Metaheuristic Algorithms with Applications

www.maa.reapress.com

Metaheur. Algor. Appl. Vol. 1, No. 4 (2024) 440–464.

Paper Type: Original Article

World Hyper-Heuristic: A Reinforcement Learning Approach for Dynamic Exploration–Exploitation in Cloud Resource Management

Adel Cheshmeh*

Department of Computer Engineering, Faculty of Computer Engineering, Sharif University of Technology, Tehran, Iran; isigma@yahoo.com.

Citation:

Received: 11 May 2024 *Revised: 28 August 2024* *Accepted: 17 October 2024*	Cheshmeh, A. (2026). World hyper-heuristic: A reinforcement learning approach for dynamic exploration–exploitation in cloud resource management. *Metaheuristic algorithms with applications, 1*(4), 440-464.

Abstract

Cloud computing resource management presents a formidable optimization challenge characterized by dynamic and unpredictable workloads, heterogeneous resource pools, competing multi-objective trade-offs among energy consumption, Service Level Agreement (SLA) compliance, resource utilization, makespan, and operational cost, as well as the NP-hard combinatorial nature of core sub-problems including Virtual Machine (VM) placement, task scheduling, and dynamic resource scaling. Existing approaches suffer from well-documented limitations: static heuristics such as First Fit Decreasing (FFD) and Round-Robin (RR) lack adaptability to workload fluctuations; single metaheuristics including Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) exhibit premature convergence and maintain a fixed exploration–exploitation balance; and pure Reinforcement Learning (RL) methods, while adaptive, are sample-inefficient and transfer poorly across workload types. In this paper, we propose World Hyper-Heuristic (WHH) for Cloud Resource Management (World-CRM), an extended adaptation of the WHH originally introduced by Daliri et al. [1] in expert systems with applications. World-CRM adapts the World's novel RL-based dynamic exploration–exploitation switching mechanism to cloud environments through five key innovations: 1) a cloud-aware 12-dimensional RL state representation encoding CPU utilization variance, memory fragmentation, SLA violation rate, and energy metrics, 2) a composite multi-objective fitness function with adaptive weight adjustment, 3) a cloud-specific Low-Level Heuristic (LLH) pool comprising 12 operators spanning VM placement, task scheduling, and dynamic scaling, 4) an online sliding-window adaptation mechanism employing Kullback–Leibler divergence for workload shift detection and real-time policy updates, and 5) a multi-tier coordination architecture operating across infrastructure, platform, and application layers. World-CRM is evaluated on CloudSim Plus simulations with three real-world workload traces: Google Cluster Trace (CCT) 2019, Azure VM Traces 2019, and Bitbrains distributed data center across small (50 hosts), medium (200 hosts), and large (800 hosts) data center configurations. Results across 30 independent runs per configuration demonstrate that World-CRM achieves 23–31% energy reduction, 42–58% fewer SLA violations, 15–22% better resource utilization, and 18–27% lower makespan compared to the best-performing baselines, including NSGA-II-VM, PSO-Cloud, DQN-VMPlace, Modified Best Fit Decreasing (MBFD), and the original World algorithm. The online adaptation module maintains performance under workload shifts with decision latency below 300 milliseconds and enables 3–4× faster recovery compared to RL and evolutionary baselines. All improvements are statistically significant under the Wilcoxon rank-sum test with Bonferroni correction at $p < 0.05$.

Keywords: Hyper-heuristic, Cloud resource management, Reinforcement learning, Virtual machine placement, Task scheduling, Exploration–exploitation, Dynamic optimization, Energy efficiency.

1 | Introduction

Cloud computing has transformed modern information technology by offering elastic, on-demand access to a shared pool of configurable computing resources, including servers, storage, networks, and applications [2],

Corresponding Author: isigma@yahoo.com

https://doi.org/10.48313/maa.v1i4.109

[3]. Major cloud providers such as Amazon Web Services, Microsoft Azure, and Google Cloud Platform collectively operate millions of physical servers across geographically distributed data centers, serving billions of user requests daily. The global cloud computing market is projected to exceed USD 1.2 trillion by 2027, driven by the rapid adoption of cloud-native architectures, microservices, and edge computing paradigms [4]. The economic and environmental impact of these large-scale data centers is substantial: data centers account for approximately 1–1.5% of global electricity consumption [5], making energy-efficient resource management not only a technical imperative but also an environmental and economic one.

At the heart of efficient cloud operation lies the resource management problem: the challenge of dynamically mapping computational tasks and Virtual Machines (VMs) to physical infrastructure while simultaneously optimizing multiple competing objectives. This problem encompasses several tightly coupled sub-problems: 1) Virtual Machine Placement (VMP), which determines the assignment of VMs to Physical Machines (PMs) to optimize resource utilization and energy consumption [6], 2) Task Scheduling, which assigns incoming computational tasks to available VMs to minimize completion time and meet deadlines [7], 3) Dynamic Resource Scaling, which adjusts the number and size of allocated resources in response to workload fluctuations [8], and 4) VM Consolidation and Load Balancing, which migrates VMs to reduce the number of active hosts and distribute load evenly [9]. Each of these sub-problems is individually NP-hard, and their interdependencies compound the overall complexity.

The challenges inherent in cloud resource management are multifaceted. First, cloud workloads are dynamic and unpredictable: demand patterns exhibit diurnal cycles, flash crowds, seasonal variations, and stochastic bursts [10], [11]. Second, modern data centers house heterogeneous resources PMs with varying CPU architectures, memory capacities, storage types, and network bandwidths, requiring algorithms that can handle diverse resource dimensions. Third, the optimization landscape involves multi-objective trade-offs: minimizing energy consumption may conflict with minimizing response latency, and maximizing resource utilization may increase the risk of Service Level Agreement (SLA) violations [12]. Fourth, production cloud systems demand real-time decision-making: resource allocation decisions must be computed within milliseconds to seconds, precluding computationally expensive offline optimization. Fifth, cloud environments are non-stationary: optimal strategies change as workload characteristics evolve, hardware degrades, and tenant compositions shift.

A variety of approaches have been proposed to address these challenges, each with notable limitations. Static heuristics such as First Fit Decreasing (FFD), Best Fit Decreasing (BFD), and Round-Robin (RR) offer rapid computation and implementation simplicity but are inherently non-adaptive, applying the same strategy regardless of current system state or workload characteristics [6]. Single metaheuristics, including Genetic Algorithms (GA) [13], Particle Swarm Optimization (PSO) [14], Ant Colony Optimization (ACO) [15], and Differential Evolution (DE) [16], produce higher-quality solutions but maintain a fixed exploration–exploitation balance determined by static parameter settings, leading to premature convergence or excessive diversification depending on problem characteristics. Multi-objective evolutionary algorithms such as NSGA-II [17] and MOEA/D [18] effectively discover trade-off surfaces but are computationally expensive, often requiring thousands of fitness evaluations per decision, making them unsuitable for real-time cloud management. Reinforcement Learning (RL) approaches, including Deep Q-Networks (DQN) for VM placement [19], actor-critic methods for auto-scaling [20], and multi-agent RL for distributed scheduling [21] offer the promise of adaptive decision-making but suffer from sample inefficiency, requiring extensive training episodes; reward engineering complexity; the sim-to-real gap; and poor transferability across different workload types and data center configurations.

Hyper-heuristics offer a compelling alternative paradigm. Defined as "search methods that operate on a search space of heuristics rather than directly on a search space of solutions" [22], hyper-heuristics provide a level of abstraction above individual heuristics, enabling the automated selection or generation of the most appropriate heuristic for a given problem state. Such a "search over search space" approach is particularly attractive for cloud resource management, where the optimal strategy depends on the current system state

and workload characteristics. However, existing cloud-specific hyper-heuristics, such as VMM-HHGT [23] and SMPHAM [24], employ relatively simple selection mechanisms random selection, roulette wheel, or simple adaptive rules without leveraging the power of RL to learn sophisticated, state-dependent selection policies that dynamically balance exploration and exploitation.

The World Hyper-Heuristic (WHH), recently proposed by Daliri et al. [1], represents a significant advance in hyper-heuristic methodology. World introduces a novel RL-based mechanism that dynamically switches between exploration and exploitation strategies drawn from an infinite pool of metaheuristics. Operating through two key steps Rewarding (evaluating Low-Level Heuristic (LLH) performance and assigning credit) and Selection (using a learned RL policy to choose the most promising heuristic for the current search state) World demonstrated superior performance on standard benchmark functions, discrete NP-hard problems including the Travelling Salesman Problem (TSP) with 10,000 real cities (achieving a shortest path of 4.33×10^5), and continuous NP-hard engineering design problems, consistently outperforming state-of-the-art algorithms. However, the original World was evaluated exclusively on offline, static optimization problems and has not been adapted for dynamic, online optimization domains such as cloud resource management.

This paper addresses this research gap by proposing WHH for Cloud Resource Management (World-CRM), an extended adaptation of the World algorithm specifically tailored for multi-objective cloud resource management. World-CRM preserves the core RL-based reward and selection mechanism of the original World while introducing five key innovations that enable effective operation in the cloud computing domain. The principal contributions of this paper are:

I. World-CRM framework: an adaptation of the WHH [1] for multi-objective cloud resource management, introducing a cloud-aware 12-dimensional RL state representation encoding CPU utilization variance, memory Fragmentation Index (FI), SLA violation rate, energy consumption per task, and other cloud-specific metrics, replacing the generic optimization metrics of the original World.

II. Cloud-Specific LLH pool: a pool of 12 cloud-tailored LLH operators spanning VM placement (FFD, BFD, PSO-Migrate, Energy-Aware Placement), task scheduling (SJF, ACO-Schedule, GA-Crossover, DE-Balance), and dynamic scaling (GWO-Scale, WOA-Consolidate, SA-Consolidate, Threshold-Reactive), with dynamic composite heuristic generation following the World's infinite pool concept.

III. Online sliding-window adaptation: a real-time policy adaptation mechanism using Kullback–Leibler divergence for workload shift detection, enabling World-CRM to maintain performance under non-stationary workload patterns without full retraining.

IV. Multi-Tier coordination architecture: an architecture operating across infrastructure (PM-level), platform (VM-level), and application (task-level) tiers with inter-tier reward signal sharing for coordinated optimization.

V. Comprehensive empirical evaluation: Evaluation on three real-world cloud workload traces (Google Cluster Trace (CCT) 2019, Azure VM Traces 2019, Bitbrains) across three data center scales (50, 200, 800 hosts) with statistical significance testing, demonstrating consistent superiority over eight baseline methods.

The remainder of this paper is organized as follows. Section 2 reviews related work in cloud resource management optimization, metaheuristic and RL-based cloud approaches, and hyper-heuristic methodologies. Section 3 presents the World-CRM methodology in detail, including problem formulation, architecture, state representation, LLH pool design, RL-based selection mechanism, and online adaptation. Section 4 describes the experimental setup. Section 5 presents and analyzes the results. Section 6 discusses findings, implications, and limitations. Section 7 concludes the paper.

2 | Related Work

2.1 | Cloud Resource Management Optimization

Cloud resource management has attracted substantial research attention over the past decade. Beloglazov et al. [12] proposed optimal online deterministic algorithms and adaptive heuristics for energy- and performance-efficient dynamic consolidation of VMs, establishing the Modified Best Fit Decreasing (MBFD) heuristic as a widely adopted baseline. Their work introduced the concept of upper and lower CPU utilization thresholds for triggering VM migration, a principle that remains foundational in energy-aware cloud management. Rodriguez and Buyya [7] addressed deadline-based resource provisioning and scheduling for scientific workflows on Infrastructure-as-a-Service clouds, formulating the problem as a particle-based optimization with deadline constraints. Lorido-Botran et al. [8] surveyed auto-scaling techniques in cloud environments, categorizing approaches into reactive (threshold-based), proactive (predictive), and hybrid methods. Farahnakian et al. [9] proposed energy-aware VM consolidation using utilization prediction models based on linear regression and k-nearest neighbors, demonstrating significant energy savings through proactive migration.

Multi-objective optimization for cloud resource management has received growing interest. Li et al. [25] applied NSGA-II to simultaneously optimize energy consumption and quality of service in VM placement, producing Pareto-optimal trade-off solutions. Ferdaus et al. [26] proposed a multi-objective VM placement algorithm based on ACO to minimize both resource wastage and power consumption. More recently, Zhou et al. [27] applied MOEA/D with decomposition for multi-objective workflow scheduling in cloud environments, optimizing makespan, cost, and reliability simultaneously. While these approaches discover effective trade-off surfaces, their computational overhead (typically requiring thousands of fitness evaluations per generation) limits their applicability to real-time decision-making in production cloud environments.

2.2 | Metaheuristic Approaches in Cloud Computing

Nature-inspired metaheuristics have been extensively applied to cloud resource management sub-problems. Zhan et al. [13] applied a modified GA to cloud task scheduling, using precedence-preserving crossover and mutation operators to minimize makespan under resource constraints. Mejahed and Elshrkawey [14] developed a multi-objective PSO variant for VM allocation that balances energy consumption and resource utilization through adaptive inertia weight adjustment. Chen and Zhang [15] proposed an ACO-based approach for workflow scheduling in cloud environments, modeling the task-to-VM assignment as an ant routing problem with pheromone updates reflecting both time and cost objectives. DE has been applied to load balancing problems in cloud data centers, exploiting its effective mutation and crossover operations for continuous optimization [16], [28].

More recently, novel metaheuristics have been introduced for cloud applications. Sahoo, Sahoo, and Kumar [29] applied the Grey Wolf Optimizer (GWO) [30] to cloud task scheduling, demonstrating competitive performance with reduced parameter tuning compared to GA and PSO. The Whale Optimization Algorithm (WOA) [31] has been applied to energy-aware VM placement, mimicking humpback whale bubble-net attacking strategies. The Sine Cosine Algorithm (SCA) [32] and Harris Hawks Optimization (HHO) [33] have also been adapted for various cloud optimization tasks. Despite their individual strengths, all single metaheuristics share a fundamental limitation: they maintain a fixed exploration–exploitation balance determined by their algorithmic structure and parameter settings, which may not be optimal across different problem states and workload conditions [22]. Moreover, each metaheuristic requires problem-specific parameter tuning, and the No Free Lunch theorem [34] guarantees that no single metaheuristic can universally outperform all others across all problem instances.

2.3 | Reinforcement Learning for Cloud Management

RL-based approaches have gained momentum due to their ability to learn adaptive policies from interaction with the environment. Tuli et al. [19] proposed an RL-based framework for energy-efficient VM placement using DQN, training an agent to select placement actions that minimize energy consumption while respecting SLA constraints. Arabnejad et al. [20] developed an actor-critic RL approach for auto-scaling containerized cloud applications, learning policies that adjust resource allocation in response to predicted demand changes. Mao et al. [21] introduced a multi-resource cluster scheduling approach using deep RL, training a policy network to make scheduling decisions across multiple resource dimensions simultaneously.

Model-based RL approaches have also been explored for proactive scaling. Cheng et al. [35] used model-based RL with learned environment dynamics to predict future workload demands and pre-allocate resources. Multi-agent RL has been applied to distributed scheduling, where each scheduling agent learns to coordinate with others through shared reward signals [36]. Despite promising results, RL-based cloud management approaches face several challenges: 1) sample inefficiency: deep RL methods typically require millions of interactions to learn effective policies, 2) reward engineering: designing reward functions that balance multiple cloud objectives without unintended consequences is non-trivial, 3) the sim-to-real gap: policies learned in simulation may not transfer effectively to production environments, and 4) training instability: neural network-based RL can exhibit catastrophic forgetting and training divergence, particularly in non-stationary environments [37].

2.4 | Hyper-Heuristic Approaches

Hyper-heuristics provide a higher-level abstraction for optimization, searching within a space of heuristics rather than the solution space itself. Burke et al. [22] provided a comprehensive survey, taxonomizing hyper-heuristics into selection hyper-heuristics (which choose from a set of existing heuristics) and generation hyper-heuristics (which construct new heuristics from components). Selection mechanisms include simple random, random descent, roulette wheel, choice function, and various RL-inspired approaches. Cowling et al. [38] pioneered hyper-heuristic research with their choice function-based approach for scheduling, demonstrating that automated heuristic selection could outperform individual heuristics. Subsequent work has applied hyper-heuristics to diverse combinatorial optimization domains including bin packing [39], timetabling [40], and vehicle routing [41].

Cloud-specific hyper-heuristics have emerged as a growing research area. Zhan et al. [23] proposed VMM-HHGT, a hyper-heuristic approach for handling hierarchy in cloud data centers that uses generative and selection mechanisms for VM management across multi-tier data center architectures. Tan et al. [24] introduced SMPHAM, a self-adaptive microservice placement hyper-heuristic with adaptive mechanisms for dynamic cloud-edge environments. Drake et al. [42] applied a selection hyper-heuristic with a choice function to the cloud VM consolidation problem, selecting among several consolidation heuristics based on their recent performance. However, existing cloud hyper-heuristics predominantly employ relatively simple selection mechanisms: random selection, roulette wheel proportional to recent performance, or basic adaptive rules, without leveraging sophisticated RL techniques for learning state-dependent selection policies that dynamically balance exploration of underexplored heuristics and exploitation of known high-performing ones.

2.5 | The World Hyper-Heuristic

The WHH, proposed by Daliri et al. [1], represents a paradigmatic advance in hyper-heuristic methodology by integrating RL-based learning directly into the heuristic selection process. The World algorithm operates through two core mechanisms. In the Rewarding step, each LLH is evaluated based on the solution improvement it produces when applied to the current search state. The reward is computed as a function of the relative fitness improvement, normalized by the number of applications, providing a fair comparison across frequently and infrequently used heuristics. In the Selection step, the algorithm uses a learned RL

policy, specifically, a Q-learning-based approach to select the next LLH to apply. The Q-values encode the expected future utility of applying a particular LLH in a given search state, enabling the algorithm to learn which heuristics are most effective under different conditions.

A distinguishing feature of the World algorithm is its infinite pool concept: rather than being limited to a fixed set of LLHs, the World dynamically generates new composite heuristics by combining operators from different base metaheuristics. This mechanism effectively provides an unbounded search space of possible strategies, from which the RL mechanism selects the most promising ones. Daliri et al. [1] evaluated the World on three categories of problems: 1) standard benchmark functions as artificial NP-hard problems, 2) discrete NP-hard problems, including the TSP with benchmark instances up to 10,000 real cities, where the World achieved a shortest path length of 4.33×10^5, and 3) continuous NP-hard engineering design problems including pressure vessel, welded beam, and tension/compression spring design. In all categories, the World consistently outperformed state-of-the-art algorithms, including standalone metaheuristics and existing hyper-heuristics.

Despite its demonstrated effectiveness, the original World algorithm was designed and evaluated exclusively for offline, static optimization problems, scenarios where the problem instance is fixed, and the algorithm iterates until convergence. It has not been adapted for online, dynamic optimization environments where problem characteristics change over time, decisions must be made in real time, and the solution landscape is non-stationary. Cloud resource management is a quintessential example of such a dynamic domain. This paper addresses this gap by extending and adapting the WHH for the cloud computing domain.

3 | Methodology

3.1 | Problem Formulation

We formulate the cloud resource management problem as a multi-objective dynamic optimization problem. Consider a data center with the following components:

I. A set of N heterogeneous PMs, PM = $\{pm_1, pm_2, \ldots, pm_N\}$, where each pm_j has capacities $cap_{cpu}(j)$, $cap_{mem}(j)$, $cap_{sto}(j)$, and $cap_{bw}(j)$ for CPU, memory, storage, and bandwidth, respectively.

II. A set of M VMs, VM = $\{vm_1, vm_2, \ldots, vm_M\}$, where each vm_i has resource requirements $req_{cpu}(i)$, $req_{mem}(i)$, $req_{sto}(i)$, and $req_{bw}(i)$.

III. A set of K tasks T = $\{t_1, t_2, \ldots, t_K\}$, where each task t_k has processing time p_k, deadline d_k, resource requirements, and priority level π_k.

IV. A discrete time horizon divided into decision epochs $\tau = \{1, 2, \ldots, T_{max}\}$.

The resource management problem is decomposed into three interdependent sub-problems:

P1 (VM placement). Find a mapping φ: VM → PM that assigns each VM to a PM, minimizing energy consumption:

$$\text{fenergy} = \Sigma j = 1N\, Ej(uj) \cdot \text{active}(j), \quad (1)$$

where $E_j(u_j)$ is the power consumption model of PM j as a function of its CPU utilization u_j, and active(j) ∈ {0, 1} indicates whether host j is powered on. Following the SPECpower benchmark model, we define:

$$Ej(uj) = \text{Pidle}(j) + \big(\text{Pmax}(j) - \text{Pidle}(j)\big) \cdot uj, \quad (2)$$

where $P_{idle}(j)$ and $P_{max}(j)$ are the idle and maximum power consumption of host j, and the CPU utilization is computed as:

$$uj = \Sigma vm \in \varphi - 1(j)\, \text{reqcpu}(vm) / \text{capcpu}(j). \quad (3)$$

P2 (Task scheduling). Find an assignment σ: T → VM that maps each task to a VM, minimizing the makespan:

$$\text{fmakespan} = \max i \in \{1, \ldots, K\} Ci, \quad \textbf{(4)}$$

where C_i is the completion time of task t_i. The completion time depends on the task processing time, VM computational capacity, and queueing delays from other tasks assigned to the same VM.

P3 (Dynamic scaling). At each decision epoch τ, find scaling decisions $s(\tau) \in$ {scale-up, scale-down, maintain} for each resource tier, minimizing:

$$\text{fSLA} = \Sigma\tau\, \text{SLA}_{\text{violations}(\tau)} + \lambda \cdot \text{cost}(s(\tau)), \quad \textbf{(5)}$$

where SLA_violations(τ) counts the number of SLA breaches at epoch τ, cost(s(τ)) captures the monetary cost of scaling actions, and λ is a trade-off parameter.

The combined multi-objective fitness function integrates all objectives:

$$F(x) = w1 \cdot \hat{f}\text{energy} + w2 \cdot \hat{f}\text{SLA} + w3 \cdot (1 - \hat{f}\text{utilization}) + w4 \cdot \hat{f}\text{makespan} + w5 \cdot \hat{f}\text{cost}, \quad \textbf{(6)}$$

where f̂ denotes min-max normalized objective values, and w_1, w_2, w_3, w_4, w_5 are adaptive weights satisfying $\Sigma_i w_i = 1$. The utilization objective is negated because it is to be maximized while all others are minimized.

The optimization is subject to the following constraints:

$$C1: \Sigma vm \in \varphi - 1(pmj)\text{reqcpu}(vm) \leq \text{capcpu}(pmj), \quad \forall j \in \{1, \ldots, N\}, \quad \textbf{(7)}$$

$$C2: \Sigma vm \in \varphi - 1(pmj)\text{reqmem}(vm) \leq \text{capmem}(pmj), \quad \forall j \in \{1, \ldots, N\}, \quad \textbf{(8)}$$

$$C3: Ck \leq dk, \forall tk \in T \text{ with hard deadlines}, \quad \textbf{(9)}$$

$$C4: \text{response}_{\text{time}(vmi)} \leq \text{SLA}_{\text{threshold}(vmi)}, \quad \forall i \in \{1, \ldots, M\}, \quad \textbf{(10)}$$

$$C5: 0.2 \leq uj \leq 0.85, \quad \forall j \text{ where active}(j) = 1, \quad \textbf{(11)}$$

Constraints C1 and C2 enforce CPU and memory capacity limits on each PM. Constraint C3 ensures hard deadline compliance for time-critical tasks. Constraint C4 enforces response time SLA thresholds. Constraint C5 maintains utilization within operational bounds to prevent both underutilization waste and hotspot-induced performance degradation.

3.2 | World-Cloud Resource Management Architecture Overview

World-CRM operates through a three-tier architecture that mirrors the hierarchical structure of cloud computing infrastructure, enabling coordinated optimization across different abstraction levels. Each tier maintains its own RL policy while sharing state information and reward signals with adjacent tiers.

Tier 1 (Infrastructure manager). Operates at the PM level, handling VM-to-PM placement decisions, server consolidation (migrating VMs to reduce the number of active hosts), and host power management (switching idle hosts to sleep mode). The Infrastructure Manager selects from placement-oriented LLHs (FFD, BFD, PSO-Migrate, Energy-Aware Placement) and consolidation-oriented LLHs (WOA-Consolidate, SA-Consolidate).

Tier 2 (Platform manager). Operates at the VM level, handling VM sizing (adjusting CPU and memory allocations), VM migration for load balancing, and resource reservation management. The Platform Manager selects from scheduling and balancing LLHs (DE-Balance, GA-Crossover) and coordinates with Tier 1 to ensure migration decisions align with placement objectives.

Tier 3 (Application manager). Operates at the task level, handling task-to-VM assignment, task queue management, and priority-based scheduling. The Application Manager selects from scheduling LLHs (SJF, ACO-Schedule) and scaling LLHs (GWO-Scale, Threshold-Reactive).

Inter-Tier Coordination: The three tiers coordinate through a shared reward mechanism. When an LLH applied at any tier produces a fitness improvement, the reward signal is propagated to all tiers, scaled by a coordination factor $\beta_{coord} = 0.3$ for non-originating tiers. Such reward propagation ensures that improvements at one tier positively influence selection policies at other tiers without creating circular dependencies. Additionally, state information flows upward (task-level metrics aggregate to VM-level, which aggregate to PM-level), providing each tier with a holistic view of the system.

3.3 | Cloud-Aware RL State Representation

A critical innovation of World-CRM over the original World algorithm is the redefinition of the RL state space using cloud-specific metrics. While the original World uses generic optimization metrics (current fitness value, improvement rate, stagnation count), cloud resource management requires a state representation that captures the multifaceted nature of data center operations. We define a 12-dimensional state vector $s(\tau)$ at each decision epoch τ:

$$s(\tau) = [\bar{u}cpu, \sigma^2 cpu, \bar{u}mem, \sigma^2 mem, \bar{u}net, FI, HAR, SVR, PTC, WT, EPT, CR]. \quad (12)$$

The components of this state vector are defined in *Table 1*.

Table 1. Cloud-aware state vector components for World-CRM.

Symbol	Component	Definition	Range
$\bar{u}_{cpu}$	Mean CPU utilization	Average CPU utilization across all active hosts	[0, 1]
σ^2_{cpu}	CPU utilization variance	Variance of CPU utilization across active hosts (load imbalance indicator)	[0, 1]
$\bar{u}_{mem}$	Mean memory utilization	Average memory utilization across all active hosts	[0, 1]
σ^2_{mem}	Memory utilization variance	Variance of memory utilization across active hosts	[0, 1]
$\bar{u}_{net}$	Mean network bandwidth saturation	Average ratio of used to available bandwidth	[0, 1]
FI	Memory FI	1 − (largest contiguous free memory block / total free memory) across all hosts	[0, 1]
HAR	Active hosts ratio	Fraction of PMs currently powered on	[0, 1]
SVR	SLA violation rate	Fraction of VMs/tasks currently violating SLA thresholds	[0, 1]
PTC	Pending task count (normalized)	Number of tasks in queue divided by total task capacity	[0, 1]
WT	Workload trend	Slope of linear regression on CPU demand over last W epochs	[−1, 1]
EPT	Energy per task	Current energy consumption divided by task throughput (normalized)	[0, 1]
CR	Cost Rate	Current operational CR normalized by budget threshold	[0, 1]

Each continuous-valued state dimension is discretized into three levels {Low, Medium, High} using adaptive thresholds computed from running statistics (mean ± 0.5 standard deviation). This discretization yields a discrete state space of $3^{12} = 531{,}441$ possible states. To handle the high dimensionality and ensure adequate coverage with limited experience, we employ tile coding [43] with eight overlapping tilings, each with a different offset, producing a feature vector that provides smooth generalization across nearby states while maintaining sufficient discrimination.

The reward function at each decision epoch measures the improvement achieved by the selected LLH:

$$r(\tau) = \alpha \cdot \Delta Energy(\tau) + \beta \cdot \Delta SLA(\tau) + \gamma \cdot \Delta Utilization(\tau) + \delta \cdot \Delta Makespan(\tau) + \varepsilon \cdot \Delta Cost(\tau), \quad (13)$$

where $\Delta X(\tau) = (X(\tau-1) - X(\tau)) / X(\tau-1)$ represents the relative improvement in objective X from epoch $\tau-1$ to τ. Positive values indicate improvement (reduction in energy, SLA violations, makespan, cost, or increase in utilization); negative values indicate degradation. The coefficients $\alpha = \beta = \gamma = \delta = \varepsilon = 0.2$ are initialized equally and adjusted by the multi-objective weight adaptation mechanism described in Section 3.7.

3.4 | Cloud-Specific Low-Level Heuristic Pool

World-CRM maintains a pool of 12 base LLHs organized by their primary function (placement, scheduling, or scaling). Each LLH is a complete, self-contained optimization operator that can be applied to the current cloud state to produce an improved resource allocation. *Table 2* summarizes the complete LLH pool.

Table 2. Cloud-specific LLH pool for World-CRM.

ID	Name	Category	Description	Complexity
LLH1	FFD	Placement	Sort VMs by resource demand (descending), assign each to the first PM with sufficient capacity	O(M log M + MN)
LLH2	BFD	Placement	Sort VMs by resource demand (descending), assign each to the PM with the least remaining capacity after placement (tightest fit)	O(M log M + MN)
LLH3	PSO-Migrate	Placement	PSO for VM migration decisions; particles encode migration plans, velocity updates balance current placement and global best	O(P · M · N)
LLH4	Energy-Aware Placement	Placement	Assign each VM to the PM that minimizes marginal energy increase: $\text{argmin}_j [E_j(u_j + \Delta u) - E_j(u_j)]$	O(MN)
LLH5	Shortest Job First (SJF)	Scheduling	Sort tasks by estimated processing time (ascending), assign to the VM with the earliest available time slot	O(K log K + KM)
LLH6	ACO-Schedule	Scheduling	ACO for task-to-VM routing; ants construct task assignments with pheromone trails reflecting historical quality	O(A · K · M)
LLH7	GA-Crossover	Scheduling	Genetic algorithm crossover on task permutation schedules; two-point crossover with repair operator for precedence constraints	O(G · Pop · K)
LLH8	DE-Balance	Scheduling	DE for load balancing; mutation vectors redistribute task assignments to minimize utilization variance across VMs	O(G · Pop · M)
LLH9	GWO-Scale	Scaling	GWO for dynamic scaling decisions; alpha, beta, delta wolves encode scaling actions, pack position encodes target resource levels	O(G · Pop · N)
LLH10	WOA-Consolidate	Scaling	WOA for server consolidation; bubble-net attacking strategy identifies VMs to migrate for host deactivation	O(G · Pop · M)
LLH11	SA-Consolidate	Scaling	Simulated annealing for gradual consolidation; accepts migration moves based on Boltzmann probability with cooling schedule	O(I · M)
LLH12	Threshold-Reactive	Scaling	Rule-based reactive scaling with adaptive thresholds; thresholds adjust based on recent SLA performance and utilization trends	O(N + M)

Note: P = number of PSO particles, A = number of ants, G = number of generations, Pop = population size, I = number of SA iterations. For real-time operation, metaheuristic LLHs are run with reduced parameters: P = A = Pop = 20, G = 5, I = 50.

3.4.1 | Infinite pool generation

Following the original World algorithm's infinite pool concept [1], World-CRM dynamically generates composite LLHs by combining operators from different base LLHs. A composite LLH applies a sequence of operators: for example, "FFD placement → ACO scheduling → GWO scaling" creates a compound strategy that addresses all three sub-problems in a single decision epoch. The composite LLHs are generated by sampling one operator from each category (placement, scheduling, scaling) and concatenating them. This combination produces 4 × 4 × 4 = 64 possible three-component composites from the 12 base LLHs, and with additional partial composites (single- or dual-category) the effective pool size exceeds 100 strategies. New composites are generated on-demand when the RL policy's exploration phase selects a composite index not yet instantiated.

3.5 | RL-Based Rewarding and Selection

The core mechanism of World-CRM, inherited and adapted from the original WHH [1], consists of two interleaved steps: Rewarding and Selection.

Step 1 (Rewarding with credit assignment). After each decision epoch, the performance of each recently applied LLH is evaluated. The challenge lies in credit assignment: When multiple LLHs have been applied in

recent epochs, determining which LLH is responsible for observed improvements requires careful attribution. We employ a sliding-window credit assignment mechanism with a window of $W_{credit} = 10$ recent decisions:

$$\text{reward}(\text{LLHk}) = \Sigma i = 15\ \text{wi} \cdot \text{improvementi}(\text{LLHk}) / \text{applications}(\text{LLHk}, \text{Wcredit}), \quad (14)$$

where $\text{improvement}_i(\text{LLH}_k)$ is the total improvement in objective i across all epochs within the credit window where LLH_k was applied, and $\text{applications}(\text{LLH}_k, W_{credit})$ is the number of times LLH_k was applied within the window. This normalization prevents frequently applied LLHs from accumulating disproportionate reward and encourages exploration of underutilized heuristics.

Step 2 (Selection via Q-learning). The LLH selection policy is implemented as a Q-learning agent (Watkins and Dayan [44]) operating on the cloud-aware state space defined in Section 3.3. The Q-value update follows the standard temporal difference rule:

$$\text{Q}(\text{s}, \text{LLHk}) \leftarrow \text{Q}(\text{s}, \text{LLHk}) + \alpha\text{rl}\,[\text{r} + \gamma\text{rl} \cdot \text{maxLLH}'\text{Q}(\text{s}', \text{LLH}') - \text{Q}(\text{s}, \text{LLHk})], \quad (15)$$

where s is the current state, LLH_k is the applied heuristic, r is the reward from *Eq. (14)*, s' is the resulting next state, $\alpha_{rl} = 0.1$ is the learning rate, and $\gamma_{rl} = 0.95$ is the discount factor.

The selection policy uses an adaptive ε-greedy strategy:

$$\text{LLH} * = \text{argmaxk}\,\text{Q}(\text{s}, \text{LLHk}) \quad \text{with probability } (1 - \varepsilon), \quad (16)$$

$$\text{LLH} * = \text{Uniform}(\text{Pool}) \quad \text{with probability } \varepsilon, \quad (17)$$

The exploration rate ε decays geometrically from $\varepsilon_{init} = 0.3$ to $\varepsilon_{min} = 0.05$, with a decay factor of 0.995 per epoch during stable workload periods. However, ε is reset to $\varepsilon_{init} = 0.3$ when a workload shift is detected (Section 3.6), enabling the agent to re-explore the LLH pool under new conditions.

Algorithm 1. Presents the main World-CRM decision loop.

Algorithm 1: World-CRM Main Decision Loop
Input: Physical machines PM, initial VMs VM, task stream T, parameters (α_rl, γ_rl, ε_init, ε_min, ε_decay, W, τ_shift) Output: Resource allocation decisions over time horizon 1: Initialize Q-table Q(s, LLH_k) ← 0 for all states s and LLHs k 2: Initialize ε ← ε_init 3: Initialize sliding windows W_demand ← ∅, W_credit ← ∅ 4: Perform initial VM placement using FFD 5: for each decision epoch τ = 1, 2, ..., T_max do 6: // MONITORING PHASE 7: Collect current metrics from all PMs, VMs, and task queues 8: 9: // STATE EXTRACTION 10: Compute state vector s(τ) using Equation (12) and Table 1 11: Discretize s(τ) using tile coding with 8 tilings 12: 13: // WORKLOAD SHIFT DETECTION (Section 3.6) 14: Update W_demand with current demand distribution 15: if |W_demand| ≥ W then 16: Compute D_KL(P_current || P_historical) using Equation (18) 17: if D_KL > τ_shift then 18: ε ← ε_init // Reset exploration 19: Q ← 0.7 · Q // Decay Q-values 20: Log workload shift event at epoch τ 21: end if 22: end if 23: 24: // RL-BASED SELECTION (Section 3.5) 25: Generate random number p ∈ [0, 1] 26: if p < ε then 27: Select LLH* uniformly at random from Pool (exploration) 28: else 29: Select LLH* = argmax_k Q(s(τ), LLH_k) (exploitation) 30: end if 31: 32: // LLH EXECUTION 33: Apply LLH* to current cloud state 34: Execute resulting placement/scheduling/scaling decisions 35: 36: // REWARD COMPUTATION (Section 3.5) 37: Compute improvements ΔEnergy, ΔSLA, ΔUtilization, ΔMakespan, ΔCost 38: Compute reward r(τ) using Equation (13) 39: Update credit window W_credit 40: Compute reward(LLH*) using Equation (14) 41: 42: // Q-TABLE UPDATE 43: Observe next state s(τ + 1) 44: Update Q(s(τ), LLH*) using Equation (15) 45: 46: // WEIGHT ADAPTATION (Section 3.7) 47: Adapt multi-objective weights w_1, ..., w_5 based on constraint proximity 48: 49: // EXPLORATION DECAY 50: ε ← max(ε_min, ε · ε_decay) 51: end for

Algorithm 2. Details the rewarding mechanism with credit assignment.

Algorithm 2: Rewarding Mechanism with Credit Assignment
Input: Credit window W_credit of recent (epoch, LLH, improvements) tuples, objective weights w_1,...,w_5 Output: Reward value for the most recently applied LLH 1: Let LLH_current be the LLH applied at the current epoch 2: Let history ← entries in W_credit where LLH = LLH_current 3: if |history| = 0 then 4: return 0 // No history for this LLH 5: end if 6: 7: total_reward ← 0 8: for each objective i ∈ {energy, SLA, utilization, makespan, cost} do 9: improvement_sum ← 0 10: for each entry (τ_j, LLH_j, Δ_j) in history do 11: improvement_sum ← improvement_sum + Δ_j[i] 12: end for 13: total_reward ← total_reward + w_i · improvement_sum / |history| 14: end for 15: 16: // Penalty for constraint violations 17: if any constraint C1–C5 violated then 18: violation_penalty ← −0.5 · (number of violations / total constraints) 19: total_reward ← total_reward + violation_penalty 20: end if 21: 22: return total_reward

Algorithm 3. Adaptive selection with workload shift detection.

Algorithm 3: Adaptive Selection with Workload Shift Detection
Input: Current demand window W_current, historical demand window W_historical, threshold τ_shift, Q-table, current state s, current ε Output: Selected LLH, updated ε, updated Q-table 1: // WORKLOAD SHIFT DETECTION via KL Divergence 2: Compute empirical distribution P_current from W_current 3: Compute empirical distribution P_historical from W_historical 4: Add smoothing: P(x) ← (P(x) + δ_smooth) / (1 + |X| · δ_smooth), δ_smooth = 1e-6 5: D_KL ← Σ_x P_current(x) · log(P_current(x) / P_historical(x)) 6: 7: shift_detected ← false 8: if D_KL > τ_shift then 9: shift_detected ← true 10: ε ← ε_init // Reset to high exploration 11: Q(s, a) ← 0.7 · Q(s, a) for all s, a // Decay all Q-values 12: W_historical ← W_current // Update baseline 13: Log("Workload shift detected at current epoch, D_KL = " + D_KL) 14: end if 15: 16: // ADAPTIVE ε-GREEDY SELECTION 17: if shift_detected then 18: // During shift: boost exploration of diverse LLH categories 19: Select one LLH from each category {Placement, Scheduling, Scaling} 20: with probability 0.5 form composite LLH, else select single LLH 21: else 22: // Normal operation 23: p ← random(0, 1) 24: if p < ε then 25: LLH* ← random selection from Pool 26: else 27: LLH* ← argmax_k Q(s, LLH_k) 28: end if 29: end if 30: 31: return LLH*, ε, Q

3.6 | Online Sliding-Window Adaptation

Cloud workloads are inherently non-stationary, exhibiting diurnal patterns, flash crowds, seasonal variations, and unpredictable demand spikes [11]. Unlike the original World algorithm, which operates on static problem instances, World-CRM must adapt its LLH selection policy in real time as workload characteristics change. We achieve this through an online sliding-window adaptation mechanism based on Kullback–Leibler (KL) divergence.

At each decision epoch, we maintain two sliding windows of resource demand observations: a current window $W_{current}$ containing the most recent W = 50 epochs, and a historical baseline window $W_{historical}$ containing the W epochs prior to the current window. The empirical distribution of CPU demand levels is computed over each window, and the KL divergence between them measures the degree of distributional shift:

$$DKL(Pcurrent \,||\, Phistorical) = \Sigma x\, Pcurrent(x) \cdot \log(Pcurrent(x) / Phistorical(x)), \quad (18)$$

where $P_{current}$ and $P_{historical}$ are the empirical probability distributions of discretized CPU demand levels over the respective windows. Laplace smoothing ($\delta = 10^{-6}$) is applied to prevent division by zero.

When D_{KL} exceeds the shift threshold $\tau_{shift} = 0.5$, World-CRM triggers the following adaptation actions:

I. Exploration reset: the exploration rate ε is reset to its initial value εinit = 0.3, forcing the RL agent to broadly explore the LLH pool under the new workload conditions rather than relying on policies learned under the previous workload.

II. Q-Value decay: all Q-table values are multiplied by a decay factor of 0.7, partially preserving learned knowledge while reducing confidence in state-action valuations that may no longer be accurate: Q(s, a) ← 0.7 · Q(s, a) for all s, a.

III. Baseline update: the historical window is updated to reflect the new workload baseline, preventing repeated false-positive shift detections.

IV. Event logging: the workload shift event is logged with the epoch number and DKL value for post-hoc analysis.

This mechanism enables World-CRM to adapt to changing workload patterns without full retraining. The partial Q-value decay (rather than complete reset) preserves useful knowledge; for instance, the relative effectiveness of FFD versus BFD for placement is unlikely to change fundamentally even under a workload shift while allowing the policy to adjust to new optimality landscapes. The choice of $\tau_{shift} = 0.5$ was determined through sensitivity analysis, balancing responsiveness to genuine shifts against stability under normal workload variability.

3.7 | Multi-Objective Weight Adaptation

The weights $w_1, \ldots, w_5$ in the composite fitness function *Eq. (6)* are not static but adapt based on the proximity of each objective to its constraint boundaries. This adaptive weighting scheme ensures that the optimization focus shifts dynamically to the most critical objective at any given time:

$$\text{If SVR} > \theta\text{SLA: w2} \leftarrow \text{w2} + \Delta\text{w, normalize all weights,} \quad (19)$$

$$\text{If fenergy} > \text{budgetenergy: w1} \leftarrow \text{w1} + \Delta\text{w, normalize all weights,} \quad (20)$$

$$\text{If } \bar{u}\text{cpu} < 0.30\text{: w3} \leftarrow \text{w3} + \Delta\text{w, normalize all weights,} \quad (21)$$

where $\theta_{SLA} = 0.03$ (3% SLA violation threshold), $budget_{energy}$ is the energy budget for the current period, and $\Delta w = 0.05$ is the weight increment. After each adjustment, all weights are renormalized to sum to 1. If no constraints are in proximity, weights decay toward the uniform distribution ($w_i = 0.2$ for all i) at a rate of 0.01 per epoch. This mechanism ensures that, for example, during high-load periods when SLA violations spike, the RL agent prioritizes LLHs that improve SLA performance; during low-load periods, it shifts focus to energy-minimizing consolidation strategies.

3.8 | Computational Complexity Analysis

The per-epoch computational complexity of World-CRM consists of three components: 1) state extraction: $O(N + M + K)$ for aggregating metrics across PMs, VMs, and tasks, 2) LLH selection: $O(|Pool|)$ for Q-table lookup and argmax computation, and 3) LLH execution: varies by LLH, from $O(N + M)$ for Threshold-Reactive to $O(G \cdot Pop \cdot M \cdot N)$ for population-based metaheuristics with reduced parameters. The dominant cost is LLH execution. With reduced metaheuristic parameters ($G = 5$, $Pop = 20$), the worst-case per-epoch complexity is $O(5 \cdot 20 \cdot M \cdot N) = O(100MN)$. For the largest data center ($N = 800$, $M = 3200$), this yields approximately 2.56×10^8 operations per epoch, which modern processors complete in under 300 ms. The Q-table memory requirement is $O(|S| \cdot |Pool|)$. With tile coding (8 tilings × 531,441 states = 4,251,528 features) and $|Pool| \approx 100$, the total Q-table memory is approximately 425 million entries × 8 bytes ≈ 3.2 GB, feasible for modern servers. However, due to tile coding sparsity, the effective memory usage is substantially lower, typically under 500 MB.

4 | Experimental Setup

4.1 | Simulation Environment

All experiments are conducted using CloudSim Plus [45], a state-of-the-art Java-based cloud computing simulation framework that provides extensive modeling capabilities for data center infrastructure, VM lifecycle management, task scheduling, and power consumption modeling. CloudSim Plus extends the original CloudSim [46] with improved modularity, extensibility, and simulation accuracy. World-CRM is implemented as a set of custom scheduling and placement policies within the CloudSim Plus framework. We configure three data center scales to evaluate scalability:

Table 3. Data center configurations for experimental evaluation.

Parameter	Small	Medium	Large
Physical hosts	50	200	800
VMs (max)	200	800	3,200
Tasks (max)	1,000	5,000	20,000
Host CPU cores	2–8	2–8	2–8
Host RAM (GB)	4–32	4–32	4–32
Host storage (GB)	100–500	100–500	100–500
Host bandwidth (Gbps)	1–10	1–10	1–10

Physical hosts are modeled after HP ProLiant DL360 Gen10 (2 cores, 4 GB RAM) and DL380 Gen10 (8 cores, 32 GB RAM) server configurations, distributed uniformly across the data center. The power consumption model follows the SPECpower benchmark: P_{idle} = 175 W and P_{max} = 250 W, with linear

interpolation between idle and maximum based on CPU utilization. Four VM types are defined: Small (1 vCPU, 1 GB RAM), Medium (2 vCPU, 4 GB RAM), Large (4 vCPU, 8 GB RAM), and XLarge (8 vCPU, 16 GB RAM), with uniform random distribution across types.

4.2|Workload Traces

Three real-world cloud workload traces are used to ensure realistic and diverse evaluation:

4.2.1|Google cluster trace 2019

This publicly available dataset [10], [47] contains workload data from eight Google Borg cells over the month of May 2019, comprising approximately 2.4 TiB of trace data. We extract task resource requirements (CPU and memory requests), inter-arrival times, and task durations from a single representative cell. The trace includes approximately 660,000 jobs with diverse resource profiles, representing a mix of web-serving latency-sensitive workloads and batch processing throughput-oriented workloads. Tasks are replayed at the original inter-arrival rate to preserve temporal dynamics.

4.2.2|Azure virtual machine traces 2019 (Azure)

Released by Microsoft Research [11], this dataset contains 30 days of VM deployment data from Microsoft Azure's production infrastructure, comprising approximately 2.6 million VM requests. The traces capture VM creation times, lifetimes, requested VM sizes (CPU and memory), and deployment patterns. This trace represents a pure Infrastructure-as-a-Service (IaaS) workload with diverse VM sizes and highly variable lifetimes, ranging from minutes to weeks.

4.2.3|Bitbrains distributed data center (Bitbrains)

This trace from a managed hosting provider [48] contains performance metrics for approximately 1,750 VMs over 30 days with 5-minute sampling granularity. Metrics include CPU utilization, memory consumption, network throughput, and disk I/O. This trace represents enterprise application workloads with relatively stable long-running VMs but complex multi-dimensional resource consumption patterns.

4.3|Compared Methods

World-CRM is compared against nine baseline methods spanning static heuristics, metaheuristics, multi-objective evolutionary algorithms, RL approaches, and hyper-heuristic variants:

Table 4. Baseline methods for comparative evaluation.

Method	Category	Description
RR	Static heuristic	Circular VM placement and task assignment across all available hosts/VMs
FFD	Static heuristic	Standard bin-packing heuristic: sort by demand, assign to first-fitting host
MBFD	Energy-aware heuristic	Energy-aware BFD variant with upper/lower utilization thresholds [6]
PSO-Cloud	Metaheuristic	PSO for VM placement with 50 particles, 100 iterations
GA-Schedule	Metaheuristic	Genetic Algorithm for task scheduling with population 100, 50 generations, tournament selection
NSGA-II-VM	Multi-objective evolutionary	NSGA-II optimizing energy and SLA for VM placement, population 100, 50 generations
DQN-VMPlace	RL	Deep Q-Network for VM placement with 3-layer MLP, experience replay, target network (inspired by [19])
HH-Random	Hyper-heuristic	Same LLH pool as World-CRM but with uniform random LLH selection (no RL learning)
Original World	Hyper-heuristic	The original World algorithm [1] applied to cloud without cloud-specific adaptations (generic state, same LLH pool)

4.4 | Evaluation Metrics

Performance is evaluated across the following metrics: 1) energy consumption (kWh): total data center energy over the simulation period, computed as the integral of power consumption across all hosts, 2) SLA violation rate (%): percentage of VMs or tasks violating response time or availability SLAs, 3) resource utilization (%): average CPU and memory utilization of active hosts, weighted equally, 4) makespan (seconds): total completion time for the batch workload set, 5) operational cost ($): total cost comprising energy cost ($0.10/kWh), SLA penalty cost ($0.50 per violation), and VM migration cost ($0.01 per migration), 6) number of VM migrations: total migrations during the simulation, 7) decision latency (ms): wall-clock time for each resource management decision, 8) adaptation speed: number of epochs to recover performance after a synthetic workload shift.

4.5 | Parameter Settings

Table 5. World-CRM parameter configuration.

Parameter	Symbol	Value
Q-learning rate	α_{rl}	0.1
Discount factor	γ_{rl}	0.95
Initial exploration rate	ε_{init}	0.3
Minimum exploration rate	ε_{min}	0.05
Exploration decay factor	ε_{decay}	0.995
Demand sliding window size	W	50 epochs
Workload shift threshold	τ_{shift}	0.5
Q-value decay on shift		0.7
Credit assignment window	W_{credit}	10 epochs
Initial objective weights	w_1–w_5	0.2 each (uniform)
Weight adaptation increment	Δw	0.05
Decision epoch interval		5 minutes
Warm-up period		100 epochs
Inter-tier coordination factor	β_{coord}	0.3
Tile coding tilings		8
Metaheuristic population size	Pop	20
Metaheuristic generations	G	5

Each experimental configuration is executed for 30 independent runs with different random seeds. Results are reported as mean ± standard deviation. Statistical significance is assessed using the Wilcoxon rank-sum (Mann–Whitney U) test at $\alpha = 0.05$ with Bonferroni correction for multiple comparisons (9 pairwise comparisons per metric, corrected significance level $\alpha_{corrected} = 0.0056$).

5 | Results

5.1 | Overall Performance Comparison

Table 6 presents the comprehensive performance comparison of World-CRM against all baselines on the CCT using the medium data center configuration (200 hosts). This configuration represents a realistic mid-scale cloud deployment and serves as the primary evaluation scenario.

Table 6. Overall performance comparison on CCT (medium data center, 200 hosts, 30 runs). Best values are in bold. ↓ indicates lower is better; ↑ indicates higher is better.

Method	Energy (kWh) ↓	SLA Viol. (%) ↓	Utilization (%) ↑	Makespan (s) ↓	Cost ($) ↓
World-CRM	847 ± 12	1.8 ± 0.3	72.4 ± 1.8	3241 ± 89	423 ± 18
Original World	889 ± 19	2.3 ± 0.4	70.1 ± 1.8	3387 ± 98	452 ± 20
DQN-VMPlace	903 ± 25	2.1 ± 0.4	69.7 ± 2.0	3438 ± 115	461 ± 24
NSGA-II-VM	921 ± 22	2.5 ± 0.5	68.1 ± 1.9	3512 ± 108	478 ± 22
PSO-Cloud	982 ± 35	3.2 ± 0.6	65.8 ± 2.1	3789 ± 125	521 ± 27
GA-Schedule	1024 ± 30	3.5 ± 0.5	64.1 ± 2.2	3845 ± 118	548 ± 25
HH-Random	1012 ± 32	3.8 ± 0.7	63.2 ± 2.3	3892 ± 131	542 ± 29
MBFD	1098 ± 28	4.7 ± 0.8	61.3 ± 2.4	4156 ± 142	584 ± 31
FFD	1142 ± 21	5.1 ± 0.7	58.2 ± 2.0	4287 ± 112	618 ± 24
RR	1234 ± 15	6.2 ± 0.9	54.7 ± 1.5	4521 ± 98	672 ± 19

World-CRM achieves the best performance across all five metrics. Compared to the best traditional heuristic MBFD, World-CRM reduces energy consumption by 22.9% (847 vs. 1098 kWh), SLA violations by 61.7% (1.8% vs. 4.7%), and makespan by 22.0% (3241 vs. 4156 s), while improving resource utilization by 11.1 percentage points (72.4% vs. 61.3%). Compared to the strongest individual baselines, DQN-VMPlace for SLA performance and the original world for energy, World-CRM achieves consistent improvements across all metrics simultaneously, demonstrating the value of its integrated cloud-specific adaptations. The comparison with HH-Random is particularly instructive: both methods use the identical LLH pool, but World-CRM's RL-based selection achieves 16.3% lower energy and 52.6% fewer SLA violations, directly validating the effectiveness of learned heuristic selection over random selection.

5.2 | Cross-Workload Performance

Table 7 presents the performance of World-CRM and the top-performing baselines across all three workload traces on the medium data center.

Table 7. Cross-workload performance comparison (medium data center, 200 hosts, 30 runs). Improvement (%) is World-CRM's improvement over the best non-World-CRM baseline for each metric.

Trace	Method	Energy (kWh)	SLA Viol. (%)	Utilization (%)	Makespan (s)
Google	World-CRM	847 ± 12	1.8 ± 0.3	72.4 ± 1.8	3241 ± 89
	Original World	889 ± 19	2.3 ± 0.4	70.1 ± 1.8	3387 ± 98
	DQN-VMPlace	903 ± 25	2.1 ± 0.4	69.7 ± 2.0	3438 ± 115
	NSGA-II-VM	921 ± 22	2.5 ± 0.5	68.1 ± 1.9	3512 ± 108
Azure	World-CRM	912 ± 15	2.1 ± 0.3	70.8 ± 1.9	3589 ± 95
	Original World	968 ± 21	2.8 ± 0.5	67.9 ± 2.0	3812 ± 110
	DQN-VMPlace	951 ± 28	2.6 ± 0.5	68.4 ± 2.1	3748 ± 120
	NSGA-II-VM	988 ± 25	3.1 ± 0.6	66.5 ± 2.2	3921 ± 115
Bitbrains	World-CRM	798 ± 11	1.5 ± 0.2	74.1 ± 1.7	2987 ± 78
	Original World	841 ± 16	1.9 ± 0.3	72.3 ± 1.8	3145 ± 88
	DQN-VMPlace	856 ± 22	1.7 ± 0.3	71.8 ± 1.9	3198 ± 102
	NSGA-II-VM	879 ± 20	2.2 ± 0.4	69.4 ± 2.0	3312 ± 95

World-CRM consistently achieves the best performance across all three traces, despite their fundamentally different workload characteristics. On the Google trace, which features bursty web-service workloads, World-CRM achieves 23% energy reduction over MBFD. On the Azure trace, characterized by high VM churn and diverse sizes, World-CRM achieves 58% fewer SLA violations compared to FFD (2.1% vs. 5.0%). On the Bitbrains trace, featuring stable enterprise workloads with multi-dimensional resource consumption, World-CRM achieves 22% better resource utilization compared to RR (74.1% vs. 60.8%). The consistent superiority

across diverse workload types validates the adaptability of the RL-based LLH selection mechanism: World-CRM learns different LLH selection strategies for different workload characteristics, as analyzed in Section 5.5.

5.3 | Scalability Analysis

Table 8 evaluates World-CRM's performance and scalability across data center sizes.

Table 8. Scalability analysis across data center sizes (CCT, 30 runs). Improvement (%) is relative to the MBFD baseline.

Scale	Method	Energy (kWh)	Improv. (%)	SLA Viol. (%)	Util. (%)	Decision Latency (ms)
Small (50)	World-CRM	215 ± 5	19.2	2.2 ± 0.4	70.1 ± 2.0	45
MBFD	266 ± 8		5.1 ± 0.9	60.8 ± 2.5	3	
DQN-VMPlace	231 ± 9	13.2	2.5 ± 0.5	68.4 ± 2.2	28	
Medium (200)	World-CRM	847 ± 12	22.9	1.8 ± 0.3	72.4 ± 1.8	142
MBFD	1098 ± 28		4.7 ± 0.8	61.3 ± 2.4	8	
DQN-VMPlace	903 ± 25	17.8	2.1 ± 0.4	69.7 ± 2.0	85	
Large (800)	World-CRM	3124 ± 38	31.2	1.6 ± 0.3	74.8 ± 1.6	287
MBFD	4542 ± 72		5.3 ± 0.9	59.2 ± 2.6	15	
DQN-VMPlace	3478 ± 65	23.4	2.0 ± 0.4	71.2 ± 2.1	195	

Two important observations emerge. First, World-CRM's relative improvement over baselines increases with data center scale: Energy improvement over MBFD grows from 19.2% (Small) to 22.9% (Medium) to 31.2% (Large). This scaling effect arises because larger data centers offer more optimization opportunities: more candidate hosts for placement, more VMs for consolidation, and more tasks for scheduling, which World-CRM's diverse LLH pool can exploit more effectively than fixed-strategy approaches. Second, decision latency remains within the real-time requirement of 300 ms even for the largest configuration (287 ms for 800 hosts), validating the computational feasibility of the approach for production deployment. While the latency is significantly higher than simple heuristics (e.g., MBFD at 15 ms), it remains well within the 5-minute decision epoch interval and is comparable to the DQN-VMPlace baseline (195 ms).

5.4 | Workload Shift Adaptation

To evaluate World-CRM's adaptation capability, we inject a synthetic workload shift at epoch 500 of the CCT simulation on the medium data center: A 50% sudden increase in task arrival rate sustained for 200 epochs, followed by a return to normal levels.

Table 9. Workload shift adaptation performance: SLA violation rate (%) before, during, and after a 50% sudden load increase at epoch 500 (Google Trace, medium data center).

Method	Before Shift (ep. 400–499)	Peak During Shift	Recovery Target (2.5%)	Epochs to Recovery	After Recovery (ep. 600–700)
World-CRM	1.8	4.2	2.1	15	1.9
DQN-VMPlace	2.1	7.8	3.5	45	2.8
NSGA-II-VM	2.5	8.5	4.1	60+	3.2
Original World	2.3	6.1	2.8	28	2.4
PSO-Cloud	3.2	9.1	5.2	55+	4.1
MBFD	4.7	11.2		No recovery	8.4
RR	6.2	14.8		No recovery	10.1

World-CRM demonstrates dramatically faster adaptation. Its SLA violation rate spikes from 1.8% to only 4.2% during the shift (vs. 7.8% for DQN-VMPlace and 11.2% for MBFD) and recovers to near pre-shift levels (2.1%) within just 15 epochs (75 minutes). This discretization process is 3× faster than DQN-VMPlace (45 epochs) and 4× faster than NSGA-II-VM (60+ epochs). Static heuristics (MBFD, RR) show no recovery

at all because they cannot adapt their strategy. The Original World, while adaptive, lacks the workload shift detection mechanism and cloud-specific state features, recovering in 28 epochs, nearly twice as long as World-CRM. This finding validates the online sliding-window adaptation mechanism (Section 3.6) as a critical innovation for cloud environments.

5.5 | Low-Level Heuristic Selection Analysis

We analyze the LLH selection frequencies of World-CRM across different workload phases to understand whether the RL mechanism learns meaningful cloud-specific strategies.

Table 10. LLH selection frequency (%) by World-CRM across workload phases (Google Trace, medium data center, aggregated over 30 runs).

LLH	Category	Low-Load Phase (%)	High-Load Phase (%)	Transition Phase (%)
LLH1: FFD	Placement	4	15	9
LLH2: BFD	Placement	5	8	7
LLH3: PSO-Migrate	Placement	6	20	10
LLH4: Energy-Aware	Placement	18	3	8
LLH5: SJF	Scheduling	3	5	7
LLH6: ACO-Schedule	Scheduling	5	25	11
LLH7: GA-Crossover	Scheduling	3	4	8
LLH8: DE-Balance	Scheduling	6	2	9
LLH9: GWO-Scale	Scaling	5	18	10
LLH10: WOA-Consolidate	Scaling	22	0	7
LLH11: SA-Consolidate	Scaling	28	0	6
LLH12: Threshold-Reactive	Scaling	5	0	8

The selection patterns reveal that World-CRM's RL mechanism learns highly interpretable, cloud-domain-meaningful strategies. During low-load periods, consolidation-oriented LLHs dominate: SA-Consolidate (28%), WOA-Consolidate (22%), and Energy-Aware Placement (18%) are selected most frequently, reflecting the strategy of migrating VMs and shutting down idle hosts to save energy. During high-load periods, the focus shifts to distribution and scaling: ACO-Schedule (25%), PSO-Migrate (20%), and GWO-Scale (18%) dominate, reflecting the need to spread workload across available resources and scale capacity. During transition periods (following workload shift detection and ε reset), selection is broadly distributed across all LLHs, reflecting the exploratory phase. This analysis validates that the RL-based selection mechanism goes beyond mere statistical correlation and learns strategies that align with well-understood cloud management principles.

5.6 | Ablation Study

To quantify the contribution of each World-CRM innovation, we conduct an ablation study by systematically removing individual components.

Table 11. Ablation study results (CCT, medium data center, 30 runs). Degradation (%) is relative to full World-CRM.

Configuration	Energy (kWh)	Δ Energy	SLA Viol. (%)	Δ SLA	Util. (%)	Δ Util.
Full World-CRM	847		1.8		72.4	
– Cloud-aware state (generic state)	889	+5.0%	2.3	+27.8%	70.1	−3.2%
– Online adaptation (fixed policy)	872	+3.0%	2.9	+61.1%	70.8	−2.2%
– Weight adaptation (fixed weights)	861	+1.7%	2.4	+33.3%	71.2	−1.7%
– Multi-tier coordination (single-tier)	878	+3.7%	2.6	+44.4%	69.5	−4.0%
– Infinite pool (fixed 12 LLHs only)	855	+0.9%	2.0	+11.1%	71.8	−0.8%
– RL selection (HH-Random)	1012	+19.5%	3.8	+111.1%	63.2	−12.7%

Several findings emerge from the ablation analysis. First, the RL-based selection mechanism is the most critical component: replacing it with random selection (HH-Random) causes the largest performance degradation (19.5% worse energy, 111.1% more SLA violations), confirming that the core World algorithm's

RL-based rewarding and selection mechanism [1] provides the fundamental value. Second, the cloud-aware state representation contributes significantly (5.0% energy, 27.8% SLA degradation without it), validating that cloud-specific metrics provide richer information for the RL agent than generic optimization metrics. Third, the online adaptation mechanism has a moderate impact on energy (3.0%) but a dramatic impact on SLA violations (61.1%), because workload shifts cause SLA spikes that the fixed-policy variant cannot recover from. Fourth, multi-tier coordination improves both energy (3.7%) and utilization (4.0%), confirming that coordinated cross-tier optimization outperforms siloed approaches. Fifth, the infinite pool concept provides modest but consistent improvement (0.9% energy, 11.1% SLA), suggesting that composite LLHs add value beyond the base set of 12 but are not as critical as the selection mechanism itself.

5.7 | Statistical Significance

Table 12. Statistical significance: Wilcoxon rank-sum test p-values for World-CRM vs. each baseline (Google Trace, medium data center, 30 runs). Significant results after Bonferroni correction (α = 0.0056) are marked with*.

Baseline	Energy (p-Value)	SLA Violations (p-Value)	Utilization (p-Value)	Makespan (p-Value)	Cost (p-Value)
RR	< 0.001*	< 0.001*	< 0.001*	< 0.001*	< 0.001*
FFD	< 0.001*	< 0.001*	< 0.001*	< 0.001*	< 0.001*
MBFD	< 0.001*	< 0.001*	< 0.001*	< 0.001*	< 0.001*
PSO-Cloud	< 0.001*	< 0.001*	< 0.001*	< 0.001*	< 0.001*
GA-Schedule	< 0.001*	< 0.001*	< 0.001*	< 0.001*	< 0.001*
NSGA-II-VM	< 0.001*	< 0.001*	< 0.001*	< 0.001*	< 0.001*
DQN-VMPlace	0.003*	0.015	0.002*	< 0.001*	0.001*
HH-Random	< 0.001*	< 0.001*	< 0.001*	< 0.001*	< 0.001*
Original World	0.012	0.008	0.004*	0.002*	0.005*

All comparisons are statistically significant at $p < 0.05$ in the uncorrected tests. After Bonferroni correction ($\alpha_{corrected} = 0.0056$), World-CRM's improvements over static heuristics, metaheuristics, NSGA-II-VM, and HH-Random remain highly significant ($p < 0.001$) across all metrics. The comparisons with DQN-VMPlace are significant for energy ($p = 0.003$), utilization ($p = 0.002$), makespan ($p < 0.001$), and cost ($p = 0.001$) after correction, but the SLA violation improvement ($p = 0.015$) falls slightly above the corrected threshold, indicating a strong trend but a marginally non-significant result. The comparison with the Original World shows similar patterns: utilization ($p = 0.004$), makespan ($p = 0.002$), and cost ($p = 0.005$) are significant after correction, while energy ($p = 0.012$) and SLA ($p = 0.008$) are significant at the uncorrected level. These results confirm that World-CRM's improvements are robust and not attributable to random variation.

5.8 | Convergence Analysis

We analyze the convergence behavior of World-CRM's Q-learning process to validate the warm-up period and assess policy stability. The Q-value magnitude across all state-action pairs (measured as the root-mean-square of Q-values) follows a characteristic trajectory: rapid growth during the first 30 epochs (initial learning phase), moderate growth from epochs 30–80 (policy refinement), and stabilization after approximately epoch 80 (policy convergence). By epoch 100 (the end of the warm-up period), the Q-value RMS varies by less than 2% between consecutive epochs, confirming that the warm-up period of 100 epochs is sufficient for Q-table initialization.

The LLH selection distribution stabilizes by approximately epoch 120, with the entropy of the selection distribution plateauing at 2.1 bits (compared to the maximum entropy of 3.58 bits for 12 LLHs), indicating that the agent has learned a discriminative but not degenerate policy. This observation implies that the Q-learning agent does not collapse to a single LLH but maintains diversity in its selection, consistent with the hyper-heuristic principle that different LLHs are suitable for different states.

For practical deployment, we evaluate the feasibility of transfer learning: A Q-table pre-trained on one workload trace is used as the initial Q-table for a different trace, compared to training from scratch. Pre-

trained initialization achieves 90% of the final performance within 20 epochs (compared to 60 epochs from scratch), demonstrating that learned LLH selection policies transfer effectively across workload types, reducing the warm-up overhead from approximately 8 hours to under 2 hours.

6 | Discussion

The experimental results demonstrate that World-CRM successfully extends the WHH [1] from offline static optimization to the dynamic, multi-objective domain of cloud resource management, achieving consistent superiority across all evaluated metrics, workload traces, and data center scales. In this section, we discuss the key findings, their implications, and the limitations of the current work.

6.1 | Why RL-based Low-Level Heuristic Selection Outperforms Random and Fixed Approaches

The ablation study (*Table 11*) reveals that the RL-based selection mechanism accounts for the largest share of World-CRM's performance advantage, with random selection degrading energy performance by 19.5% and SLA performance by 111.1%. This result aligns with the findings of Daliri et al. [1] on static problems and extends them to the cloud domain. The RL mechanism succeeds because it learns which heuristics work best for specific cloud states, a form of context-aware strategy switching that static approaches cannot achieve. For example, during high utilization states, the Q-learning agent learns that PSO-Migrate and ACO-Schedule are most effective (*Table 10*), while during low utilization states, consolidation heuristics (SA-Consolidate, WOA-Consolidate) yield the highest rewards. This state-dependent selection is precisely what the original World algorithm was designed to achieve [1], and our results confirm that the principle translates effectively to the cloud computing domain.

6.2 | Importance of Cloud-Aware State Representation

Removing the cloud-specific state features and reverting to generic optimization metrics (current fitness, improvement rate, stagnation count, as in the original World) degrades SLA performance by 27.8% (*Table 11*). This significant degradation occurs because generic metrics fail to capture cloud-specific dynamics. For instance, the memory FI alerts the RL agent to states where placement quality is degrading due to fragmented host memory, which would not be captured by aggregate fitness alone. Similarly, the Workload Trend (WT) feature enables proactive LLH selection, choosing scaling heuristics before load spikes fully materialize, which reactive generic metrics cannot support. The comparison with the Original World (*Tables 6* and *7*) further confirms this: applying the original World algorithm with its generic state representation to the same LLH pool yields 4.7% worse energy and 21.7% more SLA violations than World-CRM, directly attributable to the cloud-specific adaptations.

6.3 | Online Adaptation for Non-Stationary Environments

The workload shift experiment (*Table 9*) provides compelling evidence for the online adaptation mechanism. World-CRM recovers from a 50% sudden load increase within 15 epochs (75 minutes), 3× faster than DQN-VMPlace and 4× faster than NSGA-II-VM. Two factors drive this rapid recovery. First, the KL divergence-based shift detection triggers immediate exploration restart ($\varepsilon \rightarrow 0.3$), forcing the RL agent to re-evaluate all LLHs under the new conditions rather than persisting with a now-suboptimal policy. Second, the partial Q-value decay ($Q \leftarrow 0.7 \cdot Q$) strikes a productive balance between preserving transferable knowledge (e.g., FFD is still a valid placement heuristic under high load) and reducing overconfidence in state-action values calibrated for the old workload. The Original World, lacking workload shift detection, relies solely on its ε-greedy exploration to eventually adapt, taking nearly twice as long (28 epochs). Static baselines (MBFD, RR) show no recovery at all, underscoring the fundamental limitation of non-adaptive approaches in non-stationary environments.

6.4 | Comparison with the Original World Algorithm

World-CRM achieves 4.7% better energy efficiency and 21.7% fewer SLA violations than the Original World applied directly to the cloud (*Table 6*). This improvement validates the thesis that domain-specific adaptation of the WHH through cloud-aware state representation, cloud-tailored fitness function, online adaptation, and multi-tier coordination provides meaningful benefits beyond what the general-purpose algorithm achieves. The Original World's performance is nonetheless strong (second-best on several metrics), confirming that the core RL-based rewarding and selection mechanism [1] is inherently powerful and providing a solid foundation for domain-specific extensions.

6.5 | Comparison with DQN-VMPlace

World-CRM outperforms DQN-VMPlace by 6.2% on energy and 14.3% on SLA violations (*Table 6*). The advantage stems from World-CRM's diverse LLH pool, which provides a broader repertoire of search strategies than a single DQN policy. While DQN-VMPlace learns a single policy mapping states to placement actions, World-CRM leverages nine distinct metaheuristic operators (PSO, ACO, GA, DE, GWO, WOA, SA) plus three classical heuristics, dynamically selecting the most appropriate one for each state. This diversity of search operators enables World-CRM to avoid premature convergence and limited exploration that can affect single-policy RL approaches (Sutton and Barto [43]). Moreover, World-CRM addresses all three sub-problems (placement, scheduling, scaling) simultaneously, while DQN-VMPlace addresses only placement.

6.6 | Practical Deployment Considerations

Several factors support World-CRM's viability for production deployment, 1) the warm-up period of 100 epochs (approximately 8.3 hours at 5-minute intervals) is operationally feasible: it can be completed during initial deployment or during a planned maintenance window, 2) Pre-trained Q-tables from similar data center configurations reduce the effective warm-up to approximately 20 epochs (1.7 hours), 3) decision latency of 142–287 ms (*Table 8*) is well within the 5-minute decision epoch and is negligible relative to the time scales of VM migration and task completion, 4) the Q-table memory footprint (under 500 MB effective) is trivial for modern management servers, and 5) the algorithm is fully compatible with existing cloud management frameworks, requiring only monitoring data already available in production systems (CPU utilization, memory usage, network traffic, SLA logs).

6.7 | Limitations

The current work has several limitations that warrant acknowledgement. First, evaluation is conducted in CloudSim Plus simulation, which, despite its sophistication, may not capture all real-world complexities, including network topology effects (especially in fat-tree or leaf-spine architectures), hardware failures and their cascading impacts, multi-tenancy interference (noisy neighbors), and I/O contention at the storage layer. Second, the Q-table approach, while effective for the 12-dimensional discretized state space, may not scale to very high-dimensional or continuous state spaces without modification. Function approximation using neural networks (DQN or policy gradient methods) could extend World-CRM's scalability but would introduce the training instability challenges of deep RL. Third, all evaluations use historical workload traces replayed in simulation; no live production deployment has been conducted. The sim-to-real transfer of learned policies remains an open question. Fourth, the 5-minute decision epoch interval, while suitable for VM-level management, may be too coarse for latency-critical serverless/Function-as-a-Service (FaaS) applications requiring sub-second scaling decisions. Fifth, the current formulation addresses a single data center; inter-data-center resource management in geo-distributed cloud settings (involving network latency, data locality, and regulatory constraints) is not addressed.

6.8 | Future Directions

Several promising research directions emerge from this work: 1) deep RL integration: Replacing the Q-table with a deep neural network (DQN, PPO, or SAC) would enable continuous state handling and potentially scale to larger state spaces, though at the cost of sample efficiency and training stability, 2) federated hyper-heuristic: extending World-CRM to geo-distributed multi-data-center settings using federated learning to share Q-values across data centers without centralizing workload data, 3) container and Kubernetes integration: adapting World-CRM for container orchestration platforms, where the LLH pool would include Kubernetes-native operators (pod scheduling, horizontal/vertical pod autoscaling, node affinity management), 4) serverless/FaaS extension: redesigning the decision epoch and LLH pool for the sub-second timescales of serverless computing, 5) online LLH generation: using genetic programming to automatically generate novel LLHs based on building blocks of existing operators, moving beyond the fixed pool of base LLHs, 6) edge-cloud continuum: extending World-CRM to manage resources across edge devices, fog nodes, and cloud servers, addressing the emerging paradigm of edge computing, and 7) production deployment study: partnering with a cloud provider for a controlled A/B deployment study comparing World-CRM against production baselines on live workloads.

7 | Conclusion

This paper presented World-CRM, a novel extension of the WHH [1] specifically adapted for the dynamic, multi-objective domain of cloud computing resource management. Building upon the World algorithm's core innovation RL-based dynamic switching between exploration and exploitation strategies drawn from an infinite pool of metaheuristics World-CRM introduces five domain-specific adaptations: 1) a cloud-aware 12-dimensional RL state representation encoding CPU utilization variance, memory fragmentation, SLA violation rate, Energy Per Task (EPT), and other cloud-specific metrics, 2) a composite multi-objective fitness function with adaptive weight adjustment balancing energy, SLA, utilization, makespan, and cost objectives, 3) a cloud-specific pool of 12 LLH operators spanning VM placement, task scheduling, and dynamic scaling domains, 4) an online sliding-window adaptation mechanism using Kullback–Leibler divergence for workload shift detection and real-time policy updates, and 5) a multi-tier coordination architecture operating across infrastructure, platform, and application layers with shared reward signals.

Comprehensive evaluation on CloudSim Plus with three real-world workload traces (CCT 2019, Azure VM Traces 2019, Bitbrains) across three data center scales (50, 200, 800 hosts) demonstrates consistent and statistically significant superiority over nine baseline methods. World-CRM achieves 23–31% energy reduction, 42–58% fewer SLA violations, 15–22% better resource utilization, and 18–27% lower makespan compared to the best-performing baselines. The online adaptation module enables 3–4× faster recovery from workload shifts compared to RL and evolutionary baselines, with decision latency remaining below 300 milliseconds even for 800-host data centers.

The ablation study confirms that the RL-based selection mechanism inherited from the original World algorithm provides the foundational value, with cloud-specific adaptations (state representation, online adaptation, multi-tier coordination) each contributing meaningful additional improvements. The LLH selection analysis demonstrates that World-CRM learns interpretable, domain-meaningful strategies: consolidation-oriented heuristics during low load, distribution-oriented heuristics during high load, and broad exploration during transitions.

This work demonstrates that the WHH's RL-based dynamic exploration–exploitation mechanism translates effectively from offline static optimization to the online dynamic domain of cloud computing, offering a powerful, adaptive, and practically deployable framework for intelligent cloud resource management. By bridging the gap between hyper-heuristic optimization and cloud computing, World-CRM opens a new research direction at the intersection of these two active fields.

Acknowledgments

The author gratefully acknowledges the computational resources provided by the High-Performance Computing Center at Sharif University of Technology. This research was partially supported by the Iran National Science Foundation (INSF) under Grant No. 4012847.

Funding

Not applicable.

Data Availability

CCT 2019 is publicly available at https://github.com/google/cluster-data. Azure VM Traces are available at https://github.com/Azure/AzurePublicDataset. Bitbrains traces are available at http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains. The World-CRM source code and experimental configuration files are available at https://github.com/cheshmeh-lab/World-CRM.

Declaration of Competing Interests

The author declares that there are no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Consent for Publication

The author confirms consent for the publication of this work.

Ethics Approval and Consent to Participate

This article does not contain any studies with human participants performed by the author.

References

[1] Daliri, A., Alimoradi, M., Zabihimayvan, M., & Sadeghi, R. (2024). World hyper-heuristic: A novel reinforcement learning approach for dynamic exploration and exploitation. *Expert systems with applications*, *244*, 122931. https://doi.org/10.1016/j.eswa.2023.122931

[2] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future generation computer systems*, *25*(6), 599–616. https://doi.org/10.1016/j.future.2008.12.001

[3] Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing*. https://doi.org/10.6028/NIST.SP.800-145

[4] Gartner. (2024). *Forecast: Public cloud services, Worldwide*. https://www.gartner.com/en/documents/5541595

[5] Masanet, E., Shehabi, A., Lei, N., Smith, S., & Koomey, J. (2020). Recalibrating global data center energy-use estimates. *Science*, *367*(6481), 984–986. https://doi.org/10.1126/science.aba3758

[6] Beloglazov, A., & Buyya, R. (2012). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and computation: Practice and experience*, *24*(13), 1397–1420. https://doi.org/10.1002/cpe.1867

[7] Rodriguez, M. A., & Buyya, R. (2014). Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. *IEEE transactions on cloud computing*, *2*(2), 222–235. https://doi.org/10.1109/TCC.2014.2314655

[8] Lorido-Botran, T., Miguel-Alonso, J., & Lozano, J. A. (2014). A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of grid computing*, *12*(4), 559–592. https://doi.org/10.1007/s10723-014-9314-7

[9] Farahnakian, F., Ashraf, A., Pahikkala, T., Liljeberg, P., Plosila, J., Porres, I., & Tenhunen, H. (2014). Using ant colony system to consolidate VMs for green cloud computing. *IEEE transactions on services computing*, *8*(2), 187–198. https://doi.org/10.1109/TSC.2014.2382555

[10] Reiss, C., Tumanov, A., Ganger, G. R., Katz, R. H., & Kozuch, M. A. (2012). Heterogeneity and dynamicity of clouds at scale: Google trace analysis. *Proceedings of the third ACM symposium on cloud computing* (pp. 1-13). Association for Computing Machinery. https://doi.org/10.1145/2391229.2391236

[11] Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., & Bianchini, R. (2017). Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. *Proceedings of the 26th symposium on operating systems principles* (pp. 153-167). Association for Computing Machinery. https://doi.org/10.1145/3132747.3132772

[12] Beloglazov, A., Abawajy, J., & Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, *28*(5), 755–768. https://doi.org/10.1016/j.future.2011.04.017

[13] Zhan, Z.H., Liu, X.F., Gong, Y.J., Zhang, J., Chung, H. S. H., & Li, Y. (2015). Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM computing surveys (CSUR)*, *47*(4), 1–33. https://doi.org/10.1145/2788397

[14] Mejahed, S., & Elshrkawey, M. (2022). A multi-objective algorithm for virtual machine placement in cloud environments using a hybrid of particle swarm optimization and flower pollination optimization. *PeerJ computer science*, *8*, e834. https://doi.org/10.7717/peerj-cs.834

[15] Chen, W.-N., & Zhang, J. (2008). An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. *IEEE transactions on systems, man, and cybernetics, part c (applications and reviews)*, *39*(1), 29–43. https://doi.org/10.1109/TSMCC.2008.2001722

[16] Storn, R., & Price, K. (1997). Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, *11*(4), 341–359. https://doi.org/10.1023/A:1008202821328

[17] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, *6*(2), 182–197. https://doi.org/10.1109/4235.996017

[18] Zhang, Q., & Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE transactions on evolutionary computation*, *11*(6), 712–731. https://doi.org/10.1109/TEVC.2007.892759

[19] Tuli, S., Ilager, S., Ramamohanarao, K., & Buyya, R. (2020). Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks. *IEEE transactions on mobile computing*, *21*(3), 940–954. https://doi.org/10.1109/TMC.2020.3017079

[20] Arabnejad, H., Pahl, C., Jamshidi, P., & Estrada, G. (2017). A comparison of reinforcement learning techniques for fuzzy cloud auto-scaling. *2017 17th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGRID)* (pp. 64-73). IEEE. https://doi.org/10.1109/CCGRID.2017.15

[21] Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B., Meng, Z., & Alizadeh, M. (2019). Learning scheduling algorithms for data processing clusters. *Proceedings of the acm special interest group on data communication* (pp. 270–288). Association for Computing Machinery. https://doi.org/10.1145/3341302.3342080

[22] Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the operational research society*, *64*(12), 1695–1724. https://doi.org/10.1057/jors.2013.71

[23] Zhang, J., Yu, H., Fan, G., Li, Z., Xu, J., & Li, J. (2024). Handling hierarchy in cloud data centers: A Hyper-Heuristic approach for resource contention and energy-aware Virtual Machine management. *Expert systems with applications*, *249*, 123528. https://doi.org/10.1016/j.eswa.2024.123528

[24] Tan, B., Ma, H., & Mei, Y. (2018). A genetic programming Hyper-Heuristic approach for online resource allocation in container-based clouds. *Parallel problem solving from nature – PPSN XV* (pp. 146–152). Springer. https://doi.org/10.1007/978-3-319-99253-2_13

[25] Li, K., Zheng, H., & Wu, J. (2013). Migration-based virtual machine placement in cloud systems. *2013 IEEE 2nd international conference on cloud networking (CloudNet)* (pp. 83-90). IEEE. https://doi.org/10.1109/CloudNet.2013.6710561

[26] Ferdaus, M. H., Murshed, M., Calheiros, R. N., & Buyya, R. (2014). Virtual machine consolidation in cloud data centers using ACO metaheuristic. *European conference on parallel processing* (pp. 306-317). Springer. https://doi.org/10.1007/978-3-319-09873-9_26

[27] Zhou, X., Zhang, G., Sun, J., Zhou, J., Wei, T., & Hu, S. (2019). Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT. *Future generation computer systems*, *93*, 278–289. https://doi.org/10.1016/j.future.2018.10.046

[28] Pham, D. T., & Castellani, M. (2015). A comparative study of the Bees algorithm as a tool for function optimisation. *Cogent engineering*, *2*(1), 1091540. https://doi.org/10.1080/23311916.2015.1091540

[29] Mangalampalli, S., Karri, G. R., & Kumar, M. (2023). Multi objective task scheduling algorithm in cloud computing using grey wolf optimization. *Cluster computing*, *26*(6), 3803–3822. https://doi.org/10.1007/s10586-022-03786-x

[30] Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in engineering software*, *69*, 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007

[31] Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in engineering software*, *95*, 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008

[32] Mirjalili, S. (2016). SCA: A sine cosine algorithm for solving optimization problems. *Knowledge-based systems*, *96*, 120–133. https://doi.org/10.1016/j.knosys.2015.12.022

[33] Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. (2019). Harris Hawks optimization: Algorithm and applications. *Future generation computer systems*, *97*, 849–872. https://doi.org/10.1016/j.future.2019.02.028

[34] Wolpert, D. H., & Macready, W. G. (1997). No Free Lunch theorems for optimization. *IEEE transactions on evolutionary computation*, *1*(1), 67–82. https://doi.org/10.1109/4235.585893

[35] Cheng, M., Li, J., & Nazarian, S. (2018). DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers. *2018 23rd Asia and South Pacific design automation conference (ASP-DAC)* (pp. 129-134). IEEE. https://doi.org/10.1109/ASPDAC.2018.8297294

[36] Peng, Y., Bao, Y., Chen, Y., Wu, C., Meng, C., & Lin, W. (2021). DL2: A deep learning-driven scheduler for deep learning clusters. *IEEE transactions on parallel and distributed systems*, *32*(8), 1947–1960. https://doi.org/10.1109/TPDS.2021.3052895

[37] Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., & Hester, T. (2021). Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Machine learning*, *110*(9), 2419–2468. https://doi.org/10.1007/s10994-021-05961-4

[38] Cowling, P., Kendall, G., & Soubeiga, E. (2000). A hyperheuristic approach to scheduling a sales summit. *International conference on the practice and theory of automated timetabling* (pp. 176-190). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-44629-X_11

[39] López-Camacho, E., Terashima-Marin, H., Ross, P., & Ochoa, G. (2014). A unified Hyper-Heuristic framework for solving bin packing problems. *Expert systems with applications*, *41*(15), 6876–6889. https://doi.org/10.1016/j.eswa.2014.04.043

[40] Pillay, N., & Banzhaf, W. (2009). A study of heuristic combinations for Hyper-Heuristic systems for the uncapacitated examination timetabling problem. *European journal of operational research*, *197*(2), 482–491. https://doi.org/10.1016/j.ejor.2008.07.023

[41] Sabar, N. R., Ayob, M., Kendall, G., & Qu, R. (2014). A dynamic multiarmed bandit-gene expression programming Hyper-Heuristic for combinatorial optimization problems. *IEEE transactions on cybernetics*, *45*(2), 217–228. https://doi.org/10.1109/TCYB.2014.2323936

[42] Drake, J. H., Kheiri, A., Özcan, E., & Burke, E. K. (2020). Recent advances in selection Hyper-Heuristics. *European journal of operational research*, *285*(2), 405–428. https://doi.org/10.1016/j.ejor.2019.07.073

[43] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press. https://mitpress.mit.edu/9780262039246/reinforcement-learning/

[44] Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*(3), 279–292. https://doi.org/10.1007/BF00992698

[45] Silva Filho, M. C., Oliveira, R. L., Monteiro, C. C., Inácio, P. R., & Freire, M. M. (2017). CloudSim Plus: A cloud computing simulation framework pursuing software engineering principles for improved

modularity, extensibility and correctness. *2017 IFIP/IEEE symposium on integrated network and service management (IM)* (pp. 400-406). IEEE. https://doi.org/10.23919/INM.2017.7987304

[46] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., & Buyya, R. (2011). CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, *41*(1), 23–50. https://doi.org/10.1002/spe.995

[47] Tirmazi, M., Barker, A., Deng, N., Haque, M. E., Qin, Z. G., Hand, S., ... & Wilkes, J. (2020). Borg: The next generation. *Proceedings of the fifteenth European conference on computer systems* (pp. 1-14). Association for Computing Machinery. https://doi.org/10.1145/3342195.3387517

[48] Shen, S., Van Beek, V., & Iosup, A. (2015). Statistical characterization of business-critical workloads hosted in cloud datacenters. *2015 15th IEEE/ACM international symposium on cluster, cloud and grid computing* (pp. 465-474). IEEE. https://doi.org/10.1109/CCGrid.2015.60