The SARSAWhale algorithm integrates the Whale Optimization Algorithm (WOA) with SARSA reinforcement learning to enhance task scheduling in dynamic computing environments. In this hybrid approach, WOA drives the exploration of the solution space, generating possible scheduling configurations by mimicking the bubble-net feeding behavior of whales. These configurations are evaluated based on critical performance metrics such as makespan, energy consumption, and resource efficiency to determine their suitability.

SARSA, an on-policy reinforcement learning algorithm, complements WOA by continuously updating the action-value function based on real-time feedback from the environment. After each scheduling decision, SARSA receives a reward based on the system's performance, such as task completion time or resource utilization. This reward guides SARSA in refining its action choices, allowing it to favor those that have historically led to more efficient outcomes.

By incorporating SARSA's learning capabilities, the algorithm not only explores but also adapts based on past experiences. This dynamic learning enables the SARSAWhale algorithm to improve task scheduling decisions over time, adjusting to varying workloads and resource constraints. Unlike traditional scheduling techniques, which may be static or heuristic-based, the SARSAWhale approach actively learns and optimizes, leading to improved system performance, adaptability, and resource management in cloud and data center environments.

This synergy of WOA's global search and SARSA's learning-based refinement allows the SARSAWhale algorithm to outperform conventional methods in terms of flexibility and efficiency, positioning it as a promising solution for real-time task scheduling in complex, evolving computational systems.

The SARSAWhale and QWhale algorithms both leverage the strengths of reinforcement learning and the Whale Optimization Algorithm (WOA) to enhance task scheduling in dynamic computing environments. While SARSAWhale integrates SARSA for on-policy learning, QWhale uses Q-learning, which relies on off-policy learning to influence scheduling decisions. In QWhale, WOA explores potential

The SARSAWhale algorithm merges the on-policy reinforcement learning capabilities of SARSA with the exploratory nature of the Whale Optimization Algorithm (WOA) to tackle task scheduling in dynamic computing environments. In this hybrid approach, WOA generates potential scheduling solutions by navigating the solution space, evaluating these based on key metrics like makespan, resource utilization, and deadlines. SARSA, an on-policy learning technique, plays a crucial role in refining this process by continuously updating the action-value function during exploration. SARSA learns from the current policy and evaluates the efficacy of each generated solution, guiding WOA's exploration towards optimal outcomes by adjusting task scheduling decisions based on real-time feedback. The interaction between SARSA's learning process and WOA's optimization capabilities ensures dynamic adaptability, enabling the algorithm to efficiently respond to changing conditions and achieve better results over time.

The SARSAWhale algorithm offers several distinct advantages over conventional task scheduling algorithms and alternative hybrid methodologies:

a. **Dynamic Learning Adaptability:** SARSAWhale excels in dynamically adjusting to environmental changes by leveraging SARSA's on-policy learning approach. This allows the algorithm to continually refine its policies based on ongoing experiences, effectively adapting to shifting workloads and varying resource conditions in real-time.

b. **Integrated Policy Optimization:** Unlike traditional methods that may rely solely on exploration or exploitation, SARSAWhale integrates WOA’s global exploration capabilities with SARSA’s real-time policy adjustments. This integration ensures that the algorithm not only explores new potential solutions but also optimizes policies based on current operational feedback.

c. **Enhanced Decision-Making Efficiency:** SARSAWhale’s real-time decision-making process benefits from SARSA’s immediate feedback mechanism. This leads to more efficient decision-making compared to methods that update policies less frequently or rely on delayed feedback, improving overall task scheduling performance.

d. **Versatility in Scheduling Strategies:** The SARSAWhale algorithm can adapt its scheduling strategies to a wide range of scenarios by leveraging SARSA’s flexibility in policy updates. This versatility allows it to handle diverse scheduling problems, including those with complex constraints and dynamic resource availability.

e. **Robust Exploration of Solution Space:** By combining WOA’s robust exploration capabilities with SARSA’s on-policy learning, SARSAWhale ensures a comprehensive search of the solution space. This approach minimizes the risk of premature convergence and enhances the likelihood of discovering high-quality solutions.

f. **Real-Time Adaptation to Task Priorities:** SARSAWhale’s real-time learning mechanism allows it to promptly adjust to changing task priorities. This ensures that the algorithm can effectively reallocate resources and adjust schedules to accommodate urgent tasks or shifting deadlines.

g. **Improved Convergence Speed:** The synergy between WOA’s exploration and SARSA’s on-policy learning leads to faster convergence towards optimal solutions. SARSAWhale's ability to refine policies in real-time accelerates the process of finding high-quality task schedules compared to methods that update policies less frequently.

h. **Experimental Validation of Robust Performance:** Experimental results in various computing environments demonstrate the SARSAWhale algorithm’s ability to outperform traditional scheduling methods and other hybrid approaches. Its effectiveness in improving task scheduling efficiency and resource utilization is well-documented through empirical studies.

In comparison, the QWhale algorithm, which integrates Q-learning with WOA, also provides significant advantages but differs in its use of off-policy learning to guide decision-making.

*Below points are similar to qwhale points (message from Aakarshit)

a. **Adaptability to Dynamic Environments:** The SARSAWhale algorithm adeptly navigates the dynamic landscape of computing environments by merging the exploratory nature of WOA with the on-policy learning of SARSA. This adaptability allows the algorithm to react swiftly to workload fluctuations, resource changes, and shifting task priorities in real-time, maintaining efficiency under evolving conditions.

b. **Efficient Exploration and Exploitation:** By fusing WOA's exploration capabilities with SARSA's policy-driven exploitation, the SARSAWhale algorithm ensures a balanced exploration of the solution space. WOA uncovers promising regions, while SARSA refines decisions in real-time based on immediate feedback, facilitating convergence towards superior task scheduling solutions.

c. **Optimization of Multiple Objectives:** The SARSAWhale algorithm is proficient at optimizing multiple objectives simultaneously, such as minimizing makespan, enhancing resource utilization, and meeting deadlines. This multi-objective approach provides comprehensive task scheduling optimization, improving system performance across various metrics.

d. **Learning from Past Experiences:** Utilizing SARSA's on-policy learning, the SARSAWhale algorithm improves its decision-making process through continuous updates based on current experiences. By iteratively adjusting task scheduling policies during the exploration phase, it refines its approach over time, resulting in more informed and efficient scheduling solutions.

e. **Effective Resource Utilization:** With a focus on maximizing resource utilization, SARSAWhale efficiently allocates tasks across available computing resources. By integrating WOA's optimization strengths with SARSA's learning mechanism, it ensures optimal task-resource assignments, minimizing idle time and enhancing overall system efficiency.

f. **Scalability and Robustness:** The scalability and robustness of the SARSAWhale algorithm make it suitable for diverse and large-scale computing environments, including cloud systems. Its ability to handle complex scheduling challenges involving numerous tasks and resources ensures that it remains efficient and effective regardless of the scale of operation.

g. **Experimental Validation:** Empirical experiments conducted in dynamic computing environments validate the efficacy of the SARSAWhale algorithm. Demonstrating measurable improvements in resource utilization, reduction in makespan, and adherence to task deadlines, these results highlight its superiority over traditional scheduling methods and alternative hybrid algorithms.