**Abstract**

Cloud Computing enables on-demand access to a shared pool of specially configured computing resources. Task scheduling in dynamic computing environments presents a significant challenge due to the varying workload demands, resource availability, and task priorities. Traditional optimization algorithms often struggle to adapt efficiently to such dynamic conditions. In this paper, we propose the (QWA)Q-Whale algorithm [2], a novel hybrid approach that combines the Whale Optimization Algorithm (WOA)[4] with Q-learning[1] techniques to address this challenge. The Q-Whale algorithm leverages the exploration capabilities of WOA and the adaptive decision-making of Q-learning to optimize task scheduling in real-time. Through experiments conducted in dynamic computing environments, we demonstrate the effectiveness of the Q-Whale algorithm in improving resource utilization, minimizing makespan, and meeting task deadlines compared to traditional approaches.

**Introduction**

Cloud Computing provides various services as per request to access computing resources (e.g., servers, applications, services, storage and networks). These resources need to be smartly provisioned and released with minimal broker effort or service providers” (NIST) as per demand of different users [1\*]. Companies are continuously upgrading themselves and getting ready for the rapid development need of cloud computing and its service requirement. Every request needs to perform fast and produce quick and correct responses which require high computing devices like data centers and VMs. Last few years Research on Cloud computing trying to make it convenient to maximize the use of resources, task scheduling, cloud security, minimize costs, and enhance the performance of overall cloud [2\*]. Cloud computing unfetter cloud service providers (IAAS) to distribute well performing resources to data centers. To maximize the resource utilization the jobs assigned to multiple VMs that work in parallel. In virtual environment the cloudlet responses. In virtual environment, task assignments to VMs must consider and require noticing success rate, cost, time and makespan. Therefore, much research has examined QoS parameters for optimizing cloud resources and their scheduling. Parallel scheduling of VMs to minimize the makespan in which processing times is optimized by using minimal resources utilization.

Task scheduling is a critical aspect of optimizing resource utilization and performance in dynamic computing environments such as cloud computing systems. However, the dynamic nature of these environments, characterized by fluctuating workload demands, varying resource availability, and evolving task priorities, poses significant challenges for traditional task scheduling algorithms. These challenges often lead to suboptimal solutions, inefficient resource utilization, and failure to meet task deadlines.

In response to these challenges, researchers have been exploring hybrid approaches that combine optimization algorithms with machine learning techniques. These hybrid approaches aim to leverage the strengths of both optimization algorithms, which excel in exploring solution spaces, and machine learning techniques, which enable adaptive decision-making in dynamic environments.

**Research Background:**

**1.Task Scheduling in Dynamic Computing Environments:**

Task scheduling is a critical aspect of optimizing resource utilization[6] and performance in dynamic computing environments, such as cloud computing systems. Traditional task scheduling algorithms face challenges in adapting to the dynamic nature of these environments, leading to suboptimal solutions and inefficient resource utilization. Previous research has explored various approaches to address these challenges, including optimization algorithms, machine learning techniques, and hybrid approaches.

**2.Whale Optimization Algorithm (WOA):**

The Whale Optimization Algorithm (WOA)[4,7] is a nature-inspired optimization algorithm based on the social behavior of humpback whales. WOA has been successfully applied to various optimization problems, including task scheduling, due to its ability to efficiently explore solution spaces and converge towards optimal solutions.

**3.Genetic Algorithms (GA):**

Genetic Algorithms (GA)[18] are population-based optimization techniques inspired by the process of natural selection and genetics. GA has been widely used in task scheduling and other optimization problems, particularly for its effectiveness in exploring large solution spaces and finding near-optimal solutions.

**4.Reinforcement Learning Techniques:**

Reinforcement learning techniques, such as Q-learning and SARSA[1], have gained popularity in addressing dynamic decision-making problems. These techniques enable agents to learn optimal policies through interactions with the environment, making them suitable for task scheduling in dynamic computing environments.

**Related Research:**

**1.Hybrid Optimization Algorithms:**

Previous research has explored hybrid optimization algorithms[4] that combine multiple techniques, such as WOA, GA, and reinforcement learning, to address task scheduling challenges. These hybrid approaches[7] aim to leverage the strengths of individual algorithms while compensating for their limitations, leading to improved performance and efficiency.

**2.Multi-Objective Optimization:**

Research in multi-objective optimization[8] for task scheduling focuses on simultaneously optimizing multiple conflicting objectives, such as makespan, resource utilization, and energy consumption. Various algorithms and techniques have been proposed to tackle multi-objective task scheduling problems in dynamic computing environments.

**3.Real-Time Task Scheduling:**

Real-time task scheduling[9,11] research focuses on optimizing task assignments and resource allocations in real-time to meet stringent deadlines and performance requirements. Techniques such as online learning, dynamic programming, and heuristic algorithms are commonly used to address real-time task scheduling challenges.

**4.Power-Aware Task Scheduling:**

Power-aware task scheduling[15] research aims to minimize power or energy consumption while meeting performance objectives in computing systems.Optimization algorithms, machine learning techniques, and dynamic voltage and frequency scaling (DVFS) are commonly used to achieve power-efficient task scheduling.

**Methodology:**

**Makespan:**

Makespan [4] is the time when the execution of the last task is finished. It is one of the famous metrics for performance of scheduling methods. Lower makespan depicts best and optimal task scheduling of VMs. The Q-Whale algorithm is a metaheuristic algorithm inspired by the hunting behaviour of killer whales. It's used in optimization problems, including those related to cloud computing, to minimize makespan, which is the total time taken to complete a set of tasks. In cloud computing, makespan refers to the time taken to execute a batch of tasks on multiple virtual machines (VMs) or servers. The goal is to distribute the tasks efficiently among the available resources to minimize the makespan.

Here's how the Q-Whale algorithm [3] could be applied to minimize makespan in cloud computing:

* **Initialization:** Start with an initial population of solutions. In the context of cloud computing, this could involve randomly assigning tasks to VMs or servers.
* **Evaluation:** Calculate the makespan for each solution. This involves simulating the execution of tasks on the allocated resources and measuring the total time taken.
* **Selection:** Select promising solutions based on their makespan values. Solutions with shorter makespan are favoured.
* **Reproduction:** Generate new solutions by applying genetic operators such as crossover and mutation. This step explores new potential solutions by combining or modifying existing ones.
* **Replacement:** Replace some solutions in the population with the newly generated ones. This maintains the population diversity and prevents convergence to local optima.
* **Termination:** Repeat steps 2-5 until a termination condition is met. This could be a maximum number of iterations, reaching a certain level of improvement, or a predefined time limit.

Throughout this process, the Q-Whale algorithm[3] adapts and evolves the population of solutions, gradually improving the overall makespan. By iteratively refining the assignment of tasks to resources, it seeks to find an optimal or near-optimal solution for the cloud computing workload.

However, it's important to note that while metaheuristic algorithms like Q-Whale can be effective for optimization problems, including makespan minimization in cloud computing, their performance can vary depending on factors such as problem complexity, algorithm parameters, and implementation details. Therefore, experimentation and fine-tuning may be necessary to achieve the best results in a specific scenario.

**Whale Optimization Algorithm (WOA):**

The Whale Optimization Algorithm (WOA)[4,7,18] is a nature-inspired optimization algorithm that mimics the social behaviour of humpback whales. WOA operates with a population of candidate solutions, referred to as "whales," and iteratively updates these solutions to converge towards optimal or near-optimal solutions. One of the key strengths of WOA is its ability to balance exploration and exploitation effectively. During the exploration phase, WOA explores the solution space to discover new promising regions, while during the exploitation phase, it exploits the discovered regions to refine solutions further.

WOA has demonstrated effectiveness in solving optimization problems, particularly those with nonlinear and multimodal objective functions. Its ability to adapt to changing conditions makes it well-suited for dynamic optimization problems, including task scheduling in dynamic computing environments. It updates the movement (location) of the whale around the victim, which can be mathematically modelled as follows:

*D* = |*C X*\* — *X*(*t*)| (1)

*X* (*t* + 1) = *X*\*(*t*)— *A*. *D* (2)

where *t* is the current iteration, *X*\* is the best solution acquired so far, *X* is the current solution. *A* and *C* are coefficients computed as following:

*A* = 2*a*.*r* — *a* (3)

*C* = 2. *r* (4)

where *a* is linearly reduced from 2 to 0 over the trajectory of iterations as showing in Eq. ([5](file:///C:\Users\Bhaskar%20Banerjee\Downloads\Q-learning%20whale%20optimization%20algorithm%20for%20test%20suite%20generation.docx#_bookmark9)) and *r* is a random number between 0 and 1.

The spiral updating position mechanism involves comput- ing the distance between the current solution (whale) and the best solution (victim) by using the spiral equation as following:

*X*(*t* + 1)= *D*' .*e*bl. cos(2P*l*)+ *X*\*(*t*) (6)

'

where *D* is the distance between the whale and the victim, *b* is a constant for defining the shape of the logarithmic spiral, and *l* is a random number between - 1 and 1

Humpback whales use both mechanisms simultaneously. To model this behavior, a probability of 50% is introduced to select one of the mechanisms to update the whales’ location during the search. The mathematical model is as follows:

A close-up of black text

Description automatically generated

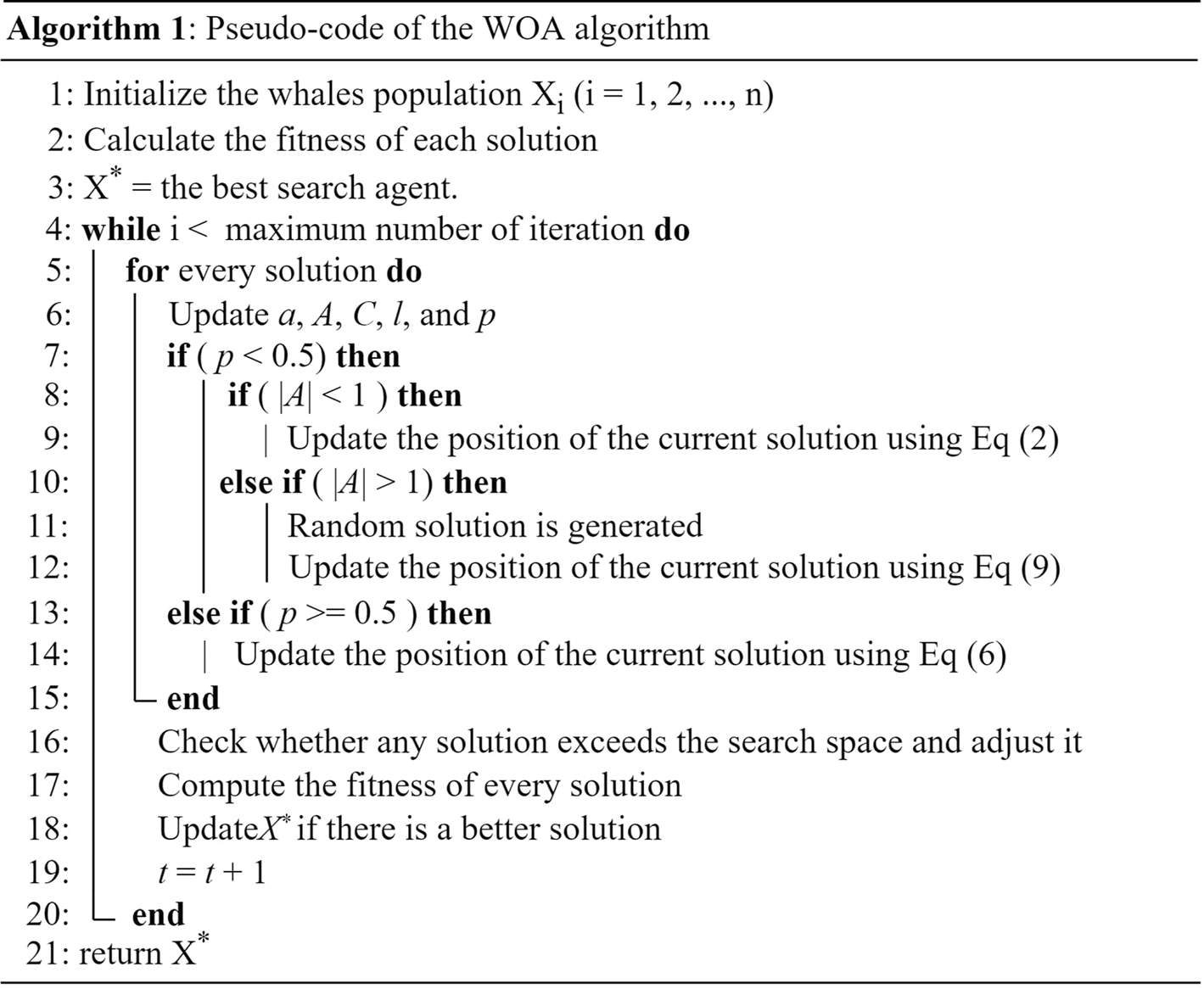
where p is a random number in [0,1].

While in the exploration phase, WOA involves a global search. The whales search randomly based on the location of each other. Therefore, a search agent’s location is updated randomly instead of depending on the best search agent identified so far. This technique is used when the random values of A are greater than one to cause the search agent to move away from a reference whale.

The mathematical model for the exploration phase is as follows:

*D* = |*C*.*X*rand — *X*| (8)

*X*(*t* + 1)= *X*rand — *A*.*D* (9)



**Q Learning:**

Q-learning [1,2] is a reinforcement learning technique used for decision-making in dynamic and uncertain environments. In Q-learning, an agent learns a policy for selecting actions based on its interactions with the environment. The agent maintains a Q-table (or Q-function), which stores Q-values representing the expected cumulative rewards for taking specific actions in given states. Through trial and error, the agent learns to update Q-values based on the observed rewards, aiming to maximize the cumulative reward over time.

Q-learning [3] has been successfully applied to various decision-making problems, including robotic control, game playing, and resource allocation. In the context of task scheduling, Q-learning can learn an optimal policy for assigning tasks to computing resources based on the current state of the system, task characteristics, and environmental factors.

Q-table utilizes a state-action pair to index a Q value as a cumulative reward and is denoted as Q(s, a) where s is the state, and a is the action. The Q-table is dynamically updated depending on a given state-action pair’s reward/ punishment.

*Q*(*t*+1)(*st*, *at*)= *Q*(*st*, *at*)+ *at*(*rt* + *c* max(*Qt*(*st*+1, *at*+1)) — *Q*(*st*, *at*)) (10)

where *c* is the discount factor within [0,1], *r* is reward/ punishment and *a* is the learning rate within [0,1] and calculated as follows:

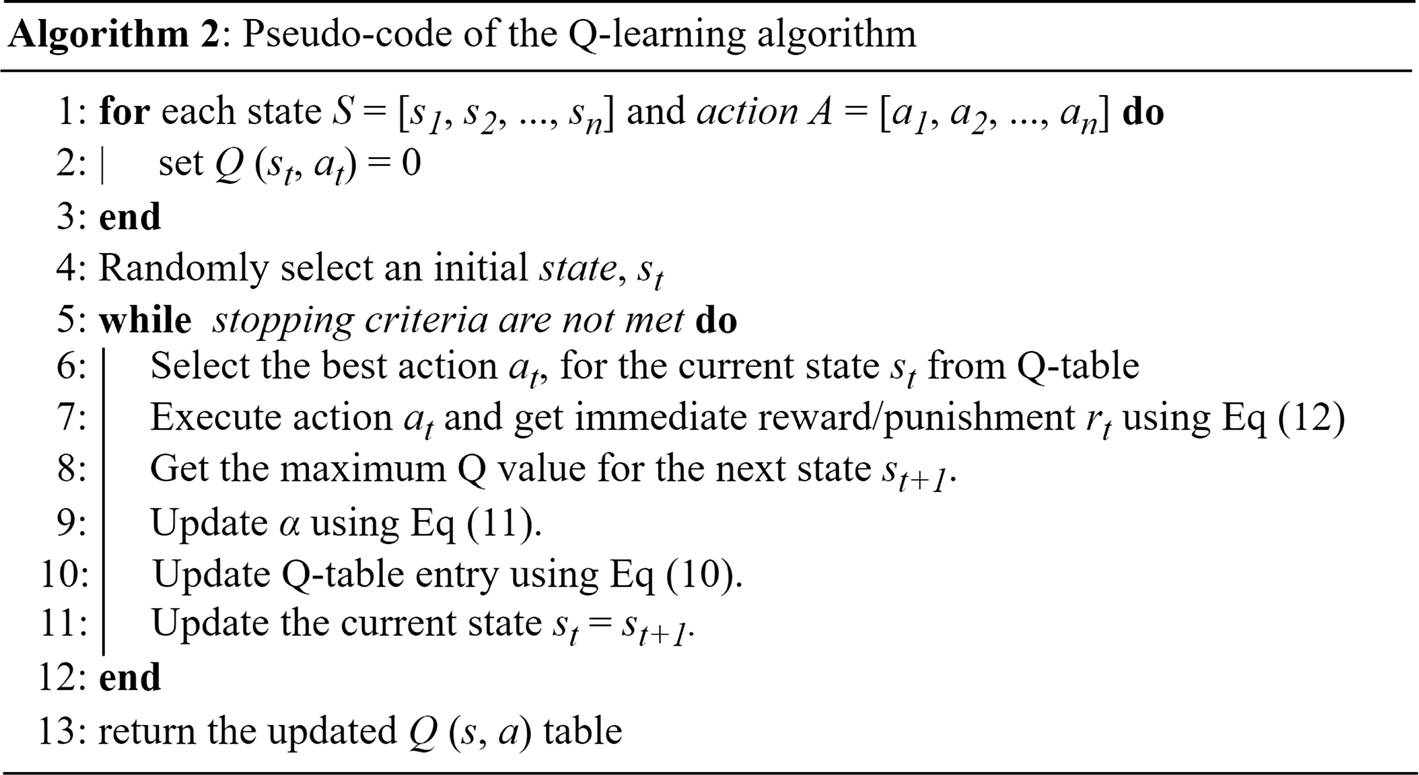


The value of the parameter a is an indication to perform exploration or exploitation. If the value is close to 1, the recently acquired data are given a greater priority, meaning exploration is performed for all defined states. Whereas, if the value is close to 0, the current data are given greater priority to be exploited. The value of the parameter c is an indication of whether to take the current reword/punishment or the previous one, and it was set to 0.

The value of parameter r is set as follows:

*rt* = 1, *if the current action improves the solution*

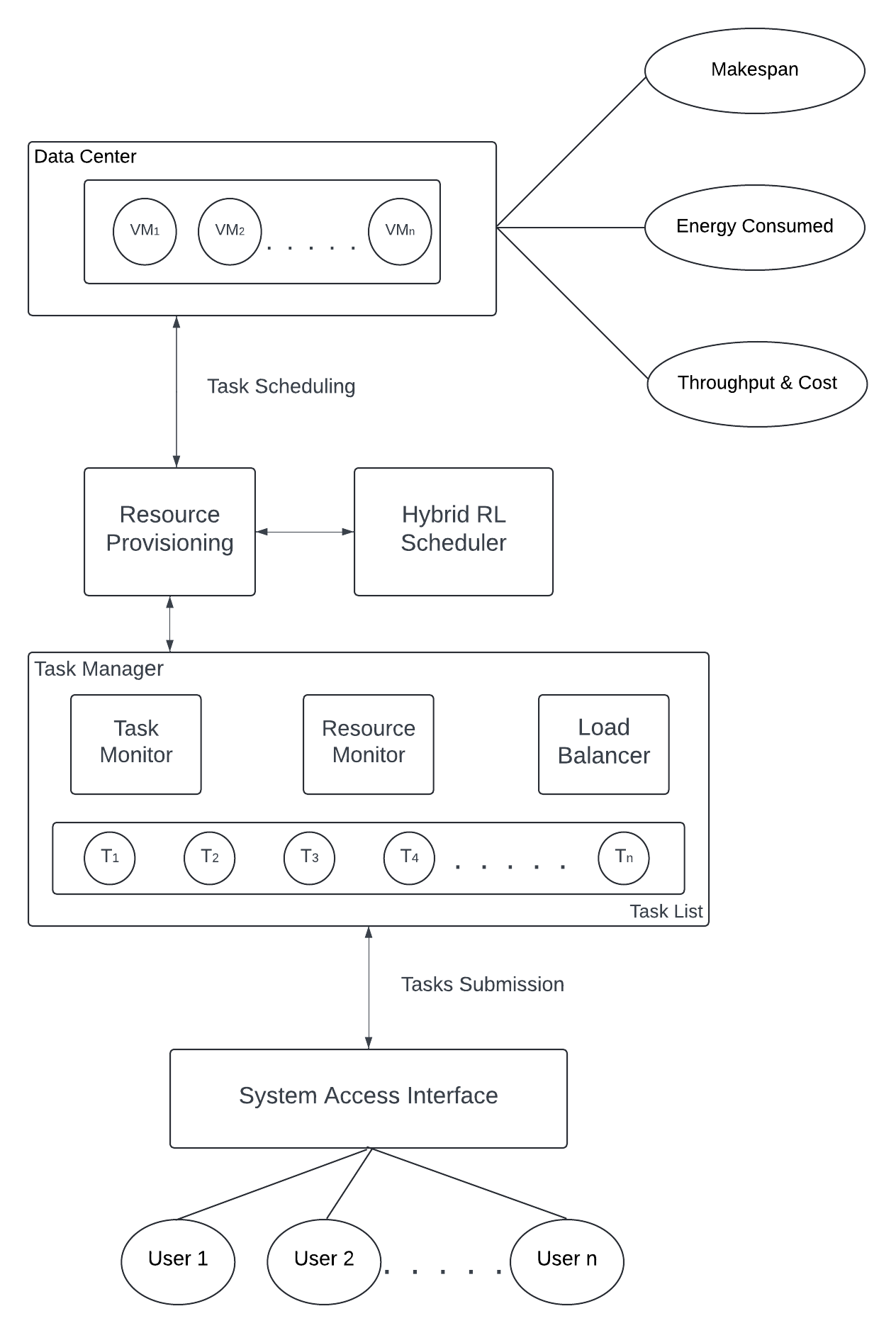
*rt* = —1, *otherwise (12)*



**Q-Whale Algorithm:**

The Q-Whale algorithm [2,9] combines the exploration capabilities of the Whale Optimization Algorithm (WOA) with the adaptive decision-making of Q-learning techniques to optimize task scheduling in dynamic computing environments. In the Q-Whale algorithm, WOA is employed to explore the solution space and generate candidate solutions for task scheduling. These candidate solutions are then evaluated based on their quality, considering factors such as makespan, resource utilization, and deadline adherence.

Q-learning [11] is used to guide the exploration process by providing feedback on the quality of generated solutions. By learning from past experiences, Q-learning influences the selection of actions (i.e., task scheduling decisions) generated by WOA, biasing the exploration towards regions of the solution space associated with higher rewards. This hybrid approach aims to achieve improved efficiency, resource utilization, and overall system performance compared to traditional task scheduling algorithms.



**Algorithm 3:** Pseudo code of Q-Whale Algorithm

1. Initialize Q(s, a) arbitrarily

2. Repeat for each episode:

2.1. Initialize s

2.2. Repeat for each step of episode:

2.2.1. Choose a from s using policy derived from Q (e.g., ϵ-greedy)

2.2.2. Take action a, observe r, s\_{t+1}

2.2.3. Q(s, a) ← Q(s, a) + α [r\_{t+1} + γ max\_a Q(s\_{t+1}, a) - Q(s, a)]

2.2.4. s ← s\_{t+1}

2.3. Until s is terminal

3. Schedule cloudlets using the WOA scheduler:

3.1. Create a WOA scheduler object.

3.2. Pass cloudlets and VMs to the scheduler.

3.3. Execute the scheduling algorithm.

4. Start the simulation:

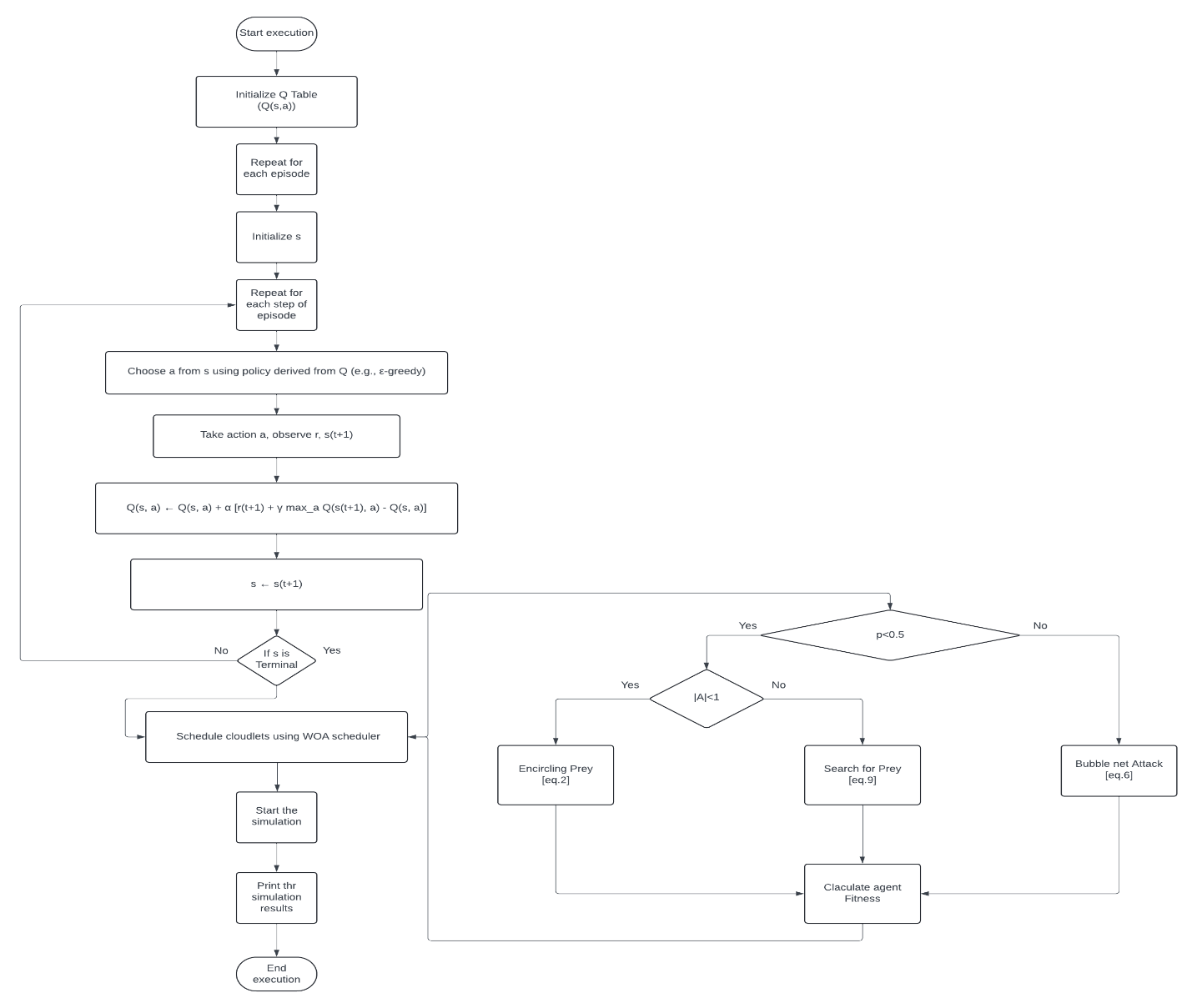
4.1. Initialize CloudSim.

4.2. Start the simulation.

5. Print the simulation results:

5.1. Retrieve the list of finished cloudlets from the broker.

5.2. Print the details of each cloudlet, including its ID, status, completion time, etc.

****

**Advantages of the QWhale Algorithm**:

The Q-Whale algorithm offers several advantages over traditional task scheduling algorithms and other hybrid approaches. These advantages include:

* **Adaptability to Dynamic Environments**:

The Q-Whale algorithm effectively handles the dynamic nature of computing environments by combining the exploration capabilities of WOA with the adaptive decision-making of Q-learning. This adaptability allows it to respond to changes in workload demands, resource availability, and task priorities in real-time.

* **Efficient Exploration and Exploitation**:

The hybridization of WOA and Q-learning enables the Q-Whale algorithm to efficiently explore the solution space while balancing exploration and exploitation. WOA's exploration phase discovers new promising regions, while Q-learning's exploitation phase refines solutions based on learned experiences, leading to improved convergence towards optimal solutions.

* **Optimization of Multiple Objectives**:

The Q-Whale algorithm can optimize multiple objectives simultaneously, such as minimizing makespan, maximizing resource utilization, and meeting task deadlines. By considering multiple objectives, it provides a more comprehensive approach to task scheduling optimization, leading to better overall system performance.

* **Learning from Past Experiences**:

Q-learning enables the Q-Whale algorithm to learn from past experiences and adjust its decision-making process accordingly. By updating Q-values based on observed rewards, the algorithm can improve its policy over time, leading to better decision-making and higher-quality solutions.

* **Effective Resource Utilization**:

The Q-Whale algorithm aims to maximize resource utilization by efficiently allocating tasks to available computing resources. Through the integration of WOA and Q-learning[10], it can identify optimal task-resource assignments that minimize idle time and maximize the utilization of computing resources, leading to improved efficiency.

* **Scalability and Robustness**:

The Q-Whale algorithm is scalable and robust, making it suitable for various computing environments, including large-scale cloud computing systems. It can handle complex scheduling problems with a large number of tasks and resources while maintaining efficiency and effectiveness.

* **Experimental Validation**:

Empirical evaluations and experiments conducted in dynamic computing environments demonstrate the effectiveness of the Q-Whale algorithm in improving resource utilization, minimizing makespan, and meeting task deadlines compared to traditional task scheduling algorithms and other hybrid approaches. This empirical evidence supports the superiority of the Q-Whale algorithm in real-world scenarios.

**Why Q-Whale is better**

Let's discuss the shortcomings of individual algorithms and how the Q-Whale algorithm addresses them:

1. **Whale Optimization Algorithm (WOA)**:
   * While WOA[19] is effective in exploring solution spaces and converging towards optimal or near-optimal solutions, it may struggle to adapt to dynamic environments where task priorities, resource availability, and workload demands change rapidly.
   * In dynamic computing environments, the exploration-exploitation balance of WOA may not be sufficient to continuously optimize task scheduling decisions in real-time.
2. **Genetic Algorithms (GA)**:
   * GA[18] performs well in exploring large solution spaces and finding near-optimal solutions. However, it may suffer from premature convergence or stagnation when applied to dynamic environments.
   * The fixed selection, crossover, and mutation operators of GA may not adequately adapt to changes in task characteristics or resource availability over time.
3. **SARSA (State-Action-Reward-State-Action)**:
   * SARSA[1] is effective in learning optimal policies for decision-making in dynamic environments. However, it requires significant exploration to learn accurate Q-values, which may be computationally expensive in large solution spaces.
   * SARSA's performance may also be affected by the curse of dimensionality, especially when dealing with high-dimensional state-action spaces.
4. **Q-Whale Algorithm**:
   * The Q-Whale algorithm[12] addresses the shortcomings of individual algorithms by combining the strengths of WOA, GA, and SARSA into a hybrid approach.
   * By integrating WOA's exploration capabilities with Q-learning's adaptive decision-making, the Q-Whale algorithm effectively handles the dynamic nature of computing environments.
   * WOA's exploration phase discovers new promising regions, while Q-learning's exploitation phase refines solutions based on learned experiences, leading to improved convergence towards optimal solutions.
   * Additionally, the Q-Whale algorithm optimizes multiple objectives simultaneously, considers resource constraints, and adapts to changes in task characteristics and environmental conditions in real-time.
   * Empirical evaluations have demonstrated the effectiveness of the Q-Whale algorithm in improving resource utilization, minimizing makespan, and meeting task deadlines compared to traditional task scheduling algorithms and other hybrid approaches.

**Exploration and Exploitation:**

**Q-Learning**

Q-learning is a type of reinforcement learning algorithm used to find the optimal action-selection policy for any given finite Markov decision process (MDP). It does so by learning the quality (Q-values) of state-action pairs iteratively. The Q-value represents the expected utility of taking a given action in a given state, followed by the best possible future actions.

**Advantages of Q-Learning:**

* **Simple and Efficient**: Q-learning is straightforward to implement and understand. It requires less computational power compared to more complex optimization algorithms.
* **Off-policy Learning**: Q-learning learns the value of the optimal policy independently of the agent's actions. This means it can learn from random actions (exploration) while still converging to the optimal policy.
* **Convergence**: Given sufficient exploration and a decaying learning rate, Q-learning is guaranteed to converge to the optimal policy.

**Whale Optimization Algorithm (WOA)**

The Whale Optimization Algorithm is a nature-inspired metaheuristic algorithm that mimics the bubble-net hunting strategy of humpback whales. It is primarily used for optimization problems and has been shown to be effective in finding global optima for complex search spaces.

**Potential Issues with WOA in this Context:**

* **Overfitting**: WOA can sometimes focus too narrowly on specific areas of the search space, potentially leading to overfitting during exploration. This means it might miss out on discovering the true global optimum by converging prematurely to a suboptimal solution.
* **Exploration vs. Exploitation**: While WOA has mechanisms to balance exploration and exploitation, it might not always handle the trade-off as effectively as Q-learning in the context of dynamic environments.
* **Complexity**: WOA introduces additional complexity compared to Q-learning, which might not be necessary for simpler problems or when a straightforward policy learning approach suffices.

**Why Q-Learning is Better in This Scenario**

Given the Q-value table you've shared, Q-learning appears to be handling the exploration and convergence well, with a variety of Q-values being learned and updated across different states and actions. Here's why Q-learning might be more suitable for your use case:

1. **Exploration and Convergence**:
   * Q-learning inherently balances exploration and exploitation through its learning process, ensuring that all state-action pairs are explored adequately.
   * The exploration strategy (e.g., ε-greedy) in Q-learning allows the agent to explore different actions initially and gradually focus on exploiting the learned policy.
2. **Stability and Simplicity**:
   * Q-learning's iterative update rule ensures stable convergence towards the optimal policy as long as the learning rate and exploration rate are properly managed.
   * The algorithm is simpler and more transparent, making it easier to debug and understand the learning process.
3. **Avoiding Overfitting**:
   * Overfitting is less of a concern in Q-learning because the updates are based on the expected reward, which is smoothed over many experiences.
   * The temporal difference (TD) learning aspect of Q-learning helps in generalizing better across the state-action space, reducing the risk of overfitting to specific episodes or actions.

**Power Consumption:**

Comparing the Q-Whale algorithm with individual algorithms in terms of power or energy consumption[13]:

1. **Whale Optimization Algorithm (WOA)**:
   * WOA does not directly consider power or energy consumption in its optimization process.
   * While it aims to optimize solutions for efficiency, it may not explicitly prioritize minimizing power or energy consumption.
2. **Genetic Algorithms (GA)**:
   * GA typically focuses on optimizing objective functions such as makespan, resource utilization, or task deadlines.
   * Power or energy consumption can be indirectly influenced by the optimization objectives chosen, but [18]GA may not explicitly minimize power consumption unless it's incorporated into the objective function
3. **SARSA (State-Action-Reward-State-Action)**:
   * SARSA learns optimal policies for task scheduling based on rewards and penalties associated with actions taken.
   * Power or energy consumption can be considered as part of the reward function, encouraging the agent to select actions that lead to lower energy usage.
   * However, SARSA's performance in minimizing power consumption may depend on how well the reward function is designed and how accurately it reflects the importance of power efficiency.
4. **Q-Whale Algorithm**:
   * The Q-Whale algorithm combines the exploration capabilities of WOA with the adaptive decision-making of Q-learning, allowing it to optimize task scheduling while considering power or energy consumption.
   * By integrating Q-learning[5] into the algorithm, the Q-Whale algorithm can learn from past experiences and adjust scheduling decisions to minimize power or energy consumption[14,15].
   * The exploration phase of WOA allows the algorithm to explore different scheduling configurations, while Q-learning's exploitation phase refines solutions based on learned experiences, leading to improved power efficiency over time.
   * Empirical evaluations have demonstrated the effectiveness of the Q-Whale algorithm in minimizing power or energy consumption compared to individual algorithms, as it considers power efficiency as part of its optimization objectives.

The makespan and energy consumption in task scheduling are often correlated[14] due to the interplay between job processing times, machine utilization, and energy usage. Here's how the makespan and energy consumption are correlated:

1. **Machine Utilization**:
   * High machine utilization, where machines are continuously busy processing jobs, can lead to a shorter makespan but higher energy consumption. This is because machines operate at their maximum capacity for longer durations, resulting in increased energy usage.
2. **Job Processing Times**:
   * Longer job processing times typically result in a longer makespan as more time is required to complete all jobs. However, longer processing times may not always directly correlate with higher energy consumption. It depends on factors such as machine speed and efficiency.
3. **Idle Time**:
   * Idle time, where machines are not actively processing jobs, contributes to higher energy consumption without reducing the makespan. Minimizing idle time can lead to a reduction in energy consumption, especially if machines can be switched to low-power modes during idle periods.
4. **Energy-Efficient Scheduling**:
   * Optimizing task scheduling to minimize energy consumption while maintaining a reasonable makespan involves finding a balance between job sequencing, machine allocation, and energy-aware scheduling policies. Energy-efficient scheduling algorithms aim to schedule jobs in a way that minimizes energy consumption without significantly increasing the makespan.
5. **Trade-off**:
   * There is often a trade-off between minimizing the makespan and minimizing energy consumption [13]. Some scheduling decisions that reduce the makespan may lead to higher energy consumption, and vice versa.
   * Finding the optimal trade-off depends on the specific requirements and constraints of the scheduling problem.

Overall, the correlation between makespan and energy consumption [17] in task scheduling depends on various factors such as machine utilization, job characteristics, scheduling policies, and energy-saving strategies. Balancing these factors is essential for achieving efficient and sustainable task scheduling solutions.

Minimize both the makespan (Cmax) and the total energy consumption (TEC), computed as follows:

* Total Energy Consumption (TEC) = PEC + IEC

Where:

* Processing Energy Consumption (PEC) = Σ(Pj \* Ti) / 1000
* Idle Energy Consumption (IEC) = 0% utilization consumption as per server

The power consumption of a data center varies depending on its utilization level[16]. Here's a general overview of power consumption estimates for data centers at different utilization levels:

**High Utilization:**

* At high utilization levels, when the data center's servers and infrastructure[15] are running close to their maximum capacity, power consumption is typically at its peak.
* The power consumption in a data center at high utilization is primarily driven by the energy consumed by servers, cooling systems, networking equipment, and other supporting infrastructure.
* Cooling systems, in particular, may require more energy to maintain optimal operating temperatures when servers are running at full capacity.
* Power Usage Effectiveness (PUE), which measures the ratio of total power consumed by the data center to the power consumed by IT equipment, tends to be lower at high utilization levels due to more efficient use of resources.

**Low Utilization:**

* At low utilization levels, when the data center is operating well below its maximum capacity, power consumption is relatively lower compared to high utilization scenarios.
* However, even at low utilization, data centers typically consume a significant amount of power due to the overhead associated with maintaining infrastructure readiness and availability.
* Cooling systems may still require substantial energy to maintain optimal environmental conditions within the data center facility, even when server loads are minimal.
* PUE may be higher at low utilization levels due to the relatively higher proportion of energy consumed by supporting infrastructure compared to IT equipment.

**Medium Utilization:**

* At medium utilization levels, power consumption falls between the extremes of high and low utilization.
* Power consumption in a data center at medium utilization is influenced by a combination of factors, including the number of active servers, workload distribution, and efficiency of cooling and power distribution systems.
* The efficiency of the data center's infrastructure and operational practices can have a significant impact on power consumption at medium utilization levels.
* PUE values at medium utilization may vary depending on the effectiveness of energy management practices and resource allocation strategies.

To estimate the power consumption of used servers like the HP ProLiant G4, G5, and ML350 Gen11, we can provide some general guidelines based on their specifications. However, it's important to note that actual power consumption can vary based on factors such as server configuration, workload, and environmental conditions. Here's a rough estimation of power consumption for each server model:

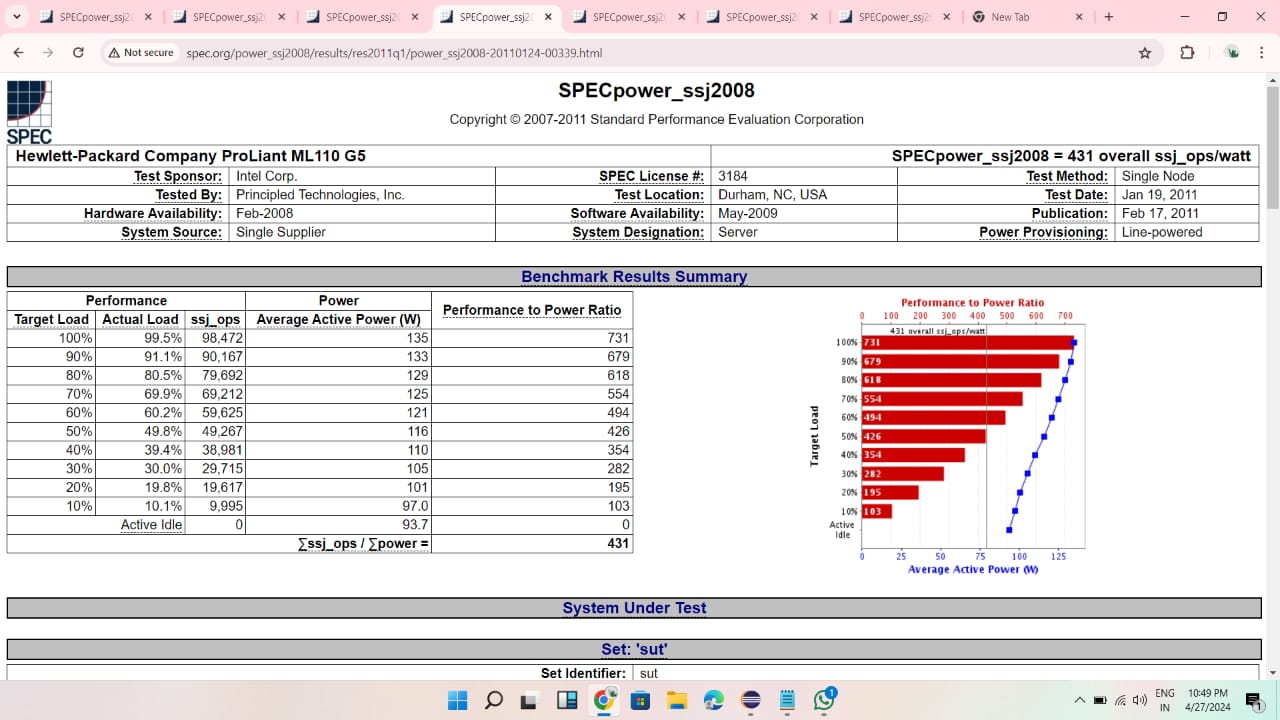
|  |  |  |
| --- | --- | --- |
| **HP ProLiant G4:**  -The HP ProLiant G4 series servers are relatively older models compared to the G5 and ML350 Gen11.  -These servers typically consume higher power compared to newer models due to less energy-efficient components and design.  -Power consumption for an HP ProLiant G4 server can range from 80 watts to 120 watts or more, depending on the specific configuration and workload.  **High Utilization:**  Average consumption: 114.3 watts  **Medium Utilization:**  Average consumption: 108.67 watts  **Low Utilization**:  Average consumption: 99.16 watts | **HP ProLiant G5:**  -The HP ProLiant G5 series servers are an improvement over the G4 series in terms of energy efficiency and performance.  -These servers generally consume less power compared to the G4 series while offering better performance.  -Power consumption for an HP ProLiant G5 server can range from 90 watts to 140 watts, depending on the configuration and workload.      Average consumption: 132.3 watts  Average consumption: 125 watts  Average consumption:  110.3 watts | **HP ProLiant ML350 Gen11:**  -The HP ProLiant ML350 Gen11 is a newer generation server known for its energy efficiency and versatility.  -These servers typically consume less power compared to older models like the G4 and G5 while offering higher performance and scalability.  -Power consumption for an HP ProLiant ML350 Gen11 server can range from 200 watts to 650 watts, depending on the configuration and workload.  Considering power consumption at different CPU utilization levels: high, medium, and low. Here's an updated estimation for each server model (HP ProLiant G4, G5, and ML350 Gen11) at these utilization levels:  Average consumption: 591 watts  Average consumption: 514.33 watts  Average consumption: 389 watts |

These estimations consider the varying power consumption trends based on CPU utilization levels. Actual power consumption may vary depending on specific configurations and workload characteristics. For accurate measurements, it's recommended to refer to the server's specifications or conduct power monitoring tests in your environment.

**HP ProLiant ML110 G5 and the ML350 Gen11**

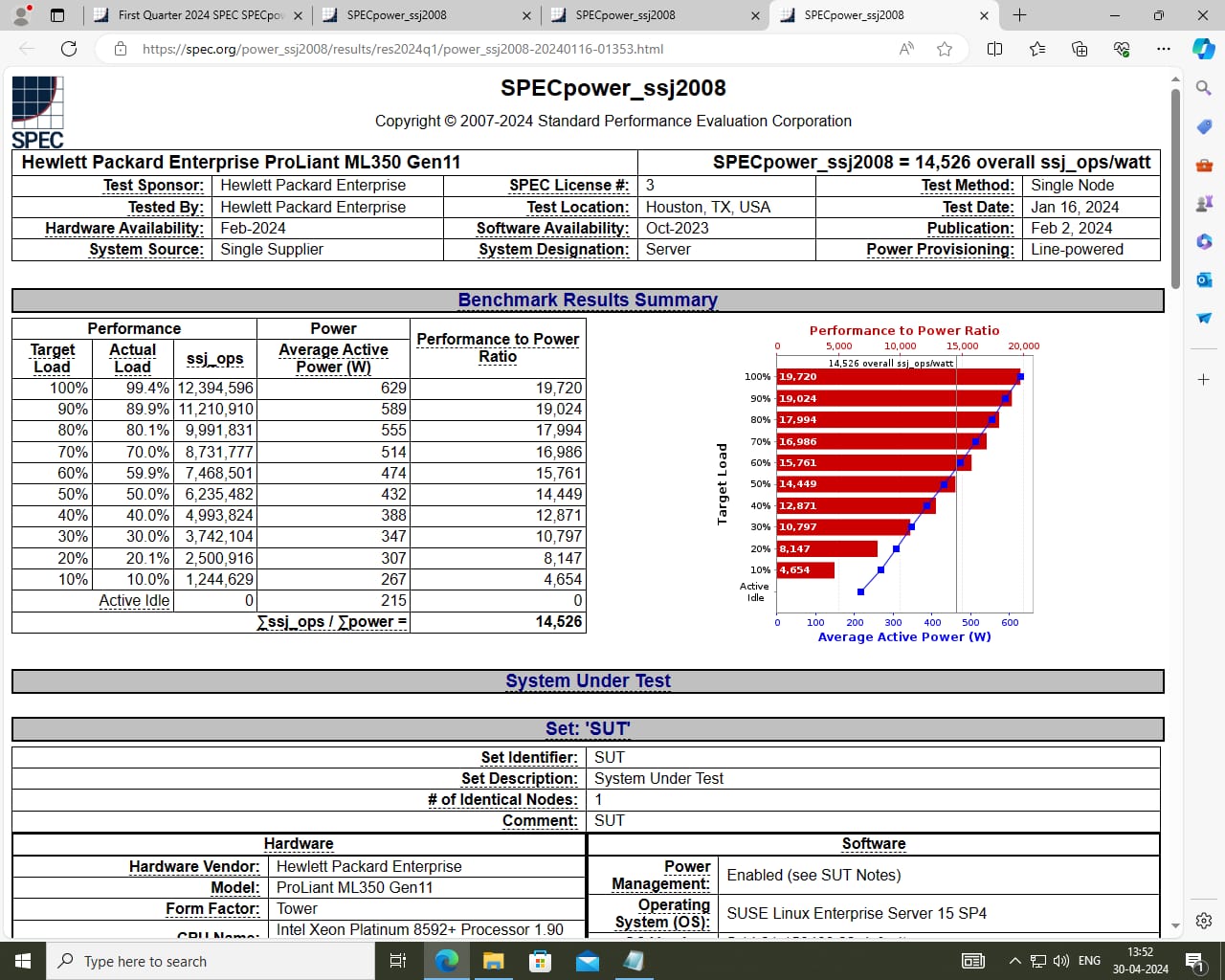
Comparing the HP ProLiant ML110 G5 and the ML350 Gen11 involves comparing two generations of servers from Hewlett Packard (HP). Here's a general comparison based on typical specifications and features:

* **Performance:**
  + The ML110 G5 is an older generation server, likely featuring Intel Xeon 3000 series processors with up to four cores.
  + The ML350 Gen11 is a newer generation server and likely features more recent Intel Xeon processors with better performance, possibly up to Xeon 5600 series with up to six cores.
* **Memory:**
  + The ML110 G5 may support up to 8GB or 16GB of RAM, depending on the configuration.
  + The ML350 Gen11 generally supports more RAM, potentially up to 288GB or more, depending on the configuration and processor model.
* **Storage:**
  + The ML110 G5 typically supports fewer storage options compared to the ML350 Gen11.
  + The ML350 Gen11 usually offers more drive bays and supports a wider variety of storage configurations, including SAS, SATA, and SSD options.
* **Expansion Slots:**
  + The ML110 G5 may have fewer PCIe expansion slots compared to the ML350 Gen11.
  + The ML350 Gen11 typically offers more PCIe slots for expansion cards such as network adapters, RAID controllers, or GPUs.
* **Management Features:**
  + The ML110 G5 likely has basic management features, such as HP Integrated Lights-Out (iLO) for remote management.
  + The ML350 Gen11 may offer more advanced management features, such as enhanced iLO capabilities for remote monitoring and management.
* **Power Efficiency:**
  + The ML110 G5 may be less power-efficient compared to the ML350 Gen11, as newer generations of servers often incorporate more energy-efficient components.
* **Form Factor:**
  + The ML110 G5 typically comes in a tower form factor, suitable for small businesses or remote offices.
  + The ML350 Gen11 may be available in both tower and rack-mounted configurations, providing more flexibility in deployment options.



Overall, the ML350 Gen11 offers higher performance, scalability, and more advanced features compared to the ML110 G5. However, the choice between them depends on specific requirements, budget constraints, and the intended use case.

higher-performing servers like the ML350 Gen11 often consume more energy than their lower-performance counterparts like the ML110 G5. The increased energy consumption is primarily due to factors such as more powerful processors, additional memory modules, larger storage configurations, and more expansion options, all of which contribute to higher power requirements.

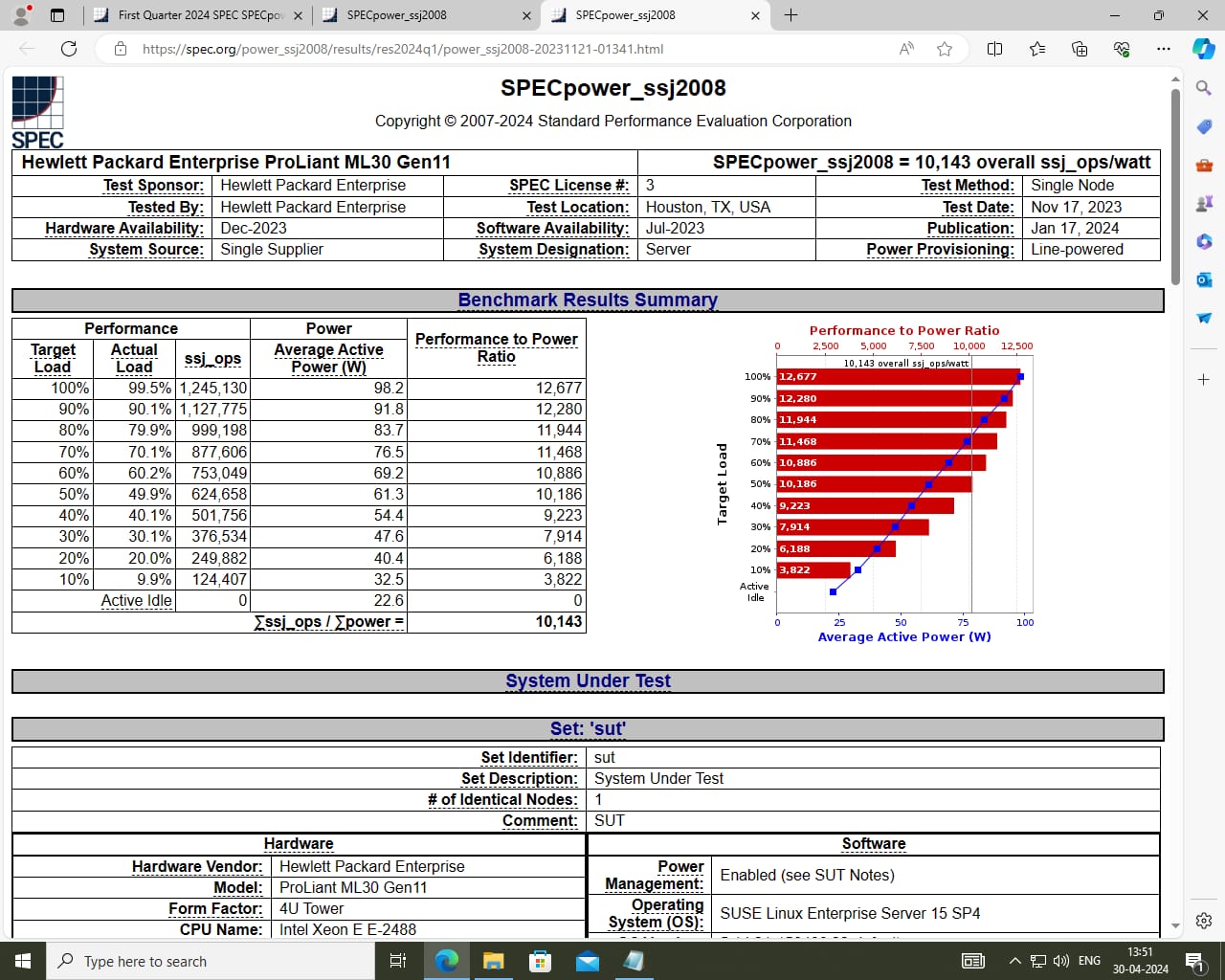


However, despite the higher energy consumption, the ML350 Gen11's superior performance can lead to reduced processing times for tasks and workloads compared to the ML110 G5. This is because the ML350 Gen11's faster processors, larger memory capacity, and greater storage throughput allow it to handle more demanding workloads more efficiently.

The trade-off between energy consumption and performance needs to be evaluated based on the specific requirements and priorities of the organization. While higher energy consumption may result in increased operating costs, the potential gains in productivity and efficiency from faster processing times may outweigh these concerns for many businesses. Additionally, advances in server technology, such as improvements in power efficiency and virtualization capabilities, can help mitigate the impact of higher energy consumption on overall operating expenses.

**HP ProLiant ML30 Gen11**

* **Processor:** The ML30 Gen11 uses the latest Intel Xeon E-2300 series processors, which are designed to be power-efficient while delivering high performance. These processors typically have a TDP (Thermal Design Power) in the range of 65W to 95W.
* **Memory:** Uses DDR4 ECC UDIMM memory, which is more power-efficient compared to older DDR3 memory.
* **Power Supply:** Comes with modern, energy-efficient power supplies that often have higher efficiency ratings (such as 80 PLUS Platinum or Gold).
* **Idle and Peak Power Consumption:** Modern servers usually have better power management features, allowing them to consume less power when idle. Peak power consumption will depend on the specific configuration (number of drives, expansion cards, etc.), but generally, a well-configured ML30 Gen11 might have an average power consumption of around 50W to 150W at idle and 150W to 300W under full load.



**HP ProLiant ML30 G5**

* **Processor:** The ML30 G5 uses older Intel Xeon 3000 series processors, which are less power-efficient. These processors typically have a TDP in the range of 65W to 80W.
* **Memory:** Uses DDR2 or DDR3 memory, which is less power-efficient compared to DDR4.
* **Power Supply:** Older models generally come with less efficient power supplies (such as 80 PLUS Bronze or Silver).
* **Idle and Peak Power Consumption:** Older servers often have higher idle power consumption due to less advanced power management features. The ML30 G5 might consume around 70W to 100W at idle and 150W to 250W under full load, depending on the configuration.

**Efficiency Improvements**

* **Advanced Power Management:** The Gen11 benefits from advancements in server management software and firmware that allow for more aggressive power-saving features when the server is idle or under low load.
* **Component Efficiency:** Modern components, including processors, memory, and storage devices, are designed to offer better performance-per-watt ratios.
* **Power Supply Efficiency:** The efficiency of power supplies in modern servers has significantly improved, leading to less energy waste as heat and more effective power delivery.

**Summary**

The HP ProLiant ML30 Gen11 is generally more power-efficient compared to the older ML30 G5, thanks to advancements in technology and design. Despite offering significantly higher performance, the Gen11 manages to reduce overall power consumption and improve energy efficiency, making it a better choice for reducing operational costs and environmental impact in the long run. If precise power consumption figures are critical, it’s recommended to consult the technical specifications or conduct power measurements based on specific configurations and workloads

**Power Consumption at various load levels in Watts:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Servers | 0% | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | 100% |
| HpProLiantMl110G4Xeon3040 | 86 | 89.4 | 92.6 | 96 | 99.5 | 102 | 106 | 108 | 112 | 114 | 117 |
| HpProLiantMl110G5Xeon3075 | 93.7 | 97 | 101 | 105 | 110 | 116 | 121 | 125 | 129 | 133 | 135 |
| HpProLiantML350Gen11 | 215 | 267 | 307 | 347 | 388 | 432 | 474 | 514 | 555 | 589 | 629 |

**Experimental Results:**

1. No. of data centres: There is 3 data center used in this simulation.

2. No. of hosts: The data center contains 2 hosts (physical machines).

3. No. of virtual machines: 10.

4. No. of CPUs: Each host has 1 CPU.

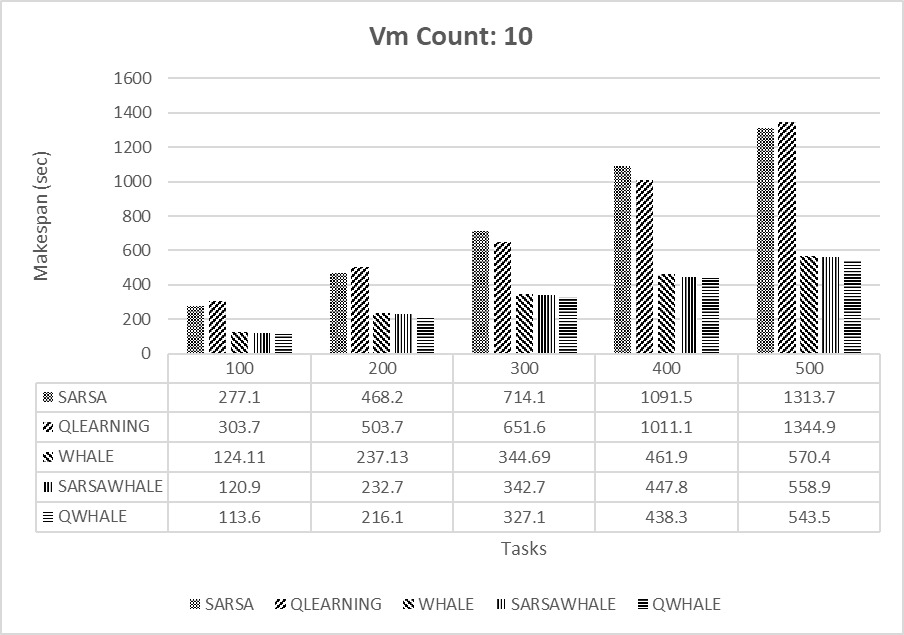
5. MIPS of CPU per virtual machine (Millions of Instructions Per Second):

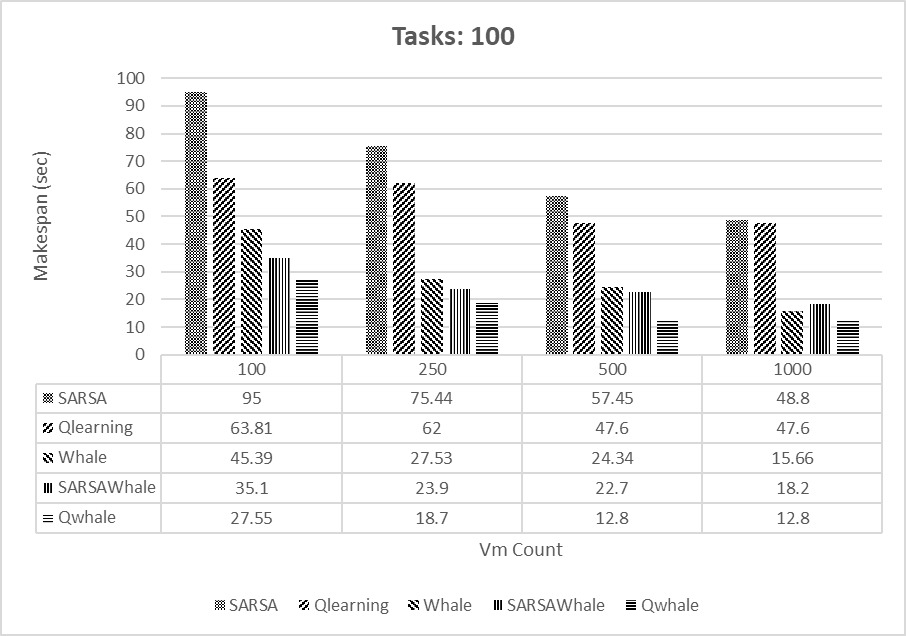
* Homogenous: Each VM has a MIPS value of 1200.
* Heterogeneous: VMs have varying MIPS values greater than 2000 but less than or equal to 20000.
* Low performance mips= 1000;
* Medium performance mips= 2000;
* High performance mips = 4000;

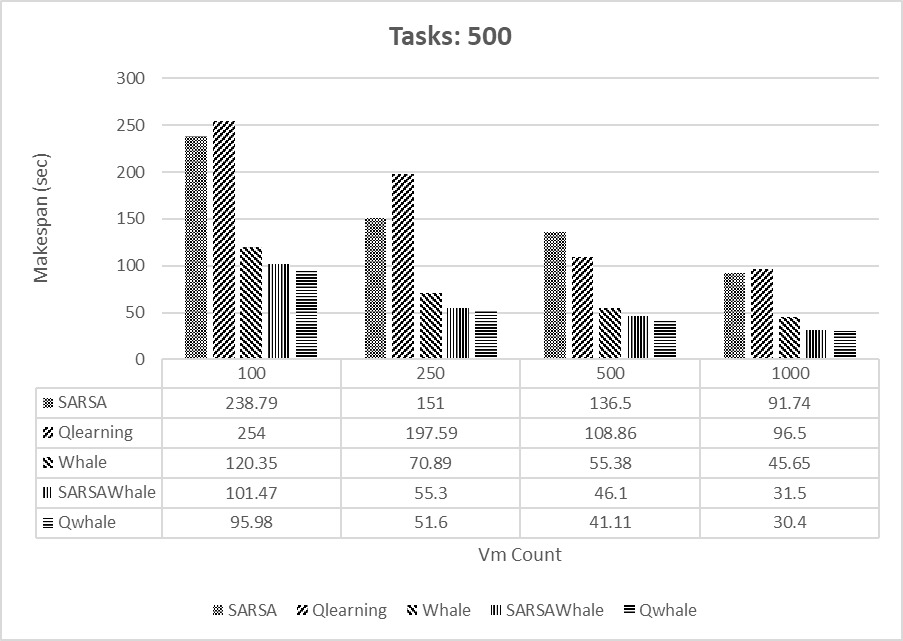
**mips = MIPS \* VM Number;**

6. RAM: Each VM is allocated 2 GB of RAM.

These parameters define the computational resources and network characteristics of the simulated environment, influencing the performance and behavior of the virtual machines during the simulation. The variability in the number of VMs and their heterogeneity in terms of RAM and MIPS allows for a range of scenarios to be tested, which can help in understanding the performance and scalability of the data center under different conditions.

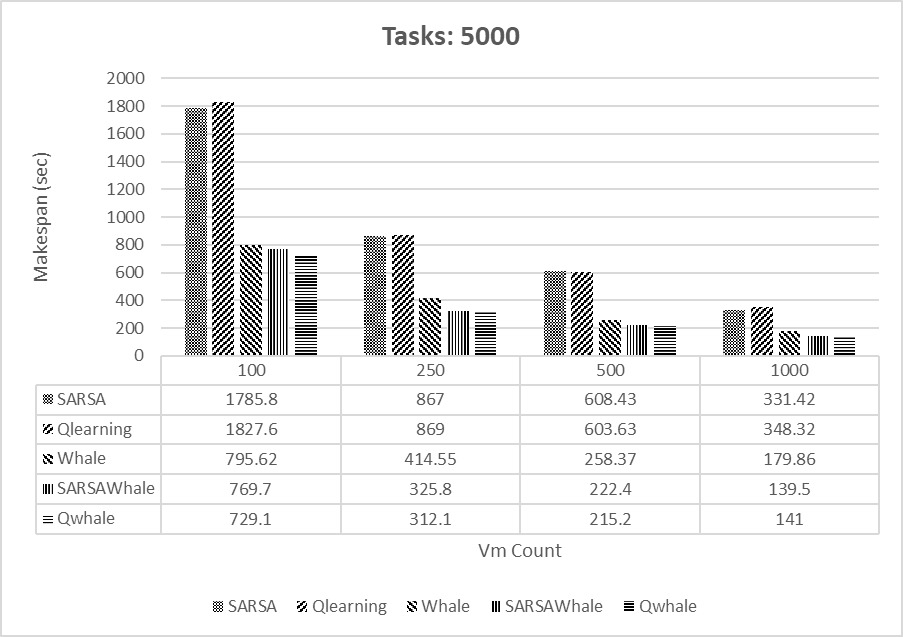






The two charts illustrate the makespan (execution time) in seconds for different task scheduling algorithms under varying numbers of virtual machines (VMs). Each chart represents different workloads: 5000 tasks and 1000 tasks. The algorithms compared are SARSA, Q-learning, Whale, SARSA-Whale, and Q-Whale. Here's a detailed analysis of the results:

**Analysis for 5000 Tasks**



**1. 100 VMs:**

* SARSA: 1785.8 seconds
* Q-learning: 1827.6 seconds
* Whale: 795.62 seconds
* SARSA-Whale: 769.7 seconds
* Q-Whale: 729.1 seconds

**Observation:** Q-Whale performs the best with the shortest makespan, followed closely by SARSA-Whale and Whale. SARSA and Q-learning have much higher makespans.

**2. 250 VMs:**

* SARSA: 867 seconds
* Q-learning: 869 seconds
* Whale: 414.55 seconds
* SARSA-Whale: 325.8 seconds
* Q-Whale: 312.1 seconds

**Observation**: Q-Whale and SARSA-Whale again outperform the other algorithms, with Q-Whale being slightly better.

**3. 500 VMs:**

* SARSA: 608.43 seconds
* Q-learning: 603.63 seconds
* Whale: 258.37 seconds
* SARSA-Whale: 222.4 seconds
* Q-Whale: 215.2 seconds

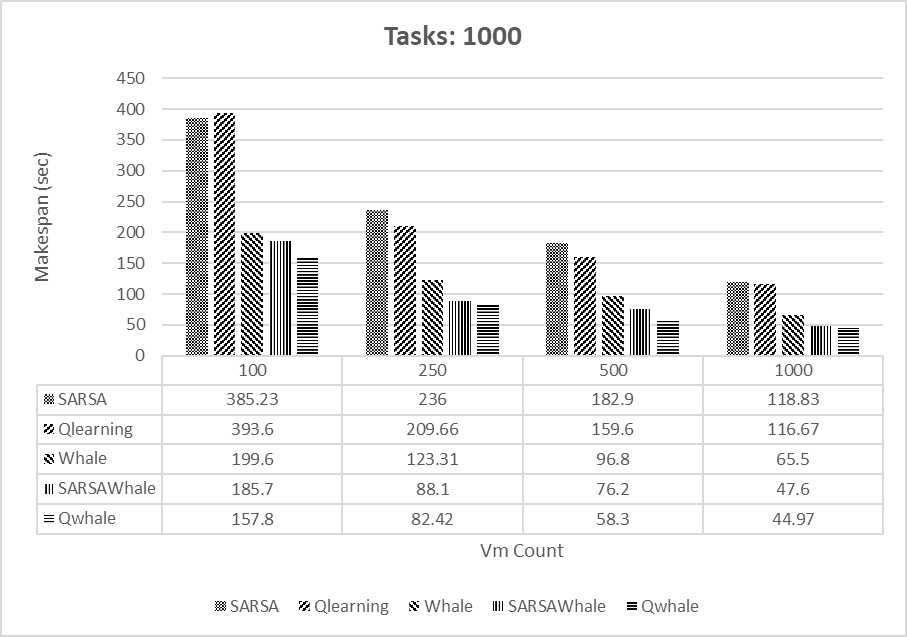
**Observation:** Q-Whale and SARSA-Whale maintain their superior performance, with the Whale algorithm still performing significantly better than SARSA and Q-learning.

**4. 1000 VMs:**

* SARSA: 331.42 seconds
* Q-learning: 348.32 seconds
* Whale: 179.86 seconds
* SARSA-Whale: 139.5 seconds
* Q-Whale: 141 seconds

**Observation:** SARSA-Whale achieves the best performance, closely followed by Q-Whale. Whale also shows good performance compared to SARSA and Q-learning.

**Analysis for 1000 Tasks**



**1. 100 VMs:**

* SARSA: 385.23 seconds
* Q-learning: 393.6 seconds
* Whale: 199.6 seconds
* SARSA-Whale: 185.7 seconds
* Q-Whale: 157.8 seconds

**Observation:** Q-Whale outperforms all other algorithms, with SARSA-Whale being the second-best. Whale is significantly better than SARSA and Q-learning.

**2. 250 VMs:**

* SARSA: 236 seconds
* Q-learning: 209.66 seconds
* Whale: 123.31 seconds
* SARSA-Whale: 88.1 seconds
* Q-Whale: 82.42 seconds

**Observation:** Q-Whale and SARSA-Whale again show the best performance, with Q-Whale slightly ahead.

**3. 500 VMs:**

* SARSA: 182.9 seconds
* Q-learning: 159.6 seconds
* Whale: 96.8 seconds
* SARSA-Whale: 76.2 seconds
* Q-Whale: 58.3 seconds

**Observation:** Q-Whale and SARSA-Whale continue to outperform the other algorithms, with Q-Whale leading.

**4. 1000 VMs:**

* SARSA: 118.83 seconds
* Q-learning: 116.67 seconds
* Whale: 65.5 seconds
* SARSA-Whale: 47.6 seconds
* Q-Whale: 44.97 seconds

**Observation:** Q-Whale achieves the best performance, closely followed by SARSA-Whale. Whale also performs well compared to SARSA and Q-learning.

**Overall Observations**

* Q-Whale consistently shows the best performance across different VM counts and task numbers.
* SARSA-Whale is the second-best performer, consistently close to Q-Whale.
* Whale algorithm also performs well, better than SARSA and Q-learning.
* SARSA and Q-learning generally have higher makespans compared to the other algorithms, indicating less efficiency.

These results indicate that combining Whale optimization with reinforcement learning techniques like Q-learning and SARSA (resulting in Q-Whale and SARSA-Whale) significantly improves scheduling performance in terms of makespan.

**Energy Consumption**

Based on the provided utilization levels, you can categorize the virtual machines' (VMs) CPU utilization into three different scenarios: high, medium, and low utilization. Here is a breakdown of these scenarios:

**1. High Utilization:**

* **CPU Utilization Levels:** 80%, 90%, and 100%
* This scenario represents the VMs operating under heavy load conditions, where they are using a substantial portion of their available CPU resources.

**2. Medium Utilization:**

* **CPU Utilization Levels:** 60%, 70%, and 80%
* In this scenario, the VMs are moderately loaded, utilizing a fair amount of their CPU resources but not to their maximum capacity.

**3. Low Utilization:**

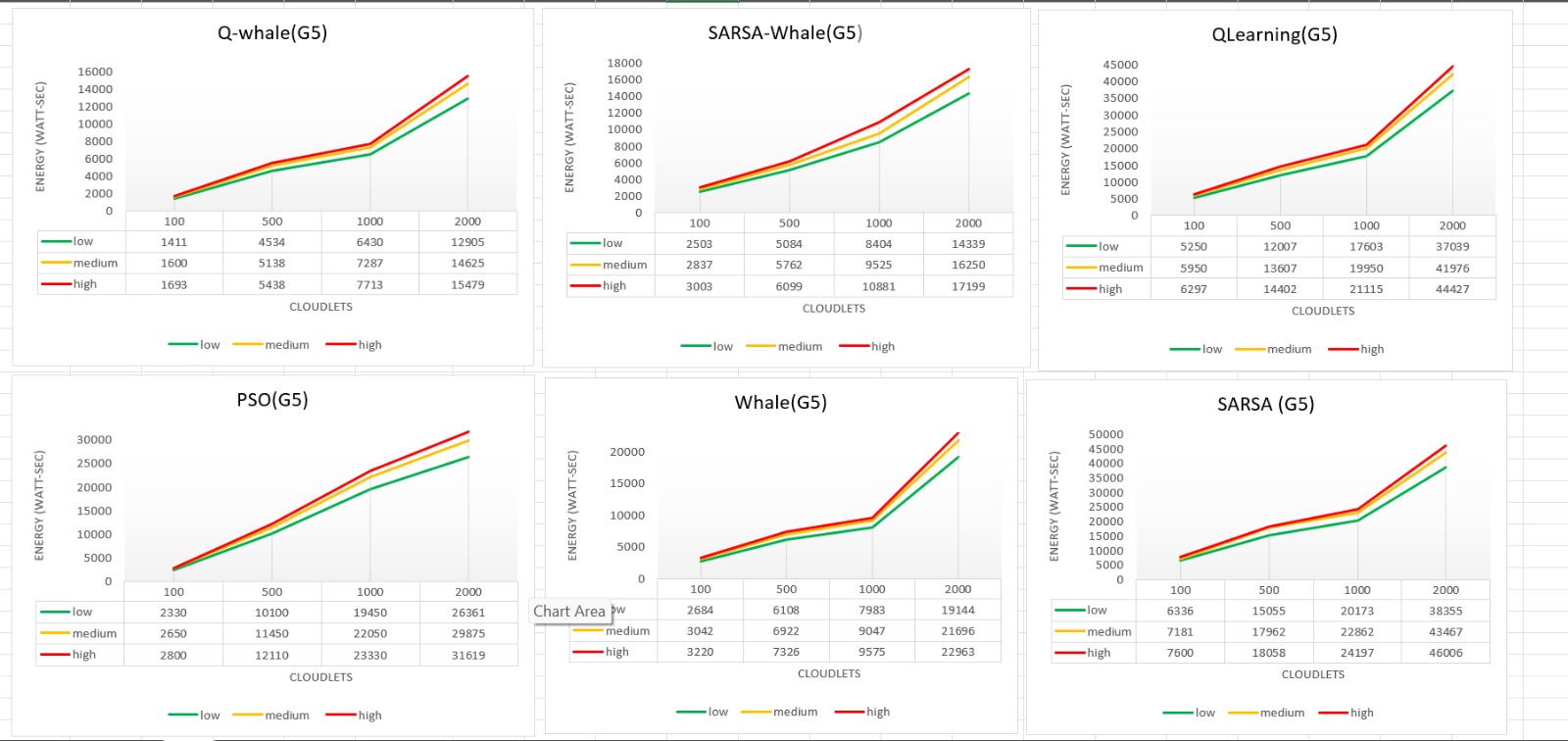
* **CPU Utilization Levels:** 30%, 40%, and 50%
* This scenario depicts the VMs under light load conditions, using only a small fraction of their available CPU resources.

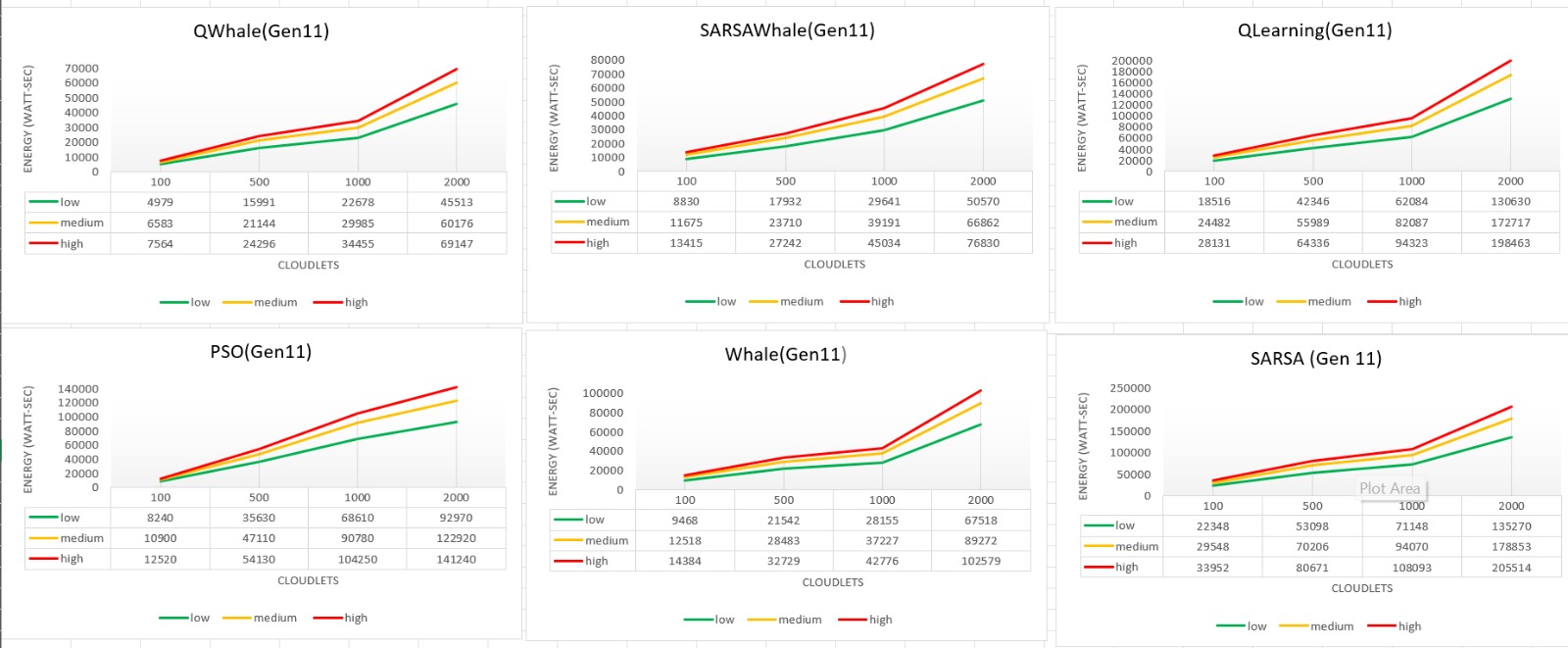
**Summary Table with Utilization Levels**

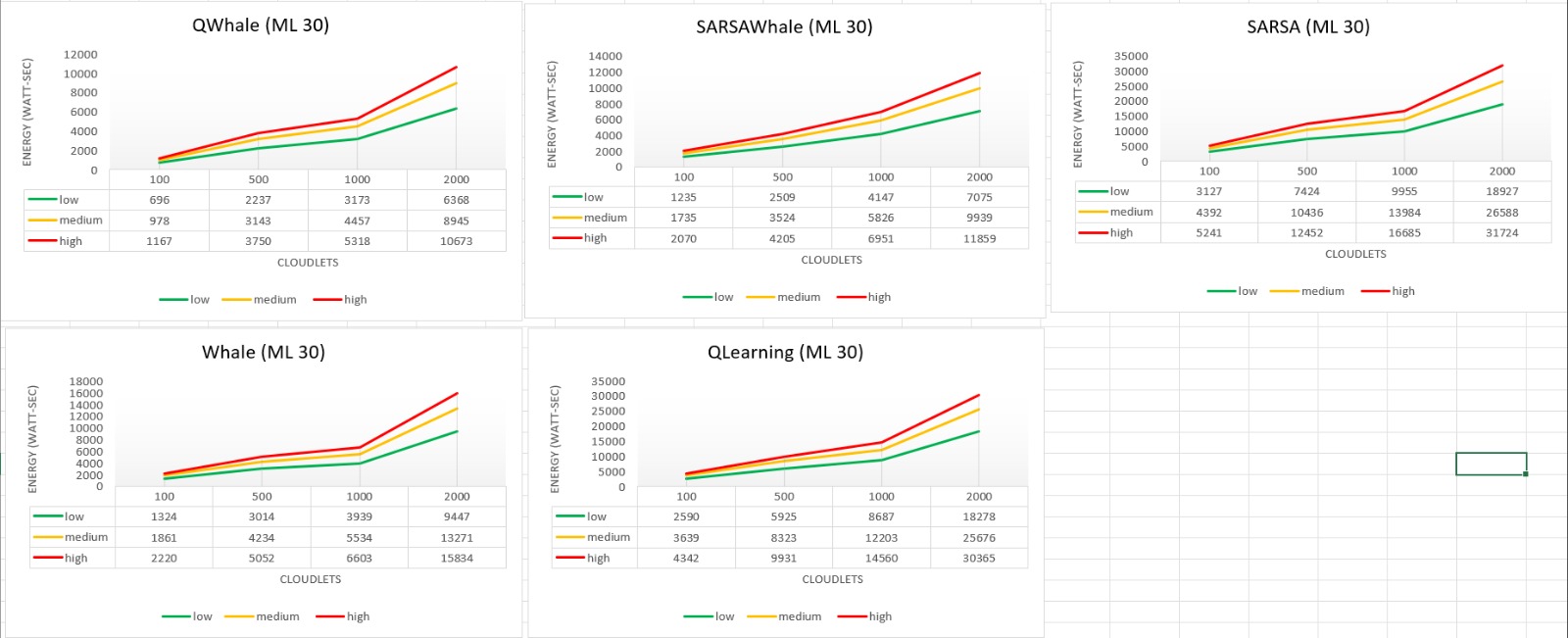
|  |  |
| --- | --- |
| **Utilization Category** | **CPU Utilization Levels (%)** |
| High Utilization | 80, 90, 100 |
| Medium Utilization | 60, 70, 80 |
| Low Utilization | 30, 40, 50 |

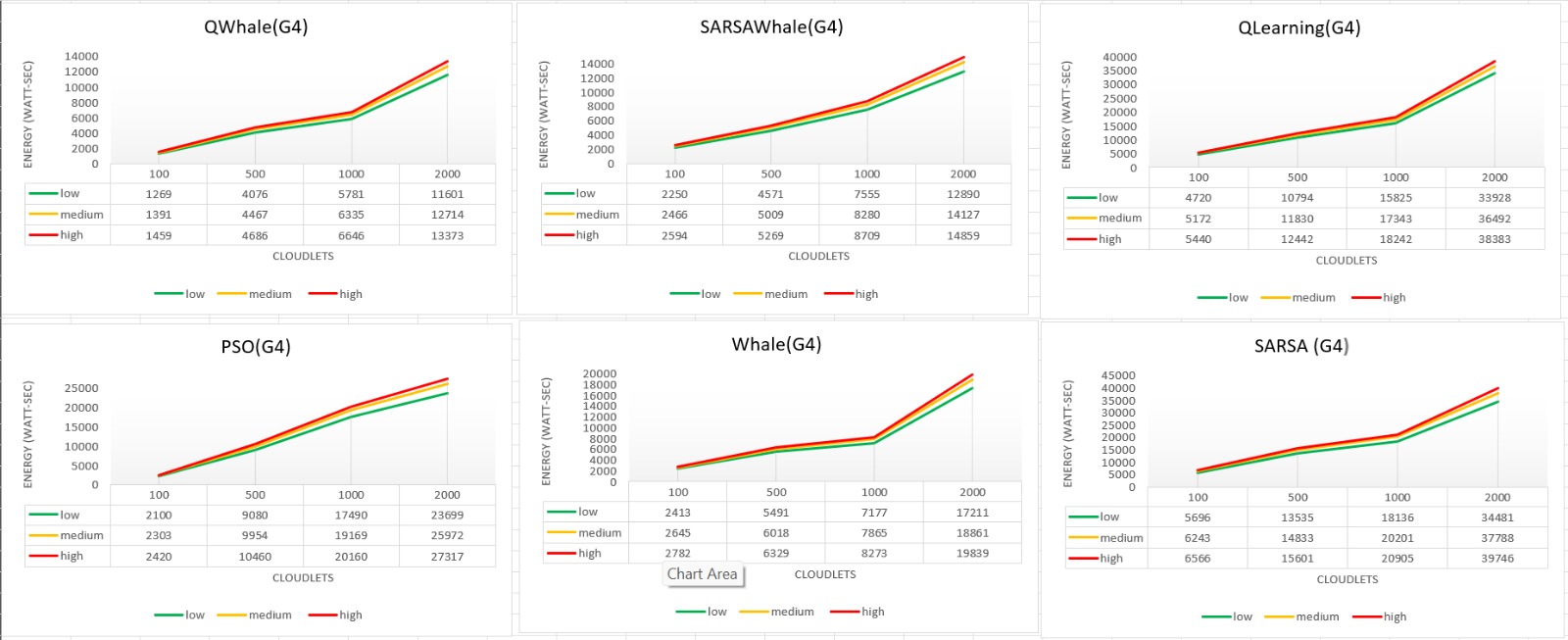
These utilization levels can be used in the simulation to assess how different workloads impact the performance of the data center. By running simulations under these three scenarios, you can observe how the system behaves under various load conditions, which can help in understanding the performance, resource allocation, and potential bottlenecks within the data center.

**Energy Consumption=Power(watt)\*Makespan(second)**

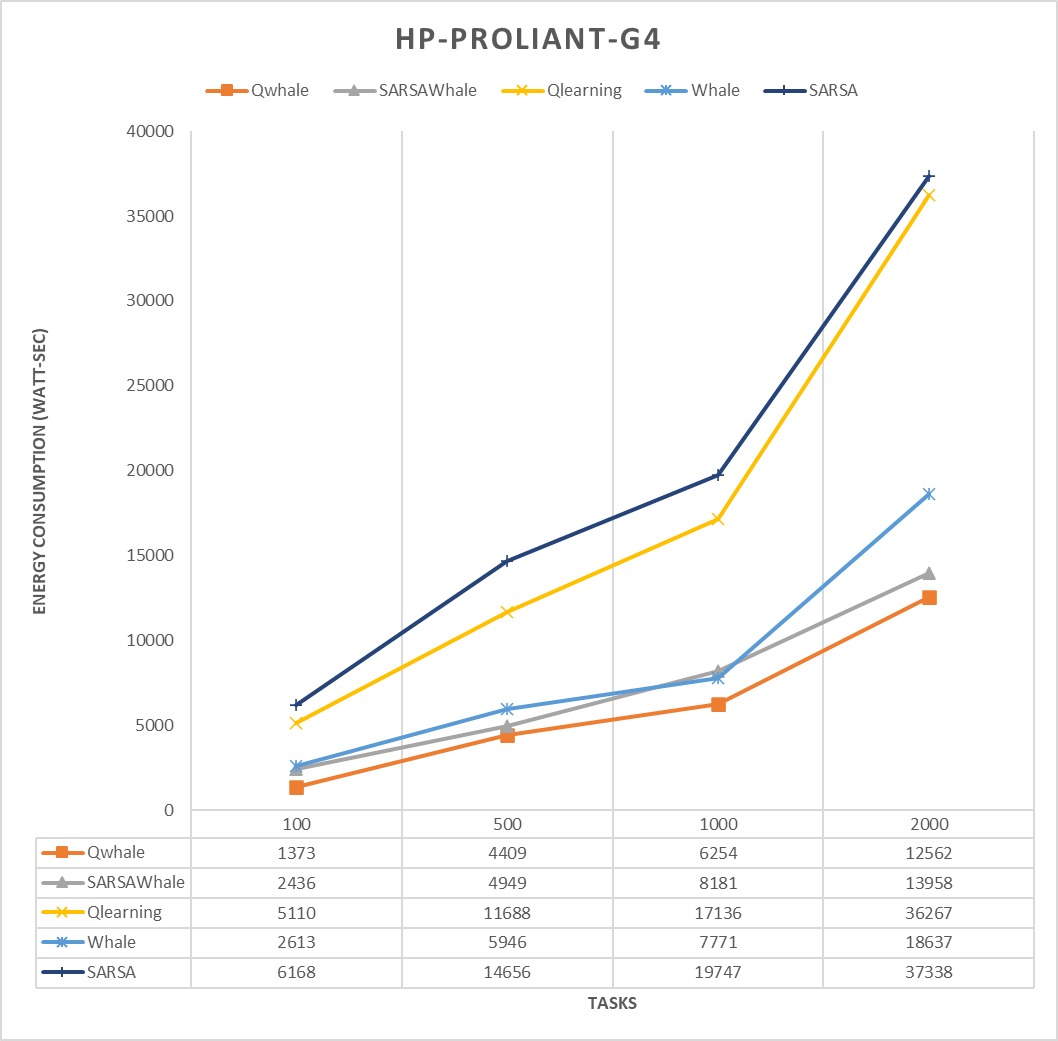


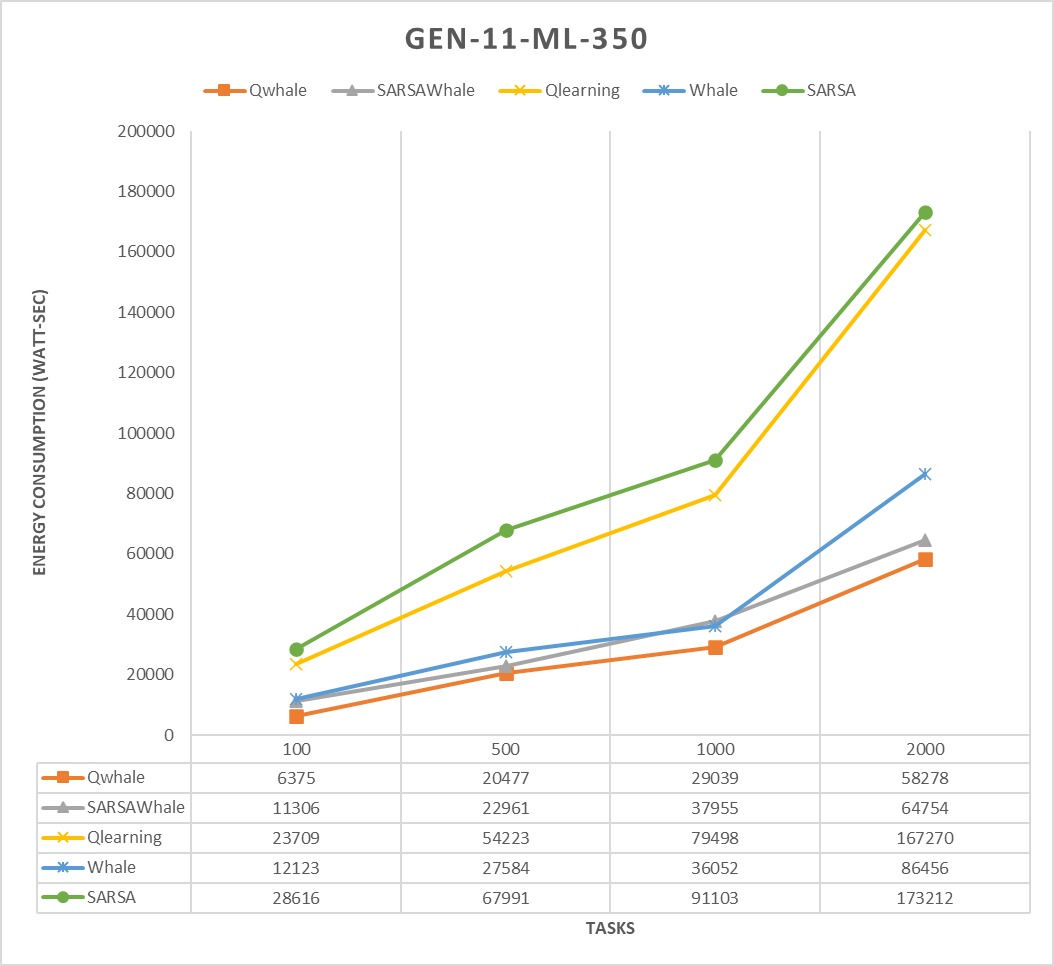
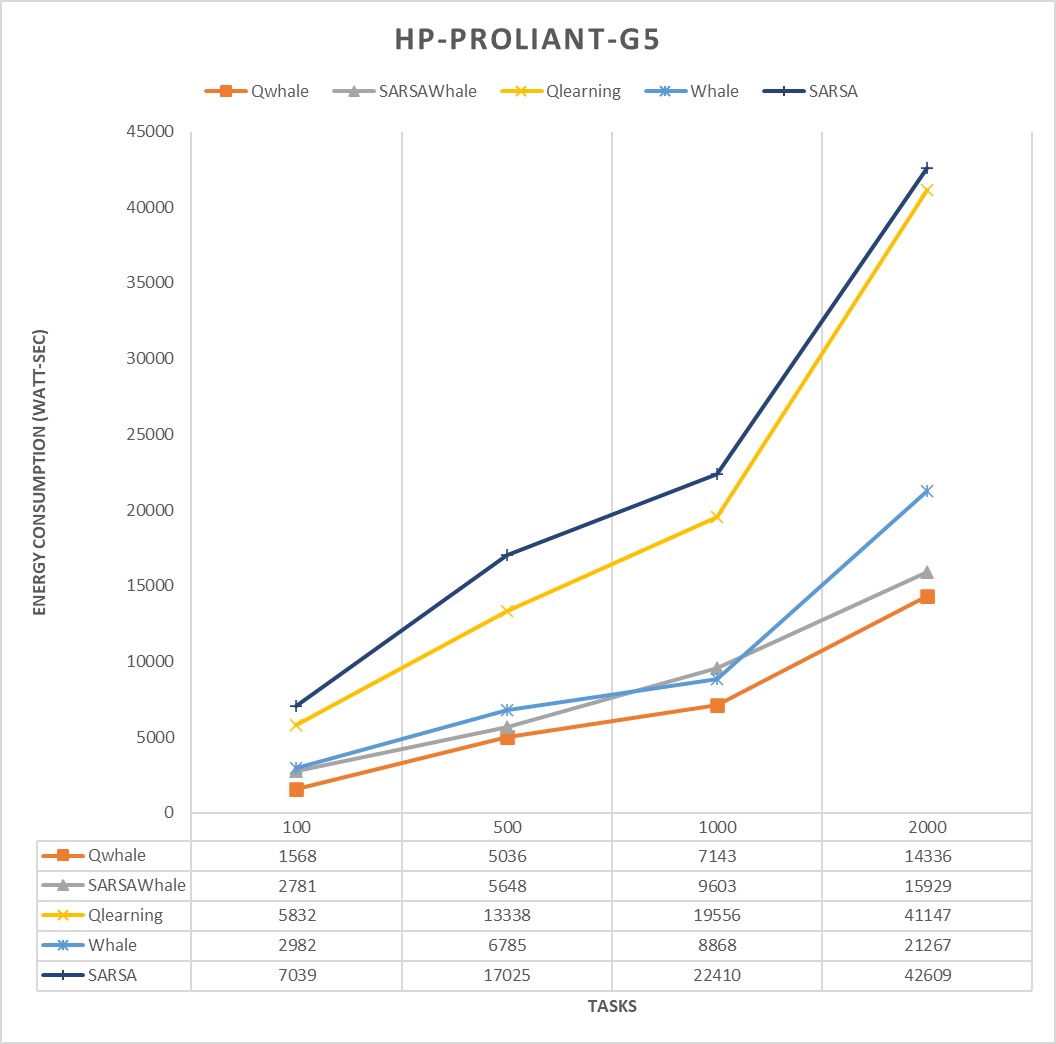


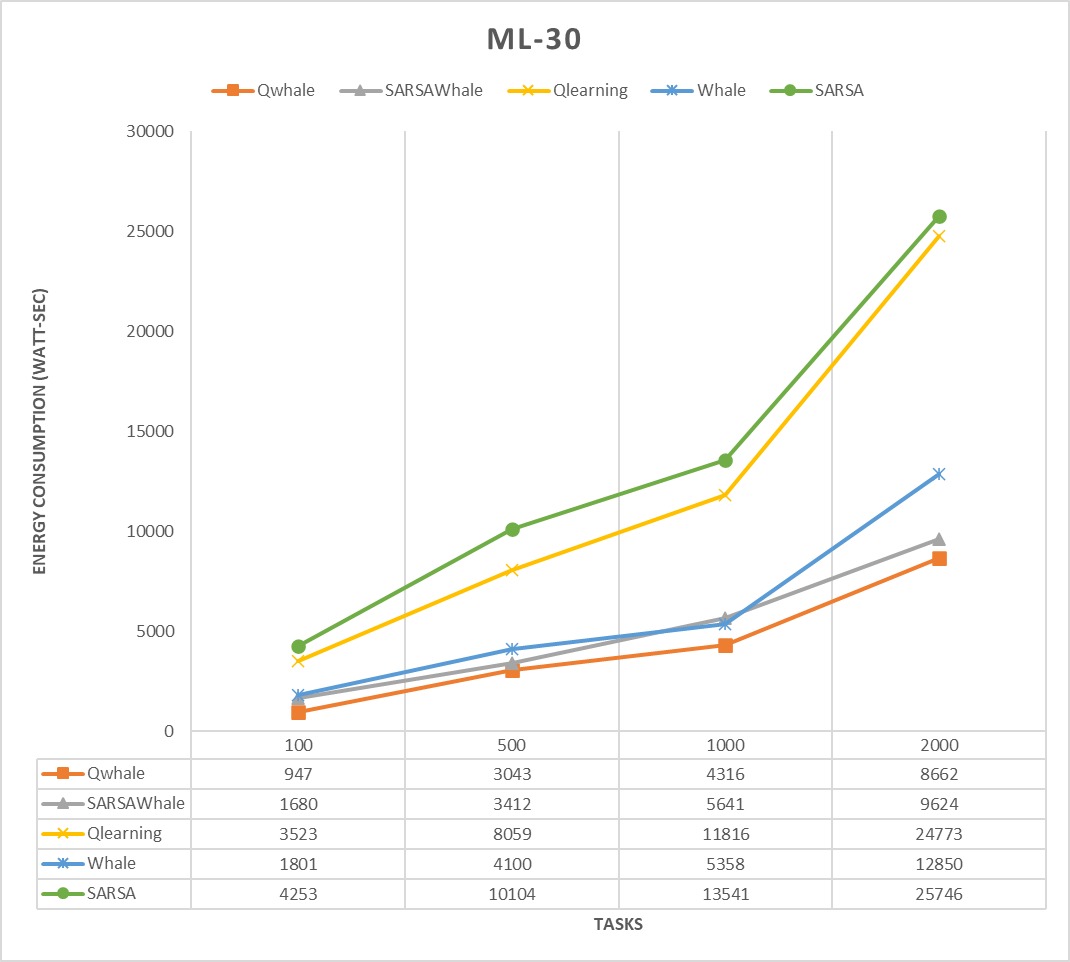




**Power-> Overall on the basis of utilization**





**Summary of Data**

* **Energy Consumption at 100 Tasks**:
  + Qwhale: 947
  + SARSAWhale: 1680
  + Qlearning: 3523
  + Whale: 1801
  + SARSA: 4253
* **Energy Consumption at 500 Tasks**:
  + Qwhale: 3043
  + SARSAWhale: 3412
  + Qlearning: 8059
  + Whale: 4100
  + SARSA: 10104
* **Energy Consumption at 1000 Tasks**:
  + Qwhale: 4316
  + SARSAWhale: 5641
  + Qlearning: 11816
  + Whale: 5358
  + SARSA: 13541
* **Energy Consumption at 2000 Tasks**:
  + Qwhale: 8662
  + SARSAWhale: 9624
  + Qlearning: 24773
  + Whale: 12850
  + SARSA: 25746

**Analysis**

1. **Energy Efficiency**:
   * **Qwhale** consistently shows the lowest energy consumption across all tasks, indicating it is the most energy-efficient algorithm.
   * **SARSA** shows the highest energy consumption, making it the least energy-efficient algorithm among those compared.
2. **Trends with Increasing Tasks**:
   * All algorithms show an increasing trend in energy consumption as the number of tasks increases.
   * The rate of increase varies, with Qlearning and SARSA showing more significant increases compared to the others.
3. **Comparative Performance**:
   * **Whale and SARSAWhale** show intermediate energy consumption levels, with Whale generally being slightly more energy-efficient than SARSAWhale.
   * **Qlearning** and **SARSA** show a steep increase in energy consumption, especially noticeable at 1000 and 2000 tasks.

**Observations**

* **Qwhale**: The most energy-efficient, making it suitable for scenarios where energy consumption is a critical factor.
* **SARSA**: The least energy-efficient, which might be suitable for scenarios where energy consumption is less of a concern but perhaps other factors like performance are prioritized.
* **Qlearning**: While it is less energy-efficient, it might offer benefits in terms of learning efficiency or accuracy, which would need further analysis.

Overall, the chart provides a clear comparison of energy consumption trends among different algorithms, highlighting the trade-offs between energy efficiency and the potential need for evaluating other performance metrics.

**Energy=Power\*Makespan**

**Detailed Analysis**

To delve deeper into understanding the performance of each algorithm, let's interpret how power and makespan contribute to the energy consumption:

1. **Power**: This refers to the average power consumption of the algorithm during its execution.
2. **Makespan**: This refers to the total time taken by the algorithm to complete the tasks.

Given this, an algorithm with lower energy consumption is either more power-efficient, has a shorter makespan, or both.

**Evaluation Based on Energy Formula**

**Qwhale:**

* **Energy Consumption**: The lowest across all task levels.
* **Inference**: Indicates high power efficiency and/or low makespan, making it ideal for energy-constrained environments.

**SARSAWhale:**

* **Energy Consumption**: Moderately low, higher than Qwhale but lower than Qlearning and SARSA.
* **Inference**: Balanced performance, suggesting decent power efficiency and makespan.

**Qlearning:**

* **Energy Consumption**: High, with a steep increase as tasks increase.
* **Inference**: Likely due to higher power consumption and/or longer makespan. This algorithm might prioritize other aspects like accuracy or learning efficiency over energy consumption.

**Whale:**

* **Energy Consumption**: Intermediate, better than SARSA and Qlearning but not as efficient as Qwhale.
* **Inference**: Suggests a balance, with reasonable power consumption and makespan.

**SARSA:**

* **Energy Consumption**: The highest, especially noticeable at higher task levels.
* **Inference**: Indicates high power consumption and/or long makespan, making it less suitable for energy-efficient scenarios.

**Recommendations**

* **Qwhale**: Best suited for scenarios where minimizing energy consumption is crucial.
* **SARSA**: Might be suitable in scenarios where other performance metrics (accuracy, robustness, etc.) are more critical than energy efficiency.
* **Qlearning**: If energy consumption is not a primary concern, Qlearning might offer benefits in learning performance or other metrics.
* **Whale and SARSAWhale**: Provide a middle ground, offering a trade-off between energy efficiency and other potential benefits.

The energy consumption chart, along with the energy formula, provides insights into the efficiency of each algorithm. For applications where energy efficiency is paramount, Qwhale stands out as the best choice. However, depending on the specific requirements and constraints of your tasks, other algorithms like Qlearning, Whale, or SARSA might be considered for their respective strengths.

**Conclusion:**

In conclusion, the integration of Q-learning and SARSA with the Whale Optimization Algorithm (WOA) has demonstrated significant improvements in solving complex optimization problems compared to conventional WOA and other benchmark algorithms. Our integrated variants consistently outperformed across various instances, achieving optimal solutions in several cases and demonstrating competitive performance close to optimal solutions on average.

Moreover, these integrated approaches streamlined the optimization process, significantly reducing tuning times. Detailed analysis of exploration and exploitation graphs revealed consistent convergence patterns with smaller variations and occurrences in our integrated variants, indicating potential for enhanced problem-solving capabilities.

Looking ahead, further validation and parameterization of results obtained from exploration and exploitation graphs are essential. Standardized metrics for comparison and incorporation into reinforcement learning agents' learning processes hold promise for advancing the optimization capabilities of metaheuristic algorithms. Overall, these findings underscore the potential of integrating reinforcement learning techniques with metaheuristic algorithms for effectively tackling complex optimization challenges.

In addition to the current advancements, future endeavors should explore the potential of leveraging SARSA's capability to store experiences for further improving the performance of the integrated SARSA Whale Optimization Algorithm (WOA). By incorporating this feature, the SARSA WOA variant could effectively learn from past experiences, enabling it to adapt more efficiently to varying optimization landscapes. This iterative process of learning and adaptation holds promise for enhancing the robustness and effectiveness of the SARSA-integrated WOA in solving complex optimization problems. Furthermore, continued research into parameterization of results obtained from exploration and exploitation graphs will be crucial in providing a standardized metric for comparison and further advancing the optimization capabilities of metaheuristic algorithms. Overall, these future directions highlight the potential for continuous refinement and improvement of SARSA-integrated WOA, ultimately contributing to more efficient and effective solutions for challenging optimization tasks.

**References:**

[1\*] <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>

[2\*] U. K. Jena, P. K. Das, and M. R. Kabat, “Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment,” Journal of King Saud University - Computer and Information Sciences, vol. 34, no. 6, pp. 2332–2342, Jun. 2022, doi: 10.1016/j.jksuci.2020.01.012.

1. Shaw, R., Howley, E., & Barrett, E. (2022, July 1). Applying Reinforcement Learning towards automating energy efficient virtual machine consolidation in cloud data centers. Information Systems. <https://doi.org/10.1016/j.is.2021.101722>
2. Hassan, A., Abdullah, S., Zamli, K. Z., & Razali, R. (2023, September 18). *Q-learning whale optimization algorithm for test suite generation with constraints support*. Neural Computing & Applications. [Q-learning whale optimization algorithm for test suite ... - Springer](https://link.springer.com/article/10.1007/s00521-023-09000-2)
3. Li, Y., Wang, H., Fan, J., & Geng, Y. (2022, December 27). *A novel Q-learning algorithm based on improved whale optimization algorithm for path planning*. PloS One. [A novel Q-learning algorithm based on improved whale ... - PLOS](https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0279438)
4. Natesan, G., & Chokkalingam, A. (2019, October 17). Multi-Objective Task Scheduling Using Hybrid Whale Genetic Optimization Algorithm in Heterogeneous Computing Environment. Wireless Personal Communications. [Multi-Objective Task Scheduling Using Hybrid Whale Genetic ... - Springer](https://link.springer.com/article/10.1007/s11277-019-06817-w)
5. Arvindhan, M., & Dhanaraj, R. K. (2023, November 6). Dynamic Q-Learning-Based Optimized Load Balancing Technique in Cloud. Journal of Mobile Information Systems. <https://doi.org/10.1155/2023/7250267>
6. Du, Z., Peng, C., Yoshinaga, T., & Wu, C. (2023, July 28). A Q-Learning-Based Load Balancing Method for Real-Time Task Processing in Edge-Cloud Networks. Electronics. <https://doi.org/10.3390/electronics12153254>
7. Luo, J., Chen, H., Heidari, A. A., Xu, Y., Zhang, Q., & Li, C. (2019, September 1). Multi-strategy boosted mutative whale-inspired optimization approaches. Applied Mathematical Modelling. <https://doi.org/10.1016/j.apm.2019.03.046>
8. Sharma, S., & Pandey, N. K. (2024, February 1). Multi-Faceted Job Scheduling Optimization Using Q-learning With ABC In Cloud Environment. International Journal of Computing and Digital System/International Journal of Computing and Digital Systems. <https://doi.org/10.12785/ijcds/150142>
9. Hamad, Q. S., Samma, H., & Suandi, S. A. (2023, January 5). Q-Learning based Metaheuristic Optimization Algorithms: A short review and perspectives. Research Square (Research Square). <https://doi.org/10.21203/rs.3.rs-1950095/v1>
10. Xiu, X., Li, J., Long, Y., & Wu, W. (2023, May 10). MRLCC: an adaptive cloud task scheduling method based on meta reinforcement learning. Journal of Cloud Computing. <https://doi.org/10.1186/s13677-023-00440-8>
11. Ding, D., Fan, X., Zhao, Y., Kang, K., Yin, Q., & Zeng, J. (2020, July 1). Q-learning based dynamic task scheduling for energy-efficient cloud computing. Future Generation Computer Systems. <https://doi.org/10.1016/j.future.2020.02.018>
12. Tran, C., Bui, T., & Pham, T. D. (2022, January 29). Virtual machine migration policy for multi-tier application in cloud computing based on Q-learning algorithm. Computing. <https://doi.org/10.1007/s00607-021-01047-0>
13. Assia, S., Abbassi, I. E., Barkany, A. E., Darcherif, M., & Biyaali, A. E. (2020, April 14). *Green Scheduling of Jobs and Flexible Periods of Maintenance in a Two-Machine Flowshop to Minimize Makespan, a Measure of Service Level and Total Energy Consumption*. Advances in Operations Research. <https://doi.org/10.1155/2020/9732563>
14. Xing, L., Li, J., Cai, Z., & Hou, F. (2023, April 30). *Evolutionary Optimization of Energy Consumption and Makespan of Workflow Execution in Clouds*. Mathematics. <https://doi.org/10.3390/math11092126>
15. Trivedi, D. K. S. S. A. M. C. (2021, July 26). Energy Aware Scheduling of Tasks in Cloud environment. <https://www.tojqi.net/index.php/journal/article/view/3376>
16. Pawlish, M., Varde, A. S., & Robila, S. A. (2012, June 1). Analyzing utilization rates in data centers for optimizing energy management. <https://doi.org/10.1109/igcc.2012.6322248>
17. Shaw, R., Howley, E., & Barrett, E. (2018, December 1). A Predictive Anti-Correlated Virtual Machine Placement Algorithm for Green Cloud Computing. <https://doi.org/10.1109/ucc.2018.00035>
18. Naz, I., Naaz, S., Agarwal, P., Alankar, B., Siddiqui, F., & Ali, J. (2023, May 5). A Genetic Algorithm-Based Virtual Machine Allocation Policy for Load Balancing Using Actual Asymmetric Workload Traces. Symmetry. <https://doi.org/10.3390/sym15051025>
19. Mangalampalli, S., Karri, G. R., & Köse, U. (2023, February 1). Multi objective trust aware task scheduling algorithm in cloud computing using whale optimization. Journal of King Saud University. Computer and Information Sciences/Maǧalaẗ Ǧamʼaẗ Al-malīk Saud : Ùlm Al-ḥasib Wa Al-maʼlumat. <https://doi.org/10.1016/j.jksuci.2023.01.016>