ &= 66.95 \end{aligned}$$

for processing unit a and

$$\begin{aligned} e_d(v_{10}) &= \alpha_d t_d(v_{10}) + \kappa_d \lambda_d d_d(v_{10}) + \mu_d \beta_d w_d(v_{10}) \\ &= 58.90 + 2 \times 1.44 + 1.45 \\ &= 63.23 \end{aligned}$$

for processing unit d. As $e_d(v_{10})$ is less than $e_a(v_{10})$, processing unit d is selected as the actual candidate processing unit for task v_{10}.

Similar calculation can be performed on task v_{15}. Energy consumptions of processing units c and d to execute task v_{15} are shown in Figure 8(c) and Figure 8(d), respectively. They are calculated from

$$\begin{aligned} e_c(v_{15}) &= \alpha_c t_c(v_{15}) + \kappa_c \lambda_c d_c(v_{15}) + \mu_c \beta_c w_c(v_{15}) \\ &= 22.40 + 0 + 0 \\ &= 22.40 \end{aligned}$$

for processing unit c and

$$\begin{aligned} e_d(v_{15}) &= \alpha_d t_d(v_{15}) + \kappa_d \lambda_d d_d(v_{15}) + \mu_d \beta_d w_d(v_{15}) \\ &= 19.22 + 0 + 1.856 \\ &= 21.076 \end{aligned}$$

for processing unit d and. As $e_d(v_{10})$ is less than $e_c(v_{10})$, processing unit d is selected as actual candidate processing unit for task v_{15}. Actual candidate processing units for all tasks in the dependent task graph G determined from EPC Algorithm 2 are shown in Figure 9.

In EPC Algorithm 3, tasks are scheduled and assigned to their corresponding actual candidate processing units identified by the previous step. For example, consider tasks v_1, v_2 and v_3 in the first level. The actual candidate processing units for tasks v_1, v_2 and v_3 are processing units a, b, and d, respectively. From Table 4, the execution time become $t_a(v_1) = 260$, $t_b(v_2) = 160$, and $t_d(v_3) = 820$. Thus, task v_2 which has the least execution time is scheduled and assigned first, followed by task v_1 and task v_3. Next, tasks whose ancestor tasks have been scheduled, in this case tasks v_4, v_5, v_6 and v_7, are considered. Because task v_2 has the least execution time compared to other scheduled tasks, it also has the earliest finishing time. Hence, task v_6 which is a descendent task of v_2 is the next task to be scheduled. Because the finished time of task v_1 is next to that of task v_2, tasks v_4 which is the descendent task of v_1 is scheduled next. The next set of tasks that reach its finished time is that of tasks v_1 and v_4. Therefore, the next task to be scheduled and assigned is task v_8 which is a descendent task of v_1 and v_4. In the case that a task shares

common ancestor tasks with other tasks, such as tasks v_5 and v_7 which are both descendent tasks of v_3, the task which has lower execution time is scheduled first. In this case, tasks v_5 is scheduled before task v_7 as $t_d(v_5) < t_a(v_7)$. The list of all tasks in the task graph G scheduled by EPC Algorithm 3 is illustrated in Figure 10(a).

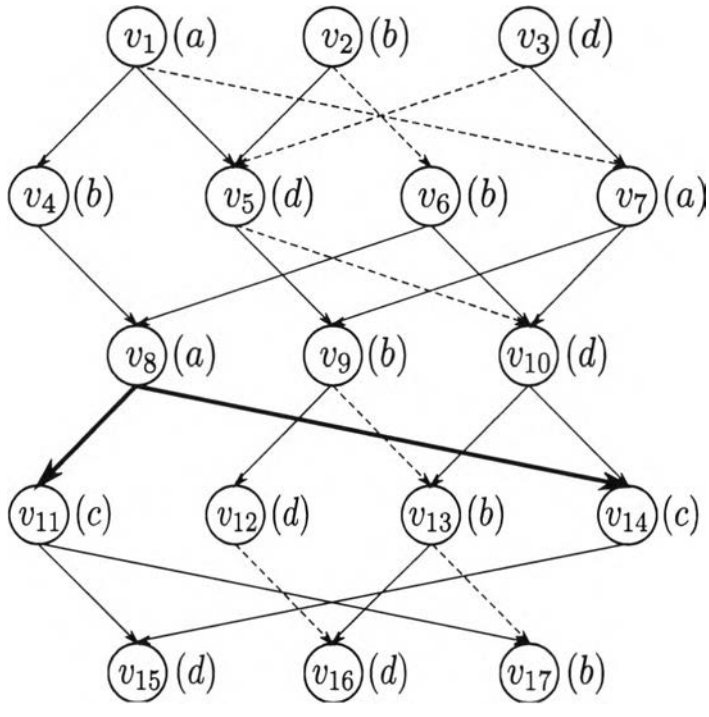
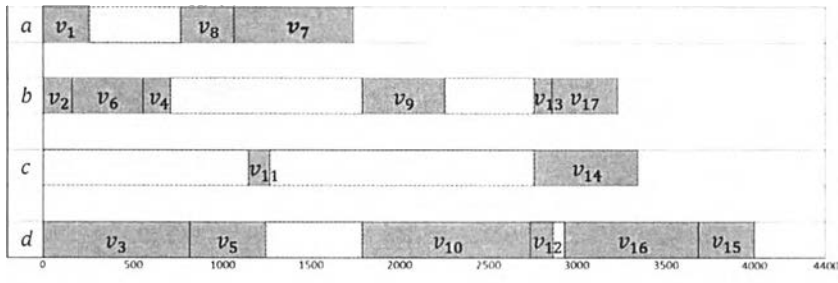
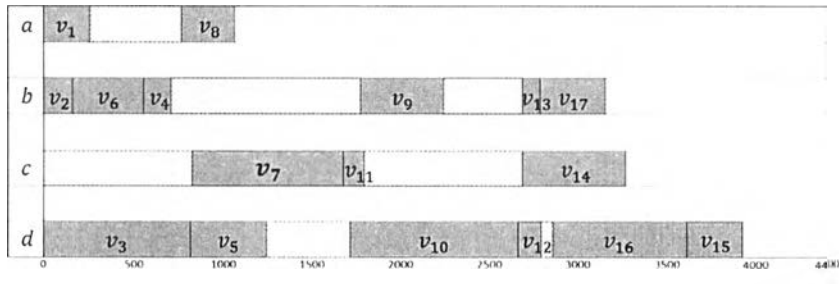


Figure 9 A depend task graph G with actual candidate processing units determined from EPC Algorithm 2. The actual candidate processing units for each task are written in the parentheses next to the graph vertices. Arrows linking tasks represent the direction of data transmission. Solid arrows represent data transmission between different processing units. Dashed arrows represent data transmission within the same processing unit. Thick arrows represent data transmission from when data are transferred to more than one a task with the same processing unit, e.g., v_8 to v_{11} and v_{14}.



(a)



(b)

Figure 10 Lists of scheduled task after (a) EPC Algorithms 3 is applied and (b) after EPC Algorithm 4 is applied. In this example, only task v_7 is removed from processing unit a and re-assigned to processing unit c.

Idle energy of the scheduled task is minimized further by reducing empty slots in the processing time-line using EPC Algorithm 4. In Figure 10(a), the latest tasks to be executed on processing unit d is task v_{15}. The algorithm thus considers v_{15} and progresses upward on the time line until it finds the first empty slot next to task v_{16}. The empty slot corresponds to idle energy between v_{16} and its ancestor tasks v_{12} and v_{13}. Tasks v_{12} and v_{13} are then re-assigned to other processing units. Although the re-assignment causes no data dependency violation, the empty slot remains and the total energy consumption increases. Thus, tasks v_{12} and v_{13} are not re-assigned. The next empty slot is found between tasks v_{10} and its ancestor tasks v_5, v_6, and v_7. In this case, the idle energy and consequently total energy is reduced when task v_7 is re-assigned to processing unit c. The assigned processing unit for task v_7 is changed from a to c. Tasks on other processing units are re-scheduled similarly and the result is depicted in Figure 10(b).

3.1.3 Experimental results

The HEFT, LSH, and EPC algorithms were tested using three study cases with different scheduling scenarios to evaluate their performance. The dependent task graph G in Figure 5 was used in the first case. A task graph in the second case was taken directly from [30] to be used as a validation benchmark. The scenario for the third case was extended from the first case to include more complex tasks graphs and shifted the emphasis from transmission load to execution of intensive scenario.

Table 5 Clock rates and power consumption at peak and idle states for benchmark desktop CPUs. The values were taken from [35] and [36] and are used to determine energy consumption of each processing unit in the experiments.

Processing unit	CPU	Clock (GHz)	Power consumption (W)	
			peak	idle
a	Intel Core i7-975 XE	3.33	240	105
b	Intel Core 2 Extreme QX6850	3.00	233	110
c	Intel Core 2 Extreme QX6700	2.66	229	106
d	Intel Core i7-920	2.66	224	105

Simulation environments for all experiments consist of four heterogeneous processing units. Each processing unit has their unique competency and different energy consumption during peak or idle states. The energy consumption for the four processing units are shown in Table 5. The values were selected from the database of experimentally acquired energy consumption for commercially available CPU [35] and [36].

To use in the calculation, the energy consumption in watts taken from the database was converted to watts per seconds. For example, the execution energy per unit time for unit a is equal to $\alpha_a = 240/3600 = 0.067$ and the waiting energy per unit time for unit a is equal to $\beta_a = 105/3600 = 0.029$. The values for transmission energy assigned to each processing unit pair were randomly generated although the generated values were constrained between the energy consumption

during peak state and the energy consumption during idle state of both processing units in each pair. The values for the execution energy per unit time, waiting energy per unit time, and transmission energy per unit data used in the experiments are shown in Table 1. Delay time during data transmission were calculated from the network transmission speed of the processing unit pairs presented in Table 2.

In the first case, the dependent task graph G, as shown in Figure 5, contains 17 tasks, v_1-v_{17}, and 4 processing units, a, b, c, and d. Each task has unique attributes and each processing unit has different competency to handle task execution. Also shown in Figure 5, in the parentheses next to each graph vertices, are initial assignment of processing units which are capable of executing the tasks denoted at the vertices. The estimated execution time of each task by each processing unit and the size of the output data defined at the beginning of the simulation are presented in Table 3. The values for total execution energy consumption of each processing unit for each task are shown in Table 4.

Although in the HEFT algorithm, energy consumption is not considered their cost functions, they have been employed as scheduling algorithms. Thus, their performance on the study cases are worth considered. The present value for the EPC algorithm is determined by setting up execution, transmission, and idle energy as cost functions of each processing unit during the scheduling process. The result scheduled tasks after applying the HEFT and LSH algorithms are depicted in Figure 11. The execution time and idle time are presented as dark and striped slot, respectively. The obtained values for system finish time are 3922, 3410, and 3924 for the LSH, HEFT, and EPC algorithms, respectively. For the performance comparison, the system finish time are not considered as important factor as it could be allocated off-line.

Table 6 to Table 8 shows the values for execution, transmission, and idle energy consumption of each processing unit obtained from the HEFT, LSH, and EPC algorithm, respectively. As seen from the tables, the HEFT algorithm mainly involves optimization of idle and execution time, it yields the least execution time. The algorithm is ranked second place when total energy consumption is considered. The

EPC algorithm focuses on optimizing server assignment to minimize both execution and transmission energy consumption. The algorithm yields reasonably low execution energy, the least idle energy and proceeded to give the lowest total energy consumption. As time-line is not a prime direction of the EPC algorithm, it is not optimized.

Table 6 Values for energy consumption of each processing unit as a result of applying the LSH algorithm to the first study case.

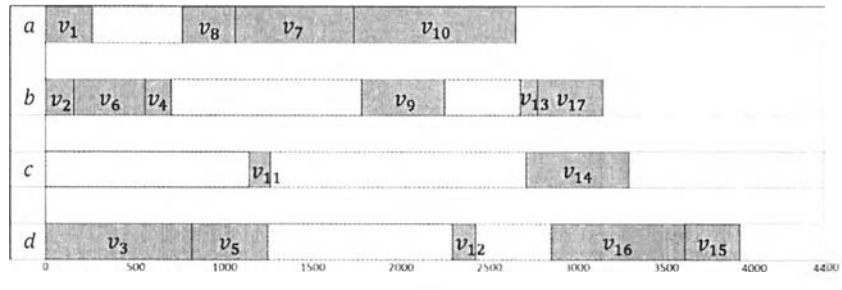
Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Total Energy
a	144.05	12.15	14.79	170.99
b	107.25	14.57	46.81	168.63
c	44.80	10.00	75.40	130.20
d	151.28	9.12	42.98	203.38
Total	447.38	45.84	179.98	673.20

Table 7 Values for energy consumption of each processing unit as a result of applying the HEFT algorithm to the first study case.

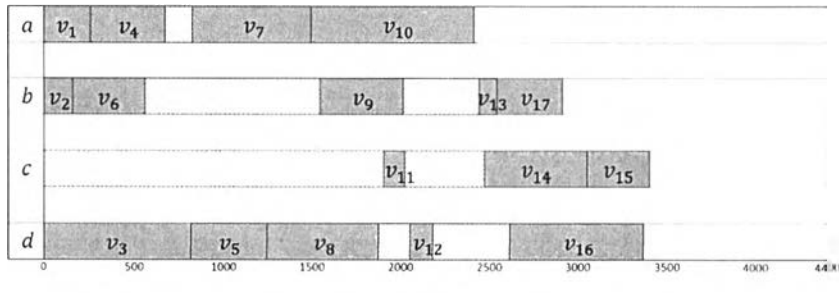
Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Total Energy
A	152.09	10.35	4.35	166.79
B	97.50	13.16	44.02	154.68
C	67.20	4.00	68.44	139.64
D	171.12	6.24	17.75	195.12
Total	487.91	33.75	134.56	656.22

Table 8 Values for energy consumption of each processing unit as a result of applying the EPC algorithm to the first study case.

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Total Energy
a	37.52	9.90	14.79	62.21
b	107.25	15.98	46.87	170.10
c	99.20	15.00	49.94	164.14
d	210.18	7.68	15.49	233.35
Total	454.15	48.56	127.09	629.80



(a)



(b)

Figure 11 List of tasks in processing units a, b, c and d of a dependent task graph G of the first study case scheduled using (a) LSH and (b) HEFT.

A task graph for the second study case was taken from [30] and is shown in Figure 12. The numbers written next to each edge of the graph represent transmission time. In this case, delay time was used to calculate transmission energy and was defined to be different for every task. The execution energy depends on the characteristic of each processing unit. Waiting energy was taken to be 50% of the obtained execution energy. The scheduling is performed under the constrained that processing units can only consume one unit of execution energy per one unit of execution time and one unit of transmission energy per one unit of transmission time.

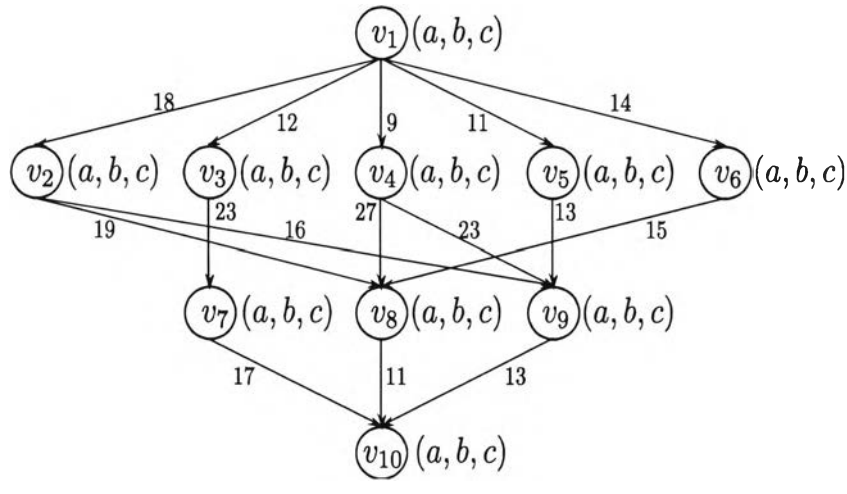


Figure 12 Dependent task graph used in the second study case [30]. The numbers written next to each edge of the graph represent transmission time and are similar for all transmission.

Table 9 Execution energy for all tasks described in the dependent task graph of Figure 12 [30] in the second study case.

Task	Execution energy		
v_i	$\alpha_a t_a(v_i)$	$\alpha_b t_b(v_i)$	$\alpha_c t_c(v_i)$
1	14	16	9
2	13	19	18
3	11	13	19
4	13	8	17
5	12	13	10
6	13	16	9
7	7	15	11
8	5	11	14
9	18	12	20
10	21	7	16

Table 10 Values for energy consumption of each processing unit as a result of applying the LSH algorithm to the second study case.

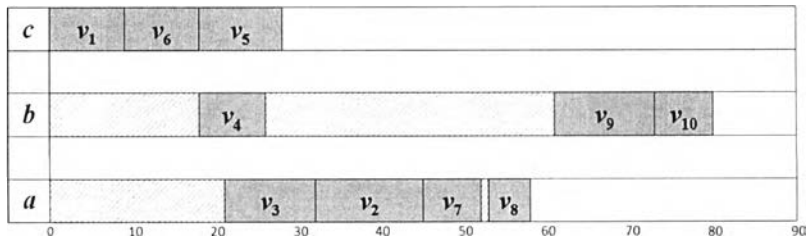
Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Total Energy
a	36.00	44.00	11.00	91.00
b	27.00	27.00	26.50	80.50
c	28.00	67.00	0.00	95.00
Total	91.00	138.00	37.50	266.50

Table 11 Values for energy consumption of each processing unit as a result of applying the HEFT algorithm to the second study case.

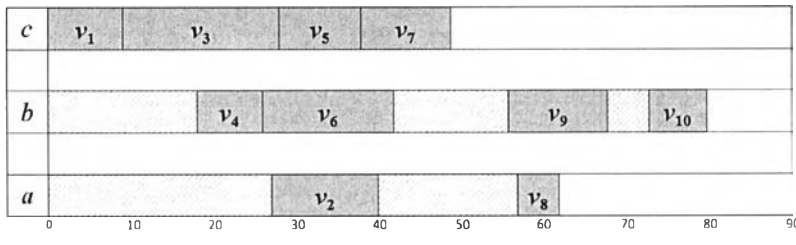
Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Total Energy
a	18.00	27.00	22.00	67.00
b	43.00	42.00	18.50	103.50
c	49.00	71.00	0.00	120.00
Total	110.00	140.00	40.50	290.50

Table 12 Values for energy consumption of each processing unit as a result of applying the EPC algorithm to the second study case.

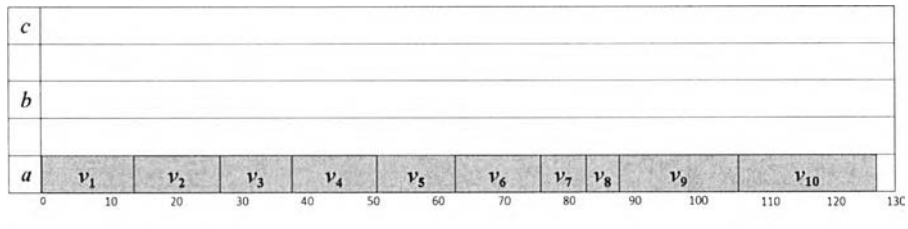
Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Total Energy
a	127.00	0.00	0.00	127.00
b	0.00	0.00	0.00	0.00
c	0.00	0.00	0.00	0.00
Total	127.00	0.00	0.00	127.00



(a)



(b)



(c)

Figure 13 List of tasks in processing units a, b, c and d of a dependent task graph G of the second study case scheduled using (a) LSH, (b) HEFT, and (c) EPC.

The LSH, HEFT, and EPC algorithms yielded system finish time of 80, 80, and 127, respectively. It is not surprising that HEFT algorithm gave the shortest computation time but failed to deliver optimal energy consumption. This is because minimizing execution time is the main focus of this algorithm while the energy consumption issue is not of concerned. An implementation of the EPC algorithm to this case gave zero transmission and idle energy consumption. Since all processing units can process any tasks, transmission and idle energy consumption can be reduced to zero if only one processing unit are selected to execute every task. Because the total execution energy consumption of processing unit a is minimum compared to those of other processing units, processing unit a is chosen by the EPC algorithm. Total energy consumption cost by this algorithm is the total execution energy consumption of processing unit a.

A task graph used in the third study case is shown in Figure 14. For this case, the transmission time was taken to be 10% of the execution time. The defined amount of transmitted data and execution time for each tasks are shown in Table 13. The estimated energy consumption and data transmission coefficients are shown in Table 1, which were used in the first case, are also adopted in this case.

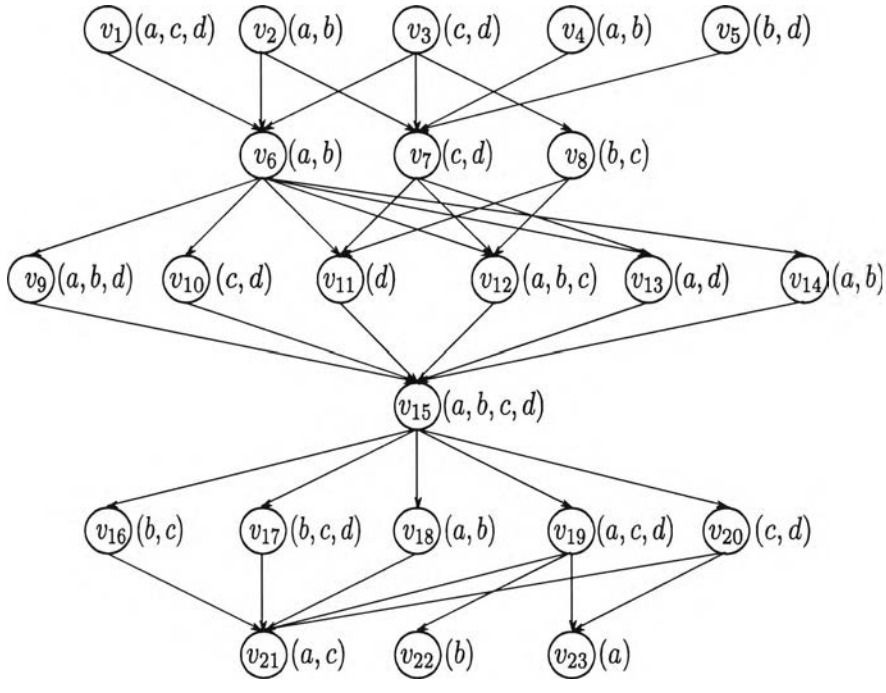


Figure 14 Dependent task graph used in the third study case.

Table 13 Amount of transmitted data $d_a(v_i)$ and amount of execution time $t_a(v_i)$ by each processing unit for all tasks v_i in the third study case.

Task	Amount of transmitted data	Amount of execution time			
v_i	$d_a(v_i)$	$t_a(v_i)$	$t_b(v_i)$	$t_c(v_i)$	$t_d(v_i)$
1	25	350	-	570	430
2	30	430	270	-	-
3	20	-	-	510	450
4	45	590	710	-	-
5	60	-	920	-	790
6	70	360	480	-	-
7	30	-	-	840	720
8	55	-	710	380	-
9	20	390	670	-	380
10	25	-	-	550	580
11	15	-	-	-	620
12	65	870	790	720	-
13	20	560	-	-	540
14	10	430	410	-	-
15	85	280	260	270	250
16	35	-	520	480	-
17	20	-	380	390	420
18	15	610	690	-	-
19	30	250	-	270	480
20	10	-	-	620	710
21	70	760	-	890	-
22	30	-	540	-	-
23	35	580	-	-	-

Table 14 Values for energy consumption of each processing unit as a result of applying the LSH algorithm to the third study case.

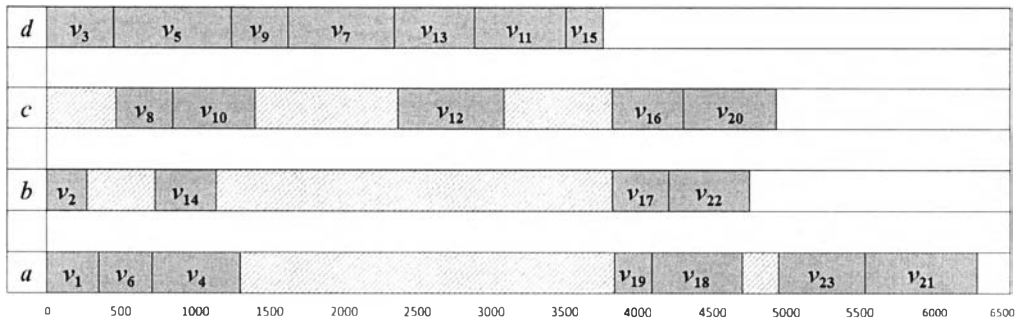
Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Total Energy
a	234.50	12.83	80.56	327.89
b	104.00	4.23	97.28	205.51
c	176.00	9.50	68.87	254.37
d	232.50	15.60	0.00	248.10
Total	747.00	42.16	246.71	1035.87

Table 15 Values for energy consumption of each processing unit as a result of applying the HEFT algorithm to the third study case.

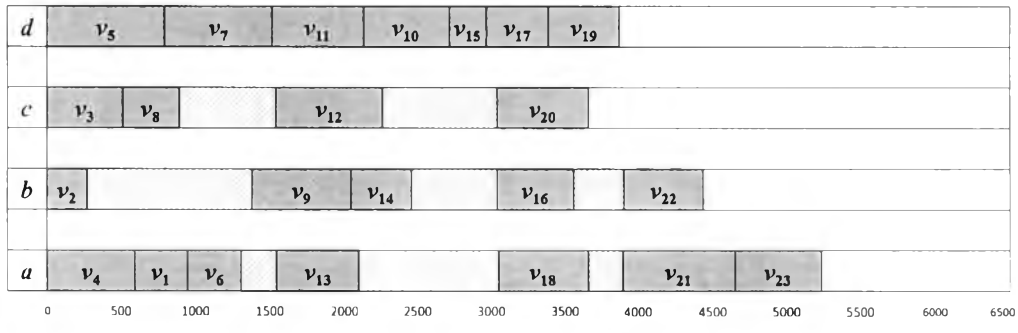
Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Total Energy
a	255.27	12.38	41.18	308.83
b	156.65	5.88	62.43	224.96
c	142.72	8.50	41.12	192.34
d	239.32	18.96	0.00	258.28
Total	793.96	45.72	144.73	984.41

Table 16 Values for energy consumption of each processing unit as a result of applying the EPC algorithm to the third study case.

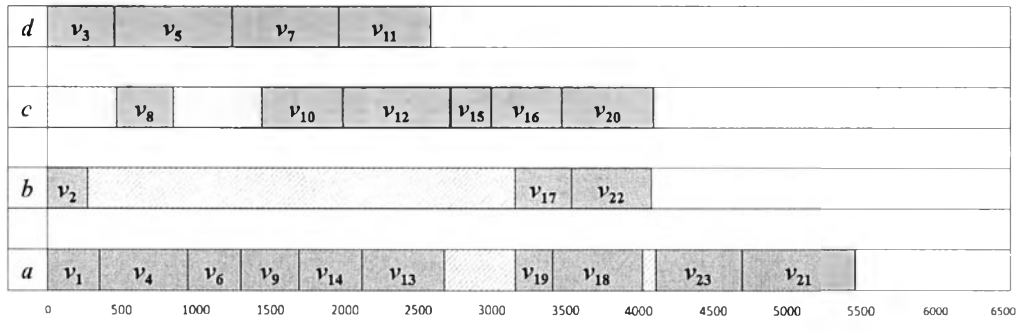
Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Total Energy
a	326.96	11.70	16.53	355.19
b	77.35	6.58	89.59	173.52
c	193.28	13.50	31.03	237.81
d	159.96	5.52	0.00	165.48
Total	757.55	37.30	137.15	932.00



(a)



(b)



(c)

Figure 15 List of tasks in processing units a, b, c and d of a dependent task graph G of the third study case scheduled using (a) LSH, (b) HEFT, and (c) EPC.

The results are illustrated in Figure 15 and the system finish time of LSH, HEFT, and EPC algorithms are 6278, 5230, and 5450, respectively. From the results, HEFT algorithm yielded the least system finish time while the EPC algorithm yielded the least total energy consumption. For the EPC algorithm, the reduction of idle energy significantly decreased the system finish time although incurred a small amount of execution and transmission energy.

It can be seen from this study case that EPC algorithm can be employed to complete the assignment using the lowest total energy consumption. However, the issues often found in real system such as the limitation to system finish time or battery supply were not investigated. These issues will be addressed in the next section.

3.2 Scheduling to minimize total energy consumption and schedule length with limited power supply

In this section, the problem constraints are extended such that it is closer the real system as much as possible. In this case, each processing unit can carry its own work load according to user's request, locally invoked applications, and resident programs. Tasks will be assigned to be executed on a designated processing unit by the scheduler. One processing unit is dedicated to be the main processing unit which handles task scheduling. Thus, apart from the compulsory execution energy, this processing unit will consume additional energy during task scheduling and additional waiting energy for the final result from other processing unit.

The proposed algorithm is aimed to find a method to schedule tasks $v_i \in \mathbf{V}$ and arrange them as a group $\mathbf{g}_a$ in such a way that the scheduling length is the shortest and the energy consumption is minimum. In other words, the objective is to optimize e_a, where e_a denotes the amount of energy consumed by processing unit a when tasks $v_i \in \mathbf{g}_a$ is executed. It is assumed in this case that the battery supply for processing units a which will perform on a dependent task graph G is limited.

3.2.1 Constraints on energy-sufficiency scheduling

The total amount of energy consumed by a processing unit is defined in the same manner as Section 3.1.1. That is, execution, transmission, and idle energy. Now inclusion of energy consideration focuses on scheduling energy and time complexity. Time complexity $T_{a\lg}$ of a scheduling algorithm is another parameter which affects the scheduling energy. The parameter refers to the amount of time taken by an algorithm to schedule a task on a processing unit. For example, $T_{a\lg}(C)$ is the amount of time taken by processing unit a in algorithm C and $\alpha_a T_{a\lg}(C)$ is defined as the energy consumed by processing unit a during task scheduling. The value of time complexity depends on the size of a task graph G which includes the number of tasks (v), the number of processing units (p), and the number of levels (l) in the task graph.

By including the scheduling energy, the total energy consumption of a processing unit is calculated from

$$e_a = \sum_{v_i \in \mathbf{g}_a} \left(\alpha_a t_a(v_i) + \eta_a \lambda_a d_a(v_i) + \mu_a \beta_a w_a(v_i)\right) + \sum_{v_j \in \mathbf{g}_a} \left(\beta_a \psi_a(v_j)\right) + \tau_a \left(\alpha_a T_{a\lg}(C) + \beta_a \psi_a(v_f)\right) \quad (2)$$

The total energy represented by equation (2) includes energy from task scheduling and task execution. Task scheduling is performed one level at a time only by the main processing unit under the prerequisite that some tasks are only executable on designated processing units. The predecessors and successors of a task are determined when the task has been assigned to a processing unit. The execution of a task is considered local if the task and its predecessors and successors are assigned to the same processing unit. In this case, no transmission energy is expended and $\eta_a = 0$. If the execution is not local, transmission energy will be included, $\eta_a = 1$. In this case, the successor has to be in a wait state before the data from the predecessors are sent over. This sets $\mu_a = 1$ as idle energy is added to the total energy cost. If there is no waiting time, μ_a is set to 0. In the case that the final leaf task v_f is not executed on the main processing unit, additional idle energy will incur on the main processing unit during the time it needs to wait for the result from the task v_f. This additional energy is represented by setting $\tau_a = 1$ in equation (2). For other processing units including the main processing unit when it is free from scheduling assignment, $\tau_a = 0$. In this case, the total energy consumption is contributed by execution energy, idle energy, and transmission energy.

In real working environment, the battery power supplies are often limited. Thus, an energy reserve option is introduced in this part of the study. The algorithm has been written based on a careful analysis of previous execution scenarios. The option is proved to handle the situation where battery power supply is limited and its scheme is outlined as follows.

In the task scheduling step, the mandate for assignment completion under limited power supply is to ensure that there will be adequate amount of energy on each processing unit to execute tasks. Because of the prerequisite that some tasks are only executable on particular processing units, battery power should be reserved for the execution of these tasks. Let $\mathbf{V}_a = \{v_i \mid v_i \in \mathbf{V} \,\&\, \mathbf{V}_a \in \mathbf{V}\}$ be the set of tasks

designated to be executed on processing unit a. The energy required to execute a task, ρ_a, is calculated from

$$\rho_a = \sum_{v_i \in \mathbf{V}_a} \left(\alpha_a t_a(v_i) + \eta_a \lambda_a d_a(v_i) \right) \quad (3)$$

In the task execution step, dependencies of the current task to its predecessor and successors are identified in order to determine the required idle energy. Define level k in a task graph G as the level under current consideration and level h as the last level where the designated task for each processing unit is. For all processing unit other than the main processing unit, the idle energy is calculated from level k to h. Because the main processing unit needs to wait until the last task is executed, its idle energy has to be calculated from level k to the last level l (leaf). Let $\sigma_{k,a}$ be the amount of idle energy reserved at level k for processing unit a, where $k, h \in [1, l]$, $average(t_a(v_i))$ be the average execution time of task v_i and p is the number of processing units. That is

$$\sigma_{k,a} = \frac{\beta_a}{p-1} \left(\sum_{v_i \in [k,h]} average(t_a(v_i)) - \sum_{v_i \in \mathbf{V}_a} t_a(v_i) \right) \quad (4)$$

The implementation of ESL algorithm will be described in the following section.

3.2.2 Energy-sufficiency level assignment (ESL) algorithm

In this section, formulation of the proposed energy-sufficiency level (ESL) assignment algorithm is described. Let P_{v_i} represent a set of processing units which is capable of processing task v_i. It is assumed that the characteristics of each task v_i are different such that the competency of each processing unit on each task is different. In other words, each processing unit requires different amount of energy to process the same task and, inversely, each task requires different amount of energy to be run on the same processing unit. The values for λ_a, β_a, $t_a(v_i)$, $d_a(v_i)$, ψ_a, and w_a for a processing unit a are also assumed prior to the application of the algorithm.

The scheduling scheme employed by ESL algorithm contains two phases. In the first phase, tasks are scheduled to attain the shortest system finish time using ESL Algorithm 1. The remaining idle slots after all tasks are scheduled are shorten in the second phase using ESL Algorithm 2 to further reduce the system finish time. The algorithm proceeds as follows.

1) Candidate processing units for each task v_i are selected from a set $\mathbf{P}_{v_i}$. The earliest finish time (EFT) of all tasks, in each level l in the dependent task graph G, on all available processing units are estimated. The first task to be assigned to its candidate processing units on each level is the task which gives the longest EFT. When the first task has been assigned, the EFT of all the remaining tasks were recalculated. The next task to be assigned is the task which gives the longest EFT in the recalculation. The process repeats until all tasks are assigned. The primary candidate processing units for a task is the processing unit which when executes the task gives the shortest EFT. The processing unit is marked as the secondary candidate processing unit for a task when it gives the second shortest EFT. In a similar manner, the processing unit is marked as the tertiary candidate processing unit if it gives the third shortest EFT. The detail of this step is given in ESL Algorithm 1. In case that battery power supply is limited, the *BatteryCheck* function is applied to reserve the energy for task execution using Equations 3 and 4. Details of the *BatteryCheck* function is described below. In case battery supply is unlimited, the *BatteryCheck* function will be skipped and the scheduling is made according to ESL Algorithm 1.
2) Reduce the remaining idle slots in some of the processing unit to minimize system finish time. An idle slot can be shortened by reassigning a task to be

executed in this slot. For example, if there is an idle slot preceding the current task induced by its immediate parent tasks assigned to another processing unit, given that the parent task can be processed at the same processing unit as the current task and the current start can be executed earlier than the original schedule, the parent task will be reassigned to this processing unit. The detail of this step is implemented in ESL Algorithm 2 which is described below.

Algorithm 1 Level-based task scheduling

Require: v_i, $\mathbf{v}_a$, and G.

1: Let $\varphi_{v_i,a}$ be the EST of v_i on processing unit a.
2: Let $\zeta_{v_i,a}$ be the EFT of task v_i on processing unit a.
3: **for** all levels l in graph G **do**
4: Let δ_l be a set of tasks v_i on level l.
5: Find task $v_i \in (\delta_l \cap \mathbf{V}_a)$ that can be executed on only one processing unit a.
6: Assign v_i to this processing unit at time slot $\varphi_{v_i,a}$ and remove v_i from δ_l.
7: Compute $\varphi_{v_i,a}$ of each $v_i \in \delta_l$.
8: **while** $\exists$ unassigned v_i in l **do**
9: **for** all tasks $v_i \in \delta_l$ **do** ▷ build EFT candidate task list
10: **for** all candidate processing unit a of v_i **do**
11: **if** v_i can be inserted in front of task v_k **then**
12: $\zeta_{v_i,a} = \zeta_{v_k,a} + (\varphi_{v_i,a} + t_a(v_i) - \varphi_{v_k,a})$
13: **else**
14: $\zeta_{v_i,a} = \varphi_{v_i,a} + t_a(v_i)$
15: **end if**
16: Marked v_i as v_k if it has the largest $\zeta_{v_i,a}$.
17: **end for**
18: **end for**
19: Let a be the processing unit that gives the shortest EFT for v_k.
20: Mark a as primary candidate processing unit.
21: Mark the processing unit with the next shortest EFT as secondary, tertiary candidate processing unit and so on.
22: Let $\mathbf{P}_{v_k}$ be the set of candidate processing units of v_k sorted in order of candidate processing unit.
23: BATTERYCHECK($v_k, \mathbf{P}_{v_k}, G$)
24: Assign v_k to the first order of candidate processing unit in $\mathbf{P}_{v_k}$ at time slot $\varphi_{v_k,a}$.
25: Remove v_k from δ_l.
26: Mark ancestor task u_{v_k} as the latest finish receiving data of task v_k.
27: **end while**
28: **end for**
29: **return** The scheduled list of tasks v_i on its assigned processing unit.

Figure 16 ESL algorithm 1: Level-based task scheduling.

function BATTERYCHECK(v_i, $\mathbf{P}_{v_i}$, G)
  for all $a \in \mathbf{P}_{v_i}$ **do**
    Temporarily insert v_i on candidate processing unit a at time slot $\varphi_{v_i,a}$.
    Let e'_a be the energy consumption after assigned task v_i to processing unit a.
    $e'_a = e_a + \sigma_{k,a} + \rho_a + (\alpha_a t_a(v_i) + \eta_a \lambda_a d_a(v_i) + \mu_a \beta_a w_a(v_i))$
    if $e'_a > \mathbf{B}_a$ **then**
      Remove a from $\mathbf{P}_{v_i}$.
    end if
  end for
  if $\mathbf{P}_{v_i}$ is empty **then**
    Halt the system.
  end if
  return $\mathbf{P}_{v_i}$.
end function

Figure 17 Function Battery check for ESL algorithm.

Algorithm 2 Reducing idle slot in task scheduling

Require: v_i, u_{v_i}, G, and scheduled list of tasks v_i.

1: Let Φ_G be the system finish time.
2: Let v_k be the latest task to finish.
3: **while** task v_k is not in the first level **do**
4: Traverse the task list upward starting at v_k until an empty slot is found and let v_i be
5: the task before this idle slot.
6: **if** u_{v_i} can be executed on the same processing unit as v_i **then**
7: Temporarily move u_{v_i} from the present slot and insert u_{v_i} at EST of processing
8: unit a.
9: Let Φ'_G be the new system finish time.
10: Let c'_a be new energy consumption after temporarily insert task u_{v_i} on processing unit a.
11: $c'_a = c_a + \alpha_a t_a(u_{v_i}) + \eta_a \lambda_a d_a(u_{v_i}) + \mu_a \beta_a w_a(u_{v_i}) - \beta_a(\Phi_G < \Phi'_G)$
12: **if** $\Phi'_G < \Phi_G$ and $c'_a < B_a$ **then**
13: Permanently assign u_{v_i} to this EST on processing unit a.
14: **else**
15: Remove u_{v_i} from processing unit a and put it back to the original processing unit.
16: **end if**
17: **end if**
18: Mark this task u_{v_i} to be v_k.
19: **end while**
20: **return** The scheduled list of tasks v_i on its assigned processing unit.

Figure 18 ESL algorithm 2: Reducing idle slot in task scheduling.

The first experimental case will be used to illustrate the implementation of the two algorithms. In this experimental case, the dependent task graph is taken from that of Figure 19 and the cost taken from Table 17. For this task graph, a root vertex is connected to its dependent vertices by direct edges. The weight on each edge denotes transmission time. The vertices are labeled by their name along with the names of the processing units which are able to execute the tasks in the parenthesis. The values for the variables which will be used to estimate the energy consumption include the amount of processing energy per unit time (α_a), waiting energy per unit time (β_a), transmission energy per unit amount of data (λ_a), and data transmission rate $r_{a,b}$. These values are given in Table 19 and Table 20.

Table 17 A list of execution time $t_a(v_i)$ for each processing unit in the dependent task graph of Figure 19 (first case).

Task	Execution time		
v_i	$t_a(v_i)$	$t_b(v_i)$	$t_c(v_i)$
1	14	16	9
2	13	19	18
3	11	13	19
4	13	8	17
5	12	13	10
6	13	16	9
7	7	15	11
8	5	11	14
9	18	12	20
10	21	7	16

Values of energy consumption are calculated in the following examples. Processing unit execution time $t_a(v_i)$ for all tasks in the dependent task graph of Figure 19 on all processing units are presented in Table 17. From the table, $t_c(v_1)=9$. Given that the execution energy per unit time $\alpha_c=0.058$, the estimated execution energy is determined from $\alpha_c t_c(v_1)=0.058\times 9=0.52$.

Transmission energy is calculated at the sender site as data are transmitted across the network. For example, transmission energy of processing unit c executing task v_1 is calculated from the results of 9 and 11 unit of data sent to v_4 and v_5 on processing unit b. In the meantime, the result of 12 unit of data sent to v_3 on processing unit a. Because v_2 and v_6 are also executed on processing unit c, no transmission energy incurred. The transmission energy expended by processing unit c is equal $\lambda_c d_c(v_1) = 0.050 \times (9+11+12) = 1.60$.

Idle energy is calculated from the time unit that a processing unit is put on wait state. For example, consider the execution of task v_4 on processing unit b. Because task v_4 is a descendent task of task v_1, processing unit b is put on a wait state for 9 units of time when task v_1 is being executed on processing unit c and 9 units of time for the delay before the result from processing unit c is sent to processing unit b. For processing unit b, the idle energy at wait state is equal to $\beta_b = 0.017$ thus the idle energy is calculated from
$\beta_b(w_b(v_1) + \psi_b(v_4)) = 0.017 \times (9+9) = 0.31$.

As the time complexity of the ESL algorithm is in the order of $\max\left(O\left(\left(v^2 \times p\right)/l\right), O\left(v^2\right)\right)$, scheduling time for the assignment using the dependent task graph of Figure 19 is calculated from $T_{alg}(ESL) = \max\left(\left(10^2 \times 3\right)/4, 10^2\right) = \max(75, 100) = 100$, where $v = 10$, $p = 3$, and $l = 4$. Thus the energy due to scheduling is equal to $\alpha_a T_{alg}(ESL) = 0.059 \times 100 = 5.90$.

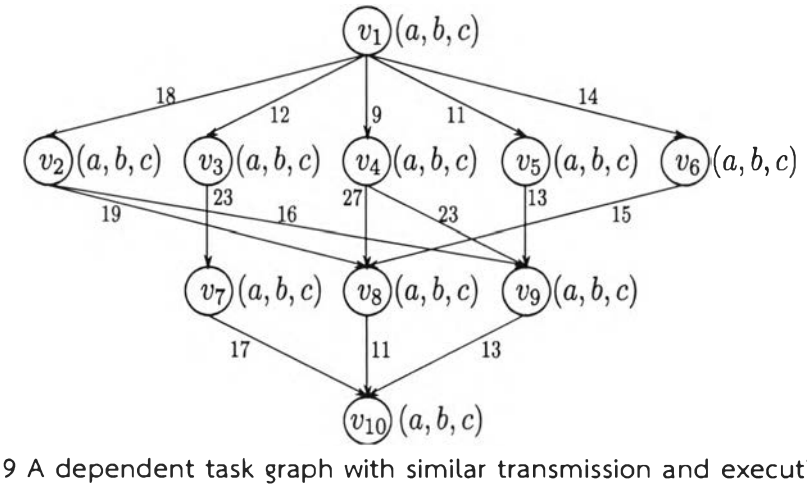


Figure 19 A dependent task graph with similar transmission and execution costs as that of [30] (first case).

In the case of no limitation to the battery supply, scheduling can be performed following ESL Algorithm 1 and 2. From the depend task graph of Figure 19, there is one unassigned task in the first level, that is $v_1 \in \delta_1$. Processing unit c is marked as a primary processing unit for task v_1 since, in comparison with processing unit a and b, processing unit c gives the shortest EFT, $\zeta_{v_1,c} = 9$ in units of time.

For the level which contains more than one task, the EFT of all tasks on each processing unit are calculated and the task with the longest EFT will be the first task to be considered. In the second level of the dependent task graph of Figure 19, there are five tasks, $v_2, v_3, v_4, v_5, v_6 \in \delta_2$. The task with the longest EFT is task v_2 executed on processing unit b, $\zeta_{v_2,b} = 46$. In comparison with other processing units, processing unit c gives the shortest EFT when executing task v_2, that is, $\zeta_{v_2,c} = 27$. Thus, processing unit c is marked as a primary processing unit for task v_2. Next, the EFT of all tasks excluding task v_2, which has already been assigned on each processing unit are recalculated. This time, the task with the longest EFT is task v_3 executed on processing unit c, $\zeta_{v_3,c} = 46$. This makes task v_3 the second task to be considered. For task v_3, the minimum EFT is achieved when it is executed on processing unit a, $\zeta_{v_3,a} = 32$, thus processing unit a is assigned as a primary processing unit of task v_3. The next longest EFT determined are that for task v_6 with $\zeta_{v_6,a} = 45$. As $\zeta_{v_6,c} = 36$ is the shortest EFT for task v_6, processing unit c is assigned as a primary processing unit of task v_6. The last task to be assigned in the second level is task v_4 with $\zeta_{v_4,c} = 44$. The shortest EFT of task v_4 is $\zeta_{v_4,b} = 26$ which defines processing unit b as a primary processing unit of task v_4. For v_5, the values of shortest EFT and primary processing unit are $\zeta_{v_5,b} = 39$ and b, respectively.

There are three tasks in the third level $v_7, v_8, v_9 \in \delta_3$. By using the same scheme as employed in the second level, the first task to be assigned is task v_9 with the longest EFT $\zeta_{v_9,c} = 72$ and the shortest EFT $\zeta_{v_9,b} = 55$. Thus, processing unit b is marked as a primary processing unit of task v_9. After the recalculation of EFT, the longest EFT is determined from task v_7 with $\zeta_{v_7,b} = 70$. As the shortest EFT for task v_7 is $\zeta_{v_7,a} = 39$, the processing a is assigned as a primary processing unit of task v_7. Following this scheme, processing unit a is assigned to task v_8 with the shortest EFT $\zeta_{v_8,a} = 58$ and, in the fourth level, processing unit b assigned to task v_{10} with $\zeta_{v_{10},b} = 76$. The result from ESL Algorithm 1 is summarized in Figure 20.

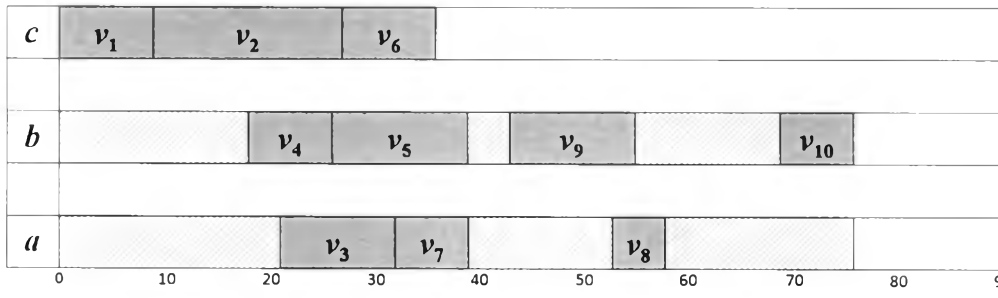


Figure 20 Results obtained from Algorithm 1 applied to the dependent task graph of Figure 19.

After the final task has been assigned, the system finish time will be reduced by employing ESL Algorithm 2. As seen in Figure 20, the final task to be executed is task v_{10} on processing unit b. Thus, task v_{10} is taken as a starting point and the ESL Algorithm 2 will traverse upwards from task v_{10}. The next empty slot preceding task v_{10} is the slot between task v_{10} and its ancestor task v_8, denoted as $u_{v_{10}}$ in ESL Algorithm 1.

The ESL Algorithm 2 will temporarily move v_8 to the earliest start time (EST) of task v_8 on the same processing unit as task v_{10}. This results in reducing the system finish time by 73 unit of time thus task v_8 is reassigned to processing unit b. In the next step, task v_8 will be taken as a starting point and the process is repeated. For this dependent task graph, no other moves can reduce the system finish time and the result from ESL Algorithm 2 is presented in Figure 21. The result energy consumption for processing units a, b, and c are $e_a = 9.12$, $e_b = 3.33$ and $e_c = 5.12$, respectively, with the total energy of 17.57.

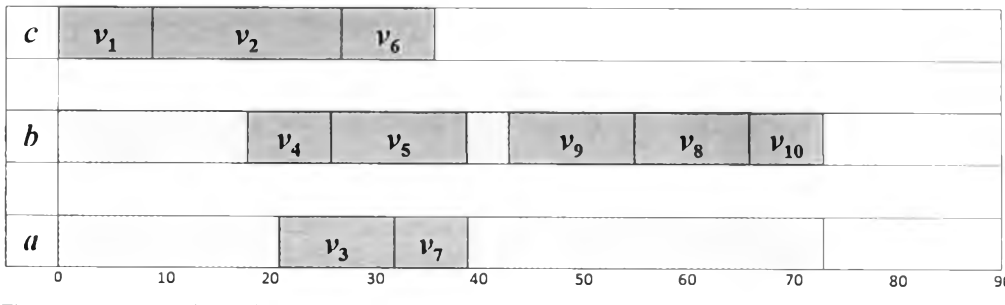


Figure 21 Results obtained from Algorithm 2 applied to the dependent task graph of Figure 19.

3.2.3 Experimental results

Experiments were conducted on six experimental cases comparing six algorithms, namely, HEFT, PETS, Lookahead, CEFT, PEFT, and the proposed ESL algorithms. Comparisons were made on the energy consumption aspect of all comparative algorithms with an emphasis on system finish time, scheduling energy, and task execution energy. The experiments were performed under the assumption of unlimited battery supply. In the case that the power supply is limited, only the ESL algorithm can be performed till completion of the assignment. The performance of ESL algorithm under limited battery supply is demonstrated in the seventh experimental case.

In the first six experimental cases, dependent task graphs and their corresponding weights were taken from the literature for fair comparison. The dependent task graph employed in case 1 was taken from the problem set up for the HEFT [30], case 2 from the problem set up for the PETS [31], case 3 from the problem set up for the Lookahead [32], case 4 from the problem set up for the CEFT [33], case 5 from the problem set up for the PEFT [34], and case 6 from [39]. The dependent task graph used in case 7 was the same as that in case 6 except for the added limitation on the battery supply. Other parameters involved in the energy consumption calculation including the execution time of each graph on each processing unit, $\iota_a(v_i)$, the weight of data transfer delay time for each edge, $w_a(v_i)$, and the amount of data transmitted between processing units, $d_a(v_i)$, were assumed at the beginning of the simulation. The same scheme as demonstrated in Section 3.1.2 was used to calculate the total energy consumption.

There are four available unique processing units in this simulation. The operating parameters of the simulation as well as the capability and power consumption for all processing units are shown in Table 19 and Table 20. For this simulation three processing unit were used for only task execution saved one processing unit which was set as the main processing unit and was used to perform scheduling as well as task execution. For example, if processing unit a is selected as the main processing unit, additional energy due to scheduling $\alpha_a T_{alg}(C)$ and addition idle energy $\beta_a \psi_a(v_j)$ will be added on top of its regular energy consumption.

Table 18 Power consumption of benchmark desktop CPUs taken from [37] and [38]. The values were used to evaluate energy consumption for peak and idle state of each processing unit in the experiment.

Processing unit	CPU	Clock (GHz)	Power consumption (W)	
			peak state	idle state
a	Intel Core 2 Extreme QX9770	3.2	211	92
b	AMD FX-8350	4.0	210	61
c	Intel Core i7-3960X	3.3	210	57
d	Intel Core i7-3930K	3.2	206	57

The values for energy consumption constants including execution energy at peak state per unit time (α_i), waiting energy at idle state per unit time (β_i), transmission energy per unit data (λ_i), and data transmission rate between processing unit a and b $(r_{a,b})$ used in this study are presented in Table 19 and Table 20. The values for α_i and β_i were determined from the power consumption presented in Table 18. For example, for processing unit a, $\alpha_a = 211 \div 3600 = 0.059$ and $\beta_a = 92 \div 3600 = 0.026$. The values for λ_i were randomly generated in the range between α_i and β_i, and the values for $r_{a,b}$ were taken from [39].

Table 19 Energy consumption constants for each processing unit $i \in \{a,b,c,d\}$. α_i is the execution energy at peak state per unit time, β_i is the idle energy at wait state per unit time, and λ_i is the transmission energy per unit data.

Energy constants	Processing Units			
	a	b	c	d
$\alpha_{i\in\{a,b,c,d\}}$	0.059	0.058	0.058	0.057
$\beta_{i\in\{a,b,c,d\}}$	0.026	0.017	0.016	0.016
$\lambda_{i\in\{a,b,c,d\}}$	0.043	0.038	0.037	0.037

Table 20 Data transmission rate $r_{a,b}$ for all processing unit pairs in unit amount of data per unit time.

Processing Unit	a	b	c	d
a	–	1	0.5	1
b	1	–	0.5	1.25
c	0.5	0.5	–	1.25
d	1	1.25	1.25	–

The dependent task graph used in the first experimental case is shown in Figure 19 and the execution time of each task on each processing unit $t_a(v_i)$ is given in Table 17. Also shown in the task graph are the values for the data transfer delay $w_a(v_i)$. It can be deduced from the task graph that the values become $v=10$, $p=3$, and $l=4$. For this experimental case, $w_a(v_i)$ were used instead of the amount of transmitted data $d_a(v_i)$ to estimate transmission energy incurred when data were transferred between processing units. The energy consumptions as a result of each algorithm can be calculated using the scheme described in Section 4.2. The results are presented in Table 21 to Table 26.

Table 21 Energy consumption in each processing unit as a result of the HEFT algorithm (first case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	1.06	1.16	1.61	17.70	21.53
b	2.49	1.60	0.63	0.00	4.72
c	2.84	2.63	0.000	0.00	5.47
Total	6.39	5.39	2.24	17.70	31.72

Table 22 Energy consumption in each processing unit as a result of the PETS algorithm (first case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	1.06	1.16	1.53	23.60	27.35
b	2.73	0.00	0.51	0.00	3.24
c	2.61	4.00	0.00	0.00	6.61
Total	6.40	5.16	2.04	23.60	37.20

Table 23 Energy consumption in each processing unit as a result of the Lookahead algorithm (first case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	3.72	2.75	0.49	132.75	139.71
b	1.57	1.03	0.94	0.00	3.54
c	0.58	0.48	0.40	0.00	1.46
Total	5.87	4.26	1.83	132.75	144.71

Table 24 Energy consumption in each processing unit as a result of the CEFT algorithm (first case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	2.95	2.88	0.81	177.00	183.64
b	2.32	1.03	0.70	0.00	4.05
c	0.52	0.56	0.45	0.00	1.53
Total	5.79	4.47	1.96	177.00	189.22

Table 25 Energy consumption in each processing unit as a result of the PEFT algorithm (first case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	2.12	1.20	1.27	17.70	22.29
b	3.60	3.15	0.39	0.00	7.14
c	0.58	0.48	0.43	0.00	1.49
Total	6.30	4.83	2.09	17.70	30.92

Table 26 Energy consumption in each processing unit as a result of the ESL algorithm (first case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	1.06	0.73	1.43	5.90	9.12
b	2.96	0.00	0.37	0.00	3.33
c	2.09	3.03	0.00	0.00	5.12
Total	6.11	3.76	1.80	5.90	17.57

Calculated energy consumptions for each algorithm in the first experimental case are shown in Figure 22. The bars represent total energy consumption which includes the execution energy, transmission energy, and idle energy of processing unit a, b, and c, and the algorithm overhead *(Alg.)* energy incurred in the main processing unit. The energy consumptions for the Lookahead and CEFT algorithms were considerably larger than those of the other algorithms which are not shown in full size of the figure due to excessive scaling.

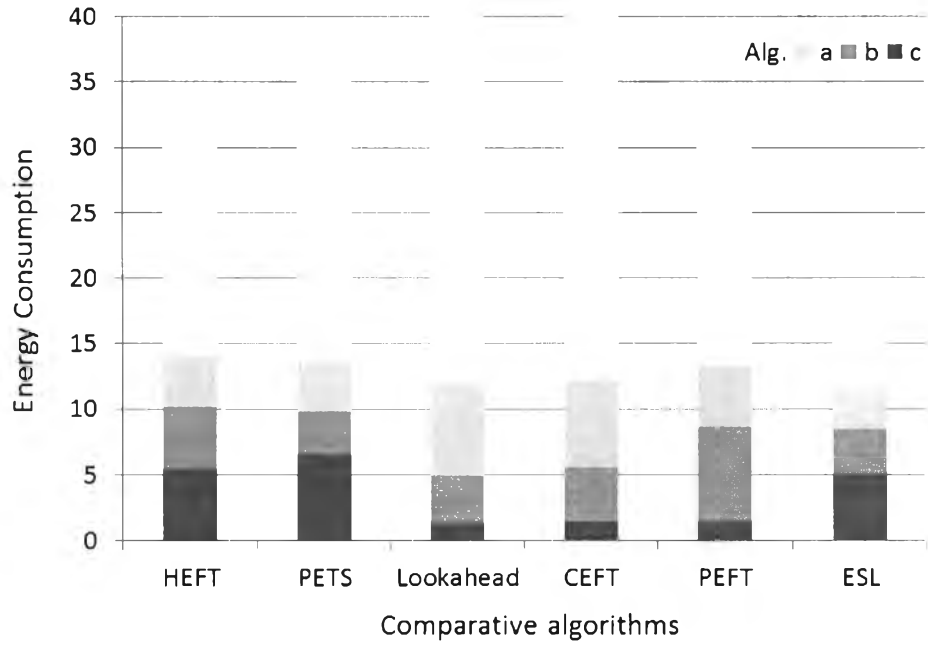


Figure 22 Results of energy consumption on all processing unit for each scheduling algorithm (first case).

Calculated energy consumptions for each algorithm in the first experimental case are shown in Figure 22. The scheduled tasks on processing unit a, b, and c using HEFT, PETS, Lookahead, CEFT, PEFT, and ESL algorithms are shown in Figure 23 to Figure 27. From the figures, system finish times are 80, 77, 82, 81, 85, and 73 units of time, respectively. It can be seen that ESL algorithm has the minimum energy consumption and the shortest system finish time.

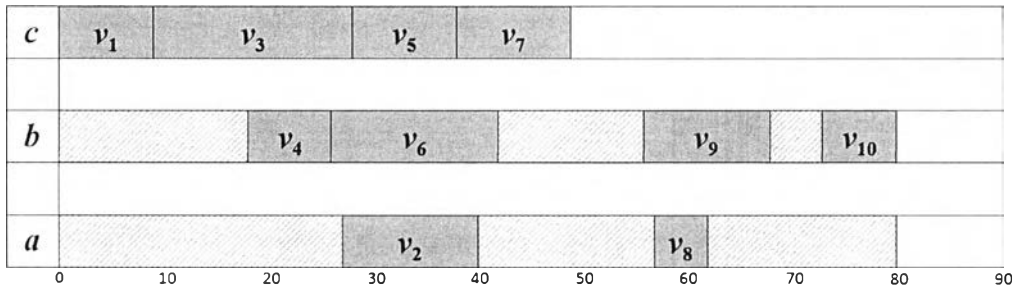


Figure 23 Results of task scheduling using the HEFT Algorithm (first case).

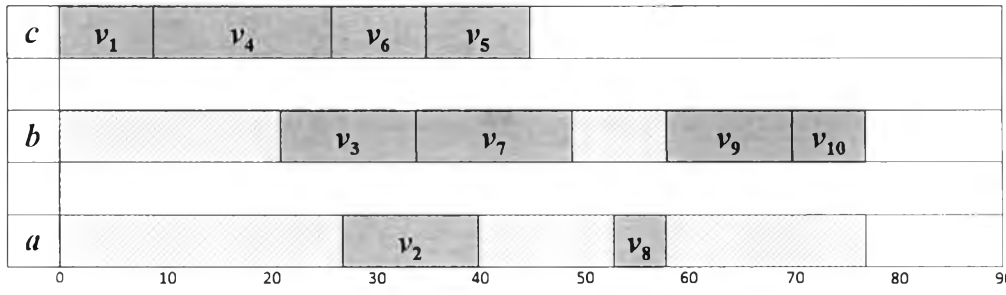


Figure 24 Results of task scheduling using the PETS Algorithm (first case).

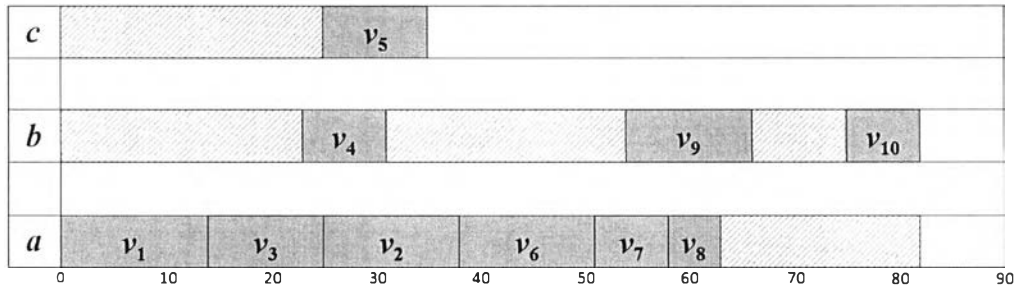


Figure 25 Results of task scheduling using the Lookahead Algorithm (first case).

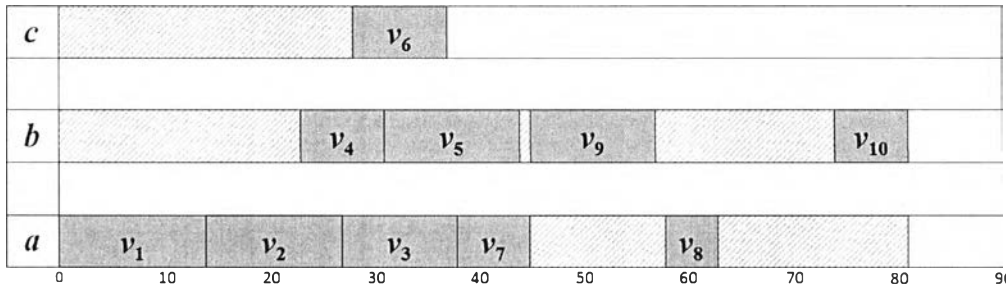


Figure 26 Results of task scheduling using the CEFT Algorithm (first case).

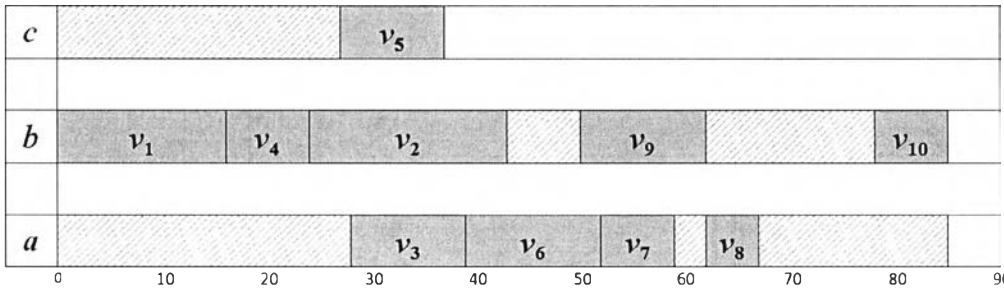


Figure 27 Results of task scheduling using the PEFT Algorithm (first case).

The dependent task graph used in the second experimental case is shown in Figure 28 and the execution time of each task on each processing unit $t_a(v_i)$ is given in Table 27. Also shown in the task graph are the values for the data transfer delay $w_a(v_i)$. It can be deduced from the task graph that the values become $v=11$, $p=3$, and $l=5$. The energy consumptions as a result of each algorithm are presented in Table 28 to Table 33.

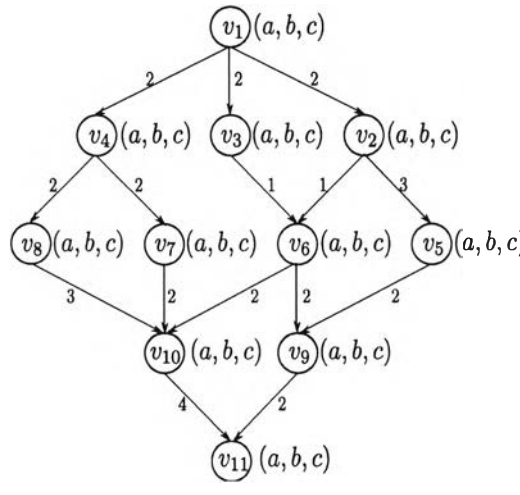


Figure 28 A dependent task graph with transmission and execution costs taken from [31] (second case).

Table 27 A list of execution time $t_a(v_i)$ by each processing unit for dependent task graph of Figure 28 (second case).

Task	Execution time		
v_i	$t_a(v_i)$	$t_b(v_i)$	$t_c(v_i)$
1	4	4	4
2	5	5	5
3	4	6	4
4	3	3	3
5	3	5	3
6	3	7	2
7	5	8	5
8	2	4	5
9	5	6	7
10	3	7	5
11	5	6	7

Table 28 Energy consumption in each processing unit as a result of the HEFT algorithm (second case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	1.30	0.52	0.18	21.42	23.42
b	0.93	0.27	0.10	0.00	1.30
c	0.35	0.15	0.13	0.00	0.63
Total	2.58	0.94	0.41	21.42	25.35

Table 29 Energy consumption in each processing unit as a result of the PETS algorithm (second case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	1.48	0.3	0.05	28.85	30.68
b	0.75	0.27	0.12	0.00	1.14
c	0.35	0.19	0.10	0.00	0.64
Total	2.58	0.76	0.27	28.85	32.46

Table 30 Energy consumption in each processing unit as a result of the Lookahead algorithm (second case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	1.71	0.22	0.05	141.35	143.33
b	0.58	0.11	0.10	0.00	0.79
c	0.41	0.26	0.14	0.00	0.81
Total	2.70	0.59	0.29	141.35	144.93

Table 31 Energy consumption in each processing unit as a result of the CEFT algorithm (second case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	1.36	0.34	0.21	235.59	237.50
b	0.93	0.23	0.14	0.00	1.30
c	0.46	0.11	0.10	0.00	0.67
Total	2.75	0.68	0.45	235.59	239.47

Table 32 Energy consumption in each processing unit as a result of the PEFT algorithm (second case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	1.59	0.30	0.00	21.42	23.31
b	0.52	0.23	0.19	0.00	0.94
c	0.35	0.15	0.10	0.00	0.60
Total	2.46	0.68	0.29	21.42	24.85

Table 33 Energy consumption in each processing unit as a result of the ESL algorithm (second case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	1.30	0.00	0.16	7.14	8.60
b	0.58	0.11	0.10	0.00	0.79
c	0.70	0.30	0.00	0.00	1.00
Total	2.58	0.41	0.26	7.14	10.39

Calculated energy consumptions for each algorithm in the second experimental case are shown in Figure 29. The scheduled tasks on processing unit a, b, and c using HEFT, PETS, Lookahead, CEFT, PEFT, and ESL algorithms are shown in Figure 30 to Figure 35. From Figure 30 to Figure 35, the system finish time are 29, 27, 31, 31, 27, and 28 units of time, respectively. Although ESL algorithm did not give the shortest finish time, the resulting time was only slightly longer than the shortest value given by the PETS and PEFT algorithms. It can be seen that ESL algorithm consumes minimum energy within acceptable system finish time.

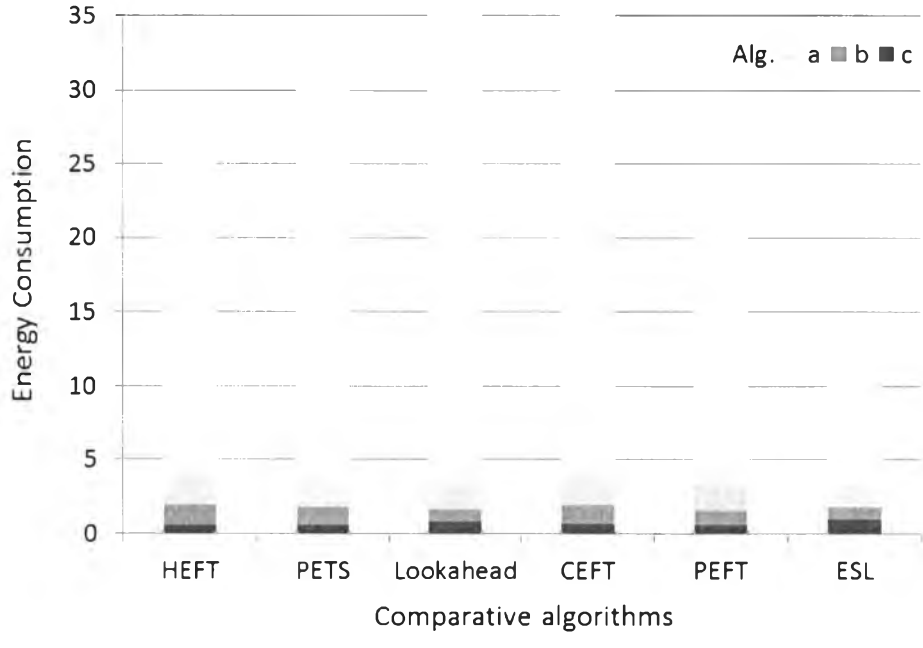


Figure 29 Result of energy consumption on each processing unit for each scheduling algorithm (second case).

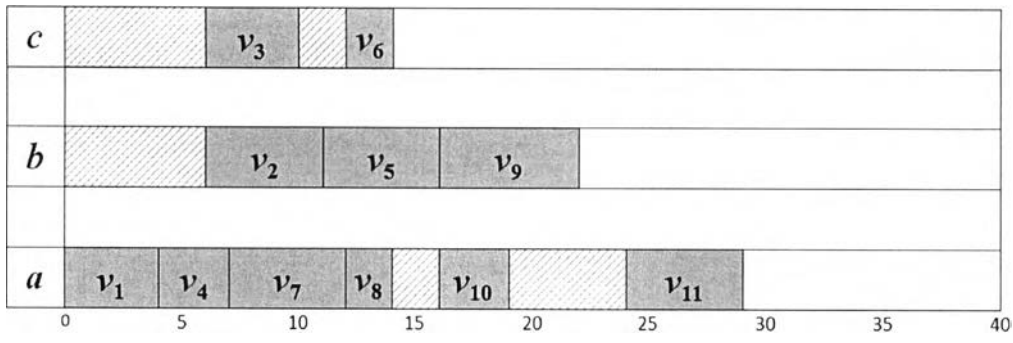


Figure 30 Results of task scheduling using the HEFT Algorithm (second case).

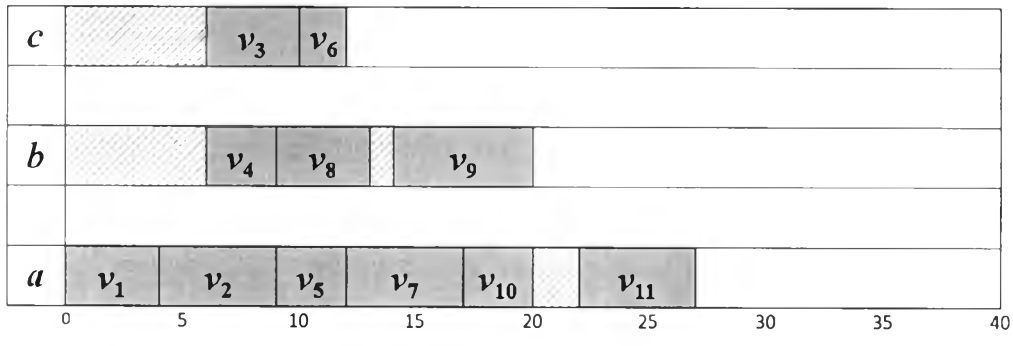


Figure 31 Results of task scheduling using the PETS (second case).

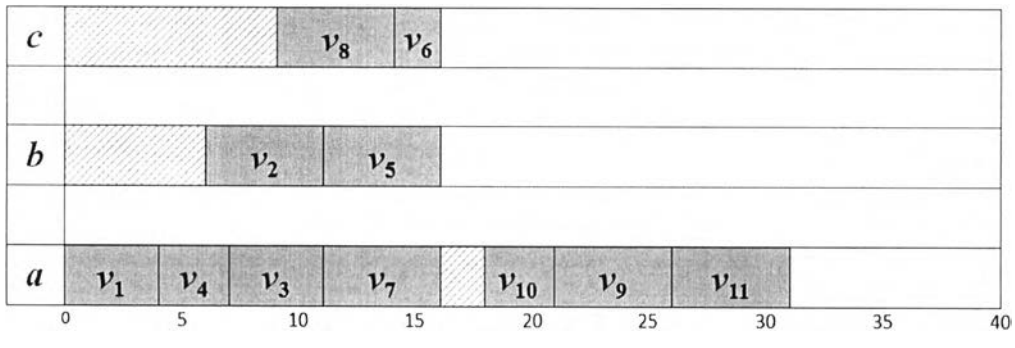


Figure 32 Results of task scheduling using the Lookahead Algorithm (second case).

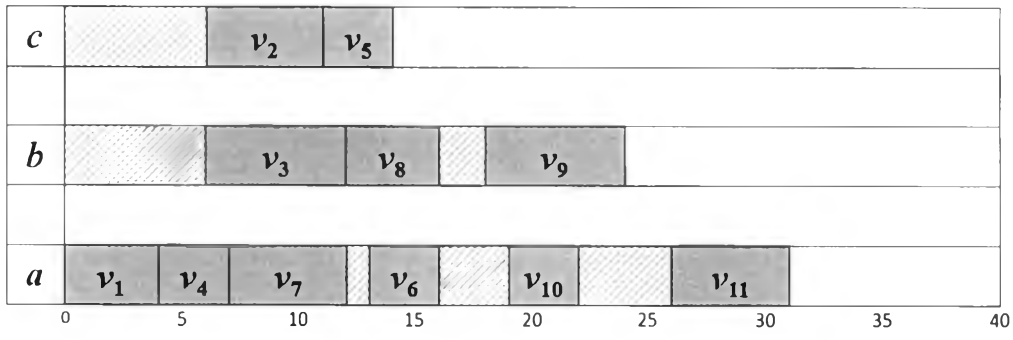


Figure 33 Results of task scheduling using the CEFT (second case).

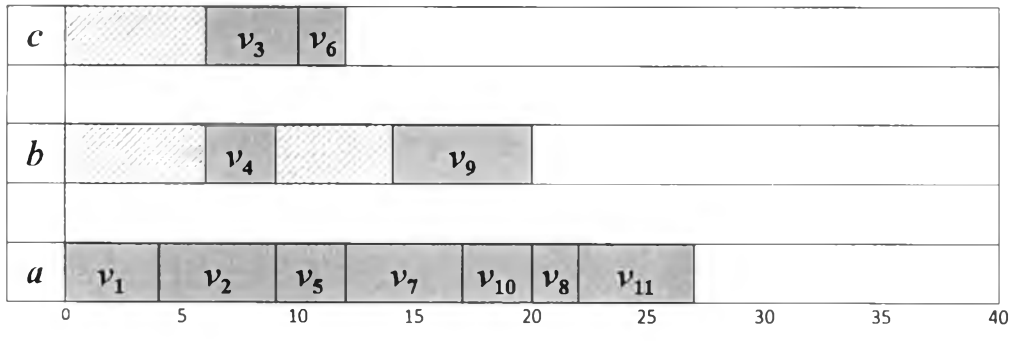


Figure 34 Results of task scheduling using the PEFT Algorithm (second case).

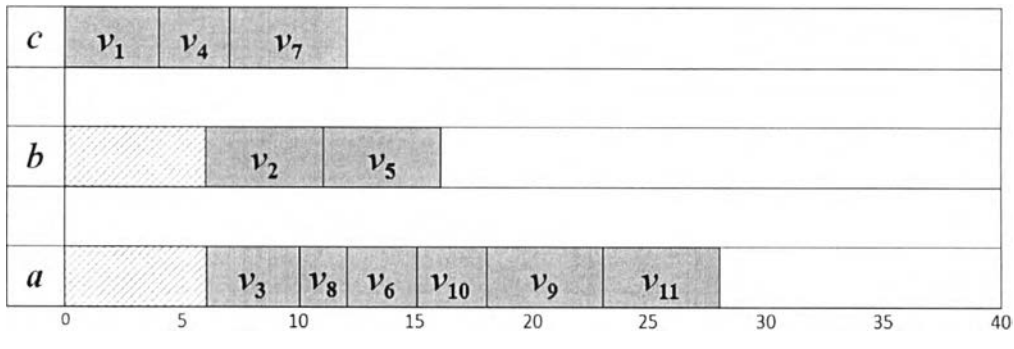


Figure 35 Results of task scheduling using the ESL Algorithm (second case).

The dependent task graph used in the third experimental case is shown in Figure 36 and the execution time of each task on each processing unit $t_a(v_i)$ is given in Table 34. Also shown in the task graph are the values for the data transfer delay $w_a(v_i)$. It can be deduced from the task graph that the values become $v=9$, $p=3$, and $l=4$. The energy consumptions as a result of each algorithm are presented in Table 35 to Table 40.

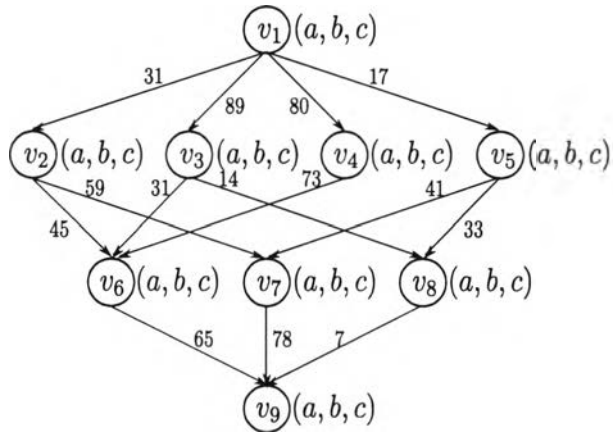


Figure 36 A dependent task graph with transmission and execution costs taken from [32] (third case).

Table 34 A list of execution time $t_a(v_i)$ by each processing unit for dependent task graph of Figure 36 (third case).

Task	Execution time		
v_i	$t_a(v_i)$	$t_b(v_i)$	$t_c(v_i)$
1	19	41	34
2	28	46	20
3	36	34	62
4	15	25	37
5	30	50	54
6	33	35	59
7	12	20	21
8	13	22	24
9	41	68	73

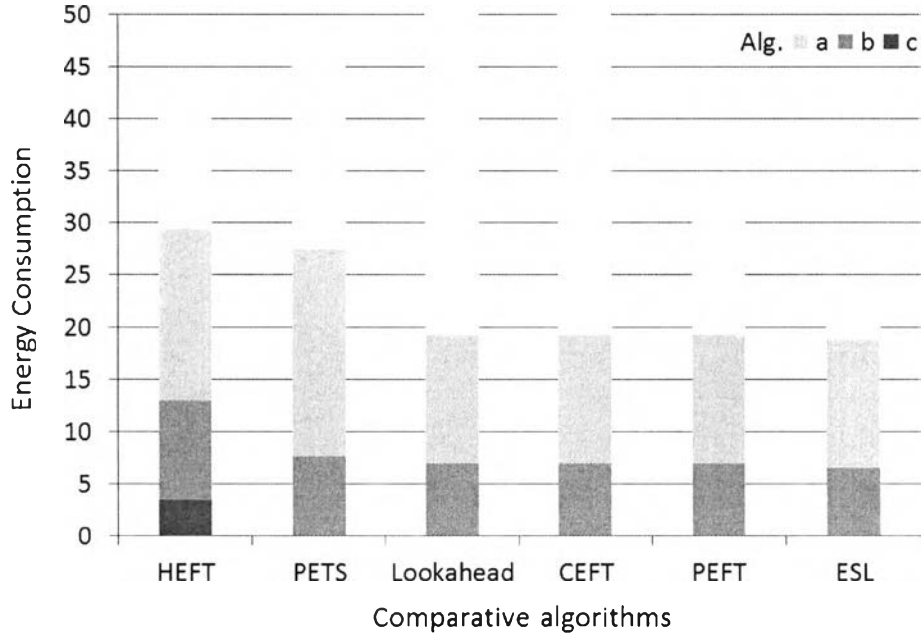


Figure 37 Result of energy consumption on each processing unit for each scheduling algorithm (third case).

Calculated energy consumptions for each algorithm in the third experimental case are shown in Figure 37. It can be seen from the figure that the total energy consumption of the Lookahead and CEFT algorithms are the largest due to their large algorithm overhead. The ESL algorithm, on the other hand, had the least algorithm overhead thus given the least total energy consumption.

The scheduled tasks on processing unit a, b, and c using HEFT, PETS, Lookahead, CEFT, PEFT, and ESL algorithms are shown in Figure 38 to Figure 43. From the figures, the system finish times are 260 and 257 for HEFT, PETS algorithms and 184 for the Lookahead, CEFT, PEFT, and ESL algorithms.

Table 35 Energy consumption in each processing unit as a result of the HEFT algorithm (third case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	10.15	3.87	2.29	14.34	30.65
b	4.06	4.22	1.21	0.00	9.49
c	1.39	0.26	1.90	0.00	3.55
Total	15.6	8.35	5.40	14.34	43.69

Table 36 Energy consumption in each processing unit as a result of the PETS algorithm (third case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	11.92	6.32	1.43	18.90	38.57
b	2.44	3.23	2.07	0.00	7.74
c	0.00	0.00	0.00	0.00	0.00
Total	14.36	9.55	3.50	18.90	46.31

Table 37 Energy consumption in each processing unit as a result of the Lookahead algorithm (third case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	10.86	1.33	0.00	96.77	108.96
b	4.18	1.82	1.05	0.00	7.05
c	0.00	0.00	0.00	0.00	0.00
Total	15.04	3.15	1.05	96.77	116.01

Table 38 Energy consumption in each processing unit as a result of the CEFT algorithm (third case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	10.86	1.33	0.00	129.03	141.22
b	4.18	1.82	1.05	0.00	7.05
c	0.00	0.00	0.00	0.00	0.00
Total	15.04	3.15	1.05	129.03	148.27

Table 39 Energy consumption in each processing unit as a result of the PEFT algorithm (third case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	10.86	1.33	0.00	14.34	26.53
b	4.18	1.82	1.05	0.00	7.05
c	0.00	0.00	0.00	0.00	0.00
Total	15.04	3.15	1.05	14.34	33.58

Table 40 Energy consumption in each processing unit as a result of the ESL algorithm (third case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	10.86	1.33	0.00	4.78	16.97
b	4.18	1.82	0.61	0.00	6.61
c	0.00	0.00	0.00	0.00	0.00
Total	15.04	3.15	0.61	4.78	23.58

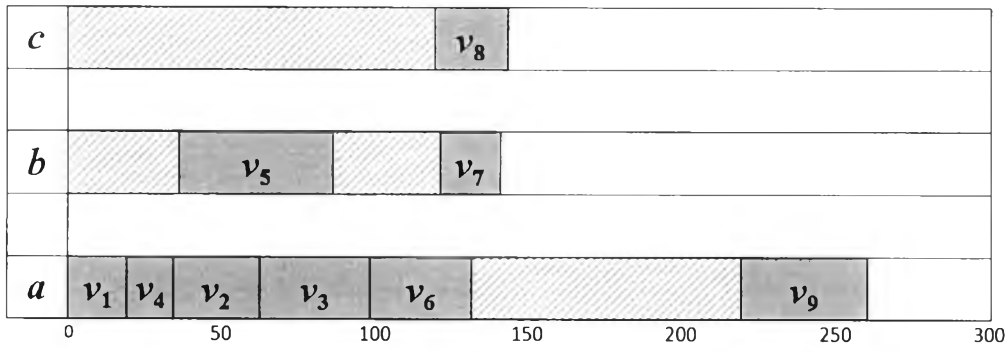


Figure 38 Results of task scheduling using the HEFT Algorithm (third case).

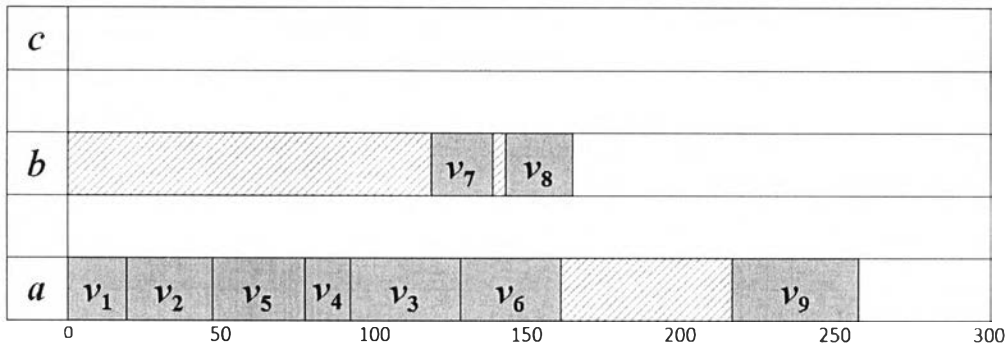


Figure 39 Results of task scheduling using the PETS Algorithm (third case).

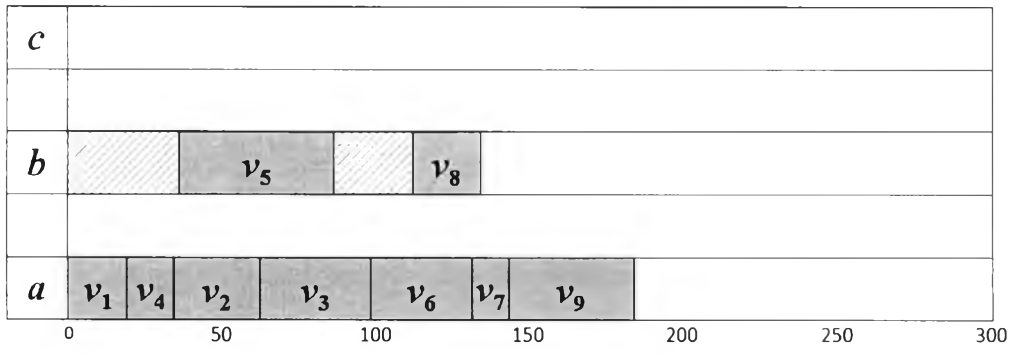


Figure 40 Results of task scheduling using the Lookahead Algorithm (third case).

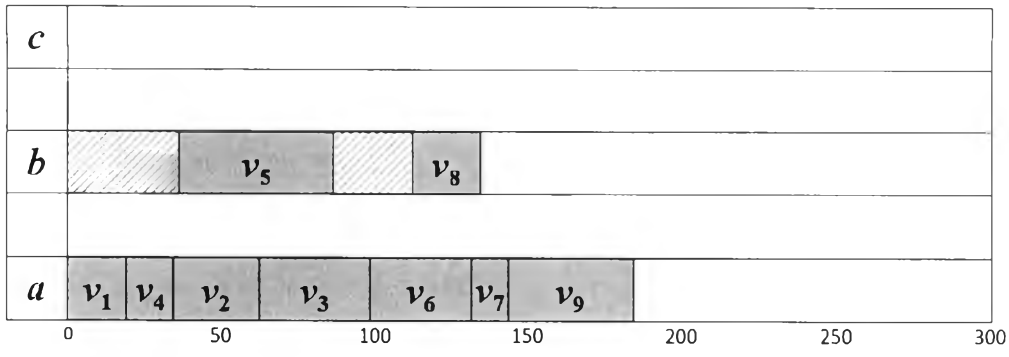


Figure 41 Results of task scheduling using the CEFT Algorithm (third case).

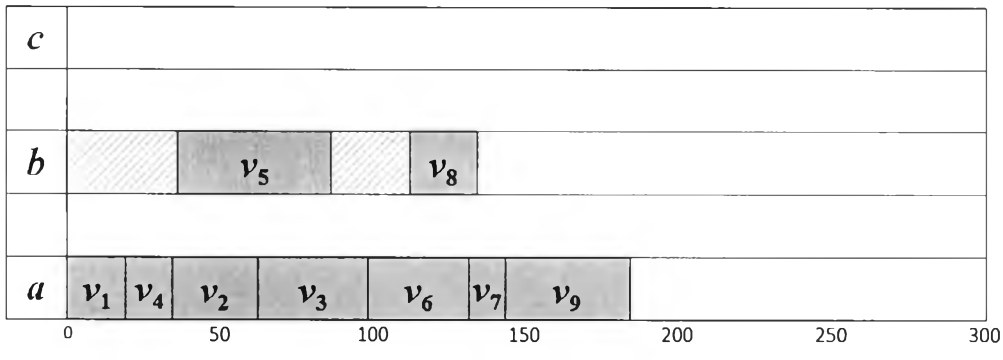


Figure 42 Results of task scheduling using the PEFT Algorithm (third case).

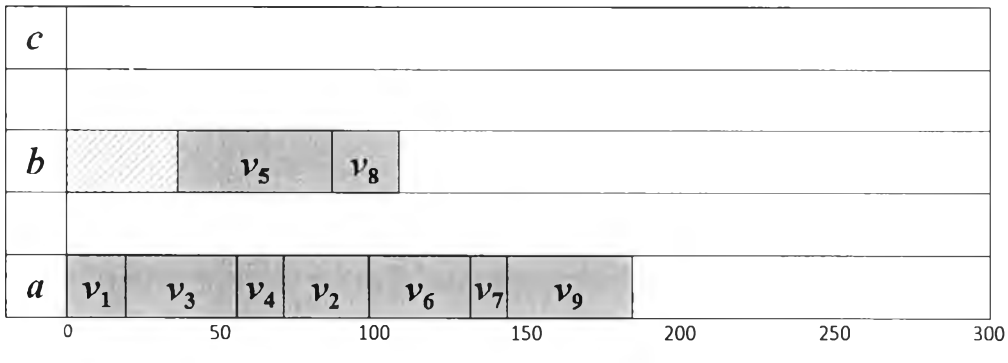


Figure 43 Results of task scheduling using the ESL Algorithm (third case).

The dependent task graph used in the fourth experimental case is shown in Figure 44 and the execution time of each task on each processing unit $t_a(v_i)$ is given in Table 41. Also shown in the task graph are the values for the data transfer delay $w_a(v_i)$. It can be deduced from the task graph that the values become $v=10$, $p=3$, and $l=5$. The energy consumptions as a result of each algorithm are presented in Table 42 to Table 47.

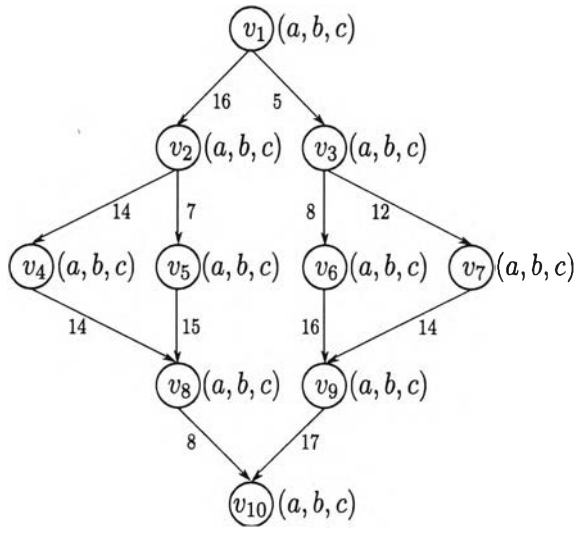


Figure 44 A dependent task graph with transmission and execution costs taken from [33] (fourth case).

Table 41 A list of execution time $t_a(v_i)$ by each processing unit for the dependent task graph of Figure 44 (fourth case).

Task	Execution time		
v_i	$t_a(v_i)$	$t_b(v_i)$	$t_c(v_i)$
1	7	8	9
2	11	14	17
3	12	15	18
4	10	8	12
5	5	7	6
6	9	7	5
7	6	8	7
8	14	12	10
9	10	8	6
10	11	13	15

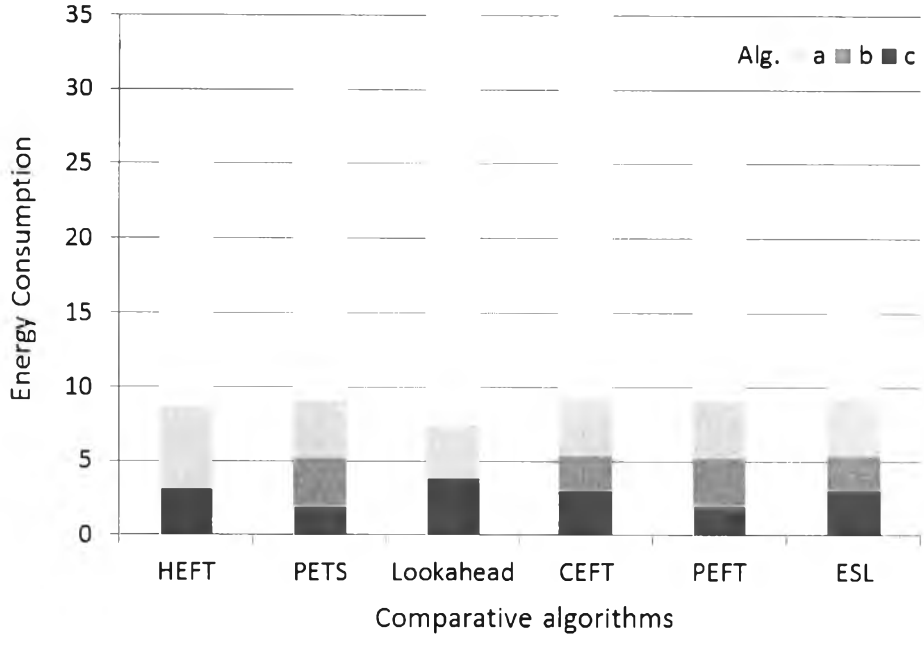


Figure 45 Result of energy consumption on each processing unit for each scheduling algorithm (fourth case).

Calculated energy consumptions for each algorithm in the fourth experimental case are shown in Figure 45. The scheduled tasks on processing unit a, b, and c using HEFT, PETS, Lookahead, CEFT, PEFT, and ESL algorithms are shown in Figure 46 to Figure 51. From Figure 46 to Figure 51, the system finish times are 85, 73, 76, 73, 73, and 73 units of time, respectively. It can be seen that ESL algorithm yielded both the lowest total energy consumption and the shortest system finish time.

Table 42 Energy consumption in each processing unit as a result of the HEFT algorithm (fourth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	2.71	1.85	1.01	17.7	23.27
b	0.00	0.00	0.00	0.00	0.00
c	2.44	0.00	0.69	0.00	3.13
Total	5.15	1.85	1.70	17.70	26.40

Table 43 Energy consumption in each processing unit as a result of the PETS algorithm (fourth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	1.65	1.12	1.17	23.60	27.54
b	2.96	0.00	0.37	0.00	3.33
c	0.93	0.30	0.58	0.00	1.81
Total	5.54	1.42	2.12	23.60	32.68

Table 44 Energy consumption in each processing unit as a result of the Lookahead algorithm (fourth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	3.01	0.00	0.65	106.20	109.86
b	0.00	0.00	0.00	0.00	0.00
c	2.61	1.22	0.00	0.00	3.83
Total	5.62	1.22	0.65	106.20	113.69

Table 45 Energy consumption in each processing unit as a result of the CEFT algorithm (fourth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	1.65	1.12	1.17	177.00	180.94
b	1.33	0.84	0.20	0.00	2.37
c	2.44	0.00	0.50	0.00	2.94
Total	5.42	1.96	1.87	177.00	186.25

Table 46 Energy consumption in each processing unit as a result of the PEFT algorithm (fourth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	1.65	1.12	1.17	17.70	21.64
b	2.96	0.00	0.37	0.00	3.33
c	0.93	0.30	0.58	0.00	1.81
Total	5.54	1.42	2.12	17.70	26.78

Table 47 Energy consumption in each processing unit as a result of the ESL algorithm (fourth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	1.65	1.12	1.17	5.90	9.84
b	1.33	0.84	0.20	0.00	2.37
c	2.44	0.00	0.50	0.00	2.94
Total	5.42	1.96	1.87	5.90	15.15

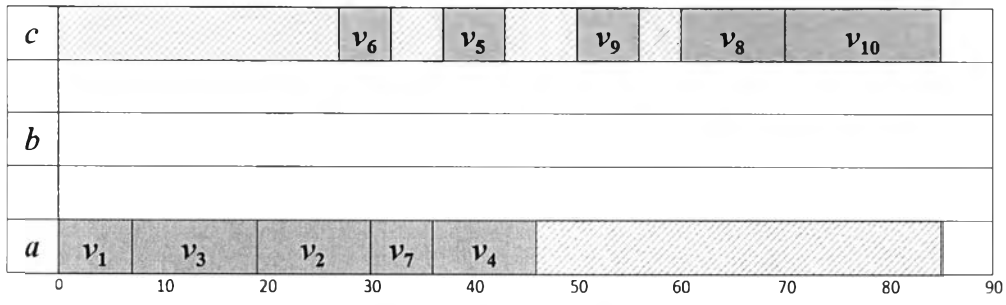


Figure 46 Results of task scheduling using the HEFT Algorithm (fourth case).

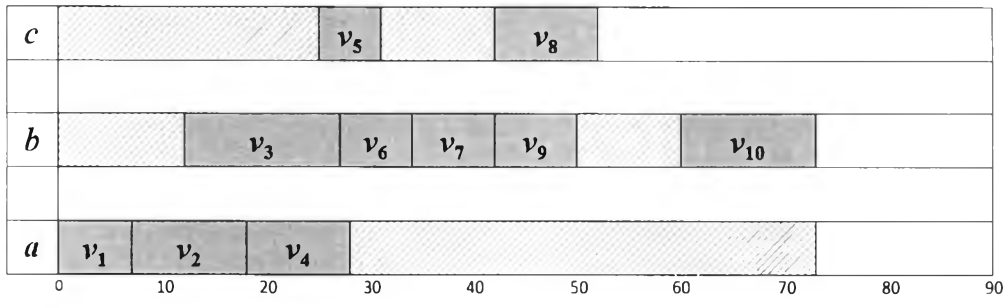


Figure 47 Results of task scheduling using the PETS Algorithm (fourth case).

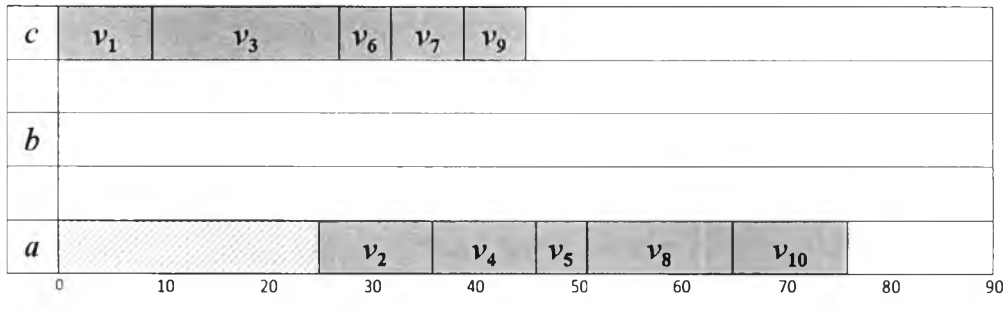


Figure 48 Results of task scheduling using the Lookhead Algorithm (fourth case).

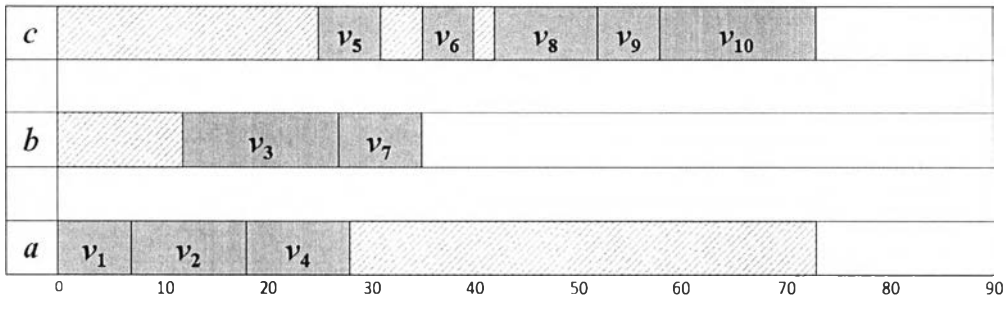


Figure 49 Results of task scheduling using the CEFT Algorithm (fourth case).

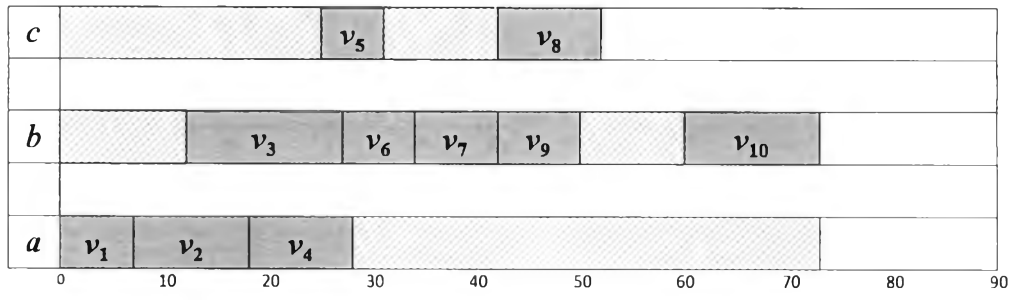


Figure 50 Results of task scheduling using the PEFT Algorithm (fourth case).

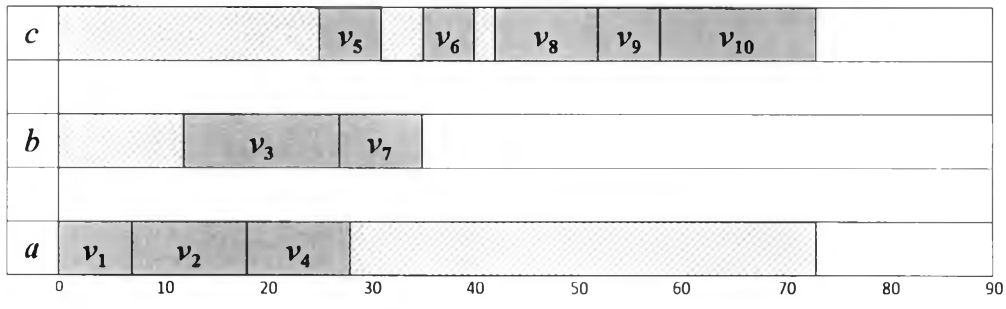


Figure 51 Results of task scheduling using the ESL Algorithm (fourth case).

The dependent task graph used in the fifth experimental case is shown in Figure 52 and the execution time of each task on each processing unit $t_a(v_i)$ is given in Table 48. Also shown in the task graph are the values for the data transfer delay $w_a(v_i)$. It can be deduced from the task graph that the values become $v=10$, $p=3$, and $l=4$. The energy consumptions as a result of each algorithm are presented in Table 49 to Table 54.

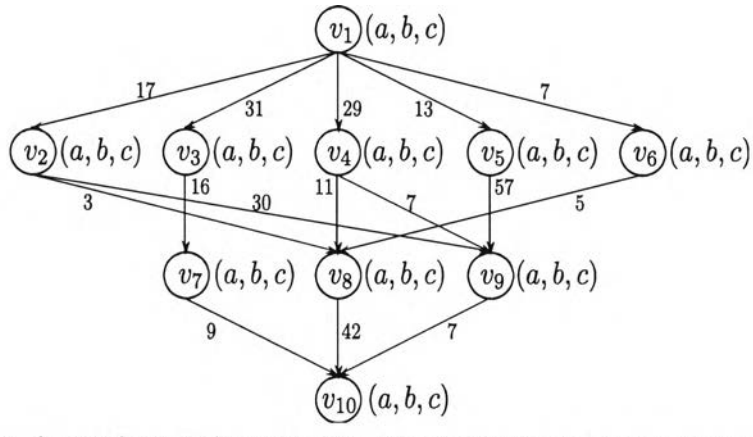


Figure 52 A dependent task graph with transmission and execution costs taken from [34] (fifth case).

Table 48 A list of execution time $t_a(v_i)$ by each processing unit for dependent task graph of Figure 52 (fifth case).

Task	Execution time		
v_i	$t_a(v_i)$	$t_b(v_i)$	$t_c(v_i)$
1	22	21	36
2	22	18	18
3	32	27	43
4	7	10	4
5	29	27	35
6	26	17	24
7	14	25	30
8	29	23	36
9	15	21	8
10	13	16	33

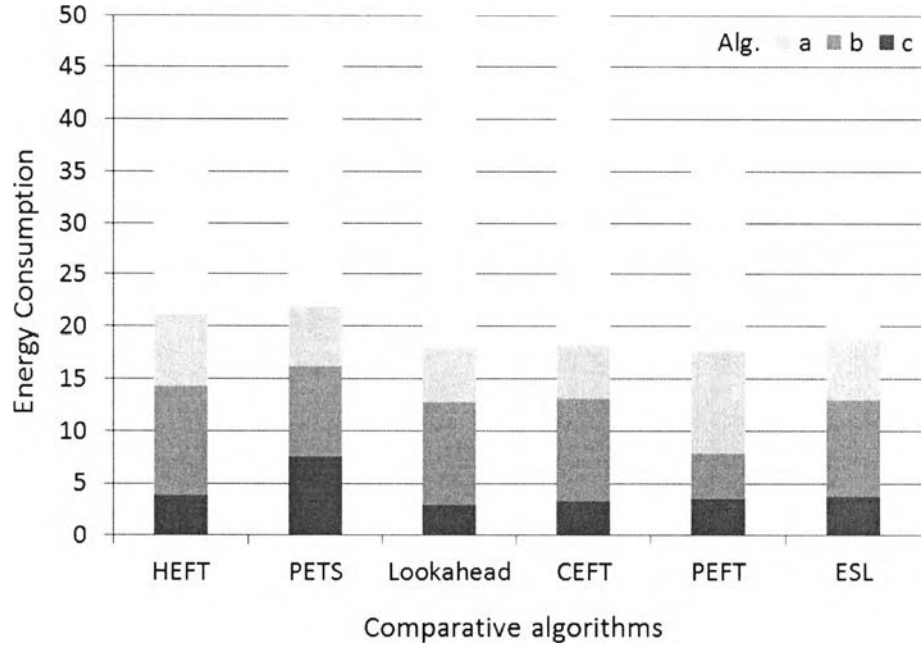


Figure 53 Result of energy consumption on each processing unit for each scheduling algorithm (fifth case).

Calculated energy consumptions for each algorithm in the fifth experimental case are shown in Figure 53. The scheduled tasks on processing unit a, b, and c using HEFT, PETS, Lookahead, CEFT, PEFT, and ESL algorithms are shown in Figure 54 to Figure 59. From Figure 54 to Figure 59, the system finish times are 133, 147, 127, 126, 122, and 124 units of time, respectively. Although ESL algorithm did not give the shortest finish time, the resulting time was only slightly longer than the shortest value given by PEFT algorithm. It can be seen that ESL algorithm consumes minimum energy within acceptable system finish time.

Table 49 Energy consumption in each processing unit as a result of the HEFT algorithm (fifth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	3.78	1.29	1.79	17.70	24.56
b	5.80	4.52	0.00	0.00	10.32
c	2.09	0.59	1.23	0.00	3.91
Total	11.67	6.40	3.02	17.70	38.79

Table 50 Energy consumption in each processing unit as a result of the PETS algorithm (fifth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	2.36	0.60	2.78	23.60	29.34
b	5.57	2.89	0.19	0.00	8.65
c	5.28	1.37	0.90	0.00	7.55
Total	13.21	4.86	3.87	23.60	45.54

Table 51 Energy consumption in each processing unit as a result of the Lookahead algorithm (fifth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	2.71	0.39	2.11	132.75	137.96
b	7.31	2.55	0.02	0.00	9.88
c	1.62	0.85	0.45	0.00	2.92
Total	11.64	3.79	2.58	132.75	150.76

Table 52 Energy consumption in each processing unit as a result of the CEFT algorithm (fifth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	2.71	0.39	2.08	177.00	182.18
b	7.31	2.55	0.00	0.00	9.86
c	1.62	0.85	0.80	0.00	3.27
Total	11.64	3.79	2.88	177.00	195.31

Table 53 Energy consumption in each processing unit as a result of the PEFT algorithm (fifth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	5.72	3.44	0.65	17.70	27.51
b	3.25	0.00	1.12	0.00	4.37
c	2.49	0.26	0.74	0.00	3.49
Total	11.46	3.70	2.51	17.70	35.37

Table 54 Energy consumption in each processing unit as a result of the ESL algorithm (fifth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	3.42	0.69	1.72	5.90	11.73
b	6.03	2.85	0.34	0.00	9.22
c	1.28	1.89	0.61	0.00	3.78
Total	10.73	5.43	2.67	5.90	24.73

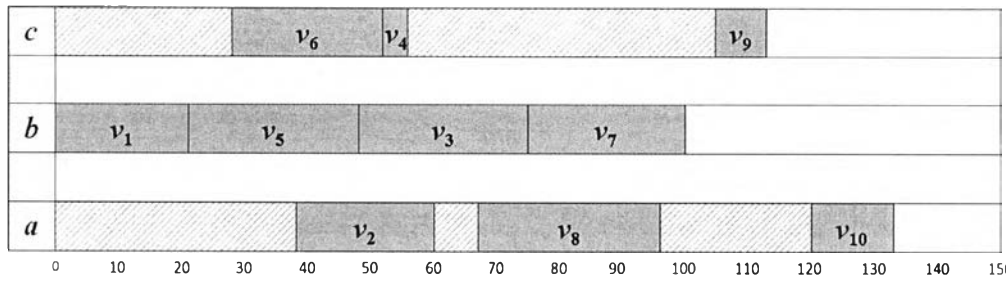


Figure 54 Results of task scheduling using the HEFT Algorithm (fifth case).

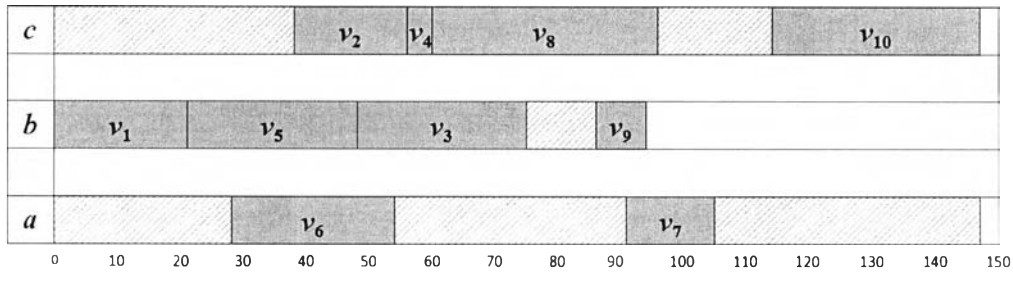


Figure 55 Results of task scheduling using the PETS Algorithm (fifth case).

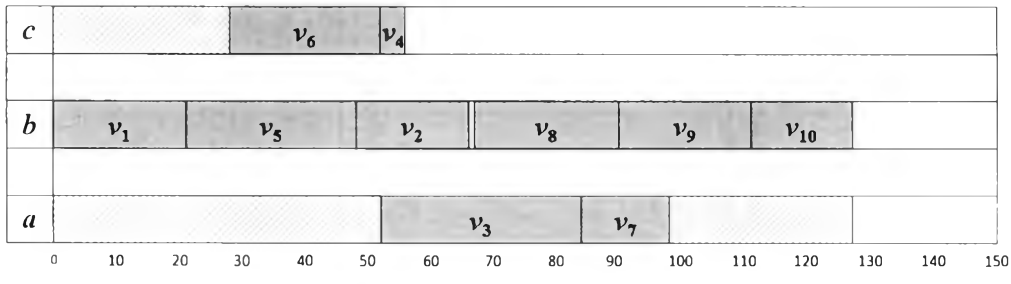


Figure 56 Results of task scheduling using the Lookahead Algorithm (fifth case).

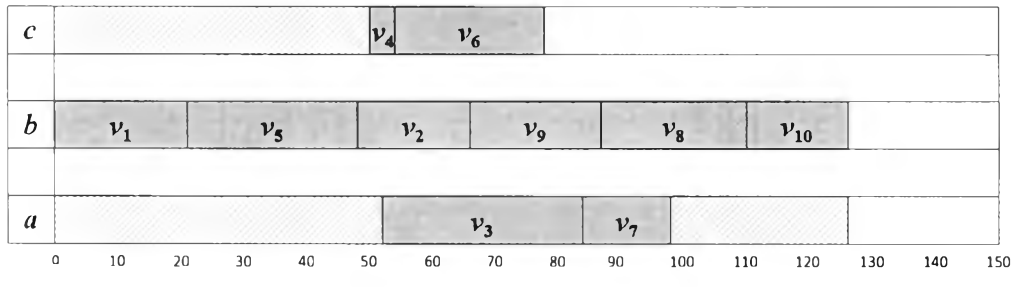


Figure 57 Results of task scheduling using the CEFT Algorithm (fifth case).

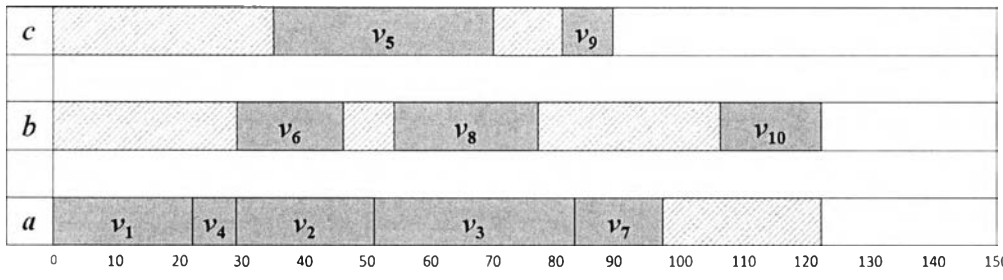


Figure 58 Results of task scheduling using the PEFT Algorithm (fifth case).

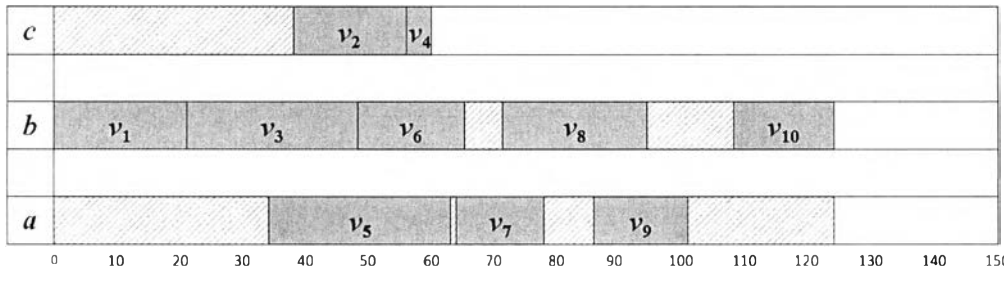


Figure 59 Results of task scheduling using the ESL Algorithm (fifth case).

The dependent task graph used in the sixth experimental case is shown in Figure 60 and the execution time of each task on each processing unit $t_a(v_i)$ is given in Table 55. It can be deduced from the task graph that the values become $v = 23$, $p = 4$, and $l = 6$. For this experimental case, additional constraints, which could come from security reason or other purposes, were imposed. The first constraint dictated each processing unit to have certain capability and security requirement such that some processing units could not execute some tasks. The second constraint defines that data transfer delay $w_a(v_i)$ of each task are calculated from the transmission rate between connecting processing units $r_{a,b}$ and the amount of transmission data $d_a(v_i)$ with the values for $r_{a,b}$ given in Table 20 and those for $d_a(v_i)$ given in Table 55. For example, if processing unit a wants to send the result after executing task v_1 to processing unit b, the amount of data being transferred from processing unit a to b, $d_a(v_1) = 25$, with the data transmission rate from a to b, $r_{a,b} = 1$. Thus, data transmission delay $w_a(v_1)$ is equal to $d_a(v_1)/r_{a,b} = 25/1 = 25$. In this way, the energy consumptions from each algorithm are calculated and the results are presented in Table 56 to Table 61.

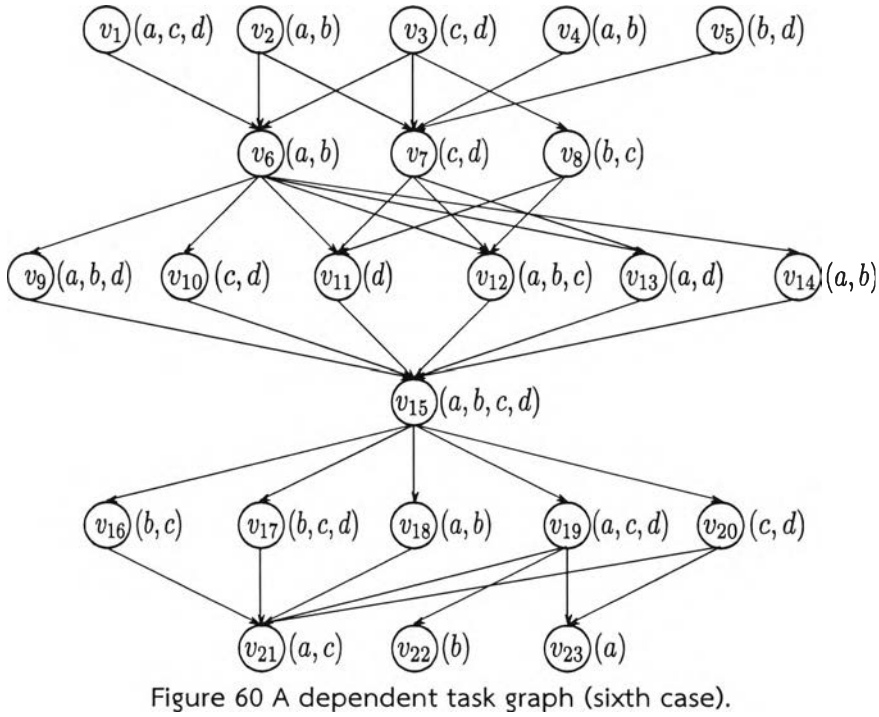


Figure 60 A dependent task graph (sixth case).

Table 55 Transmission data $d_a(v_i)$ of each task and execution time $t_a(v_i)$ required by each processing unit (sixth case).

Task	Amount of transmission data	Execution time			
v_i	$d_a(v_i)$	$t_a(v_i)$	$t_b(v_i)$	$t_c(v_i)$	$t_d(v_i)$
1	25	350	–	570	430
2	30	430	270	–	–
3	20	–	–	510	450
4	45	590	710	–	–
5	60	–	920	–	790
6	70	360	480	–	–
7	30	–	–	840	720
8	55	–	710	380	–
9	20	390	670	–	380
10	25	–	–	550	580
11	15	–	–	–	620
12	65	870	790	720	–
13	20	560	–	–	540
14	10	430	410	–	–
15	85	280	260	270	250
16	35	–	520	480	–
17	20	–	380	390	420
18	15	610	690	–	–
19	30	250	–	270	480
20	10	–	–	620	710
21	70	760	–	890	–
22	30	–	540	–	–
23	35	580	–	–	–

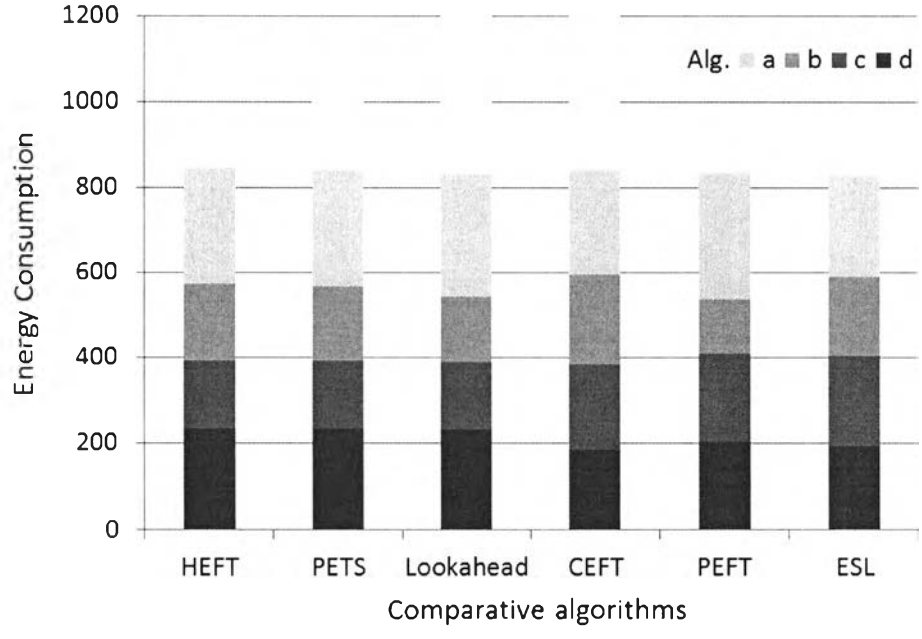


Figure 61 The result of energy consumption on each processing unit for each scheduling algorithm (sixth case).

Calculated energy consumptions for each algorithm in the sixth experimental case are shown in Figure 61. The scheduled tasks on processing unit a, b, and c using HEFT, PETS, Lookahead, CEFT, PEFT, and ESL algorithms are shown in Figure 62 to Figure 67. From Figure 62 to Figure 67, the system finish time are 5230, 5220, 5230, 4962, 5526, and 4420 units of time, respectively. The system finish times of most algorithms are considerably greater than that of the ESL. This could be because there was no provision in those algorithms to handle the applied constraints. It can be seen that ESL algorithm yielded both the lowest total energy consumption and the shortest system finish time.

Table 56 Energy consumption in each processing unit as a result of the HEFT algorithm (sixth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	224.79	11.83	36.92	124.84	398.38
b	139.78	5.89	34.24	0.00	179.91
c	129.34	6.29	22.69	0.00	158.32
d	220.02	14.62	0.00	0.00	234.64
Total	713.93	38.63	93.85	124.84	971.25

Table 57 Energy consumption in each processing unit as a result of the PETS algorithm (sixth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	224.79	11.83	36.66	167.34	440.62
b	139.78	5.89	28.53	0.00	174.20
c	129.34	6.29	22.69	0.00	158.32
d	220.02	14.62	0.00	0.00	234.64
Total	713.93	38.63	87.88	167.34	1007.78

Table 58 Energy consumption in each processing unit as a result of the Lookahead algorithm (sixth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	247.80	13.98	26.78	1914.27	2202.83
b	100.92	5.13	45.63	0.00	151.68
c	129.34	6.29	22.69	0.00	158.32
d	220.02	12.40	0.00	0.00	232.42
Total	698.08	37.80	95.10	1914.27	2745.25

Table 59 Energy consumption in each processing unit as a result of the CEFT algorithm (sixth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	184.08	12.47	47.89	2871.41	3115.85
b	182.12	6.27	20.60	0.00	208.99
c	162.98	16.65	20.35	0.00	199.98
d	176.13	3.15	6.08	0.00	185.36
Total	705.31	38.54	94.92	2871.41	3710.18

Table 60 Energy consumption in each processing unit as a result of the PEFT algorithm (sixth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	247.80	12.69	34.48	124.84	419.81
b	70.76	3.80	52.87	0.00	127.43
c	184.44	13.32	8.80	0.00	206.56
d	185.82	3.89	14.69	0.00	204.40
Total	688.82	33.70	110.84	124.84	958.20

Table 61 Energy consumption in each processing unit as a result of the ESL algorithm (sixth case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	193.52	12.69	33.28	31.21	270.70
b	164.14	6.46	14.79	0.00	185.39
c	178.64	11.29	21.44	0.00	211.37
d	176.13	12.40	5.31	0.00	193.84
Total	712.43	42.84	74.82	31.21	861.30

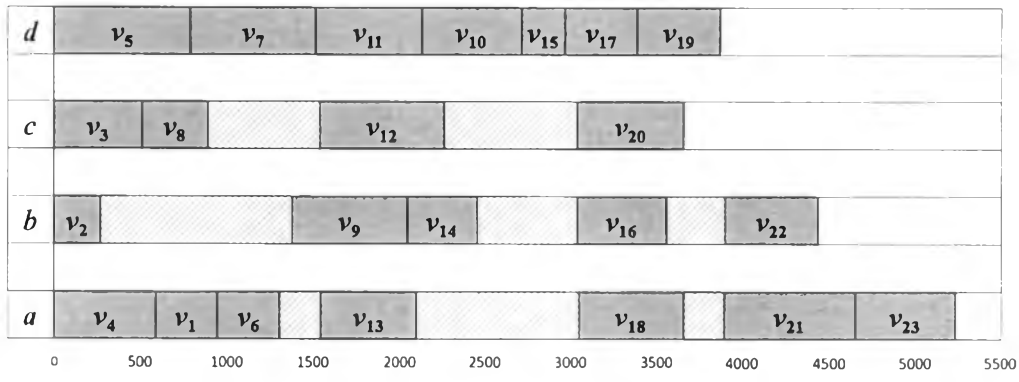


Figure 62 Results of task scheduling using the HEFT Algorithm (sixth case).

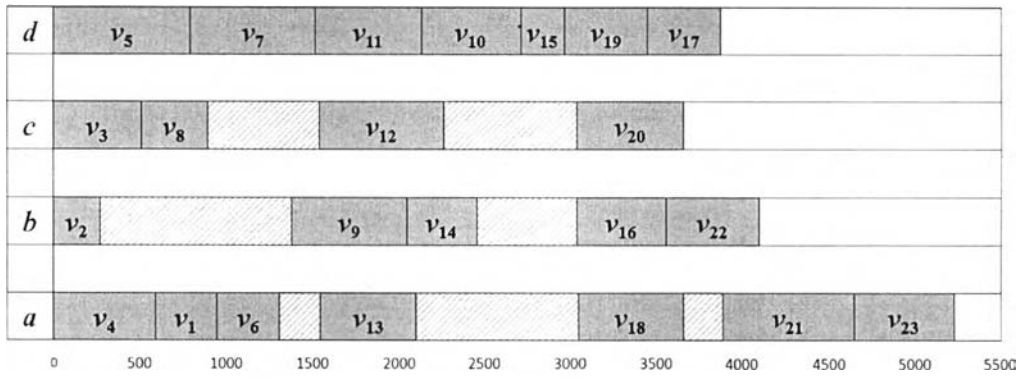


Figure 63 Results of task scheduling using the PETS Algorithm (sixth case).

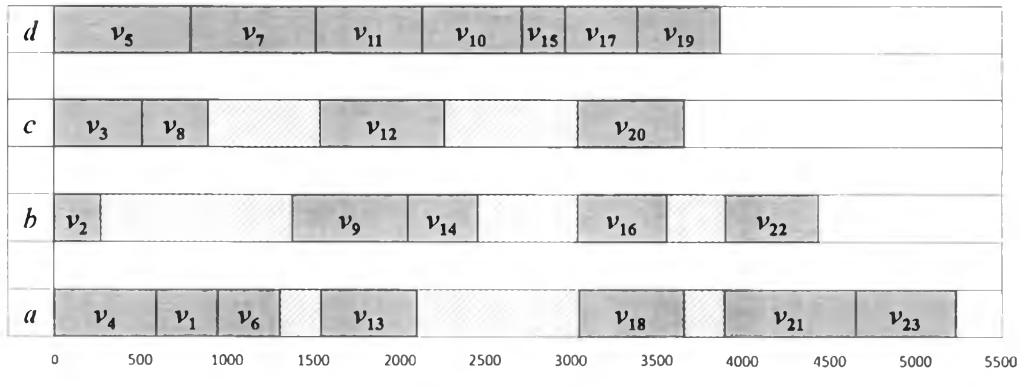


Figure 64 Results of task scheduling using the Lookahead Algorithm (sixth case).

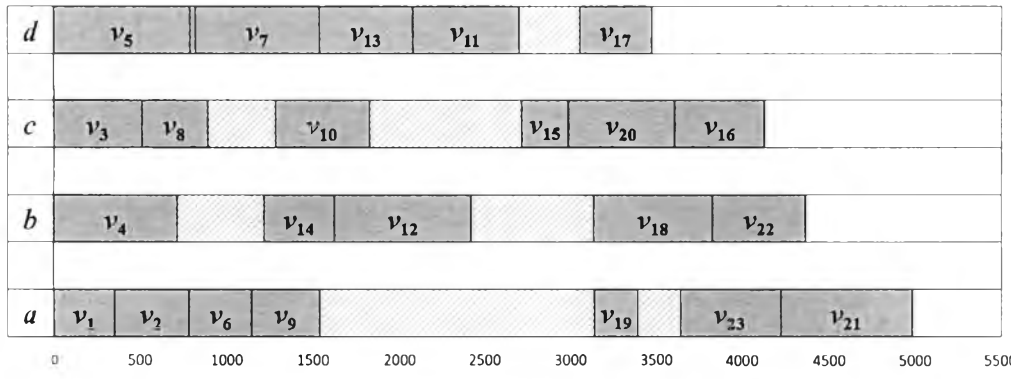


Figure 65 Results of task scheduling using the CEFT Algorithm (sixth case).

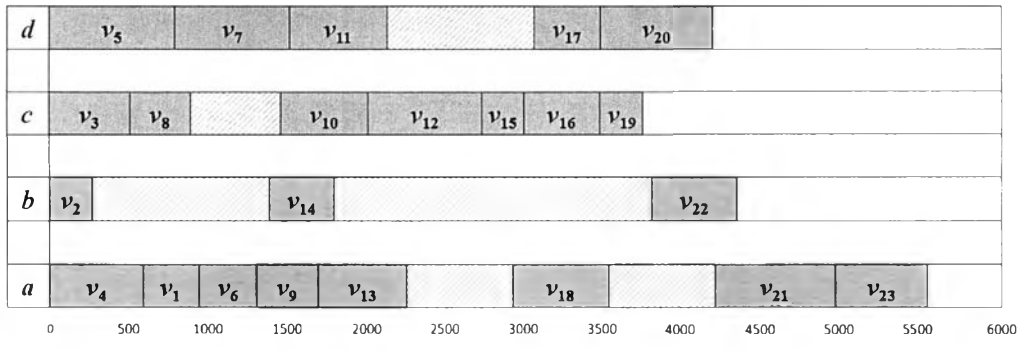


Figure 66 Results of task scheduling using the PEFT Algorithm (sixth case).

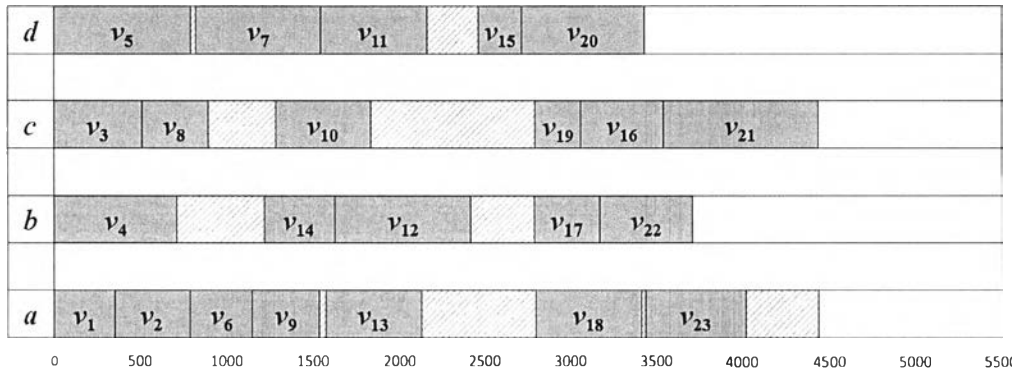


Figure 67 Results of task scheduling using the ESL Algorithm (sixth case).

Table 62 A summary of system finish time of all algorithms in the first six experimental cases. The values of the shortest system finish time in each experimental case are written in bold face and underlined.

Graph	HEFT	PETS	Lookahead	CEFT	PEFT	ESL
case 1	80	77	82	81	85	**73**
case 2	29	**27**	31	31	**27**	28
case 3	260	257	**184**	**184**	**184**	**184**
case 4	85	**73**	76	**73**	**73**	**73**
case 5	133	147	127	126	**122**	124
case 6	5230	5220	5230	4962	5526	**4420**

The system finish times determined from all comparative scheduling algorithms in the first six experimental cases are summarized in Table 62. It can be seen from the table that no algorithm can achieve the shortest system finish time in all cases. Lookahead algorithm yielded the shortest system finish time in one case, two cases for PETS and CEFT algorithms, and four cases for PEFT and ESL algorithms. Although ESL algorithm could not give the shortest system finish time in case 2 and 5, the achieved times are only slightly higher than the shortest system finished time yielded by PETS and PEFT algorithms.

Table 63 A summary of system energy consumption of all algorithms in the first six experimental cases. The values of the lowest energy consumption in each experimental case are written in bold face and underlined.

Graph	HEFT	PETS	Lookahead	CEFT	PEFT	ESL
case 1	31.72	37.20	144.71	189.22	30.92	**17.57**
case 2	25.35	32.46	144.93	239.47	24.85	**10.39**
case 3	43.69	46.31	116.01	148.27	33.58	**23.58**
case 4	26.40	32.68	113.69	186.25	26.78	**15.15**
case 5	38.79	45.54	150.76	195.31	35.37	**24.73**
case 6	971.25	1007.78	2745.25	3710.18	958.20	**861.30**

The total energy consumptions determined from all comparative scheduling algorithms in the first six experimental cases are summarized in Table 63. It can be seen from the table that ESL algorithm could achieve the lowest total energy consumption in all cases.

The dependent task graph used in the seventh experimental case is the same as that in the sixth experimental case, except the additional constraint on the battery supply. In this experimental case, performance of the energy reserve option introduced by ESL algorithm was investigated. From the sixth experimental case, the energy consumptions of the main processing unit are 398.38, 440.62, 2202.83, 3115.85, 419.81, and 270.70 for HEFT, PETS, Lookahead, CEFT, PEFT, and ESL algorithm, respectively. By applying the energy reserve option, the energy consumption of the main processing unit determined by ESL Algorithm can be further reduced to 246.75, while other comparative algorithms cannot perform the assignment when the energy supply given to the main processing unit is lower than the values quoted above.

Equations (3) and (4) are employed in the battery reserve option to calculate the energy required by each processing unit to execute the task designated to that processing unit. Consider a processing unit d in the task graph as shown in Figure 60. If task v_{11} is designated to be executed only on a processing unit d, the energy reserved by the processing unit d to execute task v_{11} can be determined from

$$\rho_d = \left(\alpha_d t_d(v_{11}) + \lambda_d d_d(v_{11})\right) = \left(0.057 \times 620 + 0.037 \times 15\right) = 35.90 .$$

As v_{11} is in the third level of the dependent task graph, there will be a period of time when processing unit d is in wait state. Thus, idle energy will also need to be reserved. The idle energy used by processing unit d through the first, second, and third level is

$$\sigma_{1,d} = \frac{\beta_d}{4-1}\left(\sum_{v_i \in [1,3]} average\left(t_a(v_i)\right) - t_d\left(v_{11}\right)\right) = \frac{0.016}{3}\left(7958.33 - 620\right) = 39.14$$

Therefore, the total energy reserved by processing unit d in the first level is $35.90 + 39.14 = 75.04$. In the case that the energy supply is greater than the energy required to be reserved by the processing unit, the excess energy can be used for task execution process. However, in case that the energy supply is less than the required amount to be reserved, the system will halt.

Table 64 Energy consumption in each processing unit as a result of ESL algorithm when the battery supply is limited to 275 unit (seventh case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	193.52	12.69	33.28	31.21	270.70
b	164.14	6.46	14.79	0.00	185.39
c	178.64	11.29	21.44	0.00	211.37
d	176.13	12.40	5.31	0.00	193.84
Total	712.43	42.84	74.82	31.21	861.30

Table 65 Energy consumption in each processing unit as a result of ESL algorithm when the battery supply is limited to 270 unit (seventh case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	172.28	14.62	45.76	31.21	263.87
b	182.12	6.27	14.79	0.00	203.18
c	185.60	9.07	21.44	0.00	216.11
d	176.13	12.40	5.31	0.00	193.84
Total	716.13	42.36	87.30	31.21	877.00

Table 66 Energy consumption in each processing unit as a result of ESL algorithm when the battery supply is limited to 265 unit (seventh case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	172.28	14.62	45.76	31.21	263.87
b	182.12	6.27	14.79	0.00	203.18
c	185.60	9.07	21.44	0.00	216.11
d	176.13	12.40	5.31	0.00	193.84
Total	716.13	42.36	87.30	31.21	877.00

Table 67 Energy consumption in each processing unit as a result of ESL algorithm when the battery supply is limited to 260 unit (seventh case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	157.53	12.04	57.67	31.21	258.45
b	182.12	6.27	16.83	0.00	205.22
c	178.64	11.29	26.69	0.00	216.62
d	200.07	9.99	5.31	0.00	215.37
Total	718.36	39.59	106.50	31.21	895.66

Table 68 Energy consumption in each processing unit as a result of ESL algorithm when the battery supply is limited to 255 unit (seventh case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	134.52	11.18	69.84	31.21	246.75
b	182.12	6.27	18.16	0.00	206.55
c	178.64	11.29	27.94	0.00	217.87
d	221.73	9.99	0.48	0.00	232.20
Total	717.01	38.73	116.42	31.21	903.37

Table 69 Energy consumption in each processing unit as a result of ESL algorithm when the battery supply is limited to 250 unit (seventh case).

Processing Unit	Execution Energy	Transmission Energy	Idle Energy	Scheduling Energy	Total Energy
a	134.52	11.18	69.84	31.21	246.75
b	182.12	6.27	18.16	0.00	206.55
c	178.64	11.29	27.94	0.00	217.87
d	221.73	9.99	0.48	0.00	232.20
Total	717.01	38.73	116.42	31.21	903.37

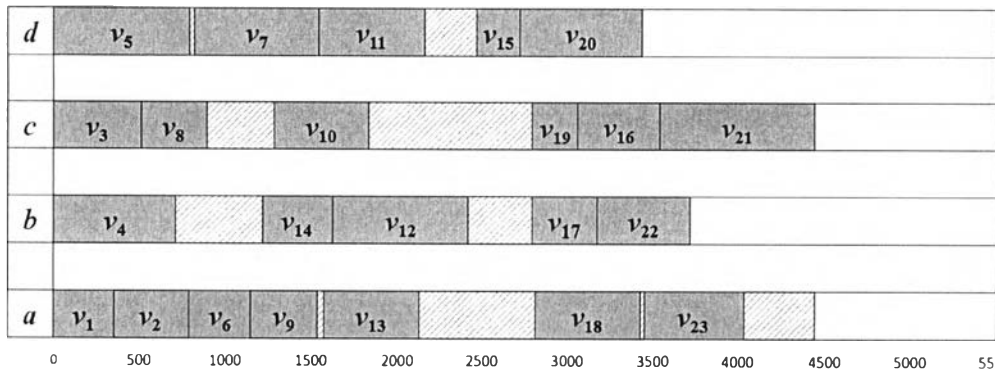


Figure 68 Results of task scheduling using ESL algorithm when the battery supply is limited to 275 unit (seventh case).

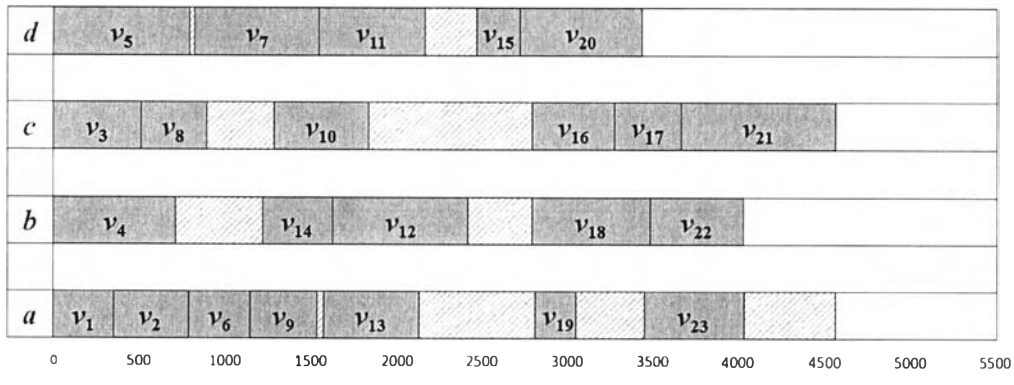


Figure 69 Results of task scheduling using ESL algorithm when the battery supply is limited to 270 unit (seventh case).

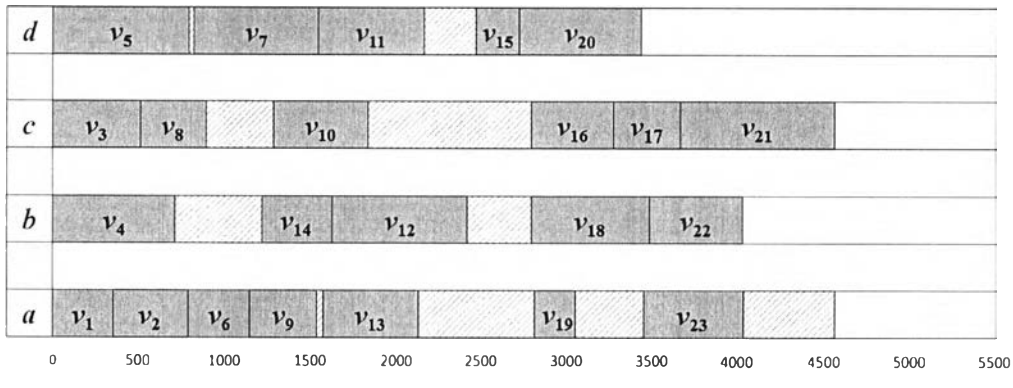


Figure 70 Results of task scheduling using ESL algorithm when the battery supply is limited to 265 unit (seventh case).

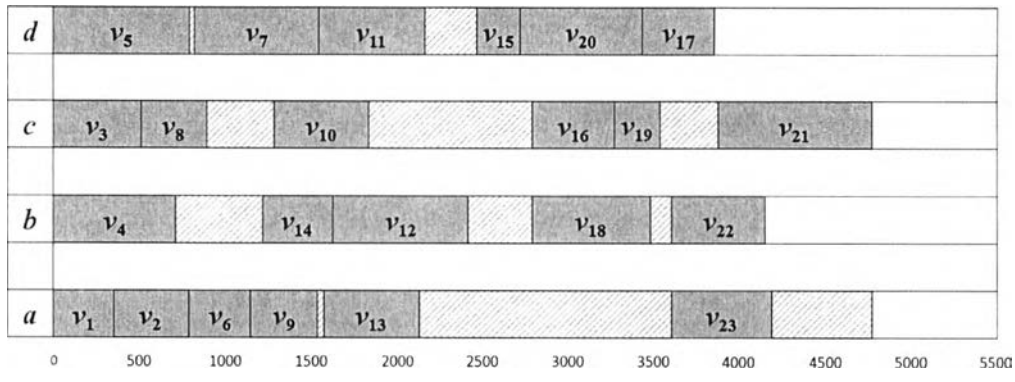


Figure 71 Results of task scheduling using ESL algorithm when the battery supply is limited to 260 unit (seventh case).

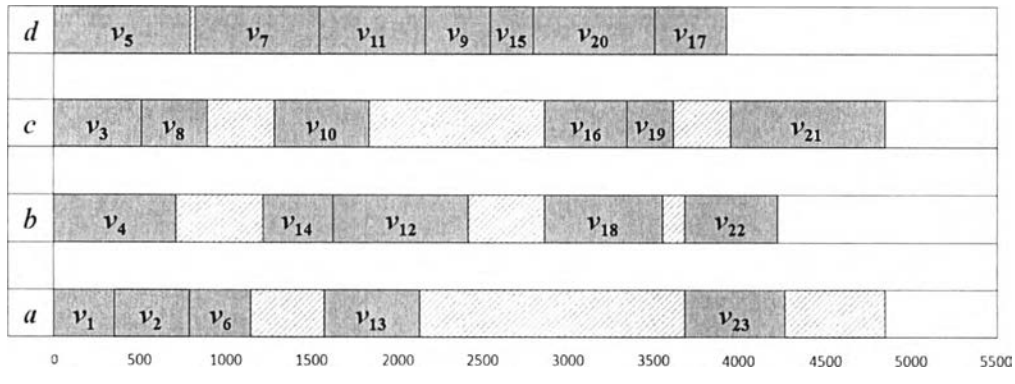


Figure 72 Results of task scheduling using ESL algorithm when the battery supply is limited to 255 unit (seventh case).

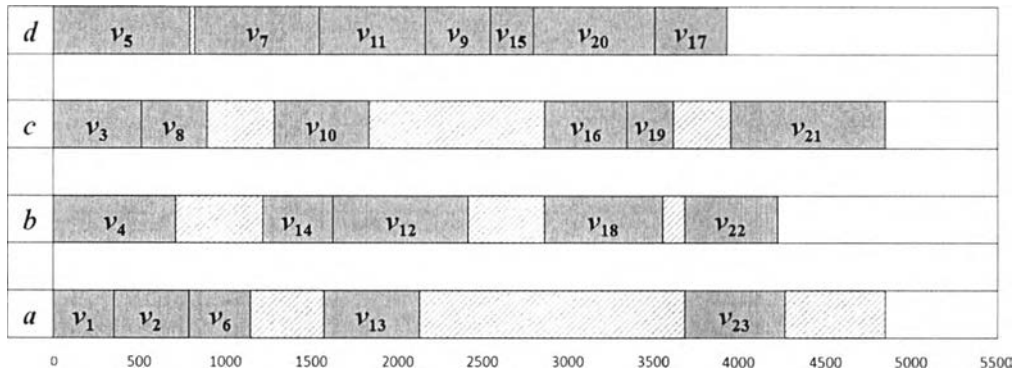


Figure 73 Results of task scheduling using ESL algorithm when the battery supply is limited to 250 unit (seventh case).

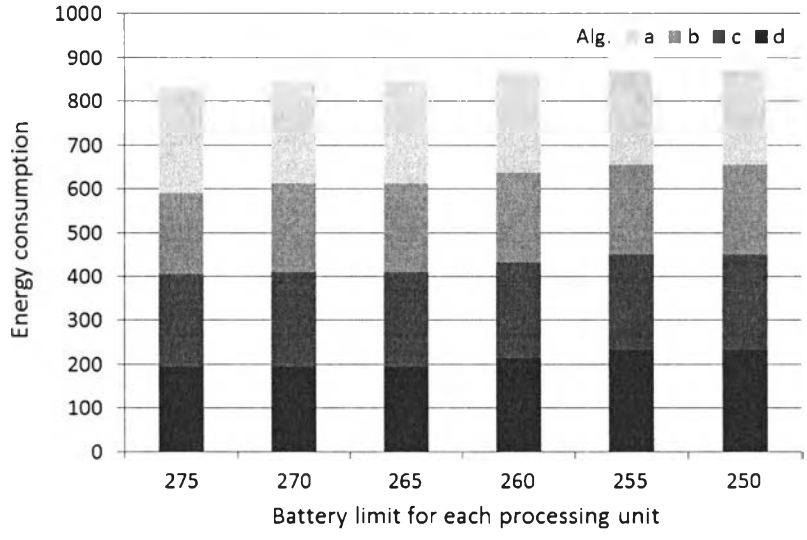


Figure 74 Energy consumption of each processing unit required by ESL algorithm as a function of the battery supply.

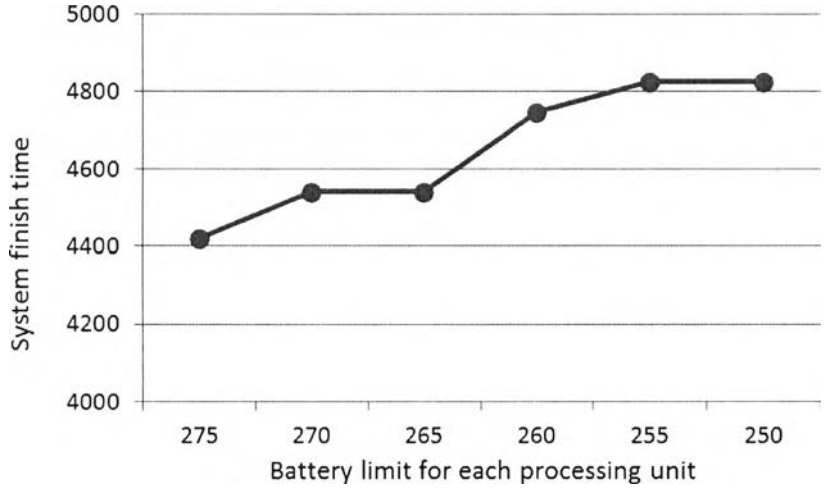


Figure 75 System finish time of ESL algorithm as a function of the battery supply.

As shown in Figure 74 and Figure 75 when the battery supplies on all processing units are greater than or equal to 275, the system finishing time remains at 4420. However, when the supplied battery energy is 250, the system finish time increases to 4826. When the supplied battery is reduced below 250, the algorithm is not able to perform the assignment and the system halts. Results of the scheduled tasks on processing unit a, b, and c using ESL algorithms under limited battery supply are shown in Figure 68 to Figure 73.