INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

ORIGINAL ARTICLE

PSTL-PPSO: A Hybrid Transformer and Swarm-Based Model for Load Balancing in Cloud Computing

V Infine Sinduja[1]*, **P Joseph Charles**[1]

1 St. Joesph's College (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli -620 002, Tamil Nadu, India

OPEN ACCESS

Received: 22/06/2025

Accepted: 09/07/2025

Published: 29/07/2025

Citation: Sinduja VI, Charles PJ (2025) PSTL-PPSO: A Hybrid Transformer and Swarm-Based Model for Load Balancing in Cloud Computing. Indian Journal of Science and Technology 18(29): 2353-2368. https://doi.org/10.17485/IJST/v18i29.1051

* **Corresponding author**.

infinesinduja@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: To improve the throughput with minimum energy consumption in cloud load balancing by using proposed Position Sensitive Transformer Learning and Parallel Particle Swarm Optimization (PSTL-PPSO).**Methods:** This work proposes Position Sensitive Transformer Learning and Parallel Particle Swarm Optimization (PSTL-PPSO). Position Sensitive Attention-based Transformer Learning is initially applied to distribute enormous volume of data, to name a few being CPU usage, memory usage, network usage and potentially user-specific information. The contribution of PSTL-PPSO improving throughput, overhead efficiency with minimum response time. The collected data is then fed as input into a Position Sensitive Attention-based Transformer Learning model to analyze the relationships and patterns to comprehend prevailing system state. Based on the Position Sensitive Attention function the Transformer Learning model predicts optimal workload distribution across servers aiming to minimize makespan and energy consumption. Next, Parallel Particle Swarm Optimization (PPSO) algorithm is applied for fine-tuning cloud load balancing where a fitness function is provided to measure quality of each load balancing configuration on the basis of response time and throughput. The PPSO algorithm fine-tunes the position of each particle in an iterative fashion, exploring distinct load balancing alternatives in parallel fashion, therefore ensuring faster convergence optimally. Simulations are performed to validate the proposed PSTL-PPSO method in Cloud Simulator and Java programming language in terms of makespan time, energy consumption, high accuracy, associated overhead and throughput. **Findings:** PSTL-PPSO method improves the throughput by 19% and reduces makespan 17%, energy consumption by 26%, overhead efficiency by 26%. **Novelty:** The integration of Position Sensitive Attention-based Transformer Learning with Parallel Particle Swarm Optimization to enable adaptive, energy-efficient, and fast-converging cloud load balancing under dynamic workloads. This hybrid model is applicable to cloud platforms, data centers, and edge computing environments, where efficient workload distribution and resource utilization are critical**.**

Keywords: Cloud Computing; Deep Learning; Position Sensitive Attention; Transformer Learning; Parallel Particle Swarm Optimization

1 Introduction

With the swift evolution of information technology (IT), cloud computing (CC) has grown to sophisticated information structure. Load is a pivotal performance metric for measuring cloud platforms performance and efficiency that encompasses a rigorous analysis of prevailing and historical resource usage patterns for anticipating future trends, in that way making effective and optimal planning of resources. In addition, prediction efficiency is another paramount element that influences cloud platforms stability. Hence, it is emerging yet demanding to design a mechanism for minimizing resource consumption and boosting the overall data centers operational efficiency.

A Wavelet decomposition-enhanced External Transformer (WETformer) was designed in [1] with the aim of capturing complicated temporal patterns therefore ensuring efficient load prediction for cloud servers. At first, discrete wavelet transform was first incorporated with the objective of extracting long-term trends, focusing the inherent features of temporal sequences. Following which a lightweight multi-head External Attention (EA) mechanism was designed parallel taking into consideration the inter associations within load sequences and the correlations across different sequences, therefore improving prediction efficiency with lesser inference time. Despite minimizing the inference time but the makepan one of the important load prediction performance metrics was not analyzed. To address on this aspect, Position Sensitive Attention-based Transformer Learning is introduced that with the aid of position sensitive attention function aids in reducing makespan with minimum energy consumption.

An innovative and efficient Improved Lion Optimization (ILO) with Min-Max Algorithm, breaking new ground in the research to use Genetic Operators for parallelization was proposed in [2]. By employing Improved Lion Optimization algorithm both load balancing & power conservation were ensured. In addition, by employing the Min-Max Algorithm across various substrate structural configurations aided in reducing maximum energy consumption along with the minimization of virtual machine migration considerably. However, the overhead incurred was not focused. To concentrate on this research gap, Parallel Particle Swarm Optimization (PPSO) algorithm that with the parallel nature not only enhances convergence speed but also minimizes overhead considerably. Accurate electrical load forecasting is advantageous for more efficient scheduling of electricity generation and reducing electrical energy. In [3], an adaptive deep-learning load forecasting framework by combining Transformer and domain knowledge (Adaptive-TgDLF) was proposed. Moreover, an adaptive learning to address the issues arising due to insufficient historical load data and load distribution change were also designed. Here using transfer learning and online learning in turn enhanced generalization ability both in terms of space and time, extensively, therefore minimizing the MSE and training time extensively. With the evolution of a new surge of applications and services via the internet, a new emergence of information dawn of information outburst is happening. Owing to this, there is a massive increase in the necessity for data storage and data processing. Cloud computing environment creates the on-demand access of for profuse computing services.

An innovative machine learning technique for ensuring high performance cloud computing and load balancing was proposed in [4] with high-cost savings. However, the load imbalance aspect was not considered. A method to focus on the load imbalance issue employing MIMO fuzzy logic was presented in [5] therefore minimize load imbalance drastically. However, prevailing load balancing techniques treated these workloads as monolithic jobs, therefore compromising diverse operational objectives. To address on this aspect, large language model was designed in [6] that in turn minimized end-to-end delay in an extensive manner. In this contemporary epoch of information technology, CC environment acts as the foundation of computing services made on-demand. Workload prediction is one type of time series forecasting. However, it has consistently been a substantial issue to impart accurate workload predictions. An efficient load balancing employing machine learning was presented in [7] toward sustainable energy management. However, it did not include both long term and short-term prediction. To focus on this aspect, a recurrent neural network-based prediction model was designed in [8] therefore ensuring accuracy. Yet another method with short term forecasting employing deep learning was proposed in [9] using k-Medoid based algorithm. Also, with the aid of this algorithm resulted in the minimization of overall training time. However, interpretability and performance aspects were not focused. To concentrate on this issue, masked encoding was employed as a pre-coding in [10] for cloud transformer. This in turn ensured both interpretability and performance aspects extensively.

Cloud load balancing is the process of distributing network traffic and workloads across numerous servers or resources in a cloud computing. A conventional Wavelet decomposition-enhanced External Transformer provides insufficient throughput. Optimization technique, deep learning gives deficient latency. Machine learning technique and fuzzy logic does not achieve higher throughput and efficiency. Load balancing and recurrent neural network does not minimize the energy consumption. In every existing method unable to focus on computation cost minimization and time minimization. To concern these issues, a new technique called Position Sensitive Transformer Learning and Parallel Particle Swarm Optimization (PSTL-PPSO) is developed for improving the load balancing efficiency in cloud with minimum time consumption. Position Sensitive Attention-based Transformer Learning is used for minimizing the energy consumption. Parallel Particle Swarm Optimization (PPSO)

algorithm is utilized for fine-tuning cloud load balancing where a fitness function is given to calculate quality of every load balancing configuration for reducing the makespan.

1.1 Major Contributions

Motivated by the above issues, like, makespan, overhead, energy consumption and throughput in cloud load balancing, in this work, transformer-based method called, Position Sensitive Transformer Learning and Parallel Particle Swarm Optimization (PSTL-PPSO) is proposed. Novelty of work are pointed below.

- To present a significant method for cloud load balancing called Position Sensitive Transformer Learning and Parallel Particle Swarm Optimization (PSTL-PPSO).
- To design Position Sensitive Attention-based Transformer Learning (PSA-TL) for load balancing with the intent of improving makepsan and energy consumption involved during the process.
- To propose a Parallel Particle Swarm Optimization-based fine-tuned cloud load balancing for making certain convergence-efficient fine-tune load balancing is made.
- Finally, performance of the proposed transformer learning based PSTL-PPSO method is compared with the conventional state-of-the-art methods.

1.2 Organization of the Paper

The rest of the paper is arranged in the following fashion. Related works are discussed in section 2. Section 3 exhibits the complete architecture of our proposed method, Position Sensitive Transformer Learning and Parallel Particle Swarm Optimization (PSTL-PPSO). Section 4 demonstrates the analysis and metrics of the experimental results following by discussion with the aid of table and graphical representation. In section 5, we conclude and summarize the whole paper. Research gaps :Cloud Computing (CC) nevertheless encounters several issues concerning load balancing. Different techniques have been applied in the literature to enhance load balancing efficiency. A Wavelet decomposition-enhanced External Transformer provides insufficient accuracy. Improved Lion Optimization (ILO) with Min-Max Algorithm gives deficient makespan. Optimization technique and genetic algorithm ,deep-learning load forecasting methods do not achieve higher throughput. In every existing method unable to focus on computation cost minimization and time minimization. To concern these issues, a new technique called Position Sensitive Transformer Learning and Parallel Particle Swarm Optimization (PSTL-PPSO) is developed for improving the cloud load balancing with minimum time consumption.

A review of machine learning techniques for cloud load balancing was investigated in [11]. However, owing to the complicated loading environment and the ineffectiveness of feature extraction process load prediction still considered as a demanding issue. A long short-term memory with map reduces was proposed in [12] to get better outcomes in resource allocation. As far as CC is concerned, making certain the high availability and reliability is predominant for significant content delivery. Nevertheless, optimal replication management while taking into consideration arbitrary changes remains a formidable issue. In order to address these aspects, Dynamic Data Replication Strategy (TD2RS) with Temporal Fusion Transformer (TFT) was proposed in [13]. This by collecting historical data on content popularity dynamically addressed availability and reliability extensively. Multivariate time series data were employed in [14] in addition to network throughput to not only reduce the error rate but also to improve throughput in an extensive manner. Yet another method to ensure low latency employing cosine similarity was proposed in [15] to accurate load balancing. Effective resource scheduling is one of the important issues in materializing CC. Taking into consideration the increasing complexity of Cloud computing, future Cloud systems will necessitate more efficient resource management methods. A review of deep learning techniques four efficient resource scheduling in CC environment was presented in [16]. A novel method combining decision function with normal distribution was proposed in [17] with minimal response and makespan time. Yet another hierarchical taxonomy classification employing deep and machine learning for load balancing was presented in [18]. However, it did not minimize response time. To focus on this response time aspect, deep reinforcement learning was applied in [19] for better reliability and efficiency. A light weight deep learning technique to focus on the throughput involved in workload distribution was proposed in [20]. Nevertheless, in energy-efficient task scheduling the centralized control architecture brings challenges. A novel method employing Entropy Oppositional Based Learning-Interpolation along with Blue Monkey Optimization Algorithm (EOBL-IBMOA) was proposed in [21] with minimal response time and high throughput. A hybrid deep learning technique with main emphasize on minimizing average error rate employing long short term and convolutional neural network was presented in [22]. By using this hybrid mechanism in turn minimized the average error rate. The pros and cons involved in the analysis of load balancing techniques were made in [23]. Yet another method employing recommendation was proposed in [24] using federated learning.

Identifying accurate short-term load data is fundamental for meticulous research and formulation of policy. A machine learning framework potential of forecasting future workloads was presented in [25] by taking into consideration both of temporal autocorrelation and inter-feature correlations. By taking into consideration both temporal and inter feature correlation in turn ensured high degree of generalizability with minimal energy consumption. A two-stage genetic mechanism for workload distribution by means of migration in cloud computing environment was presented in [26]. Here, two genetic-based methods were combined where initially virtual machines performance models were extracted and following which corresponding performance measurement were made in a cloud computing environment. Routing towards efficient workload distribution was designed in [27] to minimize transmission cost. Also, ensemble learning was applied in [28] to ensure accurate predictive performance. Yet another method based on point cloud classification was presented in [29] that by considering both the local and global feature enhanced the learning capability. In summary, the key objective behind the above-mentioned works is to minimize the makespan while predicting workloads and finally fine-tune to ensure converge-efficient load balancing. Throughput-efficient load balancing is necessary to assign resources more precisely between different cloud user requested tasks. For the above-mentioned reasons, we propose a Position Sensitive Transformer Learning and Parallel Particle Swarm Optimization (PSTL-PPSO) to gain better throughput in cloud load balancing by significantly minimizing the makespan and energy consumption. Our model can balance learning techniques.

2 Methodology

In this section a method called, Position Sensitive Transformer Learning and Parallel Particle Swarm Optimization (PSTL-PPSO) to take advantage of parallel processing capabilities in achieving faster and accurate load balancing is designed. **Figure1** shows the framework of PSTL-PPSO method.

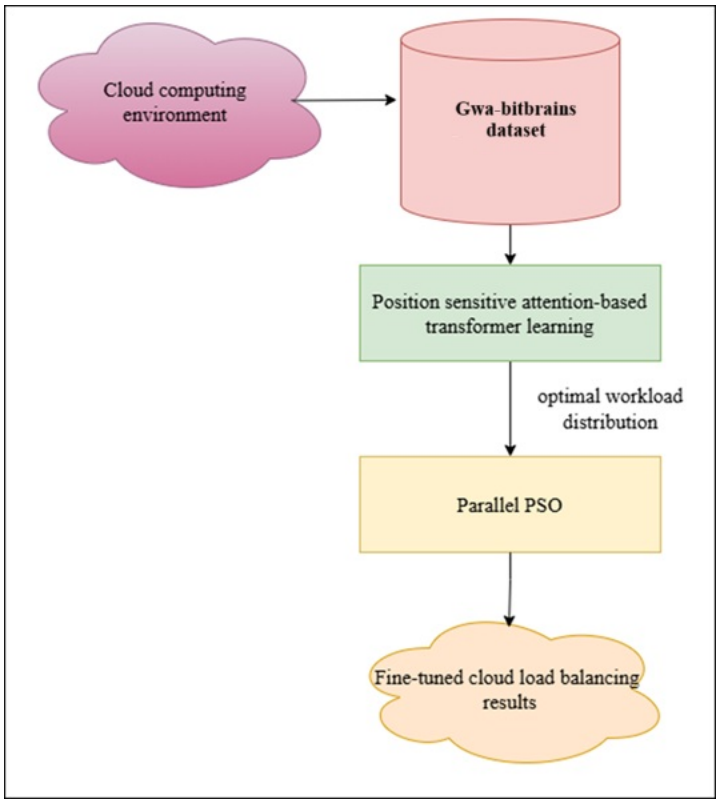

Fig 1. Framework of PSTL-PPSO method

As shown in figure 1, the framework of PSTL-PPSO method is split into three parts, namely data collection, optimal workload distribution and fine-tuned cloud load balancing results. First the raw data are obtained as input from gwa-bitbrains dataset. Following which Position Sensitive Attention-based Transformer Learning for Load Balancing based optimal workload distribution model is applied to the generated data. Here piecewise regression is performed based on the breakpoint to

generate the optimal workload distribution results with minimal makespan and energy consumption. Finally, optimal workload distribution results are subjected to Parallel Particle Swarm Optimization to generate fine-tuned cloud load balanced results accurately and precisely. The elaborate description of the PSTL-PPSO method is provided in the following sub-sections.

2.1 Position Sensitive Attention-based Transformer Learning for Load Balancing

Transformer learning in general is used in achieving dynamic balancing of load in Cloud Computing (CC) environment with the intent of boosting both utilization of resource and network performance. By employing transformer models can aid in predicting traffic patterns and then utilizing these predictions to steer load balancing decisions. Position Sensitive Attention-based Transformer Learning (PSA-TL) for load balancing in our work influences positional information (i.e. organized according by traces) to improve the attention mechanism. This permits the proposed model in better acquiring local context and relationships within the input sequence (i.e. first trace, fastStorage and second trace, Rnd) resulting in more accurate and efficient representations for load balancing in CC environment. In the proposed method, PSA-TL is initially applied to distribute enormous volume of data (i.e. increased traffic) across available cloud resources. By using the PSA-TL complex relationships between various criterions (i.e. timestamp and CPU cores) are extracted enabling load balancing decision upon comparison to straightforward threshold-based mechanism. PSA-TL model comprises of three segments. They are feature extraction, encoding and regression.

The feature extraction segment comprises of a fully connected layer along with position encoding that does the task of non-linear dimensionality reduction on the raw data obtained from Bitbrains, distributed datacenter and encompasses positional information. The Load Balancer 'LB' collects data (i.e. CPU usage, memory usage, network usage and potentially user-specific information) from different Cloud Servers 'CS ={[CS]_1, [CS]_2…, [CS]_N }'. The collected data is then fed as input into a PSA-TL model for analyzing relationships and patterns to comprehend the prevailing system state. Finally optimal workload distribution prediction across servers is made aiming at minimizing makespan. The encoding segment in addition compresses and extracts relevant features employing Position Sensitive Attention mechanism to capture more contextual information, therefore boosting the model's potentiality to include both positional and local context simultaneously. The Position Sensitive Attention mechanism enables the model to concentrate on the positional relationships of input data at each time step, thereby enhancing its capability to capture local context, where the positional relationship here refers to how the Load Balancer (LB) distributes incoming traffic to available Cloud Servers (CS) within a group. This approach measures Affinity Score between each input element (i.e. samples) and all other elements based on these affinity scores and the output is model by taking into consideration the weighted sum. By using positional information, boosts the model's potentiality in capturing local context, resulting in improved feature representations, therefore producing more efficient hidden features for accurately predicting the load. Finally, the regression segment uses the proposed Piecewise Regression to adjust incoming data based on different load conditions. Upon comparison to traditional regression model, it can more efficiently learn from hidden features, thereby minimizing energy consumption extensively.

To start with the input is constructed as given below.

$$S = \{S_1,\ S_2,\ \ldots, S_n\}; S = \begin{bmatrix} S_1F_1 & S_1F_2 & .. & S_1F_m \\ S_2F_1 & S_2F_2 & .. & S_2F_m \\ .. & .. & .. & .. \\ S_nF_1 & S_nF_2 & .. & S_nF_m \end{bmatrix} \tag{1}$$

$$PE = \{PE_1,\ PE_2,\ \ldots, PE_n\} \tag{2}$$

From the above **equations (1) and (2)** the input to the load prediction model consists of input sequence samples 'S' with relevant position encoding 'PE' where 'S_i' denotes the sample representation of the '$i-th$' element and 'PE' denotes the positional encoding of cloud sample position 'PE_i'.

The feature extraction section then comprises of fully connected (FC) layer that does the task of non-linear dimensionality reduction on the raw data employing Gaussian Radius Basis function (GRB) and incorporates positional information. The feature extraction process is then formulated as given below.

$$k\left(F_i, F_j\right) = e^{\left(-\gamma\left|F_i - F_j\right|^2\right)} = e^{\left(-\gamma\left(F_i^2 + F_j^2 - 2F_iF_j\right)\right)} \tag{3}$$

$$= e^{\left(-\gamma\left(F_i^2 + F_j^2\right) + 2\gamma F_iF_j\right)} = e^{-\gamma\left(F_i^2 + F_j^2\right)}.e^{2\gamma F_iF_j} \tag{4}$$

From the above **equations (3) and (4)** result '$k\left(F_i, F_j\right)$' by applying GRB function transform high-dimensional data into a lower-dimensional space while preserving data's essential non-linear relationships. Following which for each sample 'S_i' the affinity score of 'AS_{ij}' in the input sequence taking into consideration the impact of position encoding 'PE_{ij}' is written as given below.

$$AS_{ij}^{1} = Affinity\ \left(S_i PE_{ij}\right) = \frac{\left(S_i.\ PE_{ij}\right)}{\left(\left|S_i * PE_{ij}\right|\right)} \tag{5}$$

$$AS_{ij}^{2} = Affinity\ \left(S_i PE_{ij}\right) = \frac{\left(S_i.\ PE_{ij}\right)}{\left(\left|S_i * PE_{ij}\right|\right)} \tag{6}$$

$$AS_{ij} = Affinity \frac{\left(AS_{ij}^{1}.AS_{ij}^{2}\right)}{\left(\left|AS_{ij}^{1} * AS_{ij}^{2}\right|\right)} \tag{7}$$

Next, for each sample 'S_i', the attention weight 'W_{ij}' is measured based on the above affinity score 'AS_{ij}' results as given below.

$$W_{ij} = softmax\left(AS_{ij}\right) = \frac{\exp\left(AS_{ij}\right)}{\sum_{k=1}^{n} \exp\left(AS_{ik}\right)} \tag{8}$$

Finally, the output results are obtained by evaluating weighted sum of attention weights 'W_ij' to the input sample 'S_ij' as given below.

$$S' = \sum_{j=1}^{n} W_{ij} S_j \tag{9}$$

From the above equation results (**Equation 9**) the proposed model takes into consideration both the association between elements or samples and the impact of position encoding, generating more accurate and position-sensitive attention weights. Finally, the Piecewise Regression employed in our work uses a linear model within specific segments of workload instead of a single linear model across the entire rage with the intent of adaptively adjusting incoming data based on different load conditions. By defining these pieces and their subsequent linear models, the load balancer adjusts incoming data dynamically based on the current workload, therefore optimizing resource utilization. The Piecewise Regression analysis is based on the presence of a set of '(dep, S)' data in which 'dep' forms the dependent variable with 'S' denoting the independent variableThe least square method is applied separately to each piece, by which two regression lines are made to fit as closely as possible while minimizing sum of squares of differences between observed 'dep' and calculated 'Res' values, generating the following equations.

$$Res = C_1.S + c_1,\ if\ S < BP \tag{10}$$

$$Res = C_2.S + c_2,\ if\ S > BP \tag{11}$$

From the above **equations (10) and (11)** 'Res' represent the predicted or calculated result for a certain value of 'S' via regression coefficient 'C_1', 'C_2' and regression constants 'c_1', 'c_2' upon meeting the condition with respect to breakpoints 'BP' (i.e. mean usage or network received throughput in our work). For example, with the regression coefficient values being 3, it refers to that for every one unit increase in 'S' the predicted value of 'dep' will increase by 3 units within that piece. Also, with the assumption that the regression constant is '4' it denotes that when 'S' is zero the predicted value of 'dep' will be '4' within that piece.

Algorithm 1. Position Sensitive Attention-based Transformer Learning Input: Dataset 'DS', Features '$F = \{F_1, F_2, \ldots, F_m\}$', Samples '$S = \{S_1, S_2, \ldots, S_n\}$', Cloud Server'$CS = \{CS_1, CS_2, \ldots, CS_N\}$'
Output: Energy-efficient load balancing
Step 1: **Initialize** '$m = 11$', '$n = 50000$', '$N = 10$', regression coefficient '$C_1 = 3$', '$C_2 = 3$', regression constants '$c_1 = 4$', '$c_2 = 4$', Load Balancer 'LB'
Step 2: **Begin**
Step 3: **For** each Dataset 'DS' with Features 'F', Samples 'S', Cloud Server 'CS' and Load Balancer 'LB'
//Construction of Raw Input data
Step 4: Formulate samples to model load balancing according to (**Eq. 1**) and (**Eq. 2**)
//Feature extraction
Step 5: Perform feature extraction process by employing Gaussian Radius Basis function (GRB) according to (**Eq. 3**) and (**Eq. 4**)
//Encoding segment
Step 6: Measure affinity score with positional encoding according to (**Eq. 5**), (**Eq. 6**) and (**Eq. 7**)
Step 7: Evaluate attention weight based on the affinity score according to (**Eq. 8**)
Step 8: Measure weighted sum of attention weights according to (**Eq. 9**)
//Regression segment
Step 9: Formulate Piecewise Regression function according to (**Eq. 10**) and (**Eq. 11**)
Step 10: Generate regression results
Step 11: **End for**
Step 12: **End**

As given in the above algorithm the overall process of load balancing is split into three segments. In the first segment, raw input data are constructed employing the Grid Workloads Archive-T-12 Bitbrains dataset. Following which feature extraction is performed by applying the fully connected layer via the load balancer.

Here, Gaussian Radius Basis function (GRB) is employed that find mappings that account for complex relationships between features, therefore visualizing complex data in 3D plots extensively. This in turn aids in minimizing the overall makespan. Next, encoding process is performed using Position Sensitive Attention mechanism. Here positional encoding for each extracted feature is performed by applying Gaussian Radius Basis function (GRB). By using this GRB function aid in capturing non-linear relationships, therefore ensuring energy efficiency during the process of load balancing. Finally, Piecewise Regression function is applied to generate regression results therefore making certain optimal load prediction results is generated.

2.2 Parallel Particle Swarm Optimization-based fine-tuned cloud load balancing

Next, Parallel Particle Swarm Optimization (PPSO) algorithm is applied for fine-tuning cloud load balancing results. Each particle in the PPSO algorithm denotes a probable load balancing configuration, representing how cloud user requested tasks 'T' are allocated to distinct virtual machines 'VM'. Following which a fitness function 'fun' is provided to measure quality of each load balancing configuration on the basis of response time and throughput.

Finally, the PPSO algorithm fine-tunes the position of each particle in an iterative fashion, exploring distinct load balancing alternatives in parallel fashion, therefore ensuring faster convergence optimally. Here by taking advantage of parallel processing potentialities with the intent of fine-tuning workload distribution in a dynamic fashion across available servers, ensures faster and accurate load balancing. Figure 2 shows the block diagram of Parallel Particle Swarm Optimization-based fine-tuned cloud load balancing.

As shown in the figure 2 the main swarm is split into a number of sub-swarms where each sub-swarm functions as a single PSO and each sub-swarm runs parallel to minimize the response time for acquiring the best result. Also as illustrated above the Parallel Particle Swarm Optimization-based fine-tuned cloud load balancing model is split into two phases, namely single-evolutionary and multi-evolutionary phase.

The swarm or data center 'DC' is arbitrarily split into 'k' sub-swarms or cloud servers 'CS', during multi-evolutionary phase. Each sub-swarm or cloud servers comprises of a number of particles or virtual machines 'VMs', each of which can be evaluated independently of the swarm.

$$fun = \min\left(Throughput, RT\left[T_i, T_j\right]\right) \tag{12}$$

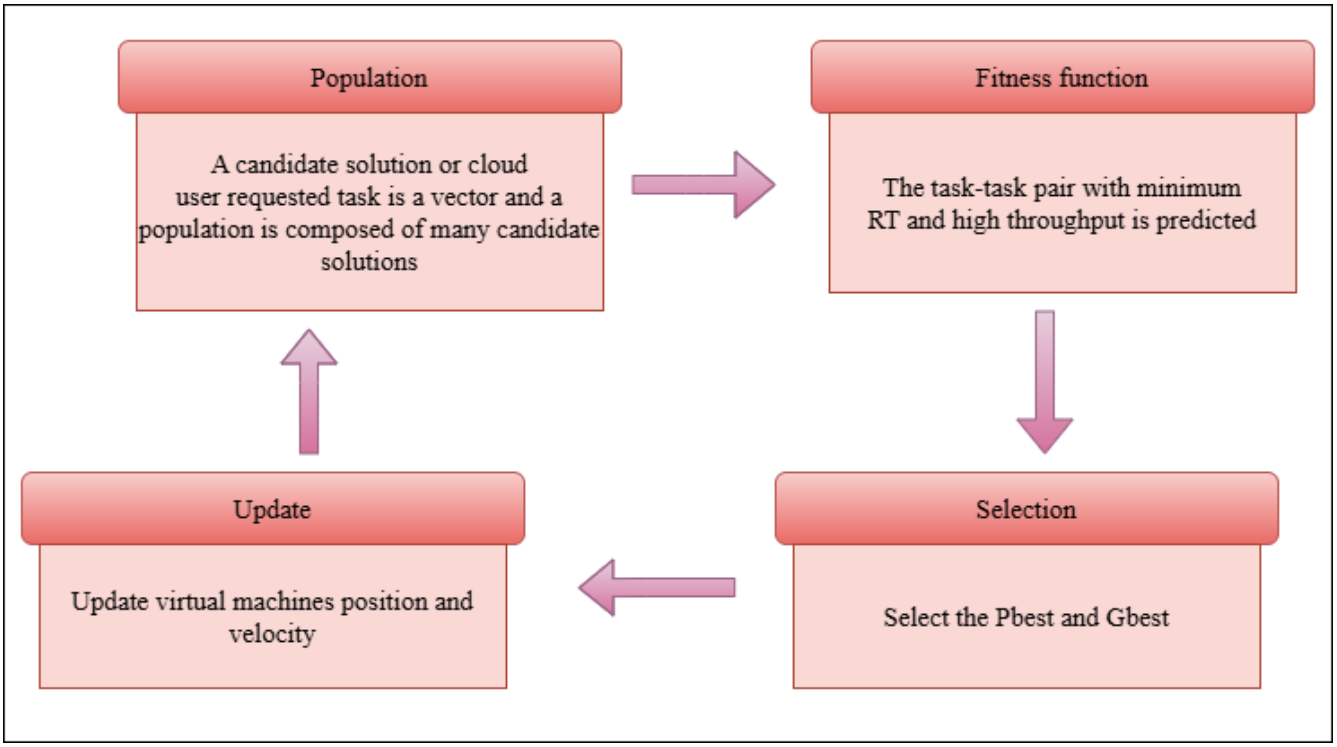

Fig 2. Structure of ensembles of a set of classifiers

From the above **equation (12)** the fitness function 'fun', result is arrived at based on the minimal response time and throughput '$\min(Throughput, RT)$' of the user requested task pair 'T_i, T_j'. Upon successful obtaining of the optimal value based on the fitness function results, each particle or virtual machine uses the equations given below to update its position and velocity based on its own and the swarms' best experiences.

$$Pos_i(t+1) = Pos_i(t) + Vel_i(t+1) \tag{13}$$

$$Vel_i(t+1) = \omega Vel_i(t) + c_1 r_1 \left(Pbest_i(t) - Pos_i(t)\right) + c_2 r_2 \left(Gbest(t) - Pos_i(t)\right) \tag{14}$$

From the above **equations (13) and (14)**, the modified position '[Pos]_i (t+1)' and modified velocity results '[Vel]_i (t+1)' are arrived at based on the current position '[Pos]_i (t)', current velocity '[Vel]_i (t)' 'i-th' virtual machine for iteration 't', personal best 'Pbest', global best 'Gbest' activated by weight 'ω', coefficient 'c_1,c_2' and random number 'r_1,r_2' respectively. Following which the sub-swarms or cloud servers 'CS' are then combined to form a single-evolutionary phase. The global best 'Gbest' is ascertained by making a comparison between the swarm best '[Sw]_best' of each sub-swarm as given below.

$$Gbest = \min\left(Sw_{best1},\ Sw_{best2},\ \ldots, Sw_{bestk}\right) \tag{15}$$

Subsequent to the identification of the global best 'Gbest' as given in the above **equation (15)**, each particle or virtual machine 'VM'is updated according to the personal best 'Pbest', swarm best '[Sw]_best' and global best 'Gbest'. Finally, the equations given below are utilized in updating the velocity and position of all particles or virtual machines 'VMs'.

$$Vel_{ij}(t+1) = \omega Vel_{ij}(t) + c_1 r_1 \left(Pbest_{ij}(t) - Pos_{ij}(t)\right) + c_2 r_2 \left(Pbest_{Swj}(t) - Pos_{ij}(t)\right) + c_3 r_3 \left(Gbest(t) - Pos_{ij}(t)\right) \tag{16}$$

$$Pos_{ij}(t+1) = Pos_{ij}(t) + Vel_{ij}(t) \tag{17}$$

From the above **equations (16) and (17**) results by performing the updates parallel not only speed ups the operation but also minimize the response time to train. The pseudo code representation of Parallel Particle Swarm Optimization-based fine-tuned cloud load balancing is given below.

Algorithm 2. Parallel Particle Swarm Optimization-based fine-tuned cloud load balancing Input: Dataset 'DS', Features '$F = \{F_1, F_2, \ldots, F_m\}$', Samples '$S = \{S_1, S_2, \ldots, S_n\}$', Cloud Server '$CS = \{CS_1, CS_2, \ldots, CS_N\}$'
Output: convergence-efficient fine-tune load balancing
Step 1: **Initialize** cloud user requested tasks '$T = \{T_1, T_2, \ldots, T_m\}$'
Step 2: **Begin**
Step 3: **For** each Dataset 'DS' with Features 'F', Samples 'S', Swarm 'Sw' [data center], Cloud Server 'CS' [sub-swarms]
//Initialize population
Step 4: **For** each cloud user requested tasks 'T'
Step 5: Evaluate fitness function according to (**Eq. 12**)
Step 6: Update position and velocity based on its own and the swarms best experiences according to (**eq. 13**) and (**eq. 14**)
Step 7: Generate global best '$Gbest$' results by making comparison between the swarm best 'Sw_{best}' of each sub-swarm according to (**eq. 15**)
Step 8: Update velocity and position of all particles according to (**Eq. 16**) and (**Eq. 17**)
Step 9: **Return** fine-tuned cloud load balanced results
Step 10: **End for**
Step 11: **End for**
Step 12: **End**

As given in the above algorithm to make certain that fine-tuned cloud load balancing results are generated, the conventional PSO in our work is replaced by parallel PSO via multi-evolutionary and single-evolutionary. Initially population initialization is performed by taking into considerations the cloud user requested tasks along with the swarm or data center, sub-swarms or cloud server. Following which fitness function is formulated based on the quality of each load balancing configuration according to response time and throughput. Finally, position of each particle is fine-tuned iteratively by investigating different load balancing alternatives parallel. Here by taking advantage of the parallel processing in a dynamic fashion across available servers, ensures enhanced throughput with improved convergence speed and accurate load balancing. As given in the above algorithm with the aid of multi-evolutionary aspect optimization problem is solved with minimal response time. Also owing to the parallelization aspects results in enhanced throughput with improved convergence speed, even in the presence of large population size.

3 Result and discussion

In this section, performance measure of the proposed Position Sensitive Transformer Learning and Parallel Particle Swarm Optimization (PSTL-PPSO) for cloud load balancing is designed. Detailed comparison is made with two existing methods, Wavelet decomposition-enhanced External Transformer (WETformer) (1) and Min-Max with Improved Lion Optimization (Min-Max with ILO) (2) are analyzed and validated in CloudSim simulation toolkit using gwa-bitbrains acquired from https://www.kaggle.com/datasets/gauravdhamane/gwa-bitbrains/data. The validation is performed in an Intel Core i5- 6200U CPU @ 2.30GHz 4 cores with 4 Gigabytes of DDR4 RAM. Simulations are analyzed using four performances parameters, makespan, overhead, energy consumption and throughput. Fair comparison is made using similar samples from both the training and testing dataset for all the three methods, PSTL-PPSO, WETformer (1) and Min-Max with ILO (2) respectively for cloud load balancing.

3.1 Performance analysis

Performance analysis was carried out based on four quality measures, i.e., makespan, energy consumption overhead and throughput. These were defined as follows.

Makespan represents the predominant quality measure as it estimates the completion time for all tasks. Makespan is employed in ascertaining the maximum completion time by evaluating the finishing time of the last task. It is measured as

given below.

$$MS(CT_{\max}) = \max \sum_{i=1}^{n} T_{i,1}F_{i,1},\ T_{i,2}F_{i,2},\ \ldots, T_{i,l}F_{i,l} \tag{18}$$

$$F_{i,l} = \begin{cases} 1,\ if\ T_i \rightarrow VM_l \\ 0,\ Otherwise \end{cases} \tag{19}$$

From the above **equations (18) and (19)** makespan 'MS' is measured based on the number of tasks and features involved in assigning corresponding task to the respective virtual machine. It is measured in terms of milliseconds (ms). Energy consumption and cloud load balancing are significantly related to each other. By distributing the workloads across multiple resources via cloud server and load balancer, overload or underutilized load are said to be prevents that in turn results to minimal energy consumption. This is mathematically formulated as given below.

$$EC_{i.l} = Pow_l EC_{T_i^j} \tag{20}$$

$$TEC_{i.l} = \sum_{i=1}^{n} \sum_{j=1}^{m} Pow_l EC_{T_i^j} \tag{21}$$

From the above **equations (20) and (22)** total energy consumption '$TEC_{i.l}$', is measured based on the power 'Pow_l' and the corresponding energy consumption of subsequent task '$EC_{T_i^j}$' respectively. It is measured in terms of joules (Joules).

As far as cloud load balancing is considered, throughput denotes the amount of data transferred within a specific time instance via a load balancer and cloud server. The throughput is considered as the chief performance indicator that in turn shows how effectively the load balancer and cloud server are handling the traffic in an efficient fashion. Finally, overhead refers to the time consumed by the load balancing procedures, specifically the power, memory and network bandwidth consumed. Also, it includes the time it consumes in distributing cloud user requested tasks across multiple cloud servers.

3.2 Results and discussion

In this section results and discussion of the proposed Position Sensitive Transformer Learning and Parallel Particle Swarm Optimization (PSTL-PPSO) for cloud load balancing along with the detailed comparison of two existing methods Wavelet decomposition-enhanced External Transformer (WETformer) (1) and Min-Max with Improved Lion Optimization (Min-Max with ILO) (2) are detailed. To make certain that fair comparisons are made same samples are applied to all the three methods and their values are substituted given in the above performance metrics. Accordingly, results are generated for 10 distinct iterations.

3.2.1. Makespan analysis

First detailing of makepsan analysis is made using the proposed PSTL-PPSO and two existing methods, WETformer (1) and Min-Max with IL (2) for cloud load balancing. Table 1 given below provides the makespan results using the PSTL-PPSO method, WETformer (1) and Min-Max with ILO (2).

Figure 3 illustrates the performance outcomes of makespan versus the number of user requested tasks. The number of user requested tasks, represented on the horizontal axis, ranged from 1000 to 10000, while the Makespan of three methods, namely the PSA-TL method with various methods, WETformer (1) and Min-Max with IL (2), is shown on the vertical axis. The overall observed results indicate that the Makespan of the PSA-TL method is minimum compared to the existing methods (1) and (2). Hence, from the simulation results, with 1000 user requested tasks considered as input, the makespan using the three methods were observed to be 24ms, 28ms and 34ms respectively. For example, let us consider 2000 user requested tasks to compute the makespan to be 28ms, 33ms, 37ms respectively. Similarly, consider 3000 user requested tasks to compute the makespan to be 35ms,40ms,44ms respectively Likewise, different performance outcomes are attained for each method. In the experiments, the same number of user requested tasks were used to compare the performance of the methods. We compared the PSTL-PPSO method separately with the two methods, (1) and (2). The results show an improvement in the performance of each algorithm upon comparison to propose PSTL-PPSO method. Also, by repeating the experiment for 10 simulation runs the average result analysis were also made to detail the effectiveness of each method. On the contrary ,in existing makespan

Table 1. Makespan analysis

Number of user requested tasks	Makespan (ms)		
	PSTL-PPSO	WETformer	Min-Max with ILO
1000	24	28	34
2000	28	33	37
3000	35	40	44
4000	45	50	54
5000	55	60	64
6000	60	65	70
7000	68	73	77
8000	53	58	62
9000	46	51	55
10000	40	45	49

is not minimized. In proposed use feature extraction, encoding and regression for attaining minimum makespan of load balancing. This was due to the incorporation of three different processes separately, namely, feature extraction, encoding and regression via feature extraction, encoding and regression using Position Sensitive Attention-based Transformer Learning (PSA-TL) for load balancing. By applying these three different processes separately, relevant features were extracted employing Position Sensitive Attention function, following which regression segment uses the proposed Piecewise Regression. With this the load balanced results were acquired for further processing that in turn minimized the makespan using PSA-TL method by 12% compared to (1) and 22% compared to (2) respectively.

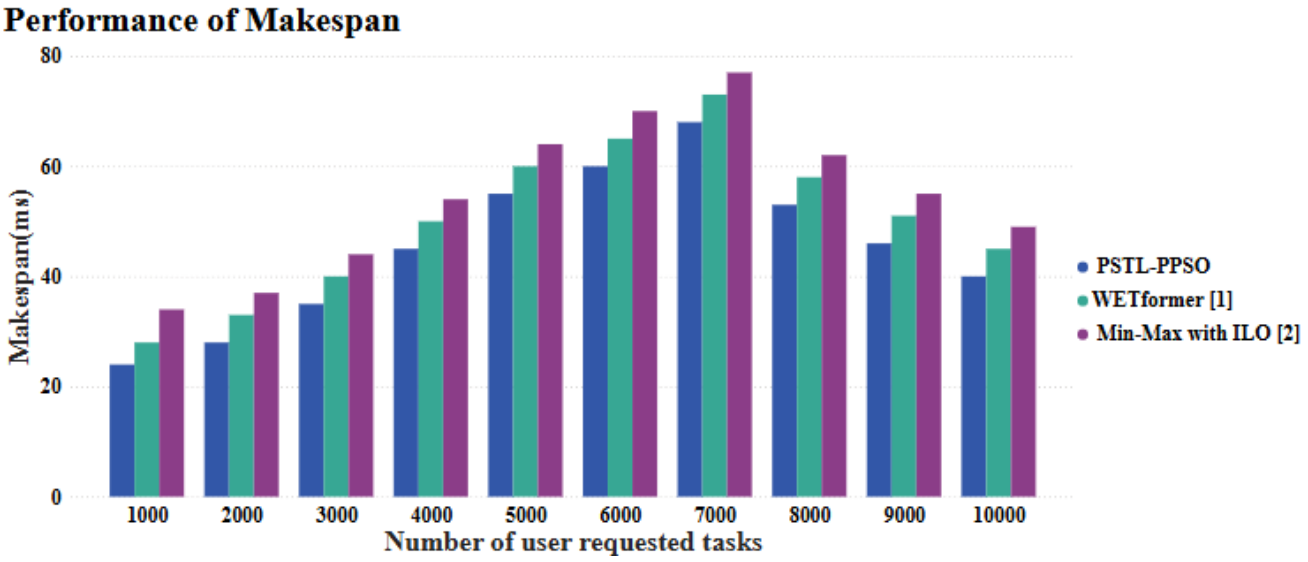

Fig 3. Variation of Makespan

3.2.2. Energy consumption analysis

Second in this section detailed analysis of energy consumption is made employing the proposed PSTL-PPSO and two existing methods, WETformer (1) and Min-Max with IL (2) in cloud computing environment with load balancing data analytics. Table 2

given below lists the tabulation results of energy consumption employing PSTL-PPSO method, WETformer (1) and Min-Max with ILO (2).

Table 2. Energy consumption analysis

Number of user requested tasks	Energy consumption (Joules)		
	PSTL-PPSO	WETformer	Min-Max with ILO
1000	27	33	35
2000	33	41	44
3000	35	43	46
4000	38	46	49
5000	42	50	53
6000	45	52	55
7000	37	45	48
8000	32	40	43
9000	35	43	46
10000	33	41	44

Figure 4 given above illustrates the experimental results of the energy consumption versus the number of user requested tasks. In addition to the efficiency, which refers to the ratio of energy consumption with the number of cloud user requested tasks used, figure depicts better results for proposed PSTL-PPSO method in comparison with two other methods, WETformer (1) and Min-Max with ILO (2). The number of user requested tasks, represented on the horizontal axis, ranged from 1000 to 10000, while the energy consumption of three methods, namely the PSA-TL method with various methods, WETformer (1) and Min-Max with IL (2), is shown on the vertical axis. The overall observed results indicate that the energy consumption of the PSA-TL method is minimum compared to the existing methods (1) and (2). The overall observed results indicate that the energy consumption of the PSA-TL method is minimum compared to the existing methods (1) and (2). The simulation results indicate that the efficiency of our proposed PSTL-PPSOmethod is significantly improved, when increasing the number of cloud user requested tasks apart from the WETformer (1) and Min-Max with ILO (2). This is evident from the simulation of 1000 cloud user requested tasks where the energy consumption using the three methods were observed to be 27Joules, 33Joules and 35Joules respectively. For example, let us consider 2000 user requested tasks to compute the energy consumption to be 33ms, 41ms, 44ms respectively. Similarly, consider 3000 user requested tasks to compute the makespan to be 35ms, 43ms, 46ms respectively Likewise, dissimilar performance outcomes are attained for each method. In contrast, existing energy consumption is not minimized. In proposed use Gaussian Radius Basis functions (GRB) for attaining minimum energy consumption of load balancing. The reason was due to the application of Position Sensitive Attention-based Transformer Learning (PSA-TL) for load balancing algorithm. By applying this algorithm by applying Gaussian Radius Basis functions (GRB) mappings that account for complex relationships between features were identified hence aiding in visualizing complicated data in an extensive manner. Also, positional encoding for each extracted feature was done by using Gaussian Radius Basis function (GRB) that in turn assists in acquiring non-linear relationships during the process of load balancing. This in turn aids in minimizing the energy consumption of PSTL-PPSO method by 22% compared to (1) and 30% compared to (2).

3.2.3. Overhead analysis

Third in this section overhead incurred in cloud load balancing for different cloud user requested tasks are made and accordingly the results are generated for three methods, PSTL-PPSO, WETformer (1) and Min-Max with IL (2). Table 3 given below emphasizes the overhead analysis results using PSTL-PPSO method, WETformer (1) and Min-Max with ILO (2).

Figure 5 shows the comparison of overhead efficiency with respect to 10000 distinct cloud user requests made via the load balancer and cloud sever. The number of user requested tasks, represented on the horizontal axis, ranged from 1000 to 10000, while the overhead of three methods, namely the PSA-TL method with various methods, WETformer (1) and Min-Max with IL (2), is shown on the vertical axis. The overall observed outcomes indicate that the overhead efficiency of the PSA-TL method is minimum compared to the existing methods (1) and (2). The overall observed results indicate that the overhead efficiency of the PSA-TL method is minimum compared to the existing methods (1) and (2). To measure the efficiency of overhead three different methods were execute and accordingly values were substituted to obtain the results. An overall of ten simulation runs

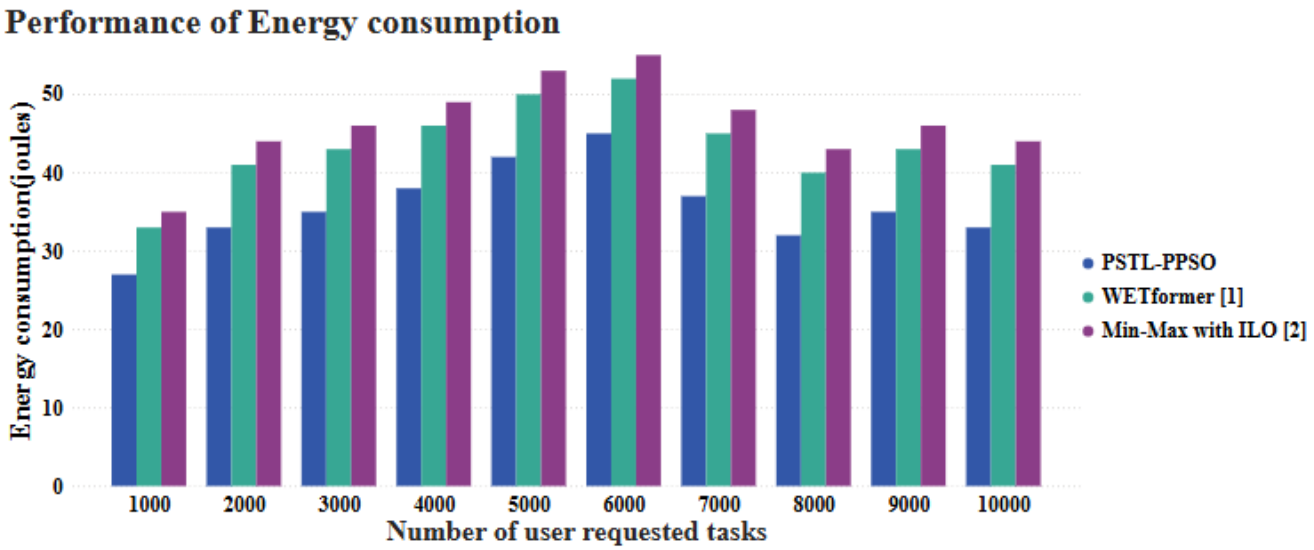

Fig 4. Variation of Energy Consumption

Table 3. Overhead analysis

Number of user requested tasks	Overhead		
	PSTL-PPSO	WETformer	Min-Max with ILO
1000	0.42	0.55	0.6
2000	0.5	0.63	0.67
3000	0.54	0.67	0.72
4000	0.56	0.7	0.75
5000	0.6	0.71	0.76
6000	0.62	0.72	0.78
7000	0.64	0.75	0.81
8000	0.66	0.78	0.83
9000	0.66	0.78	0.83
10000	0.65	0.76	0.8

were performed. It is clear that the average overhead efficiency of the proposed PSTL-PPSO method is 0.585, whereas 0.705 and 0.755 using (1) and (2) respectively. For example, let us consider 1000 user requested tasks to compute the overhead efficiency to be 0.42, 0.55, 0.6 respectively. Similarly, consider 2000 user requested tasks to compute the overhead efficiency to be 0.5, 0.63, 0.67 respectively. Dissimilar performance outcomes are attained for each method. This clearly shows the high efficiency or minimal overhead incurred during the fine-tuning of load balancing between cloud user requested tasks. In contrast , in existing overhead efficiency is not minimized. In proposed we use Particle Swarm Optimization for attaining minimum overhead efficiency of load balancing This is due to the involvement of Parallel Particle Swarm Optimization-based fine-tuned cloud load balancing algorithm. By applying this algorithm fitness function was formulated based on the minimal response time and high throughput and activated via investigating different load balancing alternatives parallel. This in turn minimized the overhead using PSTL-PPSO method by 21% compared to (1) and 30% compared to (2).

3.2.4. Throughput analysis

Finally, a holistic analysis of throughput is made using the three methods, PSTL-PPSO, WETformer (1) and Min-Max with IL (2) respectively. Finally, throughput results are tabulated in table 4 as given below.

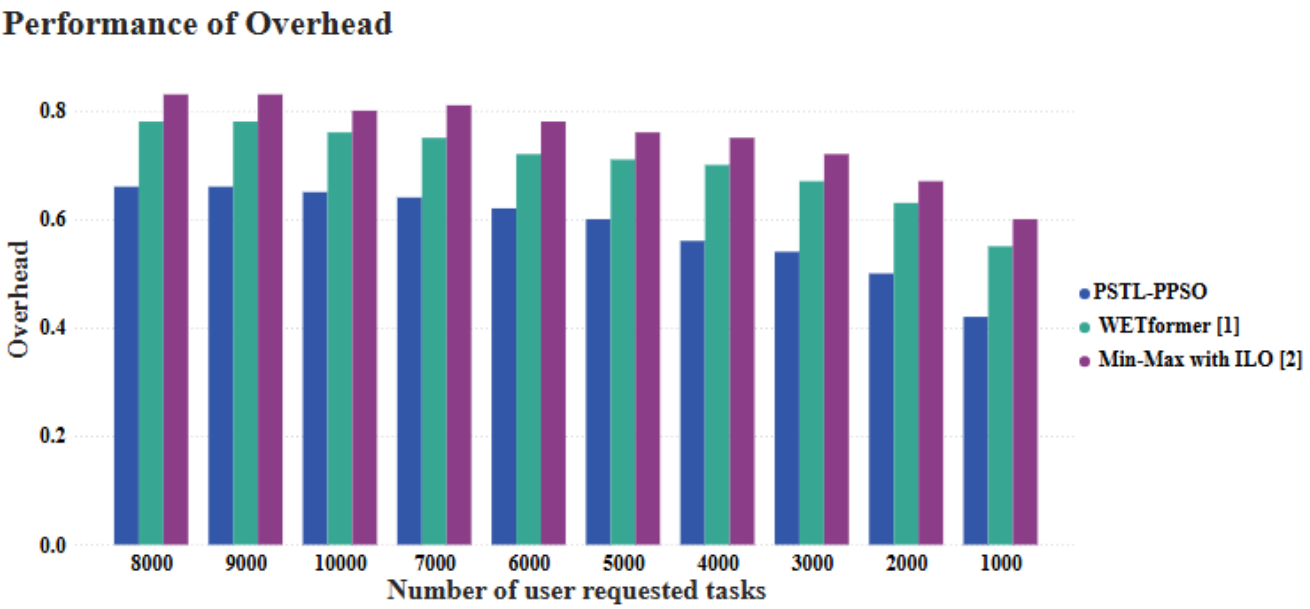

Fig 5. Variation of Overhead Analysis

Table 4. Throughput analysis

Number of user requested tasks	Throughput (bits/s)		
	PSTL-PPSO	WETformer	Min-Max with ILO
1000	715	530	490
2000	885	590	530
3000	1005	640	566
4000	1255	845	705
5000	2200	2105	2000
6000	2250	2125	2055
7000	2300	2265	2245
8000	2500	2345	2305
9000	2700	2425	2365
10000	2825	2525	2505

Finally figure 6 give above illustrates the throughput analysis using three methods, PSTL-PPSO, WETformer (1) and Min-Max with ILO (2) performed with respect to cloud user requested tasks. The number of user requested tasks, represented on the horizontal axis, ranged from 1000 to 10000, while the throughput of three methods, namely the PSA-TL method with various methods, WETformer (1) and Min-Max with IL (2), is shown on the vertical axis. The entire observed outcomes indicate that the throughput of the PSA-TL method is increased compared to the existing methods (1) and (2). The overall observed results indicate that the throughput of the PSA-TL method is improved compared to the existing (1) and (2). An increase in throughput was observed for increasing numbers of cloud user requested tasks using all the three methods. However, simulation results showed improvement using PSTL-PPSO upon comparison to (1) and (2). For example, let us consider 1000 user requested tasks to compute the throughput to be 715, 530, 490 respectively. Similarly, consider 2000 user requested tasks to compute the throughput to be 885, 590, 530 respectively. Let us assume 3000 user requested tasks to compute the throughput to be 1005, 640, 566 respectively Likewise, dissimilar performance outcomes are attained for each method. In contrast , in existing throughput is not increased. In proposed consistent throughput is achieved for load balancing. Particle Swarm Optimization is used in

proposed work. The reason would be contributed to the application of Parallel Particle Swarm Optimization-based fine-tuned cloud load balancing algorithm. By applying this algorithm population initialization was first done by considering the cloud user requested tasks in addition to swarm or data center, sub-swarms or cloud server. Second fitness function was evaluated for each cloud user requested task by taking into consideration the response time and throughput. Finally, fine-tuning was performed by exploring distinct load balancing alternatives parallel. By taking advantage of parallel processing dynamically across available servers in turn improved throughput of the PSTL-PPSO method by 17% compared to (1) and 21% compared to (2).

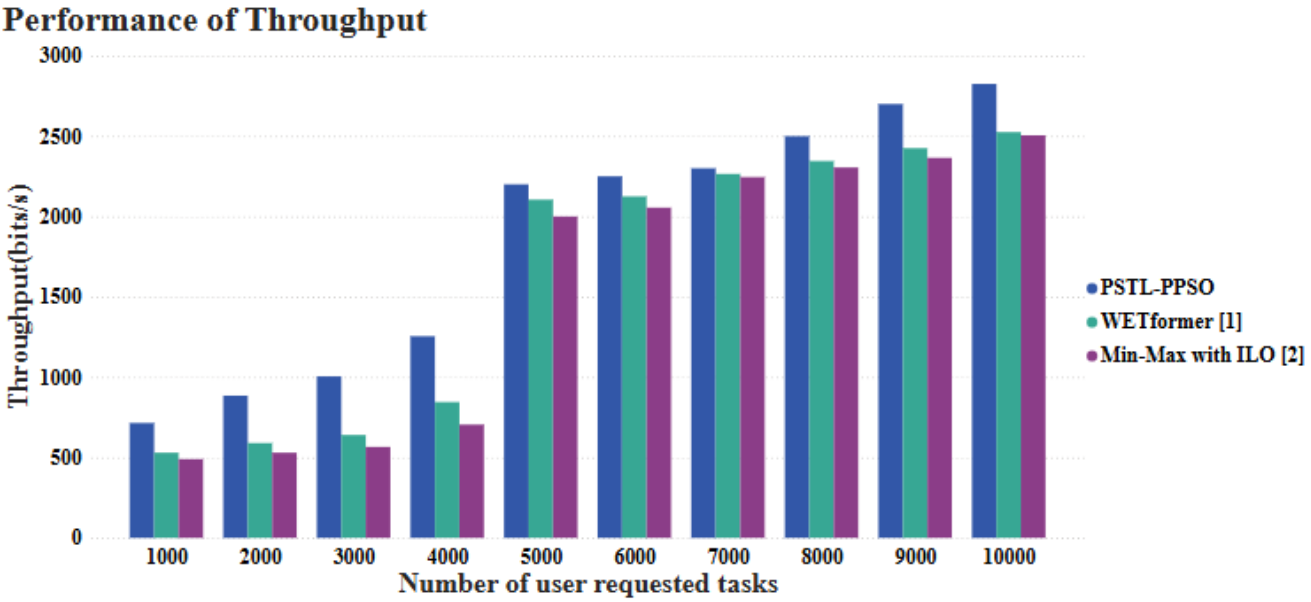

Fig 6. Variation of Throughput Analysis

4 Conclusion

Load balancing is a crucial issue in cloud computing as more number of heterogeneous cloud user requested tasks with interdependencies or workflows comes to cloud application console and it is laborious and cumbersome to balance workload appropriately between users. Ineffective load balancing led to the increase in resource utilization, minimizing scalability and reducing availability making it harder to scale up or down with the changes in demand. To tackle this situation, in this paper, we have modeled a Position Sensitive Transformer Learning and Parallel Particle Swarm Optimization (PSTL-PPSO) method which captures number of cloud user requested tasks, to measure quality of each load balancing configuration and these are fed to PPSO algorithm which is modeled with an optimization technique. The novelty of proposed PSTL-PPSO method is to improve the efficiency of load balancing in cloud with minimum response time. This load balancer needs to generate decisions based on data from different cloud servers, to name a few being CPU usage, memory usage while minimizing parameters i.e. makespan, energy consumption and improving throughput, overhead. Our proposed PSTL-PPSO method is simulated on CloudSim simulator. From results it is evident that the PSTL-PPSO method outperforms existing methods by minimizing makespan, energy consumption and also maximizing throughput considerably. PSTL-PPSO method improves the throughput by 19 % and reduces makespan 17%, energy consumption by 26%, overhead efficiency by 26%.More dataset parameters like average resource utilization, space complexity used in future.

5 Acknowledgement

The authors thank DST-FIST, Government of India, for funding towards infrastructure facilities at St.Joseph's College (Autonomous), Tiruchirappalli - 620 002.

References

1) Zhen Z, Chen X, Jinyu Z, Zhe Z, Shaohua X. When wavelet decomposition meets external attention: a lightweight cloud server load prediction model. *Journal of Cloud Computing: Advances, Systems and Applications*. 2024 Aug. https://doi.org/10.1186/s13677-024-00698-6.
2) Adaikalaraj JR, Chandrasekar C. To improve the performance on disk load balancing in a cloud environment using improved Lion optimization with min-max algorithm. *Measurement: Sensors*. 2023 Jun;27. https://doi.org/10.1016/j.measen.2023.100834.
3) Jiaxin G, Yuntian C, Wenbo H, Dongxiao Z. An adaptive deep-learning load forecasting framework by integrating transformer and domain knowledge. *Advances in Applied Energy*. 2023 Jun;10. https://doi.org/10.1016/j.adapen.2023.100142.
4) Nilayam KK, Jaroslav F, Subhendu KP, Rashmi D, Sardar MNI, Bhartia PK, et al. Machine learning model design for high performance cloud computing & load balancing resiliency: An innovative approach. *Journal of King Saud University Computer and Information Sciences*. 2022 Nov;34. https://doi.org/10.1016/j.jksuci.2022.10.001.
5) Ika NS, Harry H, Wijaya K, Anang DN, Arwin DWS, Chandra W. Advanced load balancing techniques using MIMO fuzzy logic: A panel distribution case study at state polytechnic of Malang. *MethodsX*. 2025 Jun;14. https://doi.org/10.1016/j.mex.2025.103197.
6) Kunal J, Anjaly P, Ankur M, Esha C, Xiaoting Q, Jue Z, et al. Performance Aware LLM Load Balancer for Mixed Workloads. 2025. https://doi.org/10.1145/3721146.3721947.
7) Kavitha HS, Mallikarjunaswamy S, Sharmila N. An efficient load-balancing in machine learning-based DC-DC conversion using renewable energy resources. *International Journal of Artificial Intelligence*. 2025 Feb;14:307–316. http://doi.org/10.11591/ijai.v14.i1.pp307-316.
8) Karim SDME, Mirza M, Abdullahg A. BHyPreC: A Novel Bi-LSTM Based Hybrid Recurrent Neural Network Model to Predict the CPU Workload of Cloud Virtual Machine. *IEEE Access*. 2021 Sep;9. https://doi.org/10.1109/ACCESS.2021.3113714.
9) Dabeeruddin SAG, Haitham A, Shady SR, Mahdi H, Othmane B, Santiago B. Deep Learning-Based Short-Term Load Forecasting Approach in Smart Grid With Clustering and Consumption Pattern Recognition. *IEEE Access*. 2021 Apr;9. https://doi.org/10.1109/ACCESS.2021.3071654.
10) Ioannis R, Vlassis F, Konstantinos M, Adrian M. ExpPoint-MAE: Better Interpretability and Performance for Self-Supervised Point Cloud Transformers. *IEEE Access*. 2024 Apr;12. https://doi.org/10.1109/ACCESS.2024.3388155.
11) Juliet GM, Peter MM. Machine Learning Load Balancing Techniques in Cloud Computing: A Review. *International Journal of Computer Applications Technology and Research*. 2022 Apr;11. https://doi.org/10.7753/IJCATR1106.1002.
12) Nanasaheb BK, Ghorpade GB, Bhandwalkar KT. Optimized Model for Cloud Load Prediction and Resource Allocation Using Deep Learning. *PERIODICO di MINERALOGIA*. 2024 Nov;23. https://doi.org/10.37896/pd93.4/9342.
13) Naganandhini S, Shanthi D. Temporal fusion transformer-based strategy for efficient multi-cloud content replication. *Peer Journal of Computer Science*. 2025 Mar. https://doi.org/10.7717/peerj-cs.2713.
14) Esma Y, Arafat A. Predicting Short-Term Variations in End-to-End Cloud Data Transfer Throughput Using Neural Networks. *IEEE Access*. 2023 Aug;11. https://doi.org/10.1109/ACCESS.2023.3299311.
15) Qi L, Hayata K, Lin M. A cosine similarity-based token subsampling method for vision transformer in cloud computing. *Neural Computing and Applications*. 2024 Dec. https://doi.org/10.1007/s00521-024-10718-w.
16) Guangyao Z, Wenhong T, Rajkumar B, Ruini X, Liang S. Deep reinforcement learning based methods for resource scheduling in cloud computing: a review and future directions. *Artificial Intelligence Review*. 2024 Apr. https://doi.org/10.1007/s10462-024-10756-9.
17) Seyed SS, Ahmed MN, Bahman A, Razvan C, Ciprian RC. A Probabilistic Approach to Load Balancing in Multi-Cloud Environments via Machine Learning and Optimization Algorithms. *Journal of Grid Computing*. 2025 Mar. https://doi.org/10.1007/s10723-025-09805-6.
18) Shahbaz A, Kavitha G. Load balancing in cloud computing – A hierarchical taxonomical classification. *Journal of Cloud Computing: Advances, Systems and Applications*. 2019 Dec. https://doi.org/10.1186/s13677-019-0146-7.
19) Daokun Q, Xiaojuan X, Yake T, Yuesong Z, Zhengwei G. Real-time scheduling of power grid digital twin tasks in cloud via deep reinforcement learning. *Journal of Cloud Computing: Advances, Systems and Applications*. 2024 Oct. https://doi.org/10.1186/s13677-024-00683-z.
20) Rania AI, Omneya A, Nahla EZ. A lightweight deep learning framework for transformer fault diagnosis in smart grids using multiple scale CNN features. *Scientific Reports*. 2025 May. https://doi.org/10.1038/s41598-025-96290-2.
21) Sasibhushana RP, Kalyana CC. SDN Controller Selection and Secure Resource Allocation. *IEEE Access*. 2025 May;13. https://doi.org/10.1109/ACCESS.2025.3565117.
22) Asiri MM, Ghadah A, Faiz AA, Alnfiai MM, Mohammed A, Ahmed M. Short-Term Load Forecasting in Smart Grids Using Hybrid Deep Learning. *IEEE Access*. 2024 Feb;12. https://doi.org/10.1109/ACCESS.2024.3358182.
23) Sarthak S, Bhavisha, Rupinder S, Jaskaran S. Analyzing Load Balancing Techniques for Cloud Computing: Pros, Cons, and Emerging Trends. Elsevier. 2023. https://dx.doi.org/10.2139/ssrn.4625863.
24) Sujayjumar MR, Hemanth K, Mohana SL. Transformer-Based Federated Learning Models for Recommendation Systems. *IEEE Access*. 2024 Aug;12. https://doi.org/10.1109/ACCESS.2024.3439668.
25) Wutao X. A Time-Series Model of Gated Recurrent Units Based on Attention Mechanism for Short-Term Load Forecasting. *IEEE Access*. 2024 Aug;12. https://doi.org/10.1109/ACCESS.2024.3443876.
26) Lung HH, Chih HW, Chiung HT, Hsiang CH. Migration-Based Load Balance of Virtual Machine Servers in Cloud Computing by Load Prediction Using Genetic-Based Methods. *IEEE Access*. 2021 Apr;9. https://doi.org/10.1109/ACCESS.2021.3065170.
27) Song Q, Tianping Z, Siping H. Research on intelligent routing with VAE-GAN in the internet of body. *PLOS ONE*. 2025 Feb. https://doi.org/10.1371/journal.pone.0317698.
28) Peng Z, Jialiang Z, Yi L. Research on memory failure prediction based on ensemble learning. *PLOS ONE*. 2025 Apr. https://doi.org/10.1371/journal.pone.0321954.
29) Lei WDL, Ming H, Zhenqing Y, Rui W, Dashi Q, Xingxing X, et al. LBNP: Learning features between neighboring points for point cloud classification. *PLOS ONE*. 2025 Jan. https://doi.org/10.1371/journal.pone.0314086.