IVMR suite: An Industrial-scale Virtual Machine Rescheduling Dataset and Benchmark for Elastic Cloud Service

Yupeng Zhang*
Alibaba DAMO Academy
Hangzhou, Zhejiang, China
mingliu.zyp@alibaba-inc.com

Xu Wan*
Zhejiang University
Hangzhou, Zhejiang, China
wanxu@zju.edu.cn

Xiangyun Kong*
Alibaba Cloud Intelligence Group
Hangzhou, Zhejiang, China
xiangyun.kongxy@alibaba-inc.com

Chao Yang
Alibaba DAMO Academy
Hangzhou, Zhejiang, China
xiuxin.yc@alibaba-inc.com

Binda Ma
Alibaba Cloud Intelligence Group
Hangzhou, Zhejiang, China
maxin.mbd@taobao.com

Wotao Yin
Alibaba DAMO Academy
Bellevue, Washington, United States
wotao.yin@alibaba-inc.com

Jian Zhou
Alibaba Cloud Intelligence Group
Beijing, China
xuming@alibaba-inc.com

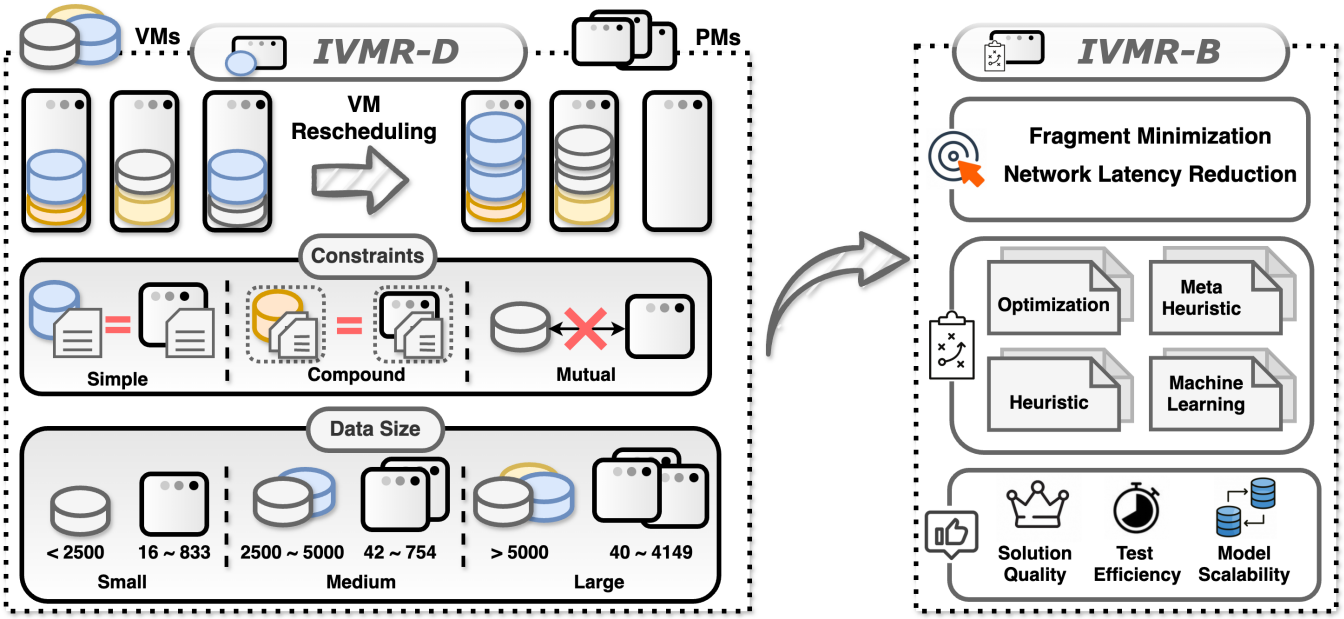

Figure 1: The framework of IVMR suite, including a dataset IMVR-D and a benchmark IMVR-B.

Abstract

Virtual Machine Rescheduling (VMR) plays a crucial role in maintaining service quality and resource efficiency in elastic cloud computing. However, existing datasets and benchmarks primarily focus on VM scheduling tasks, while lacking industrial-scale datasets and standardized evaluation for the more complex and crucial rescheduling problems. To address these challenges, we present **IVMR suite**, the first industrial-scale suite for VMR research comprising two core components: 1) *IVMR-D*, an industrial-grade VMR dataset mined from a real cloud data center, integrating complete resource specifications and complex operation constraints. The dataset is

*Both authors contributed equally to this research.

KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3737381

systematically structured based on data size and optimization objectives. 2) *IVMR-B*, a benchmark for the VMR problem that establishes seamless integration of consistent evaluation and the provision of baselines spanning optimization, metaheuristic, heuristic, and machine learning-based methodologies. Our comprehensive experimental evaluation demonstrates that all tested VMR algorithms struggle to effectively balance solution quality with computational efficiency while showing limited scalability across tasks with varying complexity levels. These findings emphasize the urgency of improving VMR algorithms for industrial deployments.

CCS Concepts

• **Computer systems organization → Cloud computing**; • **Software and its engineering → Allocation / deallocation strategies**; **Virtual machines**.

Keywords

Cloud Computing, Virtual Machine Rescheduling, Resource Management, Optimization, Meta Heuristic, Imitation Learning, Reinforcement Learning

ACM Reference Format:

Yupeng Zhang, Xu Wan, Xiangyun Kong, Chao Yang, Binda Ma, Wotao Yin, and Jian Zhou. 2025. IVMR suite: An Industrial-scale Virtual Machine Rescheduling Dataset and Benchmark for Elastic Cloud Service. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3711896.3737381

1 Introduction

Cloud data centers offer users on-demand access to computing resources via the Infrastructure-as-a-Service (IaaS) model [3]. As a foundational technology, virtualization enables multiple virtual machines (VMs) to run on a single physical machine (PM), thus improving resource-sharing efficiency. Consequently, cloud service providers, such as Amazon Web Services, Microsoft Azure, and Alibaba Cloud, can deliver scalable and flexible resources tailored to users' variable demands, with isolated, secure, and easily managed services [35]. However, with the rapid development of cloud computing, cloud service providers face an increasingly complex optimization challenge of minimizing operational costs while maintaining strict service quality guarantees for diverse customer workloads [4].

To address these challenges, *virtual machine scheduling (VMS) and virtual machine rescheduling (VMR)* have become crucial research topics. VMS primarily focuses on the initial placement of VMs onto PMs when VMs first arrive in the cloud environment [29]. This process aims to optimize initial resource utilization, considering simple constraints such as CPU usage and memory requirements. However, the initial scheduling is often sub-optimal, and the continual exiting of completed VMs results in many fragments scattered across PMs [8]. Consequently, VMR focuses on migrating VMs from their initial PMs to the new destination PMs for optimizing resource allocation [4], migration costs and system stability [10, 14]. Due to the interdependent characteristics of migration sequence generation, the process of VMR is more challenging than VMS. While researchers have made significant progress by exploring multiple solution paradigms - from heuristic algorithms [4, 16] to metaheuristic approaches [10, 14] and machine learning methods [11, 32] - two fundamental challenges persist in addressing the VMR problem.

The primary challenge is the **scarcity of available industrial-scale VMR datasets**. As illustrated in Table 1, existing public cloud datasets like Google Cluster Trace [20] and Azure Workload Dataset [7] mainly focus on VMS scenarios. They are missing vital metadata essential for VMR studies: (1) initial VM-PM allocation configurations, and (2) dynamic migration cost matrices. Although VMR2L [8] has released two VMR datasets of varying sizes, there is still a discrepancy in objective complexity and the scale of the data compared to current industrial scenarios. Consequently, many researchers are compelled to use synthetic data or simplified simulation environments [11, 19, 26, 34], which challenges the ability to accurately assess the true effectiveness of current VMR algorithms in real-world cloud environments [10, 32].

The second challenge is **the lack of standardized evaluation benchmarks for VMR methods**. Unlike VMS research [1, 15] that benefits from established cloud environments such as CloudSim [5] or DCSim [25], VMR studies suffer from inconsistent datasets and assessment metrics. For instance, IRW [31] optimizes workflow completion rates and makespan delays under VM failures, while TRM [36] focuses on minimizing network resource consumption. The diversity in evaluation further hinders the comprehensive and unified assessment of VMR algorithms. Besides, existing algorithms typically address single objectives with basic resource constraints, whereas real-world scenarios introduce more complex objectives and constraints. It motivates us to develop a benchmark for VMR tasks, enabling a unified evaluation of existing VMR algorithms using industrial-scale data. To sum up, our main contributions are:

(1) ***Industrial-scale VMR Datasets.*** First, we introduce IVMR-D, a standardized collection of real-world industrial VMR datasets. IVMR-D encompasses complete resource specifications while incorporating industrial-grade complex constraints. Furthermore, IVMR-D is systematically categorized by data size and objective functions, enabling thorough algorithm evaluation across diverse scenarios.

(2) ***Unified VMR Algorithm Benchmark.*** Second, we propose IVMR-B, the first industrial-scale benchmark of the mainstream VMR solutions based on the IVMR-D. Specifically, we implement six VMR algorithms under complex real-world constraints. We then conduct comprehensive experiments to assess the mainstream VMR algorithms' performance from the perspectives of solution quality, test efficiency, and model scalability.

(3) ***Comprehensive Performance Analysis and Insights.*** Third, the extensive evaluation presents the first comprehensive performance comparison of existing algorithms on industrial-scale VMR problems. Our experimental results reveal two critical insights: (1) none of the tested algorithms achieve a satisfactory trade-off between solution quality and testing efficiency on the "Large" category of IVMR-D; (2) for machine learning-based algorithms, the model transferability significantly deteriorates as data size increases.

These findings highlight the substantial challenges in scaling current VMR solutions to industrial requirements and provide valuable directions for future research.

Table 1: A comparison of several public datasets for VM Scheduling and Rescheduling.

Datasets	Data Source	Data Size		Constraints			Supports VMR
		VM Size	PM Size	Simple[1]	Compound[2]	Mutual[3]	
IVMR-D (Ours)	Industrial	<2500 (Small) 2500-5000 (Medium) >5000 (Large)	16-833 (Small) 42-754 (Medium) 40-4149 (Large)	✓	✓	✓	✓
VMR2L Dataset [8]	Industrial	2089 (Medium) 4546 (Large)	280 (Medium) 1176 (Large)	✓	✓	✗	✓
Google Cluster [28]	Industrial	-	-	✓	✗	✗	✗
Azure Cluster [7]	Industrial	-	-	✓	✗	✗	✗
Alibaba Cluster [6]	Industrial	-	1300, 4000	✓	✗	✗	✗
Synthetic Dataset [19]	Synthetic	Tens to thousands	Tens to hundreds	✓	✓	✓	✓

[1] Simple refers to basic resource constraints such as CPU and memory.
[2] Compound refers to composite resource constraints, specifically non-uniform memory accesses (NUMA) in our dataset.
[3] Mutual refers to the existence of mutual constraints between VMs and PMs.

2 Related Work

Existing approaches for both VMS and VMR tasks can be broadly categorized into four types: heuristic algorithms, optimization algorithms, metaheuristic algorithms, and machine learning-based algorithms.

Heuristic Algorithms. Early research mainly focuses heuristic-based approaches. The First Fit Decreasing (FFD) algorithm [16] provides a straightforward yet effective approach through multi-level classification and sorting. Beloglazov and Buyya [4] proposed a modified Best Fit Decreasing (BFD) algorithm that incorporates adaptive thresholds for host overload detection and optimized consolidation strategies. Wood et al. [29] developed the Sandpiper system with both black-box and gray-box monitoring mechanisms, enabling flexible migration decisions through multi-dimensional resource assessment. While computationally efficient, these heuristic approaches often yield suboptimal solutions in complex scenarios.

Optimization Algorithms. Mathematical programming approaches have been investigated to pursue theoretical optimality [21, 30]. Mazumdar and Pranzo [18] formulated VM consolidation as a Mixed Integer Linear Programming (MILP) problem, simultaneously optimizing energy costs, migration overhead, and workload distribution. Wei et al. [26] further modeled the problem as a three-dimensional bin packing optimization (3D BPP), incorporating comprehensive resource constraints. Duong-Ba et al. [9] proposed a Multi-level Join VM Placement and Migration (MJPM) algorithm which aims at minimizing the resource usage and power consumption in a data centre. While these methods can achieve optimal solutions through commercial solvers, their computational complexity limits scalability in large-scale deployments.

Metaheuristic Algorithms. To better handle complex optimization objectives, metaheuristic algorithms have emerged as promising alternatives [13]. Farahnakian et al. [10] leveraged Ant Colony Optimization (ACO) with pheromone mechanisms to guide VM reallocation, while Wen et al. [27] extended this approach with a distributed migration strategy to enhance scalability. Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) have also demonstrated effectiveness in multi-objective scenarios [14, 17]. For instance, Hu et al. [14] developed a GA-based framework with specialized chromosome encoding to balance load distribution and migration costs.

Machine learning-based Algorithms. More recently, machine learning techniques have shown promise in addressing the efficiency-optimality trade-off [12, 24]. Zeng et al. [32] proposed an adaptive deep reinforcement learning framework (ADVMC) that combines supervised learning for state prediction with reinforcement learning for decision-making. Guo et al. [11] integrated imitation learning (IL) with deep reinforcement learning in their DeepRM Plus solution, utilizing convolutional neural networks to capture complex resource patterns. Moreover, Ding et al. [8] developed Virtual Machines Rescheduling Using Reinforcement Learning (VMR2L), a two-stage framework specifically designed for large-scale deployments, achieving performance comparable to optimal solutions with significantly reduced computational time.

3 Virtual Machine Rescheduling

3.1 VMR Description

Given a set of VMs $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$ initially deployed on PMs $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$, VM rescheduling seeks to strategically migrate selected VMs to alternative PMs to achieve objectives such as resource utilization and quality of service. The problem is constrained by various types of basic resources $\mathcal{BR} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_\mathcal{K}\}$, typically including CPU, memory, and bandwidth. Each PM $p_j \in \mathcal{P}$ provides a maximum capacity $R_{j,k}$ for the type of resource $\mathcal{R}_k$, while each VM $v_i \in \mathcal{V}$ demands $r_{i,k}$ units of $\mathcal{R}_k$ and must be assigned to exactly one PM. Throughout the rescheduling process, the total resource demands of VMs assigned to any PM must not exceed that PM's capacity across all types of resources. Additionally, beyond the basic resources, compound resources, specifically non-uniform memory access (NUMA) which is a common resource organization mechanism in modern large-scale cloud clusters, are also taken into consideration. Each NUMA comprises multiple basic resources $\mathcal{CR} = \{\mathcal{R}_1, \ldots, \mathcal{R}_\mathcal{D}\}$. Each VM i requires p_i identical NUMAs, where each NUMA demands $u_{i,d}$ units of the d-th basic

resource. Each PM j can provide q_j NUMAs, with the d-th basic resource of the w-th NUMA having a maximum capacity $U_{j,w,d}$. Similarly, the aggregate demands of VMs' NUMA resources on any PM can't exceed that PM's NUMA capacity.

3.2 Problem Formulation

Mathematically, the problem of VMR can be formulated as an Integer Programming problem (IP), which aims to find the optimal assignment A_{opt} of m VMs to n PMs, based on the defined objective and constraints.

Minimize :

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{i,j}c_{i,j} + \sum_{i=1}^{m}(1 - x_{i,j_i^{\text{init}}})a_i \tag{1}$$

Subject to :

$$\sum_{j=1}^{n} x_{i,j} = 1 \quad \forall i \in \mathcal{V} \tag{2}$$

$$\sum_{i=1}^{m}(1 - x_{i,j_i^{\text{init}}}) \leq \mathrm{L} \tag{3}$$

$$\sum_{i=1}^{m} x_{i,j}r_{i,k} \leq R_{j,k} \quad \forall j \in \mathcal{P}, \forall k \in \mathcal{R} \tag{4}$$

$$\sum_{i=1}^{m} y_{i,j,w}u_{i,d} \leq U_{j,w,d} \quad \forall j \in \mathcal{P}, \forall w \in \{1,\ldots,q_j\}, \forall d \in C\mathcal{R} \tag{5}$$

$$\sum_{w=1}^{q_j} y_{i,j,w} = p_i x_{i,j} \quad \forall i \in \mathcal{V}, \forall j \in \mathcal{P} \tag{6}$$

$$y_{i,j,w} \leq x_{i,j} \quad \forall i \in \mathcal{V}, \forall j \in \mathcal{P}, \forall w \in \{1,\ldots,q_j\} \tag{7}$$

$$x_{i,j} \leq 1 - \text{mutex}_{i,j} \quad \forall i \in \mathcal{V}, \forall j \in \mathcal{P} \tag{8}$$

$$x_{i,j}, y_{i,j,w} \in \{0,1\} \quad \forall i \in \mathcal{V}, \forall j \in \mathcal{P}, \forall w \in \{1,\ldots,q_j\} \tag{9}$$

Here, $x_{i,j}$ is a decision variable that indicates whether VM i is deployed to PM j in the assignment (0 for No, 1 for Yes). Additionally, $y_{i,j,w}$ is another decision variable indicating whether one of VM i's NUMA is deployed to the w-th NUMA of PM j (0 for No, 1 for Yes). The objective, as defined by (1), comprises two critical components: The parameter $c_{i,j}$ in the first component denotes the assignment cost of deploying VM i on PM j. For a given VM i, smaller $c_{i,j}$ values indicate a stronger preference for migrating VM i to PM j, while larger values encourage relocating VM i away from PM j. The second component of the objective function calculates the total migration cost across all VMs, where j_i^{init} identifies the PM that initially hosts VM i, and a_i represents the migration cost associated with VM i.

Several constraints are defined to ensure the rationality and legality of the optimal assignment. Constraint (2) ensures that each VM is deployed on exactly one PM, while constraint (3) limits the maximum number of VMs that can be migrated for a given task, denoted as L. Additionally, constraint (4) restricts the resource usage by VMs on any given PM from exceeding the total capacity of that PM. Meanwhile, constraints from (5) to (7) ensure that the NUMA resources occupied by VMs do not exceed the corresponding NUMA capacities of the PMs. Constraint (8) further specifies the permissible range of PMs for the migration of each VM ensuring that real VMs do not deploy on unintended PMs, with $\text{mutex}_{i,j} \in \{0,1\}$ indicating whether VM i is prohibited from being placed on PM j.

3.3 Markov Decision Process

The VM rescheduling problem can also be modeled as the Markov Decision Process (MDP), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$. The initial state S_0 of a task corresponds to the initial assignment of VMs to PMs. Starting from this initial state, the agent selects and executes an action a_t from the action space $\mathcal{A}$ at each time step t. Specifically, in the VMR task, this action represents migrating VM i to PM j. The state S_t then transitions to the next state S_{t+1} according to the transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. Concurrently, a reward is provided by the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, which is calculated by the difference between the objective values of two consecutive states: $r_t = \mathcal{F}_t - \mathcal{F}_{t+1}$ (where $\mathcal{F}$ is defined in (1)). The goal of the agent is to maximize the cumulative return $G = \sum_{t=0}^{\infty} \gamma^t r_t$, where $\gamma \in [0,1)$ is the discount factor.

4 IVMR-D: An Industrial-grade VMR Dataset

4.1 Dataset Overview

To assess the performance of existing VMR algorithms on large-scale industrial datasets, we compiled and curated several datasets from a real-world cloud data center, IVMR-D. Specifically, IVMR-D was collected from multiple industrial cloud service clusters available on the Alibaba Cloud, capturing VMR tasks accumulated over a recent period. Each data entry in IVMR-D provides the resource demands and migration costs of VMs, the resource capacities of PMs, the initial assignments of VMs to PMs, and the assignment cost matrix. These elements collectively enable the complete construction of a realistic VMR task, as detailed in Section 3.

4.2 Dataset Classification and Organization

IVMR-D are categorized based on two primary criteria: a) data size and b) specific task.

4.2.1 ***Data Size***. IVMR-D provides three different scales of datasets based on the number of VMs: small, medium, and large. As the number of VMs increases, the number of PMs, as well as the types of VMs and PMs, also grows. Consequently, the scale and complexity of the problem increase correspondingly.

4.2.2 ***Specific Task***. IVMR-D presents two specific tasks that are both common and crucial in VM rescheduling problems: Fragment Minimization (FM) and Network Latency Reduction (NLR).

Fragment Minimization: Resource fragmentation is a phenomenon in cloud computing where the available resources on PMs are divided into small, non-contiguous segments that are insufficient to accommodate new VM instances according to the cloud provider's standard service offerings. Minimizing resource fragments is a pivotal objective in the field of VMR, essential for enhancing the efficiency and profitability of cloud service providers. By assigning negative costs to VMs that match prevalent selling specifications (denoted as S-VM) when they are not placed on their initial PMs, the objective defined in (1) encourages the migration of these S-VMs to other PMs, thereby increasing their inventory and reduce resource fragments.

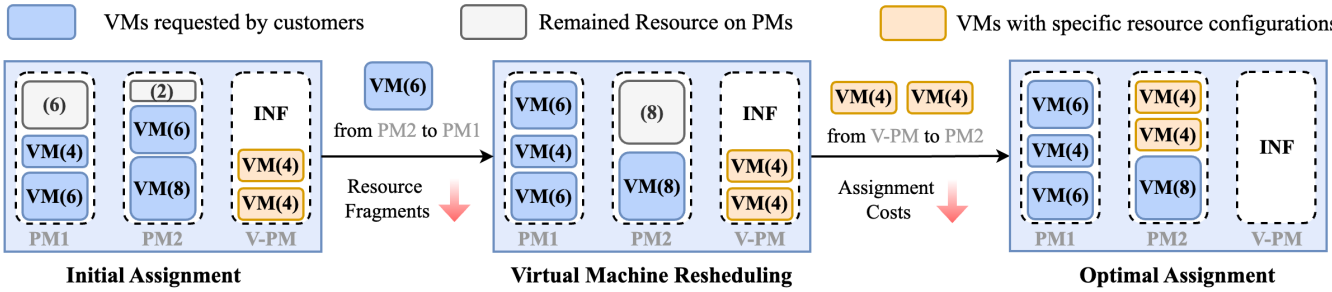

Figure 2: Illustration of Virtual Machine Rescheduling Process to Reduce Fragments. The label in each square marks resources (CPUs in this example) demanded by each VM, such as VM (4) represents this VM requests 4 CPUs. PM1 and PM2 are real Physical Machines (PMs), whereas V-PM represents a Virtual PM with infinite resources (denoted as 'INF' in the figure) that initially hosts Specific Virtual Machines (S-VMs) with particular resource configurations (such as VM(4) highlighted in orange).

Fig. 2 illustrates the task of fragment minimization. Assuming that VMs matching prevalent selling specifications, denoted as Specific VMs (S-VMs), require 4 CPUs, any remaining resources on PMs with fewer than 4 CPUs are deemed as fragments. Initially, PM 1 has 6 CPUs available, while PM 2 has only 2 CPUs left. In this scenario, only PM 1 possesses enough resources to accommodate a S-VM requiring 4 CPUs. The Fragment Rate (FR) is calculated as $((6\%4) + (2\%4))/32 = 12.5\%$. After rescheduling the VM requiring 6 CPUs from PM 2 to PM 1, PM 2 then gains the capacity to host two specific S-VMs each demanding 4 CPUs. Consequently, the resources of PM 1 are fully and efficiently utilized, and the FR is reduced to 0%, which signifies an optimal assignment. In this example, the assignment cost between the orange S-VM and the V-PM is set to 10, while the assignment costs with PM1 and PM2 are both 0. Similarly, the assignment costs for the remaining VMs to all PMs are also 0. This assignment cost matrix incentivizes the S-VM to migrate as much as possible from the V-PM to actual PMs, thereby achieving the goal of minimizing fragmentation.

Network Latency Reduction: User requests for VMs vary in their sensitivity to network latency. By migrating VMs that require high network performance to PMs equipped with substantial network bandwidth, cloud service providers can significantly enhance the quality of service (QoS) and better meet user expectations. To achieve this, different values of $c_{i,j}$ can be assigned based on the network latency requirements of VMs and the available bandwidth of PMs. Specifically, smaller $c_{i,j}$ values are assigned to VMs with higher bandwidth demands, ensuring that the objective function (1) prioritizes the migration of bandwidth-intensive VMs to PMs with substantial bandwidth.

For illustrative purposes, we continue using Fig. 2 as a reference case (note that in this context, the V-PM no longer represents a virtual PM with infinite resources but functions equivalently to PM1 and PM2, and the S-VM is non-existent). The bandwidth capacities of the three PMs follow a descending order: PM2 > PM1 > V-PM. The assignment costs for the orange VM (4) with PM1, PM2, and V-PM are -5, -10, and 10 respectively, while the assignment costs for the blue VM (6) on PM2 with PM1, PM2, and V-PM are 0, -5, and 5 respectively. This cost structure explicitly demonstrates the differential bandwidth requirements among different VMs. The configuration strategically incentivizes the orange VM (4) with higher bandwidth demands to migrate toward PM1 or PM2 with substantial network bandwidth, thereby improving service quality. Compared to the Fragment Minimization task, the Network Latency Reduction task presents a more complex challenge due to the inherent trade-off between bandwidth allocation priorities. Specifically, migrating the blue VM (6) on PM2 to alternative PMs would incur increased costs, necessitating a strategic prioritization: allocating high-capacity PMs to VMs with higher bandwidth requirements while simultaneously satisfying bandwidth demands of other VMs as possible.

4.3 Key Features of IVMR-D

Table 1 presents a comparative analysis of several publicly available datasets designed for VM scheduling or rescheduling tasks. In particular, our dataset exhibits several distinct characteristics in contrast to existing datasets:

Real-world Industrial Relevance. IVMR-D provides a realistic representation of VMR tasks commonly encountered in cloud data centers, offering researchers valuable insights into the practical challenges and complexities of industrial cloud environments.

VM Rescheduling Focus. Unlike previous datasets primarily designed for VMS, ours includes initial VM-to-PM assignments and other essential information required for VMR tasks, facilitating research on rescheduling scenarios. This focus on dynamic VM relocation presents a more complex and relevant challenge.

Scalability and Flexibility. The dataset encompasses a wide range of scales, from hundreds to tens of thousands of VMs and tens to thousands of PMs. This diversity enables algorithm testing and validation across various scales, making it suitable for both small-scale and large-scale research endeavors.

Accessibility. Our dataset is openly accessible via the provided link[1], promoting research transparency and reproducibility. Researchers can easily download and utilize the datasets without additional permissions or complex access procedures.

[1]The dataset and codes are available via https://github.com/MDrW/IVMRSuite-KDD

5 IVMR-B: The first Industrial Benchmark for VMR Algorithms

5.1 Benchmark Overview

In this section, we introduce IVMR-B, the first industrial-scale benchmark for assessing VMR algorithms based on IVMR-D. The benchmark encompasses six representative algorithms across four methodological categories. We have adapted these algorithms with specific modifications to address VMR problems effectively, ensuring they can generate feasible migration sequences while maintaining solution quality.

As shown in Fig. 3, we categorize existing VMR solutions into two paradigms based on the generation strategy of migration sequence. From this perspective, the evaluated algorithms can be classified into two paradigms:

5.1.1 ***Optimal Assignment Based Static Algorithms (OASA).*** Static algorithms focus on finding the optimal final VM-PM mappings as formulated in Section 3.2, then deriving migration sequences by comparing with the initial states. The IVMR-B benchmark includes two representative approaches of OASA:

Optimization Category: The Integer Programming Solver (IP) method [18] formulates the VMR problem as a mathematical optimization model and solves the optimization problem by commercial solvers. By comparing the optimal assignment with the initial state, we can derive migration sequences for benchmarking against other algorithms.

Metaheuristic Category: Two classical metaheuristic approaches, GA and ACO are developed in the IVMR-B benchmark. For GA [14], we incorporate specialized crossover and mutation operators that maintain feasible solutions throughout each generation. For ACO [10], we designed a repair mechanism that addresses infeasible solutions during the solution construction process.

5.1.2 ***Migration Planning Based Dynamic Algorithms (MPDA).*** Dynamic algorithms directly optimize the migration sequence through the MDP in Section 3.3 during the migration process. This category comprises:

Heuristic Category: The Greedy Algorithm (HGA) [4], as the baseline solution of VMR, employs an intuitive strategy that prioritizes migration actions based on immediate rewards to optimize the solution path.

Machine Learning-based Category: Both RL and IL approaches are implemented in the IVMR-B benchmark. For RL [8], we modified the environment modeling and observation space to ensure compatibility with IVRset specifications. Moreover, a novel IL-based VMR solution is developed that leverages migration sequences from IP solvers as expert demonstrations to train graph neural networks.

5.2 Evaluation Metrics

We evaluate algorithms from three fundamental perspectives: ***solution quality, computational efficiency, and cross-scale scalability***. Solution quality measures the accumulated reward of migration sequences across different data sizes ranging from "Small" to "Large". Computational efficiency assesses the evaluation time required by each algorithm for practical deployments in industrial settings. Cross-scale scalability evaluates the algorithms' generalization capability by training on "Small" tasks and testing performance on "Large" tasks, reflecting the adaptability to real-world industrial deployments of varying scales.

5.3 Evaluation Results

The experimental results demonstrate diverse performance characteristics across different algorithms. Notably, for machine learning-based methods, considering their requirement for consistency in data distribution, samples were randomly selected at a percentage of 10% from each task type. Separate models were then trained on these distinct task samples to assess solution quality and computational efficiency.

5.3.1 ***Solution Quality.*** We evaluate the solution quality using the Cumulative Reward (CR) metric, which is formally defined as the relative improvement of the objective function $\mathcal{F}$ (defined in (1)) between the initial state S_0 and the final state S_T. A higher CR value indicates superior performance. The IP optimization algorithm, which guarantees optimal solutions for most tasks, is employed as the performance upper bound. To ensure a clear and fair comparison, all results are normalized against the CR of the IP algorithm CR_{IP}, denoted as the Relative Cumulative Reward RCR_{algo}.

$$RCR_{algo} = \frac{CR_{algo}}{CR_{IP}} = \frac{\mathcal{F}_{algo}(S_T) - \mathcal{F}(S_0)}{\mathcal{F}_{IP}(S_T) - \mathcal{F}(S_0)} \tag{10}$$

Here, $\mathcal{F}(S_0)$ is the objective function value of the initial state, while $\mathcal{F}_{algo}(S_T)$ and $\mathcal{F}_{IP}(S_T)$ represent the objective function values of the final states achieved using algorithms *algo* and IP respectively. Fig. 4 presents the comparative results of the algorithms in terms of their RCR across the IVMR-D, revealing several key findings:

Cross-algorithm Comparison: While the IP algorithm achieves the optimal solutions across all datasets, it serves primarily as a theoretical upper bound due to its computational complexity. Among practical algorithms, IL demonstrates the strongest performance, reaching 91.5% of RC_{IP} on average, with near-optimal results on small-scale and medium-scale datasets and 81.6% performance on large-scale dataset. This suggests the potential of machine learning-based approaches when sufficient expert demonstrations are available. The algorithms RL, GA and ACO demonstrate comparable performance across various datasets, achieving average RCR of 76.7%, 77.9% and 82.1% over all datasets respectively. Notably, their RCR decreases to approximately 70% on medium-scale and large-scale datasets, indicating their limitations in handling complex tasks. The basic heuristic approach HGA performs poorly across most datasets, achieving only an average RCR_{HGA} of 63.1%, with performance dropping below 56.9% on large-scale instances.

Impact of the Data Size: As the data size increases from small to large, the relative performance of the algorithms deteriorates significantly. Specifically, the RCR metric of IL drops from 93.7% to 81.6%, RL from 80.3% to 66.4%, GA from 81.4% to 67.5%, ACO from 88.3% to 69.3%, and HGA from 63.1% to below 60%. This trend highlights the challenges these algorithms face in maintaining high performance as task complexity grows, emphasizing the need for the algorithms to handle large-scale problems effectively.

Impact of the Task: In the NLR task, compared to the FM task, the RCR of HGA increases from 60.4% to 91.2%, while RL, GA and ACO drop from 70% to below 60%. This indicates that

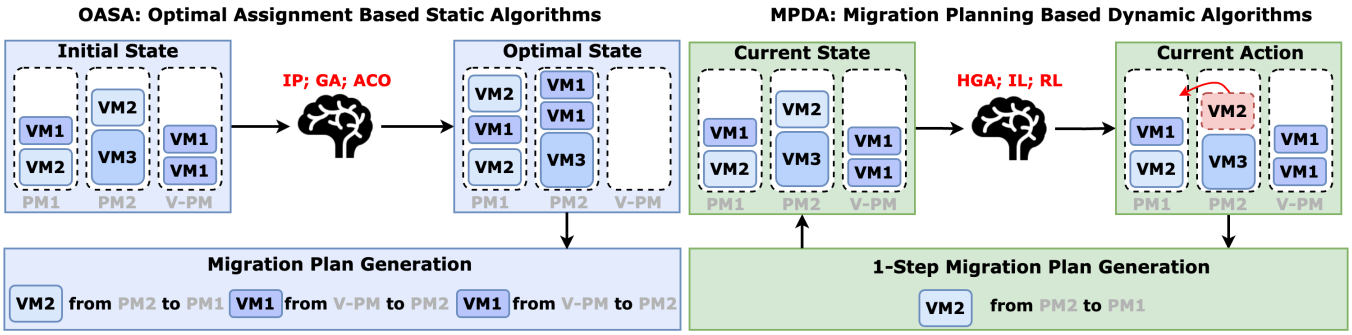

Figure 3: Principle Comparison of OASA and MPDA

RL, GA and ACO struggle to effectively migrate VMs with higher assignment costs on PMs and replace them with VMs that have lower costs, thereby failing to cumulate the positive values from these migrations. In contrast, heuristic algorithms aimed to obtain directly positive rewards perform more effective in the NLR task.

5.3.2 ***Computational Efficiency.*** We measure computational efficiency using the Evaluation Time (ET) metric, which captures the execution time ET_{algo} required to solve a task, excluding the training phase for machine learning-based methods. For the IP algorithm, ET encompasses both the modeling and solving time. The acceleration ratio AT_{algo} of an algorithm compared to the IP algorithm in terms of ET can be formally defined as:

$$AT_{algo} = \frac{ET_{IP}}{ET_{algo}} \tag{11}$$

Fig. 5 presents the ET comparison across all datasets, and several conclusions can be drawn:

Cross-algorithm Comparison: The HGA method stands out as the fastest, consistently exhibiting the shortest evaluation time across all datasets. The two machine learning-based approaches, IL and RL-based methods, demonstrate comparable efficiency and outperform other methods, except for HGA, in terms of ET. Specifically, IL achieves an average acceleration in ET that is 5.43 times faster than the IP algorithm, while RL demonstrates an even more substantial improvement, with an average acceleration of 13.1 times. The two metaheuristic algorithms, GA and ACO, exhibit similar ET across various datasets, aligning closely with the ET of IP on small and medium-scale datasets. However, on large-scale datasets, IP requires an average ET of 4000 seconds, while GA and ACO need roughly half the time, averaging around 2000 seconds.

Impact of the Data Size: As the data size expands from small to large, the evaluation time required by the algorithms correspondingly rises from tens of seconds to thousands of seconds, indicating the substantial impact of data size on computational efficiency. Specifically, the ET of MPDA (HGA, IL and RL) is less influenced by the problem scale, with ET on large-scale datasets remaining around hundreds of seconds, which is an acceptable level for practical applications. In contrast, the ET of OASA (IP, GA and ACO) escalates sharply with increasing problem scale, reaching over two thousand seconds on large-scale datasets, thereby demonstrating that these algorithms face significant efficiency bottlenecks for industrial deployment.

5.3.3 ***Cross-scale Scalability.*** For machine learning-based algorithms, scalability primarily assesses the generalization capability of models trained on small-scale datasets when applied to medium and large-scale datasets. As shown in Table 2, compared to models trained on data across all scales, the RCR of IL and RL exhibits varying degrees of decline when evaluated on medium and large-scale problems. Specifically, the RCR of IL decreases from 81.6% to 77.0% on large-scale data, and it drops significantly from 91.1% to 72.4% when evaluated on the NLR task which differs from the FM task used for training. For algorithms other than machine learning-based methods, scalability evaluates the solution quality and efficiency of these algorithms on large-scale problems. As illustrated in Fig. 4 and Fig. 5, on large-scale problems, the IP algorithm achieves high solution quality but suffers from low efficiency, while metaheuristic algorithms exhibit low solution quality and poor efficiency. This indicates that these algorithms struggle with scalability as the data size increases to large scales.

Table 2: Scalability of Machine Learning-based Methods.

Algorithms	Medium (FM)	Large (FM)	Medium (NLR)
IL	86.4%	77.0%	72.4%
RL	71.2%	62.1%	51.7%

Comprehensive Analysis. Fig. 6 illustrates the relationship between ET and RCR for each data point in IVMR-D. The x-axis represents the logarithm of ET, which provides a rough indication of the scale and complexity of the data, with higher values generally corresponding to more challenging problems. The y-axis indicates RCR, reflecting the solution quality of the algorithms. The figure displays fitted curves for the scatter points, with marked coordinates representing the start and end points of these curves. Additionally, box plots showing the upper and lower 25% around the median are included to provide a clearer visualization. From the figure, it can be observed that the solution quality of machine learning-based and metaheuristic algorithms tends to decrease as problem difficulty increases. Among them, IL demonstrates the smallest decline, highlighting its overall effectiveness and significant potential in balancing solution quality and computational

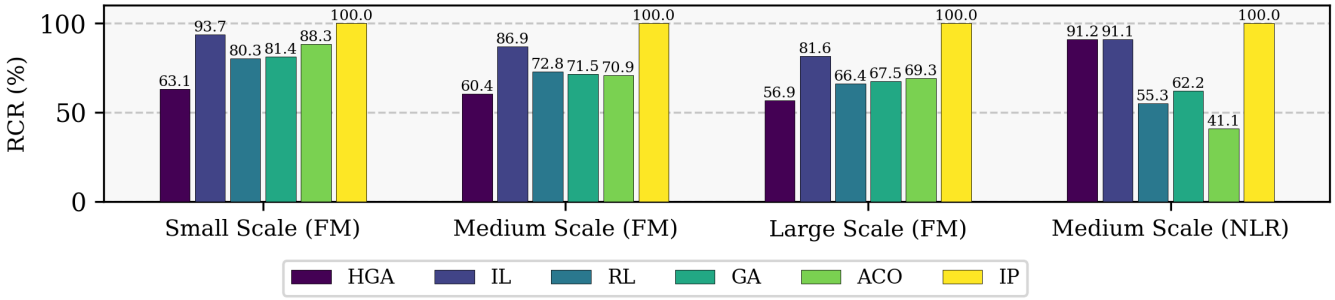

Figure 4: Relative Cumulative Reward of different Algorithms across IVMR-D

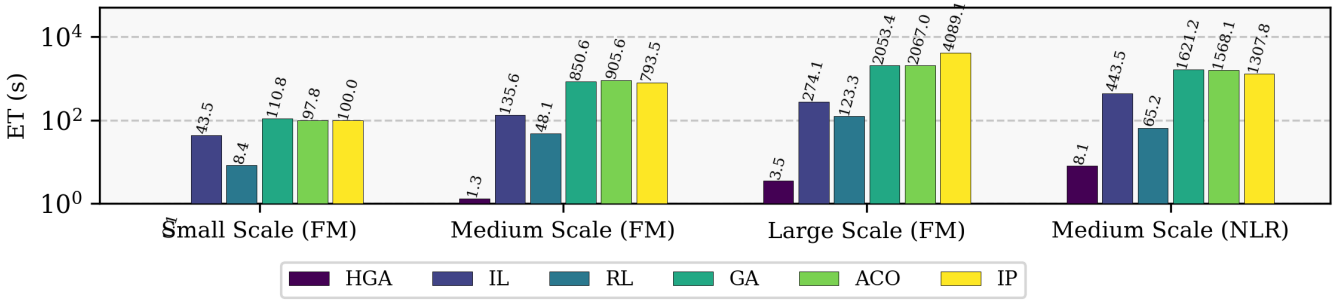

Figure 5: Evaluation Time of different Algorithms across IVMR-D

efficiency. In contrast, GA and ACO exhibit a more pronounced decline, indicating that their performance is more sensitive to problem difficulty. Furthermore, the x-axis values reveal that machine learning-based methods generally achieve better computational efficiency compared to metaheuristic algorithms. Notably, HGA shows a distinct trend compared to other algorithms. This is because the ET of HGA is more influenced by the number of positive reward actions rather than the problem difficulty. However, it is evident that HGA's solution quality remains relatively low across most datasets.

6 Challenges and Future

Based on our comprehensive evaluation using IVMR-D and IVMR-B, we identify several critical challenges in existing VMR algorithms.

Optimization. IP algorithm suffers from efficiency issues, especially when dealing with large-scale datasets, which severely limits its practical deployment in industrial settings. However, it is worth noting that with sufficient computational resources and time, the optimization algorithm can provide a theoretical upper bound for solution quality. The development of advanced acceleration methods for optimization holds the potential to enhance its efficiency, thereby facilitating broader application in industrial scenarios.

Metaheuristic. These algorithms similarly encounter efficiency challenges when tackling large-scale problems. The search efficiency within the feasible solution space is often compromised because infeasible solutions are discarded until feasible solutions are identified for updates and evolution in an iteration. This limitation underscores the need to explore methods that can conduct more efficient searches within the feasible space, thereby improving overall efficiency. Additionally, these algorithms are prone to becoming trapped in the local optima, making it difficult to escape and identify better solutions. Developing strategies to effectively escape local optima could thus significantly enhance the solution quality of metaheuristic algorithms.

Heuristic. Heuristic algorithms have high efficiency, but the solution quality is poor. Designing sophisticated heuristic algorithms tailored to the problem can improve algorithm performance and yield satisfactory solutions. For example, the finely designed heuristic algorithm running in our production cloud environment achieves an average RCR of 89.3% of IP on IVMR-D, within just tens of seconds.

Machine Learning. The performance of IL is limited by the availability and quality of expert demonstrations. When ample high-quality expert data is accessible, IL exhibits both strong effectiveness and efficiency. Conversely, when only limited expert data from small-scale datasets is available, the performance of IL becomes less satisfactory. Moreover, obtaining plenty of high-quality expert demonstrations for large-scale problems is often difficult due to resource limitations and computational efficiency, thereby posing significant challenges for the broader application of IL. In this context, RL, which does not rely on expert demonstrations, emerges as a promising alternative. However, RL underperformed in our experiments, failing to meet the expected performance. This is primarily attributable to two factors: the design of the reward function

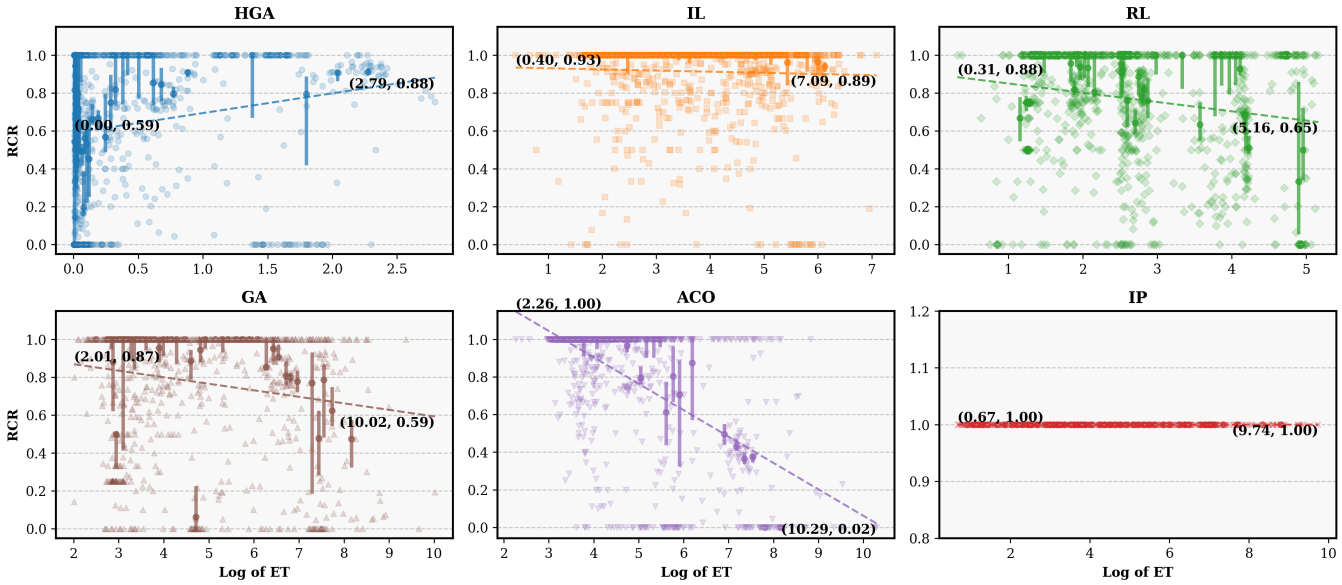

Figure 6: Comprehensive Analysis of Algorithm Performance: ET & RCR across Full Datasets

and the vastness of the action space. The reward design, based on the objective function, fails to distinguish whether an intermediate action that does not immediately produce a positive reward is necessary for subsequent positive reward relocation. As a result, by observing the sequence of decisions made by RL, it can be seen that RL generates many redundant actions. These actions neither immediately produce positive rewards nor contribute to obtaining subsequent positive rewards. This significantly undermines the cumulative reward of RL, particularly when the maximum number of actions is limited. Besides, as the scale of VMs and PMs increases, the action space grows exponentially, triggering the curse of dimensionality that undermines the effectiveness of softmax-based action selection. Consequently, the results in drastically diminished exploration efficiency and destabilized the training process. Moreover, combining IL with RL may yield better performance and offers substantial promise for effectively addressing VMR.

7 Conclusion

In this paper, we presented **IVMR suite**, the first industrial-scale suite for Virtual Machine Rescheduling research. Our contributions include IVMR-D, a comprehensive collection of real-world VMR datasets with complex constraints, and IVMR-B, a unified benchmark for evaluating VMR algorithms. Our experiments revealed significant limitations in current algorithms when applied to industrial-scale problems, highlighting the gap between academic research and industrial requirements. IVMR establishes a foundation to accelerate progress toward more effective cloud resource management solutions.

References

[1] Fasih Abdullah, Muhammad Faraz Ud Din Razi, Muhammad Aleem, Akhtar Jamil, and Alaa Ali Hameed. 2024. A Comparative Analysis of Cloud Load Balancing Algorithms Using CloudSim Simulations. In *International Conference on Advanced Engineering, Technology and Applications*. Springer, 133–150.

[2] Tobias Achterberg. 2019. What's new in Gurobi 9.0. *Webinar Talk url: https://www.gurobi. com/wp-content/uploads/2019/12/Gurobi-90-Overview-Webinar-Slides-1. pdf* 5, 9 (2019), 97–113.

[3] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. 2010. A view of cloud computing. *Commun. ACM* 53, 4 (2010), 50–58.

[4] Anton Beloglazov and Rajkumar Buyya. 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 (2012), 1397–1420.

[5] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience* 41, 1 (2011), 23–50.

[6] Yue Cheng, Zheng Chai, and Ali Anwar. 2018. Characterizing Co-located Datacenter Workloads: An Alibaba Case Study. *CoRR* abs/1808.02919 (2018). arXiv:1808.02919 http://arxiv.org/abs/1808.02919

[7] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 153–167.

[8] Xianzhong Ding, Yunkai Zhang, Binbin Chen, Donghao Ying, Tieying Zhang, Jianjun Chen, Lei Zhang, Alberto Cerpa, and Wan Du. [n. d.]. VMR2L: Virtual Machines Rescheduling Using Reinforcement Learning in Data Centers. ([n. d.]).

[9] Thuan Duong-Ba, Tuan Tran, Thinh Nguyen, and Bella Bose. 2018. A dynamic virtual machine placement and migration scheme for data centers. *IEEE Transactions on Services Computing* 14, 2 (2018), 329–341.

[10] Fahimeh Farahnakian, Adnan Ashraf, Tapio Pahikkala, Pasi Liljeberg, Juha Plosila, Ivan Porres, and Hannu Tenhunen. 2014. Using ant colony system to consolidate VMs for green cloud computing. *IEEE transactions on services computing* 8, 2 (2014), 187–198.

[11] Wenxia Guo, Wenhong Tian, Yufei Ye, Lingxiao Xu, and Kui Wu. 2020. Cloud resource scheduling with deep reinforcement learning and imitation learning. *IEEE Internet of Things Journal* 8, 5 (2020), 3576–3586.

[12] Sanjay Gupta and Sarsij Tripathi. 2024. A comprehensive survey on cloud computing scheduling techniques. *Multimedia Tools and Applications* 83, 18 (2024), 53581–53634.

[13] Bahram Hajimirzaei and Nima Jafari Navimipour. 2019. Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm. *Ict Express* 5, 1 (2019), 56–59.

[14] Jinhua Hu, Jianhua Gu, Guofei Sun, and Tianhai Zhao. 2010. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In *2010 3rd International symposium on parallel architectures, algorithms and programming*. IEEE, 89–96.

[15] Altaf Hussain, Muhammad Aleem, Muhammad Azhar Iqbal, and Muhammad Arshad Islam. 2019. Investigation of cloud scheduling algorithms for resource utilization using cloudsim. *Computing and Informatics* 38, 3 (2019), 525–554.

[16] Gastón Keller, Michael Tighe, Hanan Lutfiyya, and Michael Bauer. 2012. An analysis of first fit heuristics for the virtual machine relocation problem. In *2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualiztion management (svm)*. IEEE, 406–413.

[17] Dinesh Kumar and Zahid Raza. 2015. A PSO based VM resource scheduling model for cloud computing. In *2015 IEEE international conference on computational intelligence & communication technology*. IEEE, 213–219.

[18] Somnath Mazumdar and Marco Pranzo. 2017. Power efficient server consolidation for cloud data center. *Future Generation Computer Systems* 70 (2017), 4–16.

[19] Said Nabi, Masroor Ahmad, Muhammad Ibrahim, and Habib Hamam. 2022. AdPSO: adaptive PSO-based task scheduling approach for cloud computing. *Sensors* 22, 3 (2022), 920.

[20] Charles Reiss, John Wilkes, and Joseph L Hellerstein. 2011. Google cluster-usage traces: format + schema. In *Technical Report*. Google Inc.

[21] Maria A Rodriguez and Rajkumar Buyya. 2017. Budget-driven scheduling of scientific workflows in IaaS clouds with fine-grained billing periods. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 12, 2 (2017), 1–22.

[22] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*. Springer, 593–607.

[23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[24] Nikita Sharma and Sudarshan Maurya. 2019. SLA-based agile VM management in cloud & datacenter. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. IEEE, 252–257.

[25] Michael Tighe, Gaston Keller, Michael Bauer, and Hanan Lutfiyya. 2012. DCSim: A data centre simulation tool for evaluating dynamic virtualized resource management. In *2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualiztion management (svm)*. IEEE, 385–392.

[26] Chen Wei, Zhi-Hua Hu, and You-Gan Wang. 2020. Exact algorithms for energy-efficient virtual machine placement in data centers. *Future Generation Computer Systems* 106 (2020), 77–91.

[27] Wei-Tao Wen, Chang-Dong Wang, De-Shen Wu, and Ying-Yan Xie. 2015. An ACO-based scheduling strategy on load balancing in cloud computing environment. In *2015 Ninth international conference on frontier of computer science and technology*. IEEE, 364–369.

[28] John Wilkes. 2020. Yet more Google compute cluster trace data. Google research blog. Posted at https://ai.googleblog.com/2020/04/yet-more-google-compute-cluster-trace.html..

[29] Timothy Wood, Prashant J Shenoy, Arun Venkataramani, Mazin S Yousif, et al. 2007. Black-box and Gray-box Strategies for Virtual Machine Migration.. In *NSDI*, Vol. 7. 17–17.

[30] Jiaxi Wu, Wenquan Yang, Xinming Han, Yunzhe Qiu, and Andrei Gudkov. 2022. A robust approach for hotspots prevention and resolution in cloud services. *Available at SSRN 4289162* (2022).

[31] Guangshun Yao, Yongsheng Ding, Lihong Ren, Kuangrong Hao, and Lei Chen. 2016. An immune system-inspired rescheduling algorithm for workflow in cloud systems. *Knowledge-Based Systems* 99 (2016), 39–50.

[32] Jing Zeng, Ding Ding, Kaixuan Kang, HuaMao Xie, and Qian Yin. 2022. Adaptive DRL-based virtual machine consolidation in energy-efficient cloud data center. *IEEE Transactions on Parallel and Distributed Systems* 33, 11 (2022), 2991–3002.

[33] Mengyuan Zhang, Wotao Yin, Mengchang Wang, Yangbin Shen, Peng Xiang, You Wu, Liang Zhao, Junqiu Pan, Hu Jiang, and KuoLing Huang. 2023. Mindopt tuner: Boost the performance of numerical software by automatic parameter tuning. *arXiv preprint arXiv:2307.08085* (2023).

[34] Nan Zhang, Xiaolong Yang, Min Zhang, Yan Sun, and Keping Long. 2018. A genetic algorithm-based task scheduling for cloud resource crowd-funding model. *International Journal of Communication Systems* 31, 1 (2018), e3394.

[35] Qi Zhang, Lu Cheng, and Raouf Boutaba. 2010. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications* 1, 1 (2010), 7–18.

[36] Ao Zhou, Shangguang Wang, Ching-Hsien Hsu, Qibo Sun, and Fangchun Yang. 2016. Task rescheduling optimization to minimize network resource consumption. *Multimedia Tools and Applications* 75 (2016), 12901–12917.

A Dataset Details

IVMR-D processes and structures industrial VM rescheduling data from Alibaba Cloud into standardized, well-formatted and research-ready datasets. Each data consists of the following components:

item_infos.csv: Contains the migration cost a_i, required basic resources, and the NUMA resources (including the number of required NUMA and the demand for each basic resource within NUMA) of each virtual machine (VM).

box_infos.csv: Specifies the maximum capacities of basic resources and NUMA resources provided by each physical machine (PM).

init_placement.npz: Stores a matrix representing VM-PM initial assignments, where each element indicates whether VM i is initially placed on PM j.

item_assign_box_cost.npz: Provides an assignment cost matrix, where each element $c_{i,j}$ denotes the assignment cost of deploying VM i on PM j.

item_mutex_box.npz: Defines a mutex matrix between VMs and PMs, where each element $\text{mutex}_{i,j}$ indicates whether VM i and PM j are mutually exclusive (0 for no, 1 for yes).

config.json: Specifies the maximum number of migration L and task category (Fragment Minimization or Network Latency Reduction).

B Algorithm Details

In this section, we provide a concise summary of several algorithms in our benchmark, ranging from optimal-assignment-based static methods as well as migration-planning-based dynamic methods.

Integer Programming Solver (IP). Firstly, the VM rescheduling task is formulated as an optimization problem based on the mathematical model detailed in Section 3. Subsequently, to obtain an optimal assignment, off-the-shelf solvers such as CPLEX, Gurobi [2] and MindOpt [33] can be employed to solve this problem, leveraging their built-in exact optimization techniques such as branch and bound, and cutting planes. Specifically, this paper utilizes CPLEX to model and solve the problem.

Genetic Algorithm (GA). The genetic algorithm is an evolutionary metaheuristic that approximates the optimal VM-to-PM assignment by emulating genetic inheritance. Specifically, it begins with generating a random initial population of candidate solutions. Each solution's fitness is then assessed using a function aligned with the objective function. Superior individuals are selected as parents for the next generation. The algorithm proceeds with crossover, exchanging genetic information between parents to produce offspring, followed by mutation, which randomly alters offspring traits to maintain diversity. This process iterates until termination criteria are met.

To ensure compliance with constraints (2) through (9), we modified the crossover and mutation operations. When these operations yield infeasible solutions, we identify non-compliant PMs, de-assign VMs from these PMs, and then reassign the affected VMs to suitable PMs sequentially. If the solution remains infeasible after this process, it is discarded. This approach maintains the feasibility of the solution space within the genetic algorithm while exploring diverse VM-to-PM assignments.

Ant Colony Optimization (ACO). ACO is a metaheuristic inspired by ant foraging behavior, designed to approximate solutions to the VMR problem. It begins with a colony of artificial ants exploring the solution space of VM-to-PM assignments. Each ant constructs a solution by probabilistically selecting VMs and target PMs, guided by pheromone trails and heuristic information. After solution construction, the quality of each solution is evaluated, and pheromone trails are updated accordingly. High-quality solutions receive more pheromones, intensifying their trails, while less effective ones undergo evaporation. This process iterates, gradually converging towards optimal solutions.

The ACO algorithm involves initialization of the ant colony, solution construction, evaluation, pheromone update, and termination check. During construction, if no feasible PM is available for a VM, it is temporarily allocated to its initial PM. For infeasible solutions, a repair mechanism similar to that in GA is applied. If a solution remains infeasible after repair attempts, it is assigned an infinite objective value, ensuring only feasible solutions influence pheromone trail updates.

Heuristic Greedy Algorithm (HGA). The Greedy algorithm provides an efficient method for addressing the VMR problem by making a locally optimal choice at each step to find a global optimum. This locally optimal choice is guided by a heuristic function that estimates the quality of each candidate action.

In this paper, the heuristic is calculated by the changes in the objective function resulting from the relocation of a VM to a new PM. Utilizing this heuristic, the algorithm selects the VM for migration and the corresponding target PM that appears to be the best immediate choice. The process is repeated iteratively, with each step building upon the previous one to gradually construct a complete migration sequence.

Imitation Learning (IL). Imitation learning addresses the large-scale VM rescheduling problem by training a model to imitate optimal rescheduling strategies demonstrated by expert schedulers, such as high-performance algorithms. The main components of imitation learning for VMR problem are detailed as follows:

- **Data Collection**: We use migration sequences generated by the IP algorithm as expert data. For each VMR task, IP identifies optimal VM migrations and target PMs, creating tuples of (state, migrated VM, target PM). The state serves as the input feature of the IL model, while the migrated VM and target PM act as labels for the imitation learning model.
- **Feature Representation**: To facilitate the extraction of high-level features, the state is represented as a bipartite graph $\mathcal{G}$ with node and edge features $(\mathbf{V}, \mathbf{P}, \mathbf{E})$. On one side of the graph, nodes represent VMs, characterized by their required resources and migration costs, denoted as $\mathbf{V} \in \mathbb{R}^{m \times c}$. On the other side, nodes represent PMs, with their available resources as features, denoted as $\mathbf{P} \in \mathbb{R}^{n \times d}$. The assignment of VMs to PMs is utilized to represent the edges $\mathbf{E} \in \{0, 1\}^{m \times n}$, indicating the connections between VMs and PMs in the current state.
- **Imitation Learning Architecture**: To accommodate the varying numbers of VMs and PMs across different tasks, we employ Graph Convolutional Networks (GCNs) [22] as the foundational model for training. The model takes the state representation $(\mathbf{V}, \mathbf{P}, \mathbf{E})$ as input and performs 2-layer graph convolution, with each layer structured as two interleaved half-convolutions. Following these graph-convolutional layers, each node is enriched with information from its neighbors. Subsequently, a two-stage policy is implemented. In the first stage, a 2-layer Fully Connected Network (FCN) is applied on the VM nodes, combined with a masked softmax activation, to predict the probability distribution over the candidate VMs. In the second stage, another 2-layer FCN and a mask softmax activation are applied on the PM nodes, incorporating the features of the selected VM, to determine the target PM.
- **Training Process**: The model is updated using the Stochastic Gradient Descent (SGD) algorithm on the collected expert dataset. To minimize the divergence between the optimal actions and the model's predictions, a cross-entropy loss function is employed. Specifically, the loss function is applied to align the labels of the migrated VM with the predicted probability distribution over candidate VMs, and to align the labels of the target PM with the predicted probability distribution over candidate PMs. Consequently, the model progressively converges towards the optimal strategies through the iterative update process.

Reinforcement Learning (RL). VMR2L introduces a novel RL framework that learns optimal rescheduling policies through direct interaction with a simulated environment while maintaining strict inference time constraints. Unlike IL which requires expert demonstrations, VMR2L learns from trial and error, and unlike HGA which makes locally optimal choices, it optimizes for long-term performance. The algorithm comprises several integrated components, which are detailed as follows:

- **State-Action Representation**: Similar to IL's feature representation, the state space comprises comprehensive system information encoded in a bipartite graph structure. For each PM, the state is characterized by a 10-dimensional feature vector, including basic PM properties, resource state and migration cost statistics. For VMs, the state encompasses 12 features including basic migration properties, resource usage metrics and cost-related features. The action space is represented as a tuple (k, i), where k denotes the VM to be migrated and i indicates the target PM.
- **VMR2L Architecture**: VMR2L introduces a novel hierarchical framework that combines sparse attention mechanisms with a two-stage decision process for VM rescheduling. At its core, the model implements tree-structured attention patterns to capture the intrinsic relationships between VMs and PMs, enabling VMs to recognize other VMs co-hosted on the same PM while filtering out noise from unrelated pairs. This hierarchical attention computation serves as input to a two-stage decision pipeline: First, the VM actor stage employs a neural network to select candidate VMs for migration, utilizing the current state information to generate a probability distribution over all possible VM choices. Then, the PM actor stage determines the optimal destination PM by incorporating resource constraints through invalid PM masking.
- **Training Process**: VMR2L employs the Proximal Policy Optimization (PPO) [23] algorithm to learn the rescheduling strategy. The training objective of the actor-network is maximizing the expected long-term returns while maintaining stable policy updates through probability ratio clipping. The reward function integrates three essential components: migration costs C_m, PM utilization costs C_u, and VM-PM assignment costs C_a. The sparse attention module and two-stage framework are optimized to learn coordinated VM selection and PM placement policies. Besides, the system employs parameter sharing between the critic network and VM actor through specialized node configurations, enabling efficient value estimation while reducing overall model complexity.