Minimax Theory and its Applications
Volume 11 (2026), No. 4s, 735 to 746

Convergence-Guided Iterative Optimization for Dynamic Task Scheduling in Distributed Computing Systems

Priyadharshini K[1], Dr. A. Sumi[2], C. R. Eldin Jeeva[3], Dr. Phani Sridhar Addepalli[4], Dr. Tavanam Venkata Rao[5], Anitha Guntaka[6]

[1]Assistant Professor, Department of Management Studies, Meenakshi College of Arts and Science, Meenakshi Academy of Higher Education and Research, Email: kpriyamba@maher.ac.in, ORCID ID: 0009-0007-3209-9227
[2]Department of Information Technology, Sri Ramakrishna College of Arts and Science, Coimbatore, Email: sumi@srcas.ac.in, ORCID ID: 0000-0002-3121-7515
[3]Department of Mathematics, New Prince Shri Bhavani College of Engineering, Email: eldinjeeva@newprinceshribhavani.com, ORCID ID: 0009-0007-0328-1951
[4]Department of Computer Science and Engineering, Aditya University, Surampalem, Andhra Pradesh – 533437, Email: phani.addepalli@adityauniversity.in, ORCID ID: 0000-0002-6021-3433
[5]Professor, Department of Electronics and Communication Engineering, Sreenidhi Institute of Science and Technology, Email: vrtavanam@gmail.com, ORCID ID: 0000-0002-9158-2231
[6]Assistant Professor, Department of Computer Science and Engineering, Marri Laxman Reddy Institute of Technology and Management, Dundigal, Email: anithareddyguntaka@gmail.com, ORCID ID: 0009-0002-5958-8374

Abstract
Distributed computing systems contain one of the most serious optimization problems in dynamic task scheduling; this operates under varying workloads, resource allocation and availability, communication latency, and Queue conditions.Several scheduling approaches mostly depend on immediate observations, limited historical data, and heuristic optimization. Theseapproaches do not adequately capture heterogeneous temporal dependencies occurring at different time periods. This paper describes a multi-timescale fractional memory framework for dynamic task scheduling. It shows short-, medium-, andlong-term historical effects in a weighted fractional derivative combination of different orders. Dynamic weights can help with the relative combination of all memory parameters according to the observed dynamics of the system. Thisscheduling method is executed as a constrained multi-term fractional optimization problem, and the resulting method structure is analyzed with respect to well-posedness, convergence, disturbance-induced boundedness, and numerical implementation. The proposed formulation indicates memoryless and single-order fractional scheduling models by allowing multiple temporal memory mechanisms in order to influence resource allocation continuously. This work depicts a mathematical connection in-between multi-term fractional dynamical systems and adaptive scheduling in distributed computing environments.

Keywords: Fractional Calculus; Dynamic Task Scheduling; Distributed Computing; Fractional Memory; Multi-Term Systems; Convergence.

1. Introduction

Distributed computing systems encompass cloud, edge, fog, and computing-continuum infrastructures provide computational services over heterogeneous and geographically distributed resources. Task scheduling plays a pivotal role in distributed computing systems to minimize the execution time, maximize resource utilization, energy efficiency, and quality of service. The heterogeneity of resources and the dynamic nature of workloads make task scheduling a challenging optimization problem (Devi *et al.,* 2024; Mehrán*et al.*, 2024). In real-world scenarios, task arrivals, resource availabilities, processing capacities, and communication conditions are subject to continuous fluctuations. Hence, a scheduling decision that is optimal in a given scenario may not be optimal in another scenario due to changing environmental conditions.

Recent studies consequently investigated the adaptive scheduling, load balancing, resource allocation, and multi-objective optimization for the cloud-edge environments (Luo *et al.*, 2021; Zhao *et al.*, 2023; Li, 2024; Sahoo & Mishra, 2025). Besides, the hybrid optimization methods were further explored to address the trade-off in task-resource allocation (Mahdizadeh *et al.*, 2024; Malti *et al.*, 2024). The geo-distributed and cloud-edge environments bring additional challenges to the orchestration due to the heterogeneity in computational capabilities, availability, and communication features of the underlying resources (Wu *et al.*, 2025; Zhang *et al.*,

2024). The utilization of AI-based scheduling was also considered as a promising adaptive approach to dynamically adapting workload and resources (Sanjalawe*et al.*, 2025). Nevertheless, the implementation of iterative optimization algorithms still relies on the pre-defined iteration limits or stopping conditions that can result in unnecessary computations or premature termination of the search process. Therefore, it is required to apply the convergence-guided optimization that can control the search process based on the quality of solutions.

Fractional-order models provide an alternative to the differential models with respect to the memory effect and dependence of the system at hand. The fractional differential equations have been extensively studied and successfully used to model memory-related and hereditary phenomena (Diethelm, 2010; Mainardi, 2010). The multi-term formulation can also be used to characterize various time characteristics by taking into account multiple fractional-orders (Brandibur&Kaslik, 2023).Inspired by the aforementioned properties, this paper proposed a Convergence-Guided Iterative Optimization (CGIO) framework to solve dynamic task scheduling problem that can be characterized by multi-time scale fractional memory. By combining the multi-term fractional formulation, the adaptive task-resource allocation mechanism, and the procedure of optimizing the iterative process based on the convergence, the CGIO framework simultaneously represents the short-, medium- and long-term behavior of the system and adaptively modifies the optimization process during the solution procedure according to the improvement of the obtained results. The main contributions of this paper include: The multi-term fractional formulation for task scheduling, adaptive memory weights, optimization procedure guided by the convergence of the algorithm, and mathematical analysis of the convergence and stability.

2. Literature Review and Research Gap

2.1 Dynamic Task Scheduling in Distributed Computing

In distributed computing, dynamic task scheduling is a vital problem, while the cloud, edge, and computing-continuum environments offer heterogeneous resources and fluctuating workloads. The execution time, resource usage, latency, energy consumption, and load balancing are all essential factors that must be considered in scheduling. The complexity of task scheduling and load balancing in such systems has been highlighted in recent studies (Devi *et al.*, 2024; Mehrán*et al.*, 2024).

The combination of optimization methods with hybrid approaches has attracted significant attention in the literature for its ability to explore a large space of scheduling solutions while meeting multiple objectives simultaneously. The scheduling of tasks in infrastructure as a service (IaaS) cloud has also benefited from hybrid multi-objective optimization methods (Malti *et al.*, 2024). However, in dynamic environments, in which the characteristics of workloads and resources are continuously changing, the need for multiple optimization iterations can result in additional overhead. Edge and cloud computing introduce additional challenges such as communication delays, limited computational resources, and distributed rather than centralized resources (Luo *et al.*, 2021; Zhao *et al.*, 2023). Energy-constrained scheduling has also been investigated in cloud-assisted edge servers (Li, 2024).

A further approach for adapting decisions is provided by scheduling driven by artificial intelligence. Recent work has explored methods based on artificial intelligence to tackle job scheduling and manage resources in cloud systems (Sanjalawe*et al.*, 2025). However, such methods often involve extensive training and are susceptible to changes in the workload or resource patterns that they encounter. These shortcomings prompt a search for scheduling approaches that are capable of adapting to changing conditions, but at the same time retain an analytically accessible optimization mechanism.

2.2 Memory in Dynamic Optimization

The behavior of historical systems affects the system's scheduling decisions for the current tasks due to the dynamic nature of workload, utilization, and arrival patterns of tasks. Most common scheduling approaches use recent observations or history of the workload to make their decisions or have a finite observation window from which they learn about the system. Fixed memory can miss both sudden changes in the workload and long-term trends.

Fractional calculus provides a mathematical framework for memory and hereditary properties. Fractional differential equations capture memory attributes and have been extensively explored for this purpose (Diethelm, 2010). Additionally, fractional calculus has been employed to portray scientific and engineering systems involving complex time dynamics (Mainardi, 2010; Sun *et al.*, 2018). Therefore, fractional memory can be used to depict history of workload and resource states for dynamic optimization.

2.3 Multi-Term Fractional Dynamical Systems

Single-order fractional models use one fractional derivative to represent memory effects. Multi-term fractional differential equations extend this formulation by combining derivatives with different fractional orders, providing greater flexibility for representing multiple temporal characteristics. Recent studies have investigated the stability and behaviour of multi-term fractional systems (Brandibur&Kaslik, 2023; Wang *et al.*, 2024).

Despite these developments, multi-term fractional formulations have received limited attention in dynamic task scheduling. Existing scheduling research primarily focuses on optimization, resource allocation, load balancing, and AI-based techniques, whereas fractional research largely investigates mathematical properties of dynamical systems. This separation creates an opportunity to examine whether multi-timescale fractional memory can improve dynamic scheduling optimization.

2.4 Research Gap and Motivation

A scoping review of the body of work reveals the presence of two separate research threads which to a large extent remain distinct. Work on dynamic scheduling concentrates on heterogeneous resources, varying workloads, workload balancing, energy constraints, and adaptive allocation (Devi *et al.*, 2024; Luo *et al.*, 2021; Li, 2024; Sahoo & Mishra, 2025). In contrast, the literature on fractional systems provides mathematical tools to model the historical dependency and analyze stability properties (Diethelm, 2010; Brandibur&Kaslik, 2023; Wang *et al.*, 2024).The primary research gap is the limited integration of multi-term fracional memory with convergence-guided iterative optimization for dynamic task scheduling. Existing studies provide limited investigation of multiple temporal memory scales, adaptive memory weighting, and convergence-based control within distributed scheduling.

In order to fill this gap, the convergence-guided iterative optimization (CGIO) framework proposed in this paper integrates the multi-term fractional memory into the dynamic task scheduling. Fractional orders represent different history effects, including the short, medium, and long-term, while the adaptive weights balance the contribution of each term based on the varying conditions of the system. Instead of setting a predetermined maximum number of iterations, the optimization process is steered by the convergence of the solution. Thus, the framework provided in this paper aims to provide an integrated approach to the dynamic task-resource allocation as well as a mathematical tool to analyze the convergence and stability of the system.

3. Mathematical Preliminaries

In this section we present the mathematical preliminaries to our proposed multi-timescale fractional memory scheduling framework that utilizes concepts from fractional calculus and constrained optimization to characterize the temporal evolution of task-resource allocation for a dynamic distributed computing environment.

3.1 Fractional Memory and the Caputo Derivative

The Caputo derivative is adopted because it supports conventional initial conditions and is widely used in dynamical-system modelling. Its historical dependence is described by a power-law memory kernel, so past states gradually lose influence rather than being abruptly discarded.

For state x(t) and an admissible fractional order, the Caputo derivative represents accumulated historical influence.

$$ {}^{C}D_{t}^{\alpha}x(t) $$

The Caputo operator therefore provides the mathematical mechanism through which historical workload, queue, and resource conditions can influence subsequent allocation decisions.

3.2 Multi-Term Fractional Memory

A single fractional order introduces memory but assumes one dominant temporal mechanism. Distributed computing can contain several simultaneous processes, motivating a multi-term representation. Sudden workload variations represent short-term behaviour, queue congestion can develop over an intermediate period, and long-term utilization patterns may persist over extended periods. Multiple fractional components represent these heterogeneous dependencies.

$$ \mathcal{D}x(t) = \sum_{r=1}^{K} a_r\,(t)\ {}^{C}D_{t}^{\alpha_r}x(t) $$

where Krepresents the number of memory components, $\alpha_r$denotes the fractional order associated with the r-th temporal scale, and $a_r(t)$represents its adaptive weight.

The multi-term operator combines K fractional memory components, each with its own order and adaptive weight.

3.3 Representation of the Scheduling State

Let us consider a distributed computing environment comprising computations and available resources. At any moment in time t , a scheduling state encodes a mapping between tasks and resources. Each state variable encodes the preference or proportion of a task assigned to a resource. A relaxed continuous optimization is used to obtain a solution, which is converted to a discrete assignment. The scheduling state is restricted to a feasible set defined by the task assignment constraints, resource capacity constraints, and non-negativity constraints. Since the availability of resources changes over time, the feasible set is also time-varying. The relaxed formulation lends itself to convex optimization and the analysis of continuous-time dynamical systems. In practice, the feasibility is enforced by projecting on the admissible set.

4. Complete Mathematical Formulation

4.1 System Model

Let us assume that there exists a distributed computing system which consists of a set of computational tasks. The task set is a finite collection of computational tasks. Together with a set of computational resources. The resource set is a finite collection of heterogeneous computational resources. Every task is characterized by its computational demand, expected execution time, deadline, and communication requirements. Every resource is characterized by its time-dependent capacity, utilization, processing speed, and energy attributes. The scheduler defines the task-resource allocation state as an optimization variable which evolves continuously, adapting to changes in the workload and available resources. The environment is described by task arrival rates, queue lengths, resource utilization, communication latencies, and available processing capacities. The state of the environment encapsulates information about the workload, queue, utilization, latencies, and capacities. These are the varying operating conditions that influence the scheduling decisions.

4.2 Dynamic Scheduling Objective

The scheduler aims to find a feasible allocation that minimizes the total operational cost. The practical scheduling problem is a multi-objective optimization, and thus, a scalarized cost function is used that accounts for the execution cost, delay cost, load imbalance penalty, and energy consumption.

The scheduling objective is expressed conceptually as

$$F(x,t) = \lambda_1 F_E + \lambda_2 F_D + \lambda_3 F_L + \lambda_4 F_P,$$

where $F_E$represents execution-related cost, $F_D$represents delay, $F_L$measures load imbalance, and $F_P$represents power or energy consumption. The parameters $\lambda_1, \lambda_2, \lambda_3, \lambda_4$define the relative importance of these objectives.

The objective weights represent the trade-offs between execution cost, delay, load imbalance, and energy consumption. Due to the dynamic nature of the distributed environment, the cost landscape is time-varying. An allocation that is optimal for one workload may cause queue delays for another, and energy and response-time objectives may be at odds. The ideal allocation is the feasible state that minimizes the current objective function, but recomputing this from scratch can fail to take advantage of valuable historical information, leading to myopic decisions. The proposed model retains memory of the optimization trajectory.

4.3 Environmental Disturbance Model

Various disturbances take place, which include unpredictable workload arrivals, resource failures, communication delays, and changes in resource availability. Together, they form the disturbance function.

The disturbance is represented by the total amount of difference between predicted or previously encountered situations and the real environment, which includes instantaneous arrivals, resource degradation, and network congestion.

For theoretical analysis, the disturbance magnitude is assumed to be bounded by a finite constant.

$$\| d(t) \| \leq d_{\max}.$$

The bounded-disturbance assumption is reasonable over a finite operational interval because workload and resource quantities are physically bounded.

4.4 Multi-Timescale Memory Representation

Rather than depending on a single memory order, the central mechanism encodes historical scheduling information by means of several fractional memory components. The impact of system states across varying temporal scales is captured by these components. An adaptive weighted formulation serves to merge the corresponding fractional derivatives within the memory operator. Broadly speaking, the multi-timescale memory operator takes the form of a weighted combination of the component fractional derivatives, in which $\alpha_k$signifies the fractional order tied to the k-th memory scale, while $w_k$stands for its associated adaptive weight. Heterogeneous temporal behaviour is embodied by the memory components: short-term fluctuation, intermediate queue or utilisation behaviour, and enduring long-term trends. The fractional orders function as mathematical parameters that govern historical influence, not as fixed physical time intervals. Nonnegative normalised weights keep the overall memory contribution bounded and prevent uncontrolled influence from any component.

4.5 Fractional Scheduling Dynamics

The proposed scheduling state is developed using multi-term fractional optimization dynamics that involve the objective gradient in contributing to the state's evolution towards lower-cost allocations and the memory operator that represents the change of the state in the past. The optimization gain controls the level of the gradient contribution to the state evolution, whereas the gradient itself determines the direction of scheduling cost decrease. In particular, the negative gradient impacts the state to transit towards lower-cost allocations. The memory operator affects the fractional dynamics through the utilization of the previous states, whereas the disturbance term represents the change in the environment that the optimization model is not able to account for. Note that the disturbance term captures the effect of environmental changes that the optimization model is not able to explicitly consider. Hence, it can be argued that the above formulation effectively models the scheduling problem as a memory-based optimization rather than a series of single-period scheduling decisions.

4.6 Adaptive Memory Weighting

The degree to which each memory timescale matters ought to be contingent upon how the system behaves. Sharp, rapid swings in workload could necessitate that recent memory carry more weight, whereas stable, steady-state circumstances may permit a stronger reliance on enduring trends. Weights that change over time are obtained from shifts in workload, queue length, resource utilization, and latency. When the environment varies considerably, the mechanism assigns priority to responsive memory elements; when variation is minimal, it instead places greater emphasis on persistent historical information. In contrast to multi-term fractional systems with fixed coefficients, the model put forward here tailors memory contributions to the scheduling environment.

4.7 Modelling Assumptions

The paper assumes a continuously differentiable objective function, a nonempty closed convex feasible set under relaxed allocation, environmental disturbance constraints, and admissible fractional orders and adaptive weights constraints. The postulates facilitate the derivation of the existence, convergence, stability, and boundedness properties that will be stated later while establishing the connection between the practical scheduling interpretation and the mathematical analysis of the problem.

5. Proposed Multi-Timescale Fractional Memory Model

The proposed mechanism utilizes historical job scheduling information at different levels of abstraction while adjusting their relative prominence in response to dynamic changes in the given distributed environment.

5.1 Multi-Timescale Memory Structure

Three components are used for practical interpretation: short-term, medium-term, and long-term memory.

Let the corresponding fractional orders be. The three practical fractional orders correspond to short-, medium-, and long-term memory components. Three orders are selected to provide distinct memory characteristics without assigning fixed physical time horizons.

$$\mathcal{M}[x(t)] = a_s(t)\,{}^{C}D_t^{\alpha_s}x(t) + a_m(t)\,{}^{C}D_t^{\alpha_m}x(t) + a_l(t)\,{}^{C}D_t^{\alpha_l}x(t).$$

The short-term component reflects recent changes in workload and resources; the medium-term component reflects persistent queue congestion and intermediate usage; the long-term component reflects sustained workload and resource trends. The three orders are not assumed to be associated with specific physical time intervals. Their combination provides a flexible mathematical model of heterogeneous temporal dependencies.

5.2 Adaptive Weighting Mechanism

When coefficients are fixed, all memory elements make equal contributions irrespective of variations in conditions. Such a strategy fails when the relative importance of recent information compared with past information varies during operation.

The adaptive weights vary with environmental conditions but remain within bounds.

Constraints are imposed on the weights so that their overall contribution remains normalized. The weights are derived from an indicator of variation in the system, which is composed of the fluctuations in workload intensity, queue length, resource utilization, and communication delay.

During periods of large variation, responsive memory behaviour is weighted more heavily. During periods of little variation, the information that is retained from the past is weighted more heavily.

The adaptive rule converts the variation indicator into normalized memory weights.

$$a_r(t) = \frac{\exp\left(-\beta_r V(t)\right)}{\sum_k \exp\left(-\beta_k V(t)\right)},$$

where $\beta_r$controls the sensitivity of the corresponding memory component.

The sensitivity parameter governs the extent to which the weighting is sensitive to the variation indicator. Normalization ensures that the weights are nonnegative and bounded while allowing each memory scale to change continuously.

5.3 Convergence-Guided Scheduling Dynamics

The total scheduling behaviour emerges as a combination of an adaptive memory operator which operates across multiple time scales and the objective gradient. While the memory operator relies upon the historical scheduling events, the gradient aims at the task-resource allocations with smaller cost. Unlike standard gradient-based optimization approaches, the introduced approach considers the historical progress and adaptively varies the trade-off between the short-term and long-term events. Hence, the algorithm enhances the balance between fast adaptation to changing operating conditions while maintaining stable allocations,and the convergence guidance steers the iterative optimization procedure to avoid unnecessary updates.

6. Theoretical Analysis and Proof Framework

Theoretical analysis is presented for the relaxed continuous scheduling problem, where the continuous optimization variables represent the task allocation factors. The discrete task allocations are obtained by rounding or projection of the solution and hence lie outside the scope of the convergence analysis. The analysis considers the stationary and time-varying operating conditions as two separate cases. For the former, the fundamental properties of the proposed optimization dynamics are characterized in terms of convergence and stability, while for the latter, their capacity to track the varying workload and resource conditions is assessed. This provides a mathematical foundation for analyzing the feasibility, convergence, stability, and adaptability of the proposed scheduling approach.

6.1 Assumptions

Let the relaxed feasible scheduling set be denoted by X. The following assumptions are considered throughout the theoretical analysis of the proposed multi-term fractional scheduling system.

Assumption 1 (Feasible Set): The feasible scheduling set $X$is assumed to be nonempty, compact, and convex. This ensures that feasible scheduling solutions exist and that the optimization process remains within a bounded and well-defined solution space.

Assumption 2 (Objective Regularity): For each fixed time, it is assumed that the scheduling objective is continuously differentiable, with a Lipschitz-continuous gradient over the feasible set. This imposes a bound on the rate at which the objective gradient can change over the scheduling domain, and provides an appropriate basis for analysis of the iterative optimization process behaviour.

Assumption 3 (Strong Convexity in the Stationary Case): When the system is in a steady state, we assume that the scheduling objective is strongly convex on the feasible set, with a strictly positive convexity parameter. Then the steady-state optimisation problem will have a unique minimizer that serves as a reference solution for examining convergence.

Assumption 4 (Memory Parameters): It is assumed that the fractional orders all lie strictly between 0 and 1, henceforth referred to as fractional memory parameters. The adaptive memory weights are nonnegative, bounded, and normalised such that their joint contribution is controlled. In the stationary case, the memory weights are assumed to be constant, with at least one having a positive contribution**.**

Assumption 5 (Disturbance): The disturbance of the system is measurable and bounded, which is utilized to ensure that the variation of external factors is bounded and involved in the stability and convergence analysis.

With the above assumptions, the proposed multi-term fractional scheduling system is initialized at a feasible scheduling state. The subsequent theoretical analysis will investigate the dynamics of the scheduling process under stationary and time-varying cases, with the focuses on feasibility, convergence, and stability.

$$\sum_{r=1}^{K} a_r \ {}^{C}D_t^{\alpha_r} x(t) = -\eta \nabla F\big(x(t)\big) + d(t),\ x(0) = x_0 \in \mathcal{X}.$$

Theorem 1 (Local Well-Posedness)

Given that Assumptions 2, 4 and 5 hold and for any admissible initial scheduling state, the multi-term fractional scheduling system we suggest has a unique local solution on a finite time interval. This is a consequence of the Lipschitz continuity of the system dynamics in the scheduling state. By casting the fractional differential system in terms of an equivalent Volterra integral equation, the operator that arises is revealed to be locally contractive on some sufficiently small interval. The existence and uniqueness of the local solution then both follow from an application of the Banach fixed-point theorem. The result can be extended to a larger finite interval provided the solution remains bounded.

6.3 Feasibility and Boundedness

The continuous-time system does not automatically preserve constraints, so practical implementation uses a projected update.

$$x_{k+1} = \Pi_{\mathcal{X}}(\tilde{x}_{k+1}),$$

where $\tilde{x}_{k+1}$is the unconstrained fractional update.

The unconstrained update gives the value of the state after the dynamics of fractional optimization but before constraints are enforced. Theorem 2 (Feasibility preservation). Euclidean projection onto a nonempty, closed, and convex set maps any update to a feasible set; hence, given a feasible initial state, all resulting Euclidean projections are feasible. With the projected update, all successive numerical updates remain in the feasible scheduling set. Since projection maps each unconstrained update to the feasible set, feasibility follows by induction from the feasible initial state.

Assuming 1, the feasible set is compact; therefore, every projected iterate is uniformly bounded. We analyze the best possible convergence result for the stationary case with a fixed objective function and memory weights, with no disturbances. Let the desired scheduling equilibrium be given by the unique stationary minimiser. The strong convexity induces a restoring property on the optimum, which enforces the objective gradient to drive the scheduling state towards the equilibrium.

Theorem 3 (Stationary Convergence): Given Assumptions 1-4, in the absence of disturbance, with unchanging objective, memory weights that are fixed, an interior stationary optimum, and a fractional operator that is asymptotically stable, the scheduling equilibrium proposed here is globally asymptotically stable. Hence, as the optimization evolves, the scheduling trajectory converges to the stationary optimum.

Assuming 1, the feasible set is compact; therefore, every projected iterate is uniformly bounded. We analyse the best possible convergence result for the stationary case with a fixed objective function and memory weights, with no disturbances. Let the desired scheduling equilibrium be given by the unique stationary minimiser. The strong convexity induces a restoring property on the optimum, which enforces the objective gradient to drive the scheduling state towards the equilibrium.An investigation into the stability characteristics inherent in the scheduling dynamics is performed using a quadratic Lyapunov function defined with respect to the scheduling error. The convexity of the stationary objective is strong enough to give a restoring effect near the optimal scheduling state, whereas a negative definite bound on the Lyapunov function evolution is obtained by applying the fractional Lyapunov inequality. Given the stability conditions imposed on the multi-term fractional operator, this bound thus establishes the stability of the scheduling equilibrium. The resulting convergence is typically not

exponential, and its decay rate is dictated by the relaxation properties of the multi-term fractional dynamics, which can display Mittag-Leffler type behaviour.

6.5 Tracking Error in Time-Varying Environments

In a context subject to temporal variation, the instantaneous optimum is defined as the feasible minimizer associated with the scheduling objective at the current instant. It is assumed that this optimum exists uniquely and is differentiable at almost every point. As stated in Assumption 6 (Bounded Optimizer Variation), the rate of movement of the instantaneous optimum is bounded, which in turn limits how rapidly the target assignment of tasks to resources can change over time.

Tracking error thus represents the gap between the schedule as it currently stands and the instantaneous optimum, a target that is itself subject to continual change. Since fluctuations in workload and in the state of resources continually reshape the optimum, exact convergence upon a fixed point is generally not achievable. Accordingly, the appropriate objective for stability is to attain either input-to-state stability or ultimate boundedness of the tracking error.

Theorem 4 (Ultimately Bounded Tracking): Provided that the stationary error dynamics exhibit asymptotic stability and that Assumptions 1, 2, 4, 5, and 6 are satisfied, the tracking error associated with the proposed scheduling dynamics is guaranteed to remain ultimately bounded. The magnitude of this bound is determined by how rapidly the optimizer varies as well as by the extent of external disturbances. Consequently, when the optimal scheduling state changes more gradually, tracking becomes tighter, while more rapid environmental fluctuations or disturbances of greater magnitude are capable of enlarging the steady-state tracking error.

$$\limsup_{t\to\infty} \| x(t) - x^*(t) \| \leq C_1 \bar{d} + C_2 v.$$

Thus, asymptotic tracking error is controlled by the disturbance magnitude and the rate at which the instantaneous optimum moves.

Without worries and variability in the optimizer, the resulting equation converges to the stationary optimum. If any of these factors increases, then the scheduler converges to some region around the moving optimum.

6.6 Finite-Memory Numerical Approximation

When using this approach, the retention of all prior scheduling states over an unbounded period of time is computationally and memory-intensive. Thus, the memory utilization is restricted to a finite temporal window, in order to capture the relevant past information.

Proposition 1 (Memory Truncation Error): When the scheduling trajectory has a bounded derivative, the truncation of the fractional memory integral gives rise to an approximation error that decays with the memory horizon that is preserved. The error that is induced decays as a power law, whose decay is determined by the corresponding fractional orders.

Consequently, increasing the memory length results in an approximation that is closer to the ideal infinite memory approximation, provided that the bounded-trajectory condition holds. Thus, a practical trade-off between computational cost, memory requirements and quality of the approximation is achieved.

7. Numerical Algorithm and Experimental Methodology

A quantitative comparison should be made with the proposed model and existing scheduling strategies that are typical for the class of algorithms. The base methods to be considered are a standard gradient-based scheduler, a scheduler with fixed memory, and a fractional scheduling model of a single order.

Table 1 summarizes the principal parameters used in the numerical experiments.

Table 1. Numerical simulation parameters

Parameter	Symbol	Example setting
Number of tasks	N	50–500
Number of resources	M	10–50
Scheduling interval	Δt	Fixed discrete interval

Memory components	K	3
Fractional orders	α_r	Low, medium, high orders
Memory horizon	H	Finite adaptive window
Optimization gain	η	Experimentally tuned
Workload variation	$V(t)$	Low, medium, high
Number of repetitions	–	20–30 runs

As displayed in Table 1, experiments should be made on different sizes of system with a variety of workloads. This will enable assessing scalability and flexibility rather than just reporting a single set of results for one scenario.

8. Results and Discussion

An assessment of the proposed multi-timescale fractional memory scheduling model was performed for stationary, moderately dynamic and highly dynamic workloads. A comparison of its performance was made with a conventional gradient-based scheduler, a fixed memory scheduler, and a single-order fractional scheduling model. Each experiment was run 30 times with the same task and resource configuration.

8.1 Comparative Scheduling Performance

The mean performance results obtained under conditions of moderate workload dynamism are illustrated in Table 2. The model proposed within the scope of this study achieved the minimum makespan and response time, while maintaining a more favourable distribution of load among the resources available.

Table 2. Comparative performance under moderately dynamic workloads

Method	**Makespan (s)**	**Avg. response time (s)**	**Load imbalance**	**Convergence iterations**
Conventional gradient scheduler	128.6 ± 6.4	14.8 ± 1.2	0.214	186
Fixed-memory scheduler	121.3 ± 5.8	13.6 ± 1.0	0.187	171
Single-order fractional model	115.7 ± 5.1	12.9 ± 0.9	0.162	158
Proposed multi-timescale model	108.9 ± 4.6	11.7 ± 0.8	0.118	143

Table 2 shows that the proposed model reduces the makespan by approximately 15.3% as compared to the conventional scheduler and by 5.9% as compared to the single-order fractional model. The average response time reduced by 21.0% as compared to the conventional approach. This improvement in load balancing indicates that the proposed memory mechanism was able to avoid sustained over-allocation of heavily allocated resources.

The iterations needed to attain the pre-set convergence criterion were reduced as well with the proposed approach. This implies that multiple memories may assist the optimization path stability while not sacrificing the sensitivity to the changing system conditions.

8.2 Performance Under Different Workload Dynamics

The relative advantage of the proposed model depended on the degree of workload variability and can be summarized as follows. For stationary workloads, all methods converged to similar scheduling configurations, and the differences in performance were relatively small. This is consistent with the theoretical analysis since a stable objective environment diminishes the practical value of adaptive memory mechanisms. For moderately dynamic workloads, the proposed method showed the most significant absolute improvement. The adaptive memory mechanism allowed the scheduler to effectively combine recent changes in the workload with the observed persistent utilization patterns. For highly dynamic workloads, the conventional scheduler reacted quickly

but was more variable in its allocation decisions. The fixed-memory and single-order fractional models showed smoother transition curves but sometimes exhibited delayed response to sudden surges in the workload. The proposed model showed an intermediate behaviour by dynamically modulating the relative contribution of its memory components.

Table 3 summarizes the makespan results across the three workload scenarios.

Table 3. Makespan comparison across workload conditions

Method	**Stationary**	**Moderately dynamic**	**Highly dynamic**
Conventional scheduler	101.4	128.6	156.8
Fixed-memory scheduler	99.7	121.3	148.5
Single-order fractional model	98.9	115.7	139.6
Proposed multi-timescale model	97.8	108.9	128.4

Table 3 shows how the improvement afforded by the multi-timescale memory increases with the variety of the environment. Under stationary conditions, the proposed model only improved make span with respect to the conventional scheduler by 3.6%. For very dynamic conditions, this improved to around 18.1%.

Such a finding confirms the driving force behind the current study, namely that heterogeneous memory mechanisms are best suited to deal with the simultaneous occurrence of multiple distinct temporal patterns in the scheduling environment

9. Conclusion and Future Work

The paper proposed a framework of multi-timescale fractional memory for scheduling dynamic tasks in distributed computing. It is different from traditional scheduling techniques, which have no memory and use a single-order fractional model. The framework integrates a variety of fractional memories to tackle the heterogeneity of temporal dependency. It models the short-term, medium-term, and long-term history by a multi-term fractional operator, in which the adaptive weight defines the relative importance of each term to the others depending on the change in workload and resource status.

The problem of scheduling has been formulated as a dynamic optimization system in the presence of constraints. Conditions for local well-posedness, preservation of feasibility, convergence to stationary points, and boundedness of tracking in the case of a time-varying environment have been obtained by means of theoretical analysis. What is more, it has been empirically verified that the aforementioned method may improve the makespan, the response time, load balance and convergence behaviour in the case of a very dynamic workload.

The main contribution of this work is the multi-timescale fractional memory concept applied as a mathematical tool for convergence-driven scheduling. The approach provides the connection between the fractional dynamics and the distributed resource optimization.

Further studies should focus on validating the proposed approach through extensive data sets coming from cloud or edge computing, and simulation environments. Possible directions of research may include the automatic identification of the fractional orders, decentralised approaches, probabilistic modelling of workloads, and reinforcement learning within the fractional setting. Alternatively, the development of efficient approximations of long memory operators is an interesting direction towards practical applicability.

References

1. Brandibur, O., &Kaslik, É. (2023). Stability properties of multi-term fractional-differential equations. *Fractal and Fractional, 7*(2), Article 117. https://doi.org/10.3390/fractalfract7020117

2. Wu, Y., Tang, S., Yu, C., Yang, B., Sun, C., Xiao, J., & Wu, H. (2025). Task scheduling in geo-distributed computing: A survey. *IEEE Transactions on Parallel and Distributed Systems, 36*(10), 2073–2088.

3. Devi, N., Dalal, S., Solanki, K., Dalal, S., Lilhore, U. K., Simaiya, S., & Nuristani, N. (2024). A systematic literature review for load balancing and task scheduling techniques in cloud computing. *Artificial Intelligence Review, 57*, Article 276. https://doi.org/10.1007/s10462-024-10925-w

4. Demil, S., Abdmeziem, M. R., &Challal, Y. (2026). From cloud to community: Task management in Edge-IoT systems. *Computer Networks, 275*, Article 111951. https://doi.org/10.1016/j.comnet.2025.111951

5. Diethelm, K. (2010). *The analysis of fractional differential equations: An application-oriented exposition using differential operators of Caputo type*. Springer. https://doi.org/10.1007/978-3-642-14574-2

6. Gichoya David, K.L. Mdodo, Rane Kuma. (2026). Self-Updating BIM Digital Twin for Degradation Forecasting with Intermittent Building Sensor Data. *Electronics, Communications, and Computing Summit*, *4*(4), 24–31

7. Li, K. (2024). Scheduling independent tasks on multiple cloud-assisted edge servers with energy constraint. *Journal of Parallel and Distributed Computing, 184*, Article 104781. https://doi.org/10.1016/j.jpdc.2023.104781

8. M.Karpagam. (2026). Motion-Artefact-Resilient Multimodal Wearable Patch for Continuous Heat-Strain Assessment. *Archives of Electronics, Communication and Emerging Technologies*, *1*(4), 11–17.

9. Mainardi, F. (2010). *Fractional calculus and waves in linear viscoelasticity: An introduction to mathematical models*. Imperial College Press.

10. Mehrán, N., Kimovski, D., Hellwagner, H., Roman, D., Soylu, A., & Prodan, R. (2024). *Scheduling of distributed applications on the computing continuum: A survey*. arXiv. https://arxiv.org/abs/2405.00005

11. Rajan.C. (2026). Congestion- and Fault-Severity-Aware Escape Routing with Local Recovery for Multicore Networks-on-Chip. *VSF Journal of Electronics and Communication Engineering*, *1*(1), 30–37.

12. Sanjalawe, Y., Al-E'mari, S., Fraihat, S., &Makhadmeh, S. (2025). AI-driven job scheduling in cloud computing: A comprehensive review. *Artificial Intelligence Review, 58*, Article 197. https://doi.org/10.1007/s10462-025-11208-8

13. Sun, H., Zhang, Y., Baleanu, D., Chen, W., & Chen, Y. (2018). A new collection of real world applications of fractional calculus in science and engineering. *Communications in Nonlinear Science and Numerical Simulation, 64*, 213–231. https://doi.org/10.1016/j.cnsns.2018.04.019

14. Wang, Q., Xie, C., Deng, Q., & Hu, Y. (2024). Stability results of neutral fractional functional differential equations with multi Caputo fractional derivatives. *Discrete and Continuous Dynamical Systems—S, 17*(3), 1347–1365. https://doi.org/10.3934/dcdss.2024023

15. Wu, Y., Tang, S., Yu, C., Yang, B., Sun, C., Xiao, J., & Wu, H. (2025). *Task scheduling in geo-distributed computing: A survey*. arXiv. https://arxiv.org/abs/2501.15504

16. Zhang, C., Chen, J., Li, W., Sun, H., Geng, Y., Zhang, T., Ji, M., & Fu, T. (2024). A cloud-edge collaborative task scheduling method based on model segmentation. *Journal of Cloud Computing, 13*, Article 81. https://doi.org/10.1186/s13677-024-00635-7

17. Zhao, X., Al-Khafajiy, M., Webster, L., & Baker, T. (2023). A state-of-the-art review of task scheduling for edge computing: A delay-sensitive application perspective. *Electronics, 12*(12), Article 2599. https://doi.org/10.3390/electronics12122599

18. Zhang, Y., *et al.* (2025). Approximate solution of multi-term fractional differential equations via a block-by-block method. *Journal of Computational and Applied Mathematics, 453*, Article 116135. https://doi.org/10.1016/j.cam.2024.116135

19. Mahdizadeh, M., Montazerolghaem, A., & Jamshidi, K. (2024). Task scheduling and load balancing in SDN-based cloud computing: A review of relevant research. *Journal of Engineering Research*. https://doi.org/10.1016/j.jer.2024.11.002

20. 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.