Evaluating energy-aware cloud task scheduling techniques: A comprehensive dialectical approach

Anas Hattay, Fred Ngole Mboula, Eric Gascard, Zakaria Yahouni

Evaluating Energy-Aware Cloud Task Scheduling Techniques: A Comprehensive Dialectical Approach

Anas Hattay, Fred Ngolè Mboula
Universite Paris-Saclay
CEA, List
F-91120, Palaiseau, France
anas.hattay@cea.fr, fred-maurice.ngole-mboula@cea.fr

Eric Gascard , Zakaria Yahouni
Universite Grenoble Alpes
CNRS, Grenoble INP G-SCOP
38000 Grenoble, France
eric.gascard@grenoble-inp.fr, zakaria.yahouni@grenoble-inp.fr

***Abstract*—In this paper, we address the challenge of optimizing task scheduling in cloud environments by systematically evaluating and comparing various methodologies, namely heuristics, meta-heuristics, and deep reinforcement learning (DeepRL). The context of our study is driven by the need for efficient resource allocation and management in dynamic and large-scale cloud systems to minimize servers' power consumption. We employ the framework of Hegelian dialectics to dissect the strengths, weaknesses, and practical limitations of each approach. Our experiments, conducted using Alibaba Trace Data, provide empirical evidence that underscores the effectiveness of different methods in specific cloud scenarios. The key contributions of this study include a comprehensive analysis of each methodology's performance, findings regarding their scalability, and new perspectives for optimizing both energy consumption and maximum processing time while scheduling tasks in a cloud system.**

***Index Terms*—Task Scheduling, Cloud Computing, Heuristics, Meta-Heuristics, Deep Reinforcement Learning**

I. Introduction

The fourth industrial revolution, characterized by the rapid advancement and integration of digital technologies, has transformed industries and societies worldwide. In this new era, the proliferation of digital activities has led to an exponential increase in data generation and processing needs, placing high demands on data centers. As a result, the energy consumption of these facilities has become a critical concern. The environmental impact associated with the operation of data centers has surged, contributing significantly to global greenhouse gas emissions, by representing between 2.5% and 3.7% of all worldwide greenhouse gas emissions. This surpasses the emissions from commercial flights (approximately 2.4%) and other essential activities that power our global economy [1]. Addressing the critical issue of enhancing the energy efficiency of data centers and aligning with sustainability goals demands innovative approaches. To do so, researchers have actively intervened in cloud resource allocation using a variety of algorithms to tackle these challenges.

Our study critically examines these methodologies by selecting representative algorithms from each category: heuristic, meta-heuristic, and deep reinforcement learning. We analyze their strengths, weaknesses, and performance metrics. Specifically, we evaluate their effectiveness in optimizing energy consumption, minimizing task processing times, and consider their execution time.

The structure of the paper is organized as follows: Section II defines the problem and presents its mathematical formulation. Section III reviews existing literature on heuristic, meta-heuristic, and deep reinforcement learning approaches for task-resource allocation in cloud environments. Section IV outlines our dialectical methodology for investigating these approaches, including algorithmic modeling. Subsequently, in section V we provide implementation details and experimental results across various cloud environment scenarios. Finally, we conclude with a summary of our findings and suggest avenues for future research.

II. Preliminaries

A. Problem statement

The task scheduling process in cloud systems is illustrated in *Fig. 1* and can be summarized in three steps:

1) Resource discovery, where a data broker acting as an intermediary between users and the cloud infrastructure identifies available resources and gathers information on their capacity, processing cost, and current load;
2) Resource selection, where the scheduler chooses a suitable resource based on task requirements, resource characteristics and cloud objectives; and
3) Task submission, where the task is assigned to the selected resource, which may be a Virtual Machine

(VM) or a physical machine, depending on the cloud architecture used.

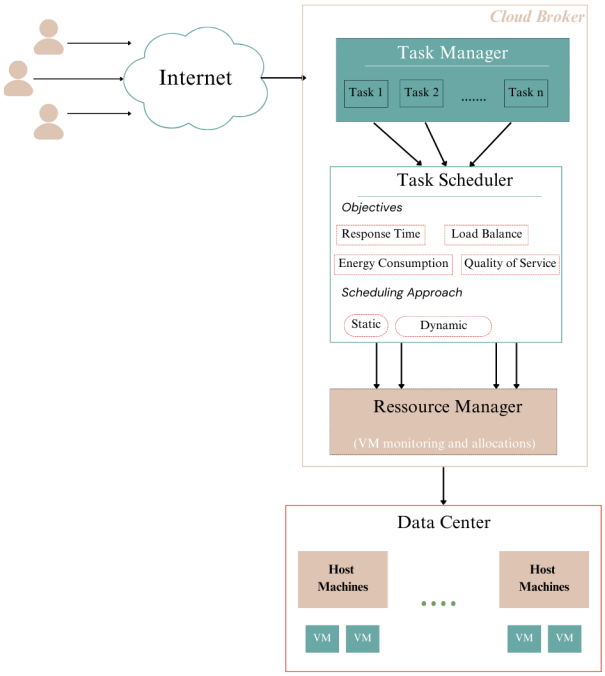

Fig. 1: Task scheduling framework in cloud.

In a concrete manner, when a user uses a cloud service and submits a task, the cloud broker, which includes a scheduling algorithm, will allocate cloud resources to tasks based on the task requirements (CPU, Memory...) and resource availability, following the logic and objective of the algorithm.

What makes this problem a complicated one, are the constraints and expectations of both the service provider and the user. Service providers want to minimize operational costs while maximizing resource utilization and hopefully minimize energy consumption. On the other hand, users expect reliable, timely, and cost-effective services that meet their specific Quality of Service (QoS) requirements. This situation clearly presents a multi-objective optimization problem under multiple constraints, with a very large combinatorial solution space, making the problem NP-hard. Among the crucial issues in cloud computing, our focus will be on energy consumption and makespan, which refers to the maximum processing time of scheduled tasks. Therefore, we must be able to achieve the near-optimal schedule in time and energy usage by balancing the desired goals.

B. Problem Formulation

Formulating the problem, we assume that each task can be allocated to any machine and must be scheduled on only one machine. Additionally, we assume that all resources are available at the beginning of the scheduling process. Moreover, all tasks are considered independent.

Let $T = \{t_1, t_2, \ldots, t_n\}$ represent a list of tasks that must be executed and $M = \{m_1, m_2, \ldots, m_k\}$ represents a list of machines , where each machine is capable of processing and executing the task assigned to it. The machines are located in cloud environments in different areas and have different processing capabilities and different power consumption rates. Thus, the process of task scheduling can be represented by $F : T \rightarrow M$.

The main objective is to build a task scheduler algorithm that achieves low energy consumption while also minimizing processing time. To address this multi-objective problem, we have chosen to use weighting between the following two objectives.

1) Energy Consumption (EC) The energy consumption EC is defined as:

$$\text{EC} = \sum_{i \in \text{machines}} \text{energy}_i$$

where energy_i is the energy consumed by machine i.

2) Makespan (MS) The makespan MS is defined as:

$$\text{MS} = \max_{i \in \text{machines}} \text{processing time}_i$$

where processing time_i is the processing time of tasks on machine i.
The processing time of tasks on a machine is defined as the sum of the processing times of different tasks scheduled on that machine. Each task's processing time $\text{processing time}_i^j$ is calculated as :

$$\text{processing time}_i^j = \sum_j (\text{end time}_i^j - \text{start time}_i^j)$$

where j denotes the tasks assigned to machine i.

The objective function combining energy consumption and makespan is:

$$\text{ObjectiveFunction} = \alpha \times \text{EC} + \beta \times \text{MS} \qquad (1)$$

where α and β are weights (between 0 and 1) that balance the importance of minimizing energy consumption and makespan, respectively.

III. Related Work

Since the matching problem to find the choice of the best pairs of tasks and resources is an NP-complete problem [2], many approximate methods have been investigated in the literature to address the task scheduling challenge. In the section below, we will explore a few of these methods.

A. Heuristics

Heuristics are methods employed in complex problems where exact methods cannot be used. They encompass simple instructions or rules to guide algorithm decisions more quickly without the need to exhaustively search every possible solution. To solve resource allocation problems in the cloud, various heuristics have been used [3, 4].

Syed et al. used the Min-Min heuristic [5] to select the task with the shortest execution time and assign it to the available resource with the shortest processing time. Moreover, authors in [6] proposed a Max-Min heuristic-based algorithm for task scheduling in the heterogeneous grid computing environment with the objective of minimizing makespan. Other works have used the Suffrage heuristic, which assigns a host to the task that would "suffer" the most if not assigned to that host [7]. For each task, its suffrage value is defined as the difference between its best Mean processing Time (MPT) and its second-best MPT. Tasks with high suffrage values take precedence. Among the most often cited, we also find the Heterogeneous Earliest Finish Time (HEFT) heuristic [8]. Additionally, other heuristics, such as Shortest Job First (SJF) [9], First Come First Serve (FCFS) [10], Tetris [11] have been widely used in the literature. Fewer works have concentrated on energy efficiency in the cloud using heuristics. For example, authors in [12] adopted the Modified Best Fit Decreasing (MBFD) heuristic for the Virtual Machines (VM) placement problem. The logic is to allocate each VM to a host that provides the least increase in power consumption due to this allocation.

Even though heuristics are usually adapted for one single objective, [2] designed a heuristic to minimize both energy consumption and task utility. While these methods can be extended to multi-objective scenarios, doing so often introduces additional complexity that these heuristics are designed to avoid.

B. Meta-heuristics

Meta-heuristic algorithms have gained significant popularity in task scheduling and multi-objective dynamic optimization due to their ability to efficiently explore search spaces and find solutions that are sub-optimal or near-optimal within reasonable time limits. In the context of cloud scheduling, researchers commonly utilize particle swarm optimization, a bio-inspired algorithm, to schedule tasks on servers while optimizing various cloud objectives [13].

Drawing inspiration from the intelligent behavior of honey bees, Jain et al. [14] employed an artificial bee colony (ABC)-based energy-aware resource allocation algorithm to assign jobs to resources in cloud environments. Genetic Algorithms (GA) [15], based on Darwin's natural selection theory, also play a significant role among promising meta-heuristics. To tackle the multi-objective nature of resource allocation problems in the cloud, the literature often employs NSGA-II, a modified genetic algorithm known for its efficient non-dominated sorting algorithm, which has become a standard approach [16]. Hybrid algorithms combining the advantages of multiple meta-heuristics have also been used to schedule tasks in cloud and fog environments [17].

C. Machine learning based methods

In light of these challenges, there has been a growing interest in applying machine learning techniques, particularly deep reinforcement learning, to task scheduling problems in cloud computing.

While some studies employ deep learning techniques [18], most research focuses on reinforcement learning to optimize various cloud objectives. Li and Hu introduced DeepJS [11], a policy gradient deep reinforcement learning method designed to schedule jobs while minimizing the makespan.

Focusing on energy efficiency, Liu et al. [19] proposed a novel deep reinforcement learning hierarchical framework to address overall resource allocation and power management in cloud computing systems. Recently, Sudhee et al. [20] employed a Deep Q-learning network model to optimize resource allocation by considering task priorities, aiming to minimize makespan, Service-level agreement (SLA) violations and energy consumption.

D. Comparative Analysis of Task Scheduling Methods

As we mentioned, the problem of cloud task scheduling has seen extensive research, with a significant focus on heuristics and meta-heuristics. Traditional heuristic methods are known for their simplicity and efficiency in certain scenarios. Meta-heuristics have been employed to address more complex scheduling problems, offering better optimization at the cost of increased computational effort. With the advent of deep learning, Deep Reinforcement Learning has emerged as a promising approach for task scheduling.

Several studies have compared these different methods, analyzing their performance and applicability in various contexts [21, 22]. However, to the best of our knowledge, our experiments are unique in comparing algorithms from three different categories of methods using data from the Alibaba Trace Data. This approach contrasts with many studies that overlook certain methods or rely solely on synthetic data, which may not accurately reflect real-world conditions.

Furthermore, our research distinguishes itself through a structured methodology for investigating, evaluating, and comparing task scheduling techniques.

IV. Methodology

A. Dialectical Framework

To investigate the most effective task scheduling methods in cloud computing, we employ a dialectical approach

inspired by Hegelian philosophy [23]. This philosophy, as outlined in Hegel's works such as *The Phenomenology of Spirit* and *Science of Logic*, posits that the development of ideas follows a dialectical process, where each idea (thesis) inevitably generates its opposite (antithesis), and the conflict between these ideas is resolved by a higher-level synthesis. By extending this concept into scientific research methodology, we begin with a thesis, then present its antithesis, highlighting the various limitations of the initial thesis. Through this tension, we synthesize the two, proposing a more refined method. This synthesis may be either a combination of the thesis and antithesis or an entirely new idea, enriched and fortified by the dialectical process. As Hegel suggests, this is a temporal and iterative process, continually evolving until we approach the "truth". Here, we define truth as a new method yielding superior results.

In a pedagogical methodology, and in order to study and enlighten the various strengths and weaknesses of each type of method used in the literature for resource allocation in the cloud, we have chosen to follow the Hegelian dialectical approach. We proceed as follows:

1) Thesis: We begin with a particular method for task scheduling, assuming initially that it is the most effective solution. For instance, we start with a heuristic-based scheduling method widely cited in the literature for its efficiency.
2) Antithesis: Next, we study this method by identifying and highlighting its limitations and scenarios where it falls short. This is done by testing the method under various conditions and workloads to observe its performance issues.
3) Synthesis: Based on the insights gained from the antithesis, we propose a new method from literature. This new method is a synthesis that aims to overcome the identified limitations.
4) Iterative Process: The synthesis itself becomes the new thesis, and the process is repeated. We put this new method to the test, identifying new limitations and proposing further refinements. This iterative approach continues, with each cycle bringing us closer to a more robust and effective task scheduling method.

B. Selected scheduling methods

To conduct a rigorous analysis, we selected a representative algorithm from each category of methods used in cloud task scheduling literature. In this section, we discuss the choice of the selected methods and the modeling employed for implementation.

1) Heuristics: We select three different heuristics with varying levels of sophistication. The simplest among them is Tetris implemented in [11]. Tetris translates task requirements and available machine resources into Euclidean space, then chooses the $\langle \text{task}, \text{machine} \rangle$ pair with the maximum dot product. Only tasks with requirements that can be met are considered.

To prioritize our primary objective of minimizing power consumption, we adopt the logic of the Modified Best Fit Decreasing (MBFD) heuristic proposed in [12]. This heuristic prioritizes tasks based on their CPU consumption and assigns them to machines that have the lowest energy consumption.

Adding another layer of complexity, we design a heuristic to address our multi-objective problem using a weighted strategy. Algorithm 1 outlines the approach.

2) Meta-heuristics: As a prominent meta-heuristic method, we opt for the Genetic Algorithm (GA), specifically the NSGA-II version, due to its robust exploration capability of the solution space, which is well-documented in the literature [24].

- *Encoding:* In the genetic algorithm, an individual represents a potential solution, i.e., an assignment of tasks to machines. The chromosome of an individual is a matrix of one-hot encoded vectors with shape (n, k), where n is the number of tasks and k is the number of machines. Each row corresponds to a task, and the column with a '1' indicates the machine to which the task is assigned. Table I provides an example of a solution represented as a one-hot encoded matrix.

TABLE I: Example of One Chromosome (GA Solution) with 3 Tasks and 4 Machines

Task	M1	M2	M3	M4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0

- *Fitness Function:* The fitness function assesses the quality of each individual solution based on two main criteria: power consumption and maximum processing time. We follow the approach outlined by [14] to estimate both values, as illustrated in Algorithm 2.
- *Crossover and Mutation:* Crossover combines the chromosomes of two parents to create offspring. For each gene (task assignment), a random probability determines whether the gene is inherited from the first parent, the second parent, or randomly reinitialized. Mutation introduces diversity by randomly altering some genes in the offspring.
- *Non-dominated Sorting:* NSGA-II employs non-dominated sorting to classify individuals into different fronts based on their Pareto dominance relationship. Pareto dominance ensures that one solution is not worse than another across multiple objectives without being worse in any. Specifically: a solution A dominates

Algorithm 1 Weighted Multi-Objective Heuristic (WMOH)

Input: Cluster of machines C, Current time t
Output: Selected machine m and task T allocation
repeat
  Initialization:
  $M_{active} \leftarrow$ List of active machines in cluster C
  $T_{pending} \leftarrow$ List of pending tasks in cluster C
  if $T_{pending}$ is empty **then**
    Wait for one timestamp
    Continue to the next iteration
  end if
  $m_{selected} \leftarrow$ None
  $minScore \leftarrow$ Large number
  $found \leftarrow$ False
  for each task T in $T_{pending}$ **do**
    for each machine m in M_{active} **do**
      if m can accommodate T **then**
        Calculate Expected processing Time of task T on machine m
        Calculate power consumption of machine m
        Calculate weighted fitness score: $S = \alpha\times$ power consumption + $\beta\times$ Expected processing Time
        if $S < minScore$ **then**
          $m_{selected} \leftarrow m$
          $minScore \leftarrow S$
          $found \leftarrow$ True
        end if
      end if
    end for
    if $found$ **then**
      return m, T
    end if
  end for
  if $m_{selected}$ is None **then**
    Wait for one timestamp
    Continue to the next iteration
  end if
until All tasks are finished and the broker no longer submits tasks

Algorithm 2 Calculate Fitness Function

Initialize $P_{\text{total}} = 0$ and Makespan $= 0$
for each machine m_j in the solution **do**
  Compute total resource utilization:
  $\text{CPU}_{\text{total}}(m_j) = \sum_{t_i \in T_j} \text{CPU}(t_i)$;
  $\text{Memory}_{\text{total}}(m_j) = \sum_{t_i \in T_j} \text{Mem}(t_i)$
  Calculate utilization ratios:
  $\text{CPU}_{\text{ratio}}(m_j) = \frac{\text{CPU}_{\text{total}}(m_j)}{\text{CPU}_{\text{cap}}(m_j)}$
  $\text{Memory}_{\text{ratio}}(m_j) = \frac{\text{Memory}_{\text{total}}(m_j)}{\text{Mem}_{\text{cap}}(m_j)}$
  Calculate power consumption $P(m_j)$:
  $P(m_j) = \text{CPU}_{\text{ratio}}(m_j) \times \text{Memory}_{\text{ratio}}(m_j)$
  Normalize power consumption:
  $P(m_j)_{\text{norm}} = \frac{P(m_j) - P(m_j)_{\text{min}}}{P(m_j)_{\text{max}} - P(m_j)_{\text{min}}}$
  Add normalized power consumption to P_{total}:
  $P_{\text{total}} = P_{\text{total}} + P(m_j)_{\text{norm}}$
  Calculate processing time :
  $\text{processing\_time} = \frac{\sum_{t_i \in T_j} \text{length}(t_i)}{\text{Processing Capacity}(m_j)}$
  Update Makespan if processing_time is greater
end for
Calculate fitness F:
$F = (\alpha \times P_{\text{total}}, \beta \times \text{Makespan})$
return F

solution B if A is at least as good as B in all objectives and strictly better in at least one objective.

- *Crowding Distance:* After non-dominated sorting, NSGA-II calculates the crowding distance for individuals within each front. Crowding distance measures the density of solutions around each individual in the objective space ; The space defined by the objectives (e.g., power consumption and makespan) where each solution is represented as a point.
- *Restart & Stop Condition:* The algorithm continues for a maximum number of generations after which the best individual based on Pareto dominance is retrieved as the optimal solution.

3) Deep Reinforcement Learning: As Deep Reinforcement Learning algorithm, we chooses for our work to design an Actor-Critic architecture. The Actor-Critic method has demonstrated proven performance in solving a variety of complex problems, including task scheduling [25], which makes it an ideal choice for our study. It combines the advantages of both policy-based and value-based methods in reinforcement learning.

- State Representation: The state is represented by the current status of both machines and tasks in the system. For each machine, we consider the track available CPU and memory. Task features include required CPU, memory and duration.
- Action Space: The action space consists of all valid task-machine pairs where a task can be scheduled on a machine without exceeding its resource capacities.
- Reward Function: The reward function is designed to ensure efficient utilization of resources reducing energy and makespan of allocating resources to tasks.
- Model Architecture: We use an actor-critic architecture, where:
  - The Actor (Policy Network) is a neural network which takes the current state representation as input and outputs logits, which are converted into a probability

distribution over actions (task-machine pairs) using the softmax function.
- The Critic (Value Network) is responsible of calculating advantages ($\hat{A}_t$) which estimate the value of taking a specific action compared to the average action taken under the current policy.

$$\hat{A}_t = Q_t - V(s_t)$$

where Q_t is the estimated return and $V(s_t)$ is the state-value function.

- Training the Agent: The agent incrementally learns by collecting trajectories, estimating advantages, and updating its Actor and Critic networks for each batch of observations.
- Inference: Each time, given a batch of pending tasks and available machines, we use the trained agent to select the best machine-task mapping decisions. We continue this process until all tasks are finished and the broker is no longer submitting tasks.

V. IMPLEMENTATION AND EXPERIMENTS

A. Implementation details

The algorithms used in this work were tested and put into place using the *CloudSimPy simulator* developed by Li and Hu [11] with Python. To generate workloads, we rely on real task allocation traces from the cloud precisely from Alibaba cluster trace data [26]. As for machines, we use heterogeneous servers with real characteristics sourced from the Standard Performance Evaluation Corporation dataset [27].

Regarding the tests conducted with the genetic algorithm, we set the parameters as follows: a maximum number of generations equal to 10, a population size of 50, a crossover probability of 0.7, and a mutation probability of 0.1. Following the training strategy outlined in [11], we train our DeepRL agent with 15 iterations and 12 episodes per batch. The learning rate is set to 1e-3, and the discount factor is 0.9. In our studies, we used a weighting strategy for the objective function to balance energy consumption and makespan. We chose $\alpha = 0.7$ and $\beta = 0.3$ to give higher priority to reducing energy consumption while still considering makespan.

B. Evaluation Metrics

- Energy consumption (Wh): We calculate energy consumption by summing the usage of each machine in the simulation environment after the execution of a batch of jobs. The energy consumption of a single machine is determined based on its CPU utilization over time. To accomplish this, we use real energy models sourced from [27]. These models provide detailed power measurements of various machines, quantifying them according to the CPU load.
- Makespan (s) : Maximum processing time across machines or resources.
- Execution Time (s) : The time needed by each algorithm to return its decisions.

C. Experiments

Given the substantial body of literature employing heuristics for task scheduling in cloud environments, we start with the thesis that heuristic methods, such as the one we selected, are optimally suited for our problem. By "optimally suited" , we mean that these methods demonstrate superior performance in accommodating the dynamic nature of cloud environments, including variations in the number and characteristics of tasks submitted, as well as the number and specifications of machines available.

Initial Thesis: Heuristics for Task Scheduling

To critically evaluate heuristics, we implement several scenarios designed to test their effectiveness and highlight their limitations. We begin our investigation with a first setup including five different lightweight workloads, all characterized by a relatively small number (50) of submitted tasks. Fig. 2 compares energy consumption across different scheduling methods, highlighting the efficiency of each algorithm in terms of power usage. We observe here that all algorithms show stable energy consumption across workloads. Fig. 3 shows that makespans increase with workload for all algorithms, with heuristic algorithms, particularly the Weighted Multi-Objective Heuristic (WMOH), consistently exhibiting lower makespans, indicating faster task processing times. Additionally, heuristics consistently exhibit lower execution times due to their low algorithmic complexity. These experimental results reinforce the initial hypothesis that heuristics are not only comparable in performance to more advanced algorithms but also offer significant advantages in execution time and energy efficiency. These characteristics make heuristics suitable for task scheduling in a cloud environment, especially in scenarios with small workloads.

Antithesis: Identifying Limitations of Heuristics

The limitations of heuristic methods become evident in more complex cloud environments. Fig. 4 and Fig. 5 illustrate the energy consumption and makespan performances of different selected methods in a second setup with scenarios with higher task density. Specifically, we generated 5 different workloads, each consisting of 1500 tasks with varying CPU and memory requirements, as well as differing lengths. The results clearly demonstrate that heuristic approaches often lead to sub-optimal performance, particularly in terms of energy consumption. This second setup reveal

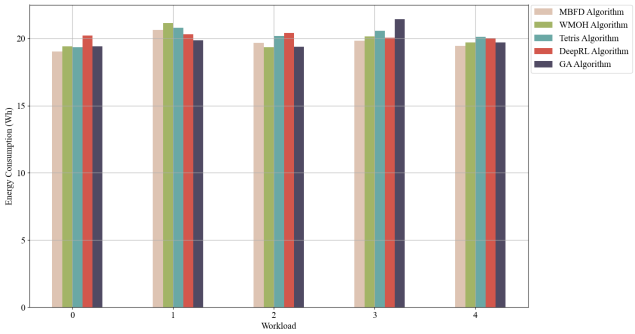

Fig. 2: Energy Consumption Comparison for Selected Scheduling Methods Applied to Initial Setup Workloads.

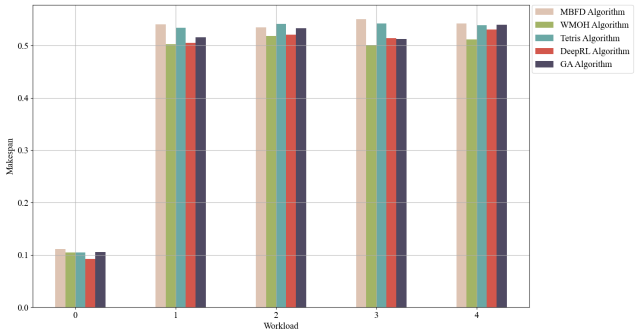

Fig. 3: Makespan Comparison for Selected Scheduling Methods Applied to Initial Setup Workloads.

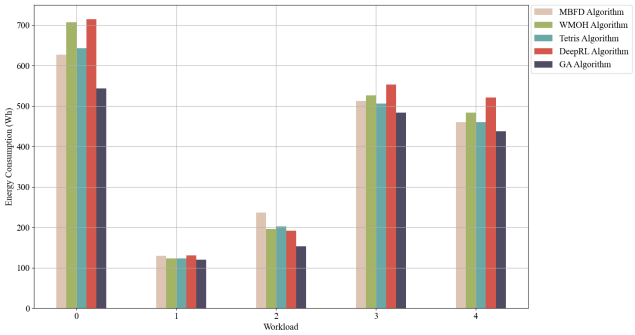

Fig. 4: Energy Consumption Comparison for Selected Scheduling Methods Applied to Second Setup Workloads.

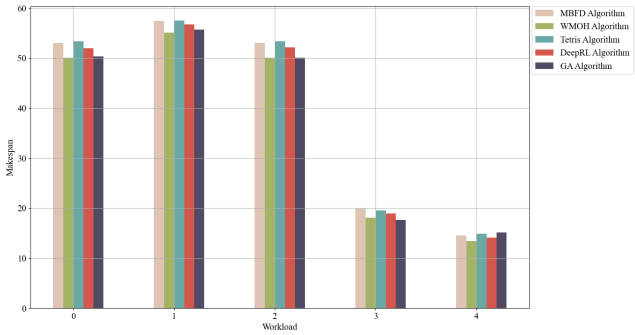

Fig. 5: Makespan Comparison for Selected Scheduling Methods Applied to Second Setup Workloads.

the shortcomings of heuristic methods, indicating that they may not be the optimal solution for complex cloud environments or particularly challenging scenarios. Therefore, it is worthwhile to explore alternative methods, such as meta-heuristics, which are well-regarded in the literature for their ability to efficiently explore search spaces and find sub- or near-optimal solutions for NP-complete problems.

Synthesis: Proposing Meta-Heuristics as an Alternative

To address these limitations, we investigate the Genetic Algorithm (GA), specifically using the NSGA-II version. We generated three more complex scenarios and observed the performance of different algorithms in terms of energy consumption, makespan and execution time.

a) First Scenario: In the first scenario, the objective is to evaluate the algorithm's performance in a resource-constrained environment with tasks of varying computational needs, simulating a real-world cloud setting with limited resources. To achieve this, we generated five different workloads, each consisting of a fixed load of 500 tasks distributed across 10 machines. The tasks exhibit high variability in CPU and memory requirements as well as differing lengths. We then analyze the mean values of energy consumption, makespan, and execution time for each algorithm across these five workloads to assess their performance comprehensively.

b) Second Scenario: In the second scenario, the objective is to test the algorithm's ability to efficiently allocate a very high volume of tasks across a relatively small number of machines, reflecting the challenges of balancing task distribution and resource utilization in such conditions. Specifically, we evaluated the algorithm's performance with 1,000 tasks distributed across 10 machines.

c) Third Scenario: In the third scenario, the objective is to examine the algorithm's capability to manage a high-density task load across a larger number of machines. Specifically, we distributed 1,000 tasks among 50 machines, with tasks varying widely in their requirements and duration. This scenario tests the limits of the algorithm's scalability and efficiency under heavy load conditions, evaluating how well it performs when both the task volume and the resource pool are substantial.

Tables II and III show the average energy consumption and makespan for different algorithms across the three scenarios, respectively. From the obtained results, we observe

that the Genetic Algorithm (GA) performs competitively in terms of energy consumption and makespan, often showing slightly better or similar performance compared to other algorithms (MBFD, WMOH, Tetris, and DeepRL).

TABLE II: Average Energy Consumption of Selected Algorithms Across Generated Scenarios

Scenarios	Average Energy Consumption (Wh)				
	GA	MBFD	WMOH	Tetris	DeepRL
1	**193.26**	195.19	196.27	198.15	193.59
2	**352.12**	357.13	357.24	357.05	355.01
3	**1757.55**	1798.24	1807.88	1799.83	1801.95

TABLE III: Average Makespan of Selected Algorithms Across Generated Scenarios

Scenarios	Average Makespan (s)				
	GA	MBFD	WMOH	Tetris	DeepRL
1	12.02	12.05	12.05	12.07	**12.00**
2	25.10	25.12	**25.00**	25.11	25.03
3	23.25	23.31	23.20	23.23	**23.06**

Perhaps this leads us to determine that GA is the best solution for achieving a balance between energy consumption and makespan.

Antithesis 2: Limitations of Meta-Heuristics

When visualising results from table IV, we can see that for Scenario 1, GA has an execution time of 3.78 seconds, significantly higher than the other algorithms. As scenarios become more complex, this disparity increases. In Scenario 2, GA's execution time rises sharply, while the other algorithms experience only minor increases. By Scenario 3, GA's execution time grows even more dramatically, highlighting its impracticality for complex scenarios. In contrast, other algorithms remain relatively efficient. **Note** that for DeepRL, the execution time reflects inference time rather than training time.

TABLE IV: Average Execution Time of Selected Algorithms Across Generated Scenarios

Scenarios	Average Execution Time (s)				
	GA	MBFD	WMOH	Tetris	DeepRL
1	3.78	**0.006**	0.023	0.011	0.065
2	23.54	**0.012**	0.029	0.027	0.16
3	71.40	**0.05**	0.13	0.12	0.5

We can conclude from these results that GA becomes increasingly computationally expensive as the solution space expands, making it impractical for time-sensitive environments such as cloud systems.

Synthesis 2: Deep Reinforcement Learning (DeepRL)

Having a more reasonable execution time compared to the genetic algorithm and very encouraging performance in multiple problems, including ours, we might consider that deep reinforcement learning can be the best solution in this context. To investigate this, we fix the number of machines and generate diverse workloads, varying this time the number of tasks, their CPU consumption, and their lengths. This procedure is applied to heterogeneous machine configurations, ranging from 5 to 30 machines.

Our analysis reveals a significant disparity in execution times (see Table V) between Genetic Algorithms (GA) , Heuristics and Deep Reinforcement Learning (DeepRL). For example, with 5 machines, the GA takes 6.11 seconds to execute, whereas DeepRL completes the same task in just 0.07 seconds. As the number of machines increases, this disparity grows even more pronounced. With 30 machines, the GA's execution time balloons to 18.91 seconds, while DeepRL's time increases to only 0.28 seconds. This trend underscores that, deep RL maintains relatively stable execution times even with more machines comparatively to the genetic algorithm.

TABLE V: Average Execution Time for Selected Algorithms Across Various Setups

No. Machines	Average Execution Time (s)				
	DeepRL	GA	WMOH	Tetris	MBFD
5	0.07	6.11	0.02	0.01	**0.01**
10	0.12	8.82	0.04	0.03	**0.01**
15	0.15	11.44	0.07	0.04	**0.02**
20	0.19	13.96	0.06	0.05	**0.02**
25	0.24	16.57	0.08	0.06	**0.04**
30	0.28	18.91	0.09	0.08	**0.03**

The results in Tables VI and VII reveal that DeepRL consistently delivers stable and balanced performance across various machine configurations; In terms of energy consumption, DeepRL is competitive across different setups. For example, while the energy consumption varies slightly with the number of machines, DeepRL generally shows efficiency similar to or slightly better than other algorithms, such as GA and Tetris. Similarly, in terms of makespan, DeepRL maintains a slight edge over other algorithms. The differences in makespan, while modest, suggest that DeepRL is capable of reducing processing time more effectively across a range of machine configurations.

This suggests that DeepRL offers a good trade-off between energy consumption, makespan, and execution time. Its performance is stable and reliable, making it a practical choice for environments with varying demands.

TABLE VI: Average Energy Consumption for Selected Algorithms Across Various Setups

No. Machines	Average Energy Consumption (Wh)				
	DeepRL	GA	WMOH	Tetris	MBFD
5	137.89	142.49	143.74	**129.77**	146.37
10	246.65	**225.17**	244.86	244.72	243.66
15	**327.76**	342.14	342.44	364.55	359.32
20	**421.26**	439.44	447.44	456.40	439.38
25	533.95	**526.75**	535.66	548.02	547.45
30	635.70	**635.41**	653.99	657.39	645.62

TABLE VII: Average Makespan for Selected Algorithms Across Various Setups

No. Machines	Average Makespan (Slowdown)				
	DeepRL	GA	WMOH	Tetris	MBFD
5	**21.94**	22.33	22.59	23.74	23.53
10	**22.66**	23.40	24.23	24.50	24.63
15	**22.55**	24.06	23.58	25.23	24.79
20	**22.72**	23.83	23.04	25.16	24.86
25	**22.97**	23.72	23.10	24.96	25.27
30	23.03	24.71	**22.76**	24.46	25.21

Antithesis 3: Limitations of DeepRL

Nevertheless, DeepRL faces several notable limitations. First, the training process for DeepRL is both time-consuming and computationally intensive. This extensive training requirement leads to high energy consumption. Secondly, DeepRL can encounter significant transferability issues. When the characteristics of a workload differ substantially from those used during training, the model's performance may degrade. This poses a particular challenge in dynamic or evolving environments where workload characteristics can change frequently. Without adequate training on diverse and representative workloads, DeepRL may produce sub-optimal results. Conversely, excessive training to cover a wider range of scenarios can, as we mentioned, exacerbate the problem of high energy consumption, as the computational demand and associated energy usage increase.

Thirdly, DeepRL models often suffer from a lack of interpretability. The complexity of these models can make it challenging to understand their decision-making processes. This lack of transparency is particularly problematic in a cloud system because it hinders users from trusting the model's decisions and makes it difficult to diagnose and address issues when the model's performance is subpar.

Final Synthesis

The dialectical process we undertook systematically led us through a series of insights regarding different algorithmic approaches in cloud computing task scheduling. Initially, our investigation revealed that heuristics, while effective for certain tasks with limited complexity, struggle when confronted with large-scale resource allocation problems involving numerous tasks. Transitioning to meta-heuristics, such as genetic algorithms, we found that while they offer promising performance improvements over heuristics, they introduce new challenges. Meta-heuristics often require extensive computational resources and time to converge on optimal solutions, which can be impractical in environments demanding rapid decision-making and responsiveness. Similarly, deep RL, known for its ability to learn complex decision-making policies, raises concerns about the interpretability and transferability of learned models across different operational contexts. Considering these result, it becomes clear that no single algorithmic approach emerges as universally superior in all cloud computing scenarios. Instead, the effectiveness of each method depends heavily on the specific characteristics of the workload, the dynamics of the cloud environment, and the performance metrics prioritized (such as energy efficiency, makespan and execution time).

VI. Conclusion and Future Works

Our investigation into the Hegelian dialectical framework for evaluating green cloud task scheduling techniques has highlighted the complexity and contextuality of optimizing such systems. The research shows that no single method—heuristic, meta-heuristic, or deep reinforcement learning is always ideal to optimize resource allocation in the cloud. Each approach has distinct advantages and limitations dependent on specific workload characteristics, cloud environment dynamics, and prioritized performance metrics such as energy efficiency, makespan, and execution time.

To address this multifaceted challenge, future work should focus on developing adaptive hybrid models that leverage the strengths of each method while mitigating their weaknesses. One promising direction is the creation of an intelligent algorithm selector that dynamically chooses the most suitable scheduling strategy based on real-time analysis of task and resource conditions. This selector would function by continuously monitoring cloud resources and task metrics to gather data on CPU usage, memory consumption, task arrival rates, and execution times. It would process this data to extract relevant features that inform the decision-making process, such as average task duration, peak load times, and energy consumption patterns. The prediction model would analyze historical and real-time data to predict which scheduling approach will yield the best results under the given conditions. The selector would dynamically choose and deploy the optimal scheduling algorithm, continuously learning and updating its decision-making criteria to improve over time. Integrating a feedback mechanism to evaluate the performance of the chosen algorithm and refine the

prediction model could be useful for ensuring continuous improvement and adaptability to changing conditions.

In the end, our dialectical analysis not only gave us a clear picture of current methods but also opened up new possibilities for innovative solutions in green cloud computing task scheduling. This helps us move towards both greater technical efficiency and ecological sustainability.

ACKNOWLEDGMENT

This work was supported by a grant from the French government, managed by the National Research Agency under the France 2030 initiative: DCARBO project within the PEPR Spleen program, reference number 22-PESP-0002. For open access purposes, a CC-BY 4.0 license has been applied by the authors to this document and will be applied to any subsequent version up to the author-accepted manuscript resulting from this submission.

REFERENCES

[1] Alexandre Monnin. *Lean ICT: Towards Digital Sobriety*. The Shift Project, 2019. URL: https://theshiftproject.org/en/article/lean-ict-towards-digital-sobriety/.

[2] David Fernández-Baca and A. Medepalli. "Allocating modules to processors in a distributed system with limited memory". In: *Computer Science* (1992), pp. 335–347.

[3] Muthucumaru Maheswaran et al. "Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems". In: Feb. 1999, pp. 30–44.

[4] H. Casanova et al. "Heuristics for scheduling parameter sweep applications in grid environments". In: *Proceedings 9th Heterogeneous Computing Workshop (HCW 2000)*, pp. 349–363.

[5] Syed Arshad Ali and Mansaf Alam. "Resource-Aware Min-Min (RAMM) Algorithm for Resource Allocation in Cloud Computing Environment". In: *CoRR* abs/1803.00045 (2018).

[6] George Amalarethinam and Vaaheedha Kfatheen. "Max-min Average Algorithm for Scheduling Tasks in Grid Computing Systems". In: 2012.

[7] Ali Aldailamy, Jamilu Yahaya Maipan-uku, and Abdullah Muhammed. "Heuristic Task Scheduling Algorithms for Optimal Resource Utilisation in Grid Computing". In: *INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN ENGINEERING TECHNOLOGY* 11 (Dec. 2020), pp. 1099–1107.

[8] Henan Zhao and Rizos Sakellariou. "An Experimental Investigation into the Rank Function of the Heterogeneous Earliest Finish Time Scheduling Algorithm". In: *Euro-Par 2003 Parallel Processing*. Ed. by Harald Kosch, László Böszörményi, and Hermann Hellwagner.

[9] Nobo Chowdhury et al. "Performance Evaluation of Various Scheduling Algorithm Based on Cloud Computing System". In: *Asian Journal of Research in Computer Science* (Oct. 2018), pp. 1–6.

[10] A K M Mashuqur Rahman Mazumder et al. "Dynamic task scheduling algorithms in cloud computing". In: *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*. 2019, pp. 1280–1286.

[11] Fengcun Li and Bo Hu. "DeepJS: Job Scheduling Based on Deep Reinforcement Learning in Cloud Data Center". In: Association for Computing Machinery, 2019.

[12] Anton Beloglazov, Jemal H. Abawajy, and Rajkumar Buyya. "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing". In: *Future Gener. Comput. Syst.* 28 (2012), pp. 755–768.

[13] R. Valarmathi and T. Sheela. "A comprehensive survey on Task Scheduling for parallel workloads based on Particle Swarm optimisation under Cloud environment". In: *2017 2nd International Conference on Computing and Communications Technologies (ICCCT)*. 2017, pp. 81–86.

[14] Nidhi Jain and Inderveer Chana. "Artificial bee colony based energy-aware resource utilization technique for cloud computing". In: *Concurrency and Computation: Practice and Experience* 27 (May 2014).

[15] Seyedali Mirjalili and Seyedali Mirjalili. "Genetic algorithm". In: *Evolutionary algorithms and neural networks: theory and applications* (2019), pp. 43–55.

[16] Qi Liu et al. "A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment". In: *Security and Communication Networks* 9 (Aug. 2016).

[17] Jafar Meshkati and Faramarz Safi. "Energy-aware resource utilization based on particle swarm optimization and artificial bee colony algorithms in cloud computing". In: *The Journal of Supercomputing* 75 (May 2019).

[18] Samar Shaban and Fatma Omara. "A deep learning based framework for optimizing cloud consumer QoS-based service composition". In: *Computing* 102 (May 2020).

[19] Ning Liu et al. "A Hierarchical Framework of Cloud Resource Allocation and Power Management Using Deep Reinforcement Learning". In: (Mar. 2017).

[20] Sudheer Mangalampalli et al. "DRLBTSA: Deep reinforcement learning based task-scheduling algorithm in cloud computing". In: *Multimedia Tools and Applications* 83 (2023), pp. 8359–8387.

[21] Filippo Poltronieri et al. "Reinforcement Learning vs. Computational Intelligence: Comparing Service Management Approaches for the Cloud Continuum". In: *Future Internet* 15.11 (2023), p. 359.

[22] Yaoyao Ping et al. "Deep reinforcement learning-based multi-task scheduling in cloud manufacturing under different task arrival modes". In: *Journal of Manufacturing Science and Engineering* 145.8 (2023), p. 081003.

[23] Georg Wilhelm Friedrich Hegel, Arnold V Miller, and John Niemeyer Findlay. "Phenomenology of spirit". In: (1977).

[24] K. Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197.

[25] Guilherme Piêgas Koslovski, Kleiton Pereira, and Paulo Roberto Albuquerque. "DAG-based workflows scheduling using Actor–Critic Deep Reinforcement Learning". In: *Future Gener. Comput. Syst.* 150.C (2024), pp. 354–363.

[26] Alibaba Group. *Alibaba Cluster Trace Data*. https://github.com/google/cluster-data. 2017.

[27] Standard Performance Evaluation Corporation. *SPECpower*. http://www.spec.org/power/.