Karaelmas Science and Engineering Journal
Journal home page: https://dergipark.org.tr/tr/pub/karaelmasfen
DOI: 10.7212/karaelmasfen.1794485

Research Article
Received / Geliş tarihi : 01.10.2025
Accepted / Kabul tarihi : 24.11.2025

Digital Twin-Enabled Resource Allocation and Performance Optimisation in Cloud-Based Complex Data Workflows: A Particle Swarm Optimisation Approach

Bulut Tabanlı Karmaşık Veri İş Akışlarında Dijital İkiz Destekli Kaynak Tahsisi ve Performans Optimizasyonu: Parçacık Sürüsü Optimizasyonu Yaklaşımı

Ümit Demirbaga*

Bartın University, Engineering, Architecture and Design Faculty, Department of Computer Engineering, Bartın, Türkiye

Abstract

In large-scale cloud-based big data systems, optimal resource allocation remains challenging due to the variety of workloads and limited computational resources. Using digital twin technology, we demonstrate our methodology through an optimisation approach to resource allocation and performance tuning for a cloud-based, complex data processing application. Contrary to the conventional heuristic or genetic algorithms popular in the literature, this paper applies Particle Swarm Optimisation (PSO) to efficiently solve the task-to-resource assignment problem. We obtain a synthetic yet realistic workload that includes task execution times, resource usage profiles, and data locality characteristics. The model above addresses the issue by minimising the makespan while imposing penalty constraints for infeasible scheduling and minimising CPU and memory utilisation. Experimental results show that the proposed PSO solution achieves better makespan reduction and improved resource utilisation than the baseline greedy allocation method across various multi-VM configurations. The proposed approach offers a novel and valuable optimisation tool that can be widely applied to cloud workload management by integrating digital twin frameworks.

Keywords: Cloud computing, digital twin, particle swarm optimisation, performance optimisation, resource allocation.

Öz

Büyük ölçekli bulut tabanlı büyük veri problemlerinde, çeşitli iş yükleri ve sınırlı hesaplama kaynakları nedeniyle optimal kaynak tahsisi hâlâ önemli bir zorluktur. Bu çalışmada dijital ikiz teknolojisini kullanarak, kaynak tahsisi ve performans ayarlaması için bir bulut tabanlı karmaşık veri işleme uygulamasında optimizasyon temelli bir yöntem önerilmiştir. Literatürde yaygın olarak kullanılan geleneksel sezgisel veya genetik algoritmaların aksine, bu çalışmada görev-kaynağa atama problemini etkin bir şekilde çözmek amacıyla Parçacık Sürüsü Optimizasyonu (PSO) uygulanmıştır. Gerçekçi ve sentetik bir iş yükü veri seti oluşturularak görev yürütme süreleri, kaynak kullanım profilleri ve veri yerelliği özellikleri dikkate alınmıştır. Geliştirilen model, uygun olmayan planlamalar için ceza kısıtları altında toplam işlem süresini minimize etmeyi ve aynı zamanda CPU ve bellek kullanımını dengelemeyi hedeflemektedir. Deneysel sonuçlar, önerilen PSO çözümünün, çoklu VM konfigürasyonları altında, temel greedy tahsis yöntemine kıyasla daha iyi makespan azaltımı ve kaynak kullanım dengesi sağladığını göstermektedir. Önerilen yaklaşım, dijital ikiz çerçeveleri ile bütünleştirilerek bulut iş yükü yönetimi alanında geniş uygulama potansiyeline sahip yeni ve etkili bir optimizasyon aracı sunmaktadır.

Anahtar Kelimeler: Bulut bilişim, dijital ikiz, parçacık sürüsü optimizasyonu, performans optimizasyonu, kaynak tahsisi.

1. Introduction

The widespread deployment of cloud computing technologies has significantly changed how data-intensive systems are supported, deployed, and optimised across many industrial domains (Demchenko et al., 2017). The ever-increasing volumes of big data generated from scientific experiments, sensor networks, industrial Internet of Things (IoT) devices, smart cities, and health monitoring systems pose enormous challenges in big data storage, resource management, and task scheduling (Demirbaga et al., 2024). Although cloud architectures enable highly flexible computing environments, resource allocation remains a major issue, directly affecting system throughput, makespan, quality of service (QoS), and energy usage (Ghafari et al., 2022).

*Corresponding author: udemirbaga@bartin.edu.tr

Ümit Demirbaga orcid.org/0000-0001-5159-0723

Resource allocation for cloud-based big data processing systems determines how to map heterogeneous tasks to various virtual machine (VM) configurations, thereby affecting computing capacity, memory bandwidth, and network resources (Dua et al., 2024). Poor resource allocation may result in several performance problems, including resource deficiency, under-utilisation, service-level agreement (SLA) violations, and inefficient processing, ultimately contributing to increased operational costs (Zeng et al., 2024). These issues become even more challenging to manage in dynamic workloads, where work items arrive with varying resource demands and lifetimes. Many studies in the literature have addressed the cloud resource allocation task with various methods, from elementary heuristics to complex meta-heuristic algorithms. Traditional scheduling algorithms, such as First-Come-First-Served (FCFS), Round Robin, and Greedy heuristics, are conceptually simpler to implement and computationally less expensive. However, they are often unable to manage complex resource requests effectively. In contrast, meta-heuristic methods, such as those based on the Genetic Algorithm (GA), Ant Colony Optimisation (ACO), Simulated Annealing (SA), and hybrid models, have shown higher adaptability to heterogeneous workload patterns (Jomah & Aji, 2024). However, such methods often suffer from premature convergence, oversensitivity to hyperparameter configurations, low computational efficiency, and limited scalability for handling massive data.

Digital twin technology has recently emerged as a promising paradigm for simulating, controlling, and optimising complex systems in real time (Aggarwal et al., 2024). Derived from industry, and first applied in sectors such as manufacturing, predictive maintenance, and smart energy systems, digital twins replicate the operational conditions of physical systems to support predictive analytics, real-time optimisation, and improved decision-making (Kabir et al., 2024). Nevertheless, applying digital twin ideas to cloud resource scheduling for data processing workflows remains a relatively under-researched area. The complex dependency relationships among task execution behaviour, resource consumption patterns, and data locality factors can be captured by digital twins, providing significant guidance for resource scheduling decisions.

In the cloud resource allocation problem, PSO has demonstrated strong performance in optimising complex multi-objective problems (Malti et al., 2024). By enabling the optimisation of different scheduling configurations, PSO can identify better task-to-resource optimisation to trade off utilisation and minimise both make span and resource utilisation. The parallel nature of PSO algorithms offers additional benefits when applied in large-scale distributed cloud frameworks. Although digital twins and PSO-based methods have attracted increased attention in the literature, few studies have applied both techniques to resolve resource allocation problems in cloud-based big data workflows. Most current work focuses on energy-efficient scheduling or anomaly detection in cloud systems (Singh & Chaturvedi, 2024; Ni et al., 2025). However, there are few efforts to achieve complete performance-oriented scheduling integrated with digital twin modelling.

To address these gaps, this paper introduces a new digital twin-enabled resource allocation and performance optimisation scheme tailored for the Cloud-based complex data workflows. The proposed approach combines the predictability features of digital twins with the global search strength of PSO to address resource allocation for extremely heterogeneous workloads. We create a set of synthetic yet realistic workload traces that include task execution profiles, resource consumption patterns, and data locality. The issue is formulated to minimise the makespan and includes penalty constraints on prohibited task-resource assignments, while balancing CPU and memory usage.

We conduct extensive simulation experiments to compare the proposed approach with naive greedy allocation schemes across various VM settings. Results show that the PSO-based algorithm significantly outperforms the greedy scheduler-based method in terms of both makespan reduction and balanced resource utilisation. In addition, digital twin modelling provides insight into system behaviour and will be useful for more intelligent and adaptive resource management in cloud-based big data.

The main contributions of this study can be summarised as follows:

- We propose a digital twin-powered resource allocation policy for cloud-native data processing workflows, including virtual modelling and resource optimisation.
- A PSO-based workload-aware algorithm is used for the task-to-resource mapping problem in heterogeneous workloads.
- A novel artificial workload generation is established by taking task execution time, resource usage pattern, and data locality into consideration in realistic scheduling.

- Comparative simulation experiments have demonstrated that the proposed PSO-based approach outperforms greedy allocation methods in reducing the make-span and balancing resource utilisation.
- The proposed model offers a pragmatic and flexible optimisation model to act as the grounding for extension of various facets such as: (i) energy-efficient scheduling, (ii) SLA fulfilment, and (iii) real digital twin world integration.

2. Related Work

In cloud computing, the use of digital twin technologies for resource scheduling and system optimisation has recently attracted significant interest. Digital twin models enable the creation of virtual representations of real-world systems, providing predictive analytics and facilitating adaptive decision-making (Tao et al., 2019). Qiu et al. (2022) presented a digital twin-assisted edge computing resource allocation scheme with an enhanced whale optimisation algorithm to minimise power usage and optimise resource distribution in edge networks. Although they have successfully addressed the edge layer's energy efficiency, their model does not focus on big-data processing workflows at the cloud level, where workload heterogeneity and makespan optimisation are crucial. Li et al. (2022) proposed a digital twin-based computing resource management mechanism for vehicular networks that adopts a two-layer digital twin architecture to collect dynamic network status and service requirements. Although they demonstrated scalability and adaptability in the vehicular cloud, their design remains domain-specific. It does not directly handle heterogeneous big data workloads in most general-purpose cloud data centres. Our work targets these general-purpose cloud systems, which support dynamic, mixed workloads.

PSO has been widely adopted for cloud resource allocation problems due to its simplicity, global search ability, and insusceptibility to local optima (Kennedy & Eberhart, 1995). Alkayal (2018) utilised multi-objective PSO to optimise cloud resource provisioning while considering task scheduling goals across multiple dimensions. Their work, however, does not include digital twin models or task-level workload simulators with execution profiles and data locality, which are essential for capturing realistic workflow behaviours. Our contribution is the synergistic combination of empirical workload modelling with digital twin simulation and materialised optimisation with PSO. Prasad et al. (2025) proposed a hybrid algorithm based on PSO and Grey Wolf Optimisation for cloud task scheduling. Their work improved load balancing across VMs and reduced task completion times. However, the lack of workload- and motion-specific digital representations prevents the modelling system from capturing features of actual task performance. We explicitly model task-level resource consumption, execution time, and data locality in our study, and our scheduling framework is more practical and adaptive.

Saxena & Singh (2025) proposed a self-healing, fault-tolerant cloud-based digital twin processing control structure that focuses on dependability through failure prediction and resource demand estimation. Although useful for guaranteeing system robustness, their model was not aimed at performance optimisation measures (such as makespan and resource utilisation balance), which are the core of our work.

In summary, these experiments demonstrate the promise of digital twin modelling and PSO for cloud resource allocation. However, most currently available solutions concentrate on energy consumption, fault tolerance, or specific domain workloads, rather than overall performance optimisation. Unlike existing literature, we propose a novel digital twin-enabled resource allocation framework coupled with PSO to optimise task-to-resource mapping, accounting for realistic workload distributions. A synthetic dataset representing complex big data workflows is generated, which includes task execution times, resource utilisation profiles, and data locality properties. The targeted optimisation of makespan minimisation in their proposed scheme, coupled with balanced CPU and memory provisioning at virtual machine nodes, makes this framework a practical and generalised approach to performance optimisation of cloud-based big data processing systems. A comparative summary of existing digital twin-related studies and the distinct characteristics of our proposed framework is presented in Table 1.

3. Methodology and System Design

This section presents the entire process for simulating and optimising a cloud-based big data processing system powered by digital twin technology. The model uses real-world telemetry as input to synthesise the dataset and shape resource-aware optimisation criteria, which are used with PSO and a greedy scheduling baseline. Figure 1 illustrates the digital twin architecture, where data collected from diverse sources are processed within a cloud-based big data system to enable resource allocation and performance optimisation through coordinated twin entities.

Table 1. Comparison of digital twin-related studies and the proposed work.

Study	Domain / System	Use of Digital Twin	Real-Time Feedback	Optimisation Method	Key Limitation	Contribution Compared to Our Work
Tao et al. (2019)	Industry / Manufacturing	Conceptual DT model	Varies	Not optimisation-focused	General framework; not cloud workflow-oriented	Defines the DT concept, but not applicable to cloud scheduling
Qiu et al. (2022)	Edge computing	DT-assisted allocation	Yes	Whale Optimisation Algorithm	Edge domain only; not cloud big data workloads	Our model targets general-purpose heterogeneous cloud systems
Li et al. (2022)	Vehicular networks	Two-layer DT architecture	Yes	DT-driven adaptive management	Domain-specific; not general cloud workloads	Our framework applies to large-scale mixed cloud workloads
Saxena & Singh (2025)	Cloud reliability	DT for failure prediction	Yes	Not optimisation-focused	Focus on dependability, not performance	Our work focuses on makespan & resource balance optimisation
Alkayal (2018)	Cloud scheduling	No	No	Multi-objective PSO	Lacks DT-based modelling	We integrate DT modelling with PSO for realistic scheduling
Prasad et al. (2025)	Cloud scheduling	No	No	Hybrid PSO–GWO	No task-level DT or data locality representation	We model execution time, locality and resource usage with DT

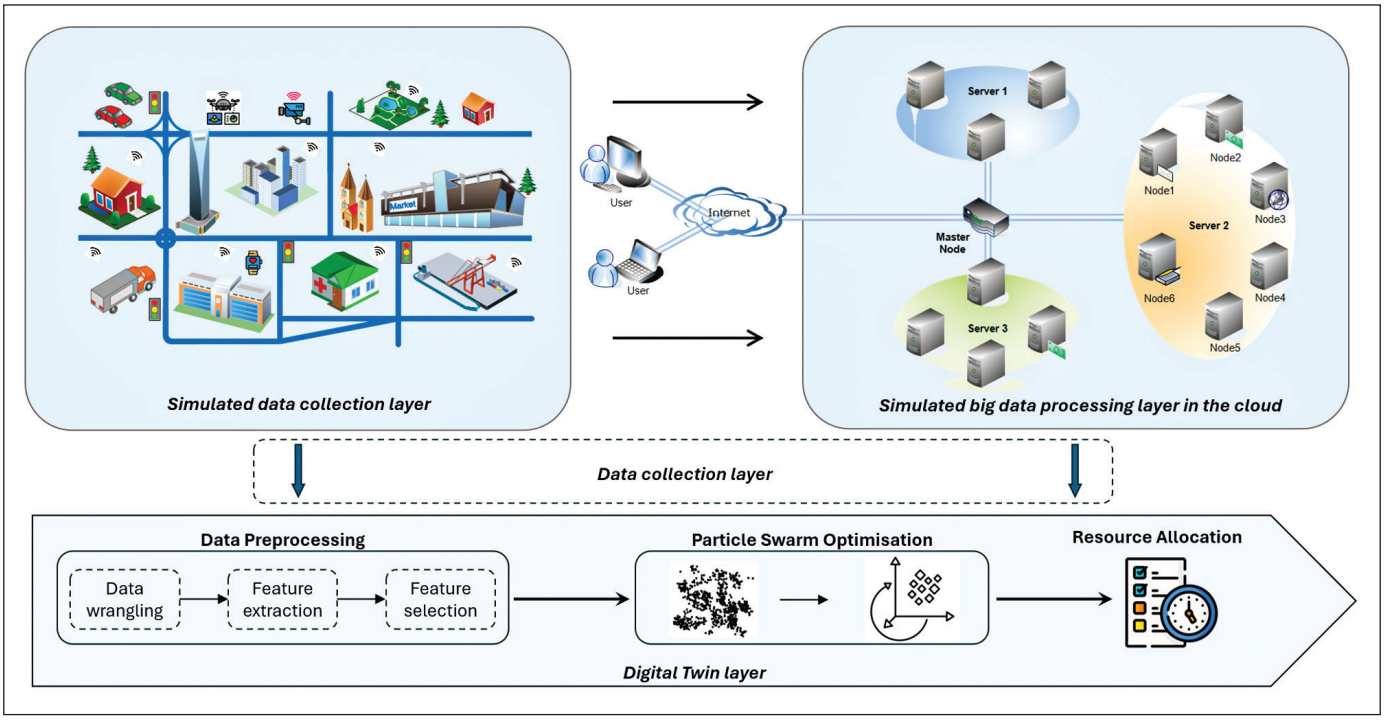

Figure 1. Digital twin framework for workload simulation and resource optimisation.

3.1. Real-Time Data Collection from Large-scale Cluster

We collected real-time data from a production-scale Hadoop cluster with 31 AWS EC2 nodes to obtain realistic workload patterns. These nodes are configured with different (CPU and memory) resources to simulate a heterogeneous cloud environment. All nodes ran with SmartMonit (Demirbaga et al., 2024), a lightweight telemetry agent that records system metrics with fine-grained detail.

The monitored metrics included:

- The duration of task execution (map, shuffle, reduce)
- CPU and memory usage
- Input data locality flags
- Network traffic and latency
- VM specifications (vCPU and RAM)

The structured logs were used as empirical evidence to create an in-control synthetic data set.

3.2. Synthetic Dataset Construction

Based on the observed telemetry, a synthetic dataset was gained from 25 different experiments containing around **1000 job records**, each with the following features:

- *map_time*, *shuffle_time*, *reduce_time* (seconds)
- *cpu_demand* and *ram_demand* (normalised between 0 and 1)
- *dataLocality* (1 = data is local to assigned VM, 0 = remote)
- *total_job_time* = *map* + *shuffle* + *reduce*
- *assigned_vm* and *vm_types* (configurable VM pool)

To incorporate realism, **dataLocality** directly affects map duration. Specifically:

- If *dataLocality* = 1, then *map_time* ∈ [25, 35] sec
- If *dataLocality* = 0, then *map_time* ∈ [35, 50] sec (due to network delay)

Similarly, the shuffle and reduce phases were randomly sampled within controlled ranges based on empirical observations:

- *shuffle_time* ∈ [2, 5] sec
- *reduce_time* ∈ [5, 9] sec

3.3. Problem Definition

In a cloud system, resource allocation is crucial, as it assigns jobs to available virtual machines (VMs) and optimises overall system performance. In our problem, performance is measured by minimising makespan, although balancing CPU and memory usage is also a secondary objective. Furthermore, the system should not suffer from a resource violation, in which a job's resource demand exceeds the capacity of its assigned VM.

3.3.1. Job and VM Definitions

Let us define:

- $J=\{j_1,j_2,\ldots,j_n\}$ be the set of jobs (tasks), where each job j_i is characterised by:
  - T_i: total execution time
  - d^i_{cpu}: CPU demand
  - d^i_{ram}: RAM demand
  - $l_i \in \{0,1\}$: data locality indicator (1 if data is local, 0 otherwise)
- $V=\{v_1,v_2,\ldots,v_m\}$ be the set of available VMs, each with:
  - r^k_{cpu}: CPU capacity
  - r^k_{ram}: RAM capacity

Each job must be assigned to exactly one VM such that its resource demands do not exceed the VM's capabilities.

3.3.2. Objective Function

The primary objective is to minimise the makespan M, defined as the maximum completion time among all VMs, as shown in Equation 1:

$$M = \max_{vk \in V} \sum^{n}_{ji \in Ak} Ti \tag{1}$$

where A_k is the set of jobs assigned to VM v_k.

The secondary objective is to maintain balanced resource usage across the VM pool, defined through the average utilisation, as indicated in Equation 2:

$$U_{cpu} = \frac{1}{m}\sum_{k=1}^{m} \frac{\sum_{j_i \in A_k} d^i_{cpu}}{r^k_{cpu}}, U_{ram} = \frac{1}{m}\sum_{k=1}^{m} \frac{\sum_{j_i \in A_k} d^i_{ram}}{r^k_{ram}} \tag{2}$$

To discourage invalid assignments where a job is assigned to a VM that lacks sufficient resources, we define a penalty term P as in Equation 3:

$$P = \alpha \cdot \sum_{j_i \in J} 1\left(d^i_{cpu} > r^{f(i)}_{cpu} \vee d^i_{ram} > r^{f(i)}_{ram}\right) \quad (3)$$

where f(i) is the index of the VM to which job j_i is assigned, 1(·) is an indicator function returning 1 if the condition is true and 0 otherwise, and α is a large constant penalisation weight.

3.3.3. Combined Cost Function

To combine the goals into a single optimisation criterion, we define the total cost function C as in Equation 4:

$$C = M + P \quad (4)$$

This structure allows the algorithm to prioritise makespan minimisation while also rejecting infeasible solutions due to over-allocation.

3.3.4. Data Locality Impact

Data locality plays a direct role in execution time, particularly for the **map** phase:

- If l_i=1, then t^i_{map} ~U(25,35) seconds
- If l_i=0, then t^i_{map} ~U(35,50) seconds

Thus, the **execution time of a job** is defined as in Equation 5:

$$T_i = t^i_{map} + t^i_{shuffle} + t^i_{reduce} \quad (5)$$

3.3.5. Optimisation Challenge

The space of solutions is the set of all job-to-VM assignments possible under capacity constraints and optimised for an imbalanced load. It renders the problem NP-hard, making metaheuristics like PSO suitable for generating near-optimal solutions within a reasonable time.

Our proposed model leverages:

- Workload profiles (synthetic) from real workloads derived from system logs,
- Adaptive behaviour resulting from the digital twin simulation loop,
- An efficient search method based on PSO does not carry the limitations of heuristic/greedy-only methods.

4. Proposed Model

4.1. Overview of the Approach

The proposed model combines a workload-aware scheduler and a digital twin that virtually replicates the task behaviour observed on the real cluster. The digital twin consumes the traces of task executions (including the duration of map, shuffle, and reduce), CPU and memory consumption, network traffic, and data locality, and replays workloads in a controlled simulation loop that statistically resembles the traces. For a given task, in this loop, we consider two solvers: (a) a light-weight baseline solver called the Greedy Assignment Algorithm, and (b) PSO as the main optimiser. Both solvers minimise the composite objective defined in Section 3.3.2—namely, the makespan with a penalty term for infeasible allocations—while we also report CPU and memory utilisation to assess balance across VMs.

Design assumptions are as follows: tasks are non-preemptive; each task is placed on exactly one VM; a task's map time is affected by data locality; CPU and RAM capacities are hard constraints; the penalty weight α is chosen sufficiently large to discourage violations.

4.2. Greedy Assignment Algorithm

Greedy scheduling is one of the most effective weighting-based heuristics for resource allocation problems. Its operation involves processing each task and assigning it to the first suitable virtual machine in the virtual machine set, based on its CPU and memory demands. This method is simple, computationally less expensive, and easy to implement; however, it is myopic, as it does not consider the entire workload distribution and does not aim to minimise the makespan globally. The greedy algorithm is not presented as a candidate solution in this study, but it is considered merely a baseline method. The incentive for this choice is that we seek a relatively easy and easy-to-reproduce benchmark from which the improvements achievable through our proposed optimisation, article swarm optimisation, can be assessed.

The greedy baseline in Algorithm 1 initialises the residual CPU and RAM of each VM and sets the completion time for all tasks to zero. Then it reads the job list once. For example, the algorithm traverses the VMs in a static order for each job and selects the VM with the highest idle CPU and RAM capacity. Once a feasible fit is found, it records the placement, updates the VM's residual capacities, and increases the VM's completion time by the job's

Algorithm 1: Greedy assignment for task–VM mapping	
Input:	*- J: set of jobs; V: set of VMs* *- d^i_{cpu} i ,d^i_{ram}: resource demands of job j_i* *- r^k_{cpu} , r^k_{ram} : capacities of VM v_k* *- T_i : total execution time of job j_i*
Output:	π: mapping $j_i \mapsto v_k$; per-VM completion times C_k
1	*// Initialise residual capacities and completion times*
2	***for** each VM v_k in V **do***
3	*$cap_{cpu[k]} \leftarrow r^k_{cpu}$; $cap_{ram[k]} \leftarrow r^k_{ram}$; $C_k \leftarrow 0$*
4	*// Sequentially place each job*
5	***for** each job j_i in V **do***
6	*placed ← false*
7	***for** each VM v_k in V **do***
8	*If $d^i_{cpu} \leq cap^k_{cpu}$ and $d^i_{ram} \leq cap_{ram[k]}$ then*
9	*$\pi(i) \leftarrow k$*
10	*$cap_{cpu[k]} \leftarrow cap_{cpu[k]} - d^i_{cpu}$*
11	*$cap_{ram[k]} \leftarrow cap_{ram[k]} - d^i_{ram}$*
12	*$C_k \leftarrow C_k + T_i$*
13	*placed ← true; Break*
14	*If placed = false then*
15	*$\pi(i) \leftarrow \perp$ // mark infeasible (penalised in Section 3.3.2)*
16	*Return π, {C_k}*

processing time. There is no available VM to receive the job, which is flagged as infeasible and subject to penalisation (Section 3.3.2). It is a greedy and deterministic approach, but it is computationally light since it reasons locally and in sequence; however, it can potentially overload early VMs, leading to unbalanced utilisation and longer makespan.

4.3. Particle Swarm Optimisation

PSO is a population-based metaheuristic that mimics the social behaviour of birds and fish. It is widely used to solve NP-hard scheduling problems and has been adopted in cloud task scheduling due to its balance between exploration and exploitation (Lin et al., 2019). Its effectiveness has also been demonstrated in large-scale optimisation scenarios, where improved PSO variants achieve stable convergence (Xue et al., 2020). This paper uses PSO as the original optimiser for a power plant. Each particle contains an entire mapping of jobs to VMs, and the swarm evolves through successive updates to these mappings using inertia, personal best, and global best. By employing the objective function from Section 3.3.2, which incorporates both makespan and penalty elements, PSO systematically seeks efficient solutions. In our simulation environment supporting a digital twin, PSO is used to demonstrate how a global learning-and-adapting optimiser can outperform basic heuristics in minimising makespan and better utilising CPU and memory resources on heterogeneous VMs, in line with recent digital twin-driven optimisation approaches (Zhang et al., 2021).

The formulation follows the standard PSO updates introduced by Kennedy & Eberhart (1995) and later refined through inertia-weight and constriction strategies (Shi & Eberhart, 1998; Clerc & Kennedy, 2002). The PSO optimiser shown in Algorithm 2 encodes a candidate solution as an integer vector of size equal to the number of jobs, where each integer in the vector corresponds to the selected VM for that job. The swarm is initialised with random mappings and velocity vectors set to zero; the fitness of each particle is then represented as the sum of its makespan and a penalty value for a capacity violation. At each iteration, particles adjust their velocities based on inertia and follow their own historical best solution, as well as the global swarm's best solution, to advance them by one step in the search space.

Algorithm 2: PSO for task–VM mapping	
Input:	*- J, V V, fitness F (π) = M + α P (Section 3.3.2)* *- swarm_size, max_iter, w (inertia), c_1, c_2 (cognitive/social)* *Encoding:* *- Particle position x ∈ { 1 , … , |V| }$^{\|J\|}$ (index of VM per job)* *- Velocity v stored in $R^{|J|}$ and discretised by rounding*
Output:	- π*: best mapping found (global best)
1	*// Initialise swarm*
2	***for** each particle p = 1 … swarm_size **do***
3	*x_p ← random integers in [1, |V|] for all jobs*
4	*v_p ← 0 vector ; evaluate F(x_p)*
5	*$pbest_p$ ← x_p*
6	*gbest ← $argmin_p$ F(x_p)*
7	*// Main loop*
8	***for** t = 1 … max_iter **do***
9	*for t = 1 … max_iter do*
10	*// Velocity–position update*
11	*v_p ← w·v_p + c1·r1·($pbest_p$ − x_p) + c2·r2·(gbest − x_p)*
12	*x_p ← round(x_p + v_p)*
13	*x_p ← clip_to_domain(x_p, 1, |V|) // repair if needed*
14	*Evaluate F(x_p) // computes makespan + penalty*
15	*If F(x_p) < F($pbest_p$) then $pbest_p$ ← x_p*
16	*gbest ← $argmin_p$ F(x_p)*
17	*Return gbest as π**

Since assignments are discrete, updated states are not updated instantly. They are rounded (to the nearest valid VM ID), clipped to the boundaries, and a lightweight repair is performed if necessary. The fitness is reconsidered, the pbest is updated, and the gbest is determined. When the stopping criterion is fulfilled, the best global mapping is given. In contrast to the greedy algorithm, PSO balances exploration and exploitation to escape poor local optima and drive the search to lower makespan with more balanced CPU/RAM scheduling.

In selecting the PSO hyperparameters, we first considered the parameter ranges commonly adopted in prior optimisation studies. To validate the robustness of our configuration, we conducted a sensitivity assessment using w ∈ {0.3, 0.5, 0.7} and c_1, c_2 ∈ {1.0, 1.5, 2.0}. The results showed that w = 0.5 and c_1 = c_2 = 1.5 deliver the most stable convergence behaviour while preserving adequate swarm diversity. Parameter settings with excessively high acceleration coefficients (≥2.0) resulted in oscillatory swarm motion, whereas lower inertia weights (<0.4) accelerated premature convergence and reduced search coverage. Therefore, the adopted configuration represents a balanced and empirically validated choice aligned with PSO convergence theory and practical recommendations.

4.4. Implementation Details

The model is written in Python, with simulations and data manipulation performed with NumPy and Pandas, and plotting with Matplotlib. The digital twin replays task profiles from the SmartMonit traces to maintain the observed distributions for execution times, resource demands, and data locality. The cost function C=M+αP of Section 3.3.2 is uniformly used in the two solvers. Unless otherwise stated, PSO parameters are set to: swarm size = 50, iterations = 100, w = 0.5, c_1= c_2 = 1.5. The penalty weight is taken to be α

= 1000 x maxiTi to heavily penalise violations while retaining a scaling that makes span and other large constants of the same order work as well. The termination criteria are either the number of iterations or a patience mechanism that does not improve the global best within a fixed window. All experiments are conducted with the same digital twin seeds to ensure fair, repeatable comparisons between greedy and PSO.

4.5. Baseline Comparison with Greedy Allocation

A greedy allocation algorithm was used as a benchmarking method, which iterates over the job list in random order and schedules each job to the first VM whose residual CPU and memory capacity are sufficient for the job request. This method is efficient in terms of time (time complexity O(|J|·|V|)), but naive from an algorithmic perspective. It does not consider optimisation principles, such as load distribution among VMs or minimising the makespan.

In our framework, greedy allocation is also used as a reference to compare and validate the performance of the developed PSO algorithm. The comparison is made based on the following performance parameters:

- **Makespan:** The overall elapsed time until the last job finishes is computed with greedy and PSO allocations. Due to its local decision-making, the greedy method overcommits some early VMs and undercommits others, resulting in a higher makespan more frequently.
- **CPU Utilisation:** We measure the percentage of CPU time taken on each VM. Greedy allocation often results in inhomogeneous utilisation distributions, with some VMs overloaded and others underloaded.
- **Memory Efficiency:** We also monitor the RAM usage. Since greedy ignores balancing strategies, memory resources could be distributed and fragmented across all VMs, decreasing effective utilisation.

The systematic exploration of PSO for greedy allocation across these dimensions demonstrates the efficacy of our proposed mechanism in terms of efficiency (makespan) and balance (CPU/RAM utilisation).

4.6. Digital Twin System Integration

We use an optimisation integrated in a digital twin of the to-be-learnt cloud system. The digital twin receives as input telemetry streams—task execution traces, CPU and memory usage, network bandwidth consumption, and data locality characteristics—collected from an actual Hadoop cluster. These input streams are pre-processed and played in the virtual model, synchronised with the physical system at the workload state and VM resource profile levels.

We continue these arguments in the next subsection, explaining that adding the digital twin would also support three key functionalities:

1. **Performance Prediction under Unseen Workload:** Synthetic workloads generated statistically from real traces enable the digital twin to perform performance prediction for various workloads unseen by the actual system. Load spikes or changes in data proximity can, for example, be simulated before they occur.
2. **Adaptive Optimisation:** The adaptive twin self-modifies its internal models based on the real systems' current telemetry. The PSO scheduler can re-focus its work assignments as the workload fluctuates. These adaptive properties enable the optimisation to be used in cases of limited workload homogeneity and dynamic demand.
3. **Scenario Testing for Orchestration Strategies:** The digital twin is the sandbox to play with what-if scenarios, such as adding new VM scaling policies, introducing heterogeneous VM types or restructuring data placement strategies – effectively, the framework acts as a schedule optimiser for proactive cloud orchestration.

To elaborate further on the integration workflow, we highlight that the digital twin is updated with up-to-the-minute telemetry from the physical cloud cluster via the SmartMonit monitoring pipeline. The metrics to be fed into the twin, such as task execution time, CPU and memory usage, network latency, and data locality, are also added to the twin in a regular updating cycle. Such live updates update the virtual system state and workload distributions, thereby bringing the synthetic environment up to date with the physical one. Whenever the digital twin detects significant differences in workload intensity, resource saturation, or locality behaviour, the PSO optimiser is triggered to recompute and then replay the scheduling on the updated state, enabling the simulation to surpass static offline scheduling and perform adaptive, feedback-driven optimisation, thereby enhancing the operational connection between simulation and reality.

By incorporating optimisation, the developed model goes beyond static offline scheduling. It produces a dynamic, autoregressive decision-making loop that accounts for

workload variability and resource heterogeneity, making it practically implementable for the management of real cloud systems.

5. Results and Discussion

In this section, we present the analysis of the experimentation results generated by our digital twin-based cloud resource allocation framework. They measure performance based on several factors: job completion times, workload phase distribution, resource usage, and VM utilisation. In this section, we compare the greedy-based allocation to the optimal allocation and reveal their trade-offs. Each figure is accompanied by a description to provide an intuitive interpretation of workload characteristics, algorithm effectiveness, and cloud workload management optimisation.

Such influence of data locality on map task run time is illustrated in Figure 2. When executed on non-local nodes (i.e., data are moved from source to destination), the map phase is significantly slower than before, as is local execution. This is a theoretical expectation in a distributed setting due to the overhead introduced by network transfer for non-local execution, which results in higher latency. On the other hand, the transmitted data is small since local execution is moved off, and thus computation time is low; processing time and performance predictability become more favourable. These results demonstrate that data locality is crucial for big-data processing in the cloud, and even a small additional map time can be significant across thousands of tasks, magnifying the effect. Our digital twin system explicitly models this effect to realistically capture workload dynamics, whereas abstract schedulers often overlook it.

Figure 3 shows the run-time distributions for the three critical stages of big data processing: map, shuffle, and reduce. The shuffle and reduce phases are also well clustered, with low variance, indicating that Python is lightweight and has predictable performance. Yet, the map phase exhibits a wide spectrum with two evident peaks for local and non-local runs. This bimodal pattern arises from data locality, which causes non-local execution to incur higher network latency. The contrast in map performance highlights the importance of the map stage in achieving workload efficiency. Realising these heterogeneous distributions in our digital twin model enables us to appropriately simulate cloud workflows, thereby increasing the applicability of optimisation methods that can leverage it.

The distribution of CPU and RAM requirements for submitted jobs jointly is plotted in Figure 4. This scatter plot

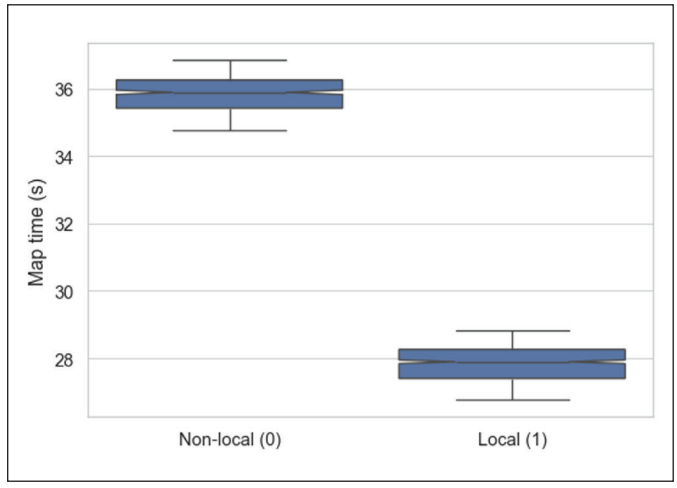

Figure 2. Map execution time by data locality.

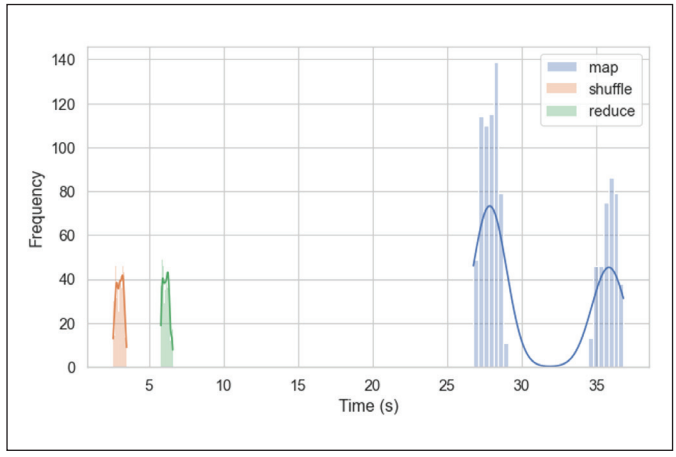

Figure 3. Job phases distribution over time.

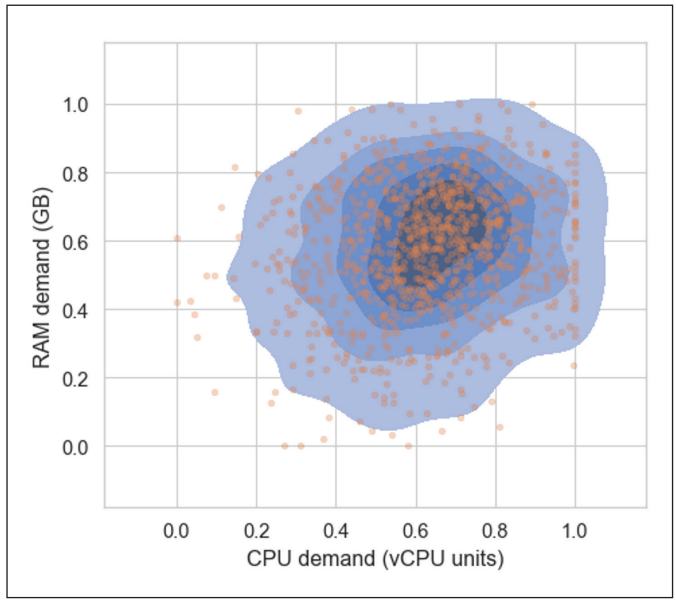

Figure 4. Job resource demand scatter.

with density contours shows that almost all jobs are located at or near medium-demand levels (i.e., approximately half of the total available vCPU and RAM capacity). However, fewer employees (though still some) have significantly excessive CPU or RAM usage – outliers that cause more scheduling trouble for us. This diversity emphasises the scheduling challenge in cloud systems: it is hard to serve a high volume of modest-demand jobs while avoiding corner cases that can overload resources. The digital twin plant evaluates scheduling approaches for workload variation to replicate such realistic demand profiles. The presence of overlapping density regions provides the rationale for high-performance heuristics (e.g., PSO), self-tuning application profiles, and adaptively overlaying them on heterogeneous VM capabilities to utilise resources.

Figure 5 shows the makespan of the greedy baseline and PSO. The greedy algorithm achieves a significantly lower completion time than PSO, with a 44.7% improvement, by assigning tasks to primary VMs at the earliest available slot in their lifetimes. This improvement is due to PSO's ability to search the space globally: a task is evenly distributed across multiple machines rather than concentrated on a few nodes. PSO does not fall into any bottlenecks because it can and does consider execution profiles or resource limits. The result confirms the applicability of optimisation-driven schedule construction to digital twin environments, where fine-grained modelling of task characteristics enables more informed resource management. These results indeed verify the reduction in total execution time achieved by PSO and its ability to enhance the scalability and efficiency of large-scale cloud workflows.

The total completion time of all the VMs under greedy scheduling and PSO is depicted in Figure 6. The greedy distribution is highly unbalanced, with nearly all jobs assigned to VM3 and only a few underloaded on VM1. As mentioned before, this unfairness results in a high makespan because heavy loads create a bottleneck on the machines. In comparison, PSO provides a more balanced load allocation by assigning jobs to all available VMs; thus, both VM1 and VM2 contribute equally to job processing, with VM3 contributing equally. Overall, it means resources are utilised more efficiently; that is, fewer resources become idle, resulting in better system throughput. Our results also verify that not only is the JIT scheduling important for the purpose of minimising makespan, but it's crucial to ensure symmetric load distribution so that imbalance and over subscription can be avoided, or otherwise will incur high cost in losing

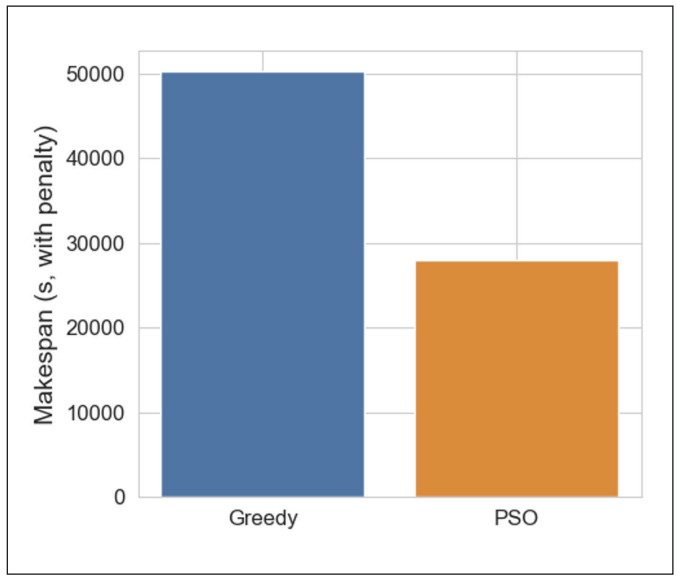

Figure 5. Makespan comparison.

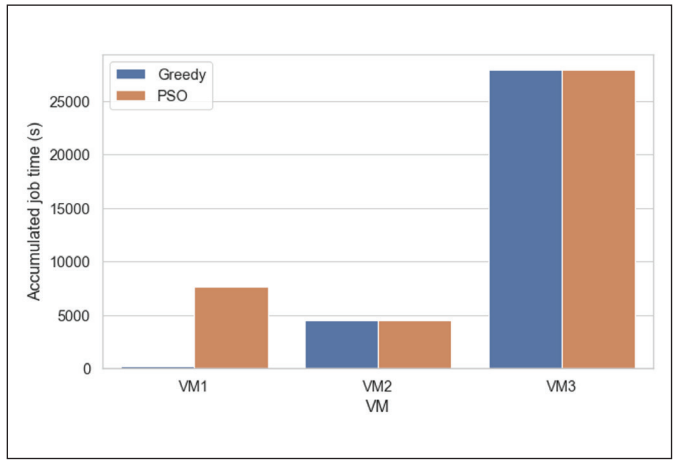

Figure 6. VM-wise accumulated completion times.

performance, dissipating this undesirable electrical waste occurring in commercially-deployed real-world cloud architectures.

We also compare CPU and memory utilisation between greedy and PSO scheduling, as shown in Figure 7 and Figure 8. This greedy approach causes VM1 to be underutilised and VM3 to be overloaded, resulting in unbalanced utilisation, which extends the utilisation time and leads to stale systems. PSO, however, yields a more uniform distribution because it effectively balances processor demands and virtual-machine capacities through its global search. This results in flatter usage across all VMs, avoiding underutilization and minimising the chance of hotspots. These results

validate that PSO not only minimises makespan but also enhances overall system performance by providing more balanced and adaptive resource scheduling.

We note that the performance gaps between PSO and the greedy strategy are only moderate in our experiments. However, this is driven by the uniformity of the workload assumptions and their execution-time ranges, which limits the maximum optimisation potential. Moreover, the penalty term is deliberately constructed to discourage aggressive exploration of infeasible or highly imbalanced mappings, thereby establishing a natural limit on the magnitude of possible improvements. In these balanced environments, PSO continues to offer benefits from both exploratory and CRD perspectives, but it is unlikely that very large gains will be achieved.

Figure 9 illustrates the job migration matrix, comparing task assignments under the greedy baseline with those generated by PSO. A considerable number of jobs initially assigned to VM3 by the greedy algorithm are redistributed by PSO to other VMs, particularly VM2, thereby reducing overload on a single node. Moreover, jobs that were marked as "unsuitable" under greedy allocation—because no VM satisfied their resource requirements—are successfully assigned to VM1 by PSO. This demonstrates the optimisation algorithm's capacity to explore a wider search space and identify feasible placements that greedy scheduling overlooks. By redistributing workloads and recovering infeasible allocations, PSO not only achieves lower makespan but also ensures higher resource utilisation and system stability. These results emphasise the added value of optimisation-driven scheduling, which enables more adaptive and reliable task-to-VM mapping in heterogeneous cloud environments.

Comparisons of renewable generation scheduling results based on greedy allocation and PSO are given by Figure 10 and Figure 11. In the greedy case (see Figure 10), tasks are not assigned evenly, likely resulting in VM1 being starved and receiving only light tasks, significantly fewer than the other two VMs, which receive heavily saturated tasks. This mismatch is at the root of bottlenecks, as overloaded machines delay job completion, thereby extending the makespan. On the other hand, the PSO schedule snapshot (Figure 11) shows a uniform distribution of jobs across the three VMs. Every VM is utilised effectively at all times, avoiding idle time, and no machine becomes a performance bottleneck, indicating the advantage of the PSO algorithm

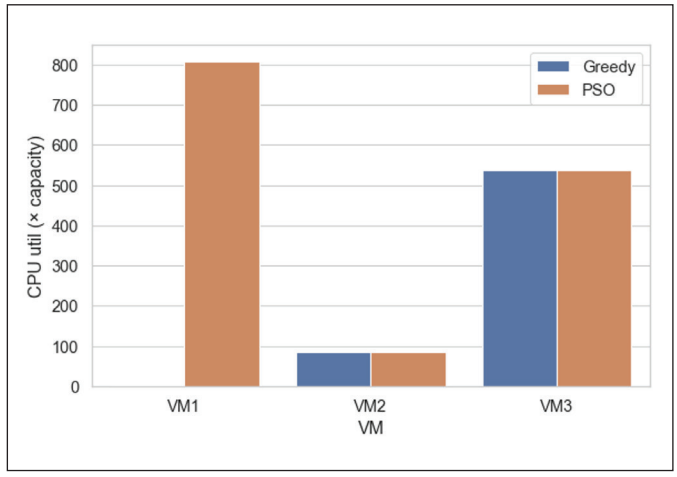

Figure 7. CPU utilisation per VM.

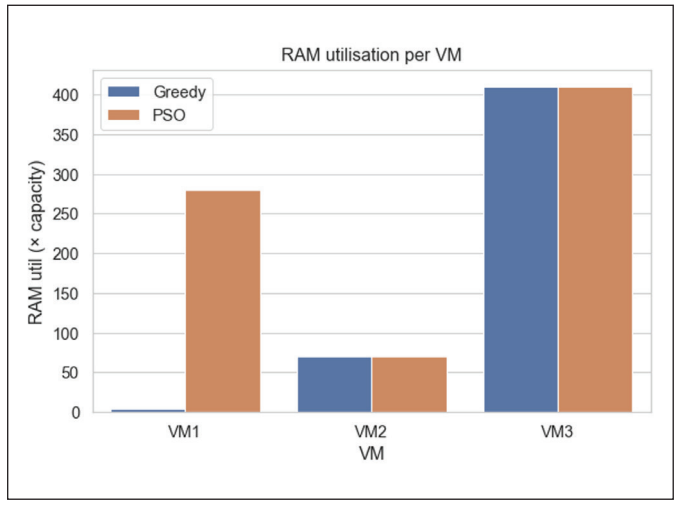

Figure 8. Memory utilisation per VM.

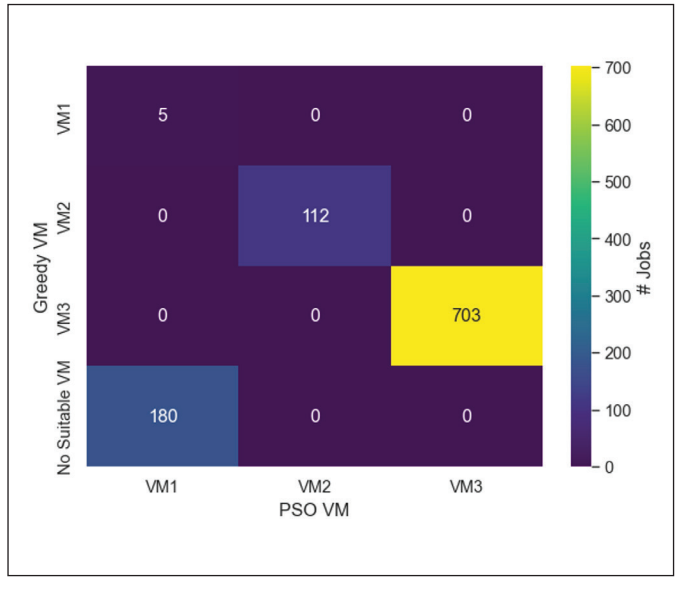

Figure 9. Job migration matrix (Greedy --> PSO).

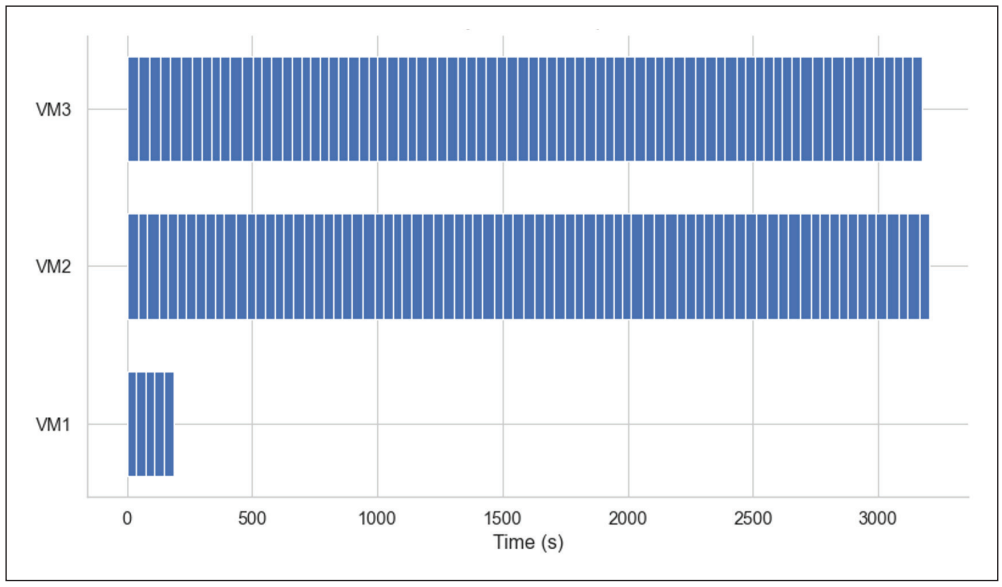

Figure 10. Greedy schedule snapshot.

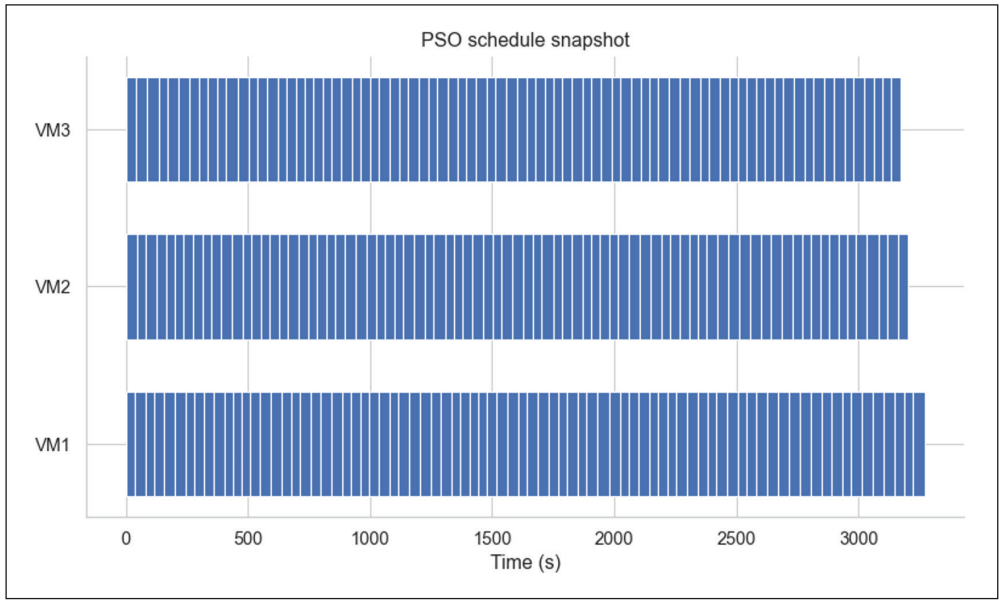

Figure 11. PSO schedule snapshot.

in exploring a larger scheduling space to perform load balancing. Consequently, the system throughput increases and the execution latency decreases.

Figure 12 shows the convergence performance of the PSO algorithm in solving the task–VM allocation problem. The global best cost, a combination of makespan and penalties, decreases rapidly in the first iterations, indicating that the initial random solutions are explored thoroughly and then improved efficiently. After 5–10 iterations, the curve stabilises, and the swarm converges towards an optimal or near-optimal solution. This result demonstrates the effectiveness of PSO for searching the intricate solution space of heterogeneous workloads while preserving the feasibility and performance optimisation.

6. Conclusion and Suggestions

In this paper, a digital twin-assisted optimisation framework is developed for resource allocation of cloud-based big data workflows. We simulated a complex system by modelling workloads in terms of execution time, resource consumption, and data locality, building a virtual environment that reflects the nuances of the concrete systems. In this

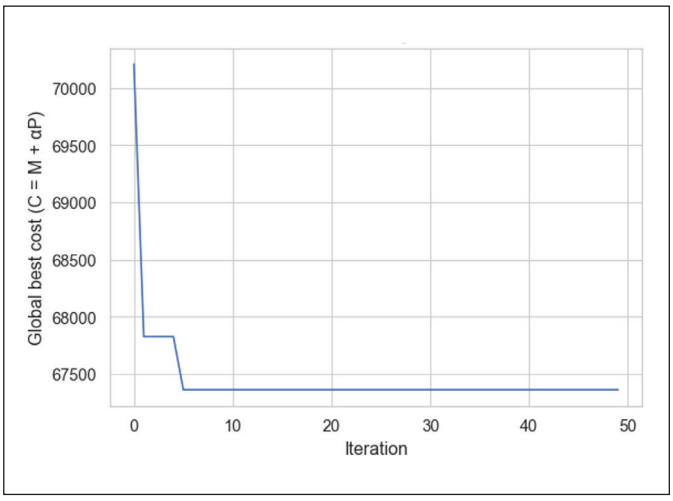

Figure 12. PSO convergence.

environment, we employed a PSO-based approach to tackle the task-to-resource assignment problem and compared its behaviour with an allocation policy that greedily assigns homogeneous tasks. The experimental results demonstrated that the presented method yielded a better balance between declining solvespan and workload distribution than previous methods, thereby improving overall resource utilisation. The PSO algorithm was more effective at ensuring optimal utilisation of all jobs across VMs, while sparing resources. On the other hand, a greedy schedule concentrates jobs on VMs and makes the resources inefficient.

The results show that combining digital twin modelling and meta-heuristic optimisation can mitigate production issues arising from heterogeneous, dynamic cloud workloads. In contrast to previous techniques tailored to specific scenarios (such as energy conservation, error recovery, or application domains), our current framework is comprehensive and suitable for cloud computing. The combination of simulation/optimisation/performance analysis within a single digital twin will enable predictive capabilities for simultaneously considering proactive resource management options. Practically, the digital-twin-driven PSO scheduler presented in this paper can be deployed in actual cloud systems by incorporating the optimisation module into existing autoscaling or scheduling pipelines. As the twin runs on real-time telemetry, it can integrate with monitoring services, such as AWS CloudWatch or Azure Monitor, to drive adaptive rescheduling, offering a viable deployment option without requiring changes at the infrastructure level.

In the future, we will aim to scale our evaluation by studying larger and more diverse cloud infrastructures using real production workload trace data. The optimisation framework can also be extended to include other explicit concerns, such as energy and carbon, to make it more relevant to green computing. Another exciting route ahead is integrating PM and abnormal injection features into a digital twin, which can facilitate scheduling optimisation, predict and prevent breakdowns.

Author contributions: Software, data collection, methodology design, analysis, and manuscript writing were carried out by the single author.

Ethics committee approval: This study does not require ethics committee approval, as it does not involve any human or animal experiments, and no personal data were collected.

References

Aggarwal, S., Kumar, N., Singh, A., & Aujla, G. S. (2024). Blocktwins: Blockchain empowered supply chain digital twins in metaverse. 2024 IEEE International Conference on Communications Workshops (ICC Workshops), 1456–1461.

Alkayal, E. (2018). Optimizing resource allocation using multi-objective particle swarm optimization in cloud computing systems. Doctoral Thesis, University of Southampton, UK.

Clerc, M., & Kennedy, J. (2002). The particle swarm – explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation, 6(1), 58–73.

Demchenko, Y., Turkmen, F., de Laat, C., Hsu, C. H., Blanchet, C., & Loomis, C. (2017). Cloud computing infrastructure for data intensive applications. In Big Data Analytics for Sensor-Network Collected Intelligence (pp. 21–62). Elsevier.

Demirbaga, Ü., Aujla, G. S., Jindal, A., & Kalyon, O. (2024). Big data analytics. In Big Data Analytics: Theory, Techniques, Platforms, and Applications (pp. 31–42). Springer.

Dua, A., Aujla, G. S., Jindal, A., & Sun, H. (2024). Green AutoML: Energy-efficient AI deployment across the edge–fog–cloud continuum. 2024 IEEE Globecom Workshops (GC Wkshps), 1–6.

Ghafari, R., Kabutarkhani, F. H., & Mansouri, N. (2022). Task scheduling algorithms for energy optimization in cloud environment: A comprehensive review. Cluster Computing, 25(2), 1035–1093.

Jomah, S., & Aji, S. (2024). Meta-heuristic scheduling: A review on swarm intelligence and hybrid meta-heuristics algorithms for cloud computing. Operations Research Forum, 5(4), 94.

Kabir, M. R., Halder, D., & Ray, S. (2024). Digital twins for IoT-driven energy systems: A survey. IEEE Access, 12, 177123–177143.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks, 1942–1948. https://doi.org/10.1109/ICNN.1995.488968

Li, M., Gao, J., Zhou, C., Shen, X., & Zhuang, W. (2022). Digital twin-driven computing resource management for vehicular networks. GLOBECOM 2022 - 2022 IEEE Global Communications Conference, 5735–5740.

Lin, W., Wu, M., Deng, D., & Chang, W. (2019). An energy-efficient task scheduling algorithm based on PSO in cloud environments. Journal of Systems Architecture, 98, 14–25.

Malti, A. N., Hakem, M., & Benmammar, B. (2024). A new hybrid multi-objective optimization algorithm for task scheduling in cloud systems. Cluster Computing, 27(3), 2525–2548.

Ni, C., Wu, J., & Wang, H. (2025). Energy-aware edge computing optimization for real-time anomaly detection in IoT networks. Applied and Computational Engineering, 139, 42–53.

Prasad, R., Roy, A., & Kumari, S. (2025). Enhancing cloud task scheduling using a hybrid particle swarm and grey wolf optimisation approach. International Journal of Cloud Applications and Computing (IJCAC), 15(1), 32–47. https://doi.org/10.4018/IJCAC.326106

Qiu, S., Zhao, J., Lv, Y., Dai, J., Chen, F., Wang, Y., & Li, A. (2022). Digital-twin-assisted edge-computing resource allocation based on the whale optimization algorithm. Sensors, 22(23), 9546. https://doi.org/10.3390/s22239546

Saxena, D., & Singh, A. K. (2025). A self-healing and fault-tolerant cloud-based digital twin processing management model. IEEE Transactions on Industrial Informatics.

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. 1998 IEEE International Conference on Evolutionary Computation, 69–73.

Singh, G., & Chaturvedi, A. K. (2024). A cost, time, energy-aware workflow scheduling using adaptive PSO algorithm in a cloud–fog environment. Computing, 106(10), 3279–3308.

Tao, F., Zhang, H., Liu, A., & Nee, A. Y. C. (2019). Digital twin in industry: State-of-the-art. IEEE Transactions on Industrial Informatics, 15(4), 2405–2415. https://doi.org/10.1109/TII.2018.2873186

Xue, F., Zhou, Y., & Li, J. (2020). Improved particle swarm optimisation for large-scale scheduling problems. Applied Soft Computing, 92, 106–122.

Zeng, C., Wang, X., Zeng, R., Li, Y., Shi, J., & Huang, M. (2024). Joint optimization of multi-dimensional resource allocation and task offloading for QoE enhancement in Cloud-Edge-End collaboration. Future Generation Computer Systems, 155, 121–131.

Zhang, H., Guo, S., & Zhang, J. (2021). Digital twin-driven resource optimisation in intelligent computing systems. Future Generation Computer Systems, 123, 1–12.