An Intelligent Task Scheduling Model for Hybrid Internet of Things and Cloud Environment for Big Data Applications

Souvik Pal 1,2, N. Z. Jhanjhi 3,*, Azmi Shawkat Abdulbaqi 4, D. Akila 5, Faisal S. Alsubaei 6 and Abdulaleem Ali Almazroi 7

1 Department of Computer Science and Engineering, Sister Nivedita University, New Town 700156, India
2 Post-Doctoral Researcher, Sambalpur University, Sambalpur 768019, India
3 School of Computer Science, SCS Taylors University, Subang Jaya 47500, Malaysia
4 Department of Computer Science, College of Computer Science and Information Technology, University of Anbar, Baghdad 55431, Iraq
5 Department of Computer Applications, Saveetha College of Liberal Arts and Sciences, SIMATS Deemed University, Chennai 602105, India
6 Department of Cybersecurity, College of Computer Science and Engineering, University of Jeddah, Jeddah 23218, Saudi Arabia
7 Department of Information Technology, Faculty of Computing and Information Technology in Rabigh, King Abdulaziz University, Rabigh 21911, Saudi Arabia

* Correspondence: noorzaman.jhanjhi@taylors.edu.my

Abstract: One of the most significant issues in Internet of Things (IoT) cloud computing is scheduling tasks. Recent developments in IoT-based technologies have led to a meteoric rise in the demand for cloud storage. In order to load the IoT services onto cloud resources efficiently even while satisfying the requirements of the applications, sophisticated planning methodologies are required. This is important because several processes must be well prepared on different virtual machines to maximize resource usage and minimize waiting times. Different IoT application tasks can be difficult to schedule in a cloud-based computing architecture due to the heterogeneous features of IoT. With the rise in IoT sensors and the need to access information quickly and reliably, fog cloud computing is proposed for the integration of fog and cloud networks to meet these demands. One of the most important necessities in a fog cloud setting is efficient task scheduling, as this can help to lessen the time it takes for data to be processed and improve QoS (quality of service). The overall processing time of IoT programs should be kept as short as possible by effectively planning and managing their workloads, taking into account limitations such as task scheduling. Finding the ideal approach is challenging, especially for big data systems, because task scheduling is a complex issue. This research provides a Deep Learning Algorithm for Big data Task Scheduling System (DLA-BDTSS) for the Internet of Things (IoT) and cloud computing applications. When it comes to reducing energy costs and end-to-end delay, an optimized scheduling model based on deep learning is used to analyze and process various tasks. The method employs a multi-objective strategy to shorten the makespan and maximize resource consumption. A regional exploration search technique improves the optimization algorithm’s capacity to exploit data and avoid becoming stuck in local optimization. DLA-BDTSS was compared to other well-known task allocation methods in accurate trace information and the CloudSim tools. The investigation showed that DLA-BDTSS performed better than other well-known algorithms. It converged faster than different approaches, making it beneficial for big data task scheduling scenarios, and it obtained an 8.43 percent improvement in the outcomes. DLA-BDTSS obtained an 8.43% improvement in the outcomes with an execution time of 34 s and fitness value evaluation of 76.8%.

Keywords: big data; cloud computing; Internet of Things; scheduling
1. Importance of Task Scheduling for Hybrid Internet of Things and Cloud Environment

To improve production processes and produce higher quality products at lower prices, the manufacturing process incorporates ample information, analytic capabilities, high-performance computation, and the commercial Internet of Things (IoT) into production systems and sectors [1]. Nowadays, factories, factories, gadgets, machines, and procedures are linked, monitored, and optimized to increase production and efficiency. Today, the use of these systems in manufacturing is a critical facilitator for tackling issues with scheduling, order fulfillment, transportation, and supply management. When sensors are used to monitor manufacturing lines in real-time, they save the performance data and then analyze the data (founded on the observations) to generate the planned or anticipated production schedule.

For the increasing computational demands of big data cloud computing (CC) applications, a smarter approach is required. The big data and cloud computing applications are widely used in digital society. The combined form of application is utilized in the business sector. By providing scalable and malleable computing devices/resources that can be utilized on a pay-as-you-go basis, cloud computing presents a substantial increase in the number of Internet of Things (IoT) applications. Many Big-Data-related tasks for these IoT apps are performed in the cloud as a service to the cloud, and their results are then shared amongst many devices. Big-Data-related tasks manage all types of datasets and analyze and evaluate the complete structure of data. Big data are implemented in IoT application by the transfer of data among wearable sensors, smart devices. However, another obvious difficulty is estimating how many devices will be needed to carry out a cloud-based set of extremely large-scale operations.

The public cloud is gaining popularity as a new form of distributed computation in the digital age. By 2019, the cloud provider will have processed over 70% of the calculations. Cloud computing offers consumers effective, secure computation and communication services, but it has driven significant economic growth in various industries [2–5]. The interconnection of particles is becoming more extensive and healthier, the devices connected to networks are exploding, and many devices at the edge require a low delay and high serviceability due to the fast expansion of connected phones, smart systems, and other industrial IoT [5].

Although cloud technology has a large-scale parallelization design, some problems, such as low resource utilization and an imbalanced load, still exist and significantly impact task efficiency. The rise of the cloud environment is crucial at this time because large-scale data transport uses up a lot of bandwidth and adds complexity to datacenters; because of the various terminal requirements, the necessities for assets are also diverse, and the long-distance communication system produces a significant delay, making it impossible to satisfy time-sensitive customer requirements.

The most fundamental issue with cloud computing task scheduling is matching requirements with available resources. In IoT environments, the user requirements vary, and the performance of the cloud computing integrated services node is noticeably worse than that of the combined services node. Therefore, it is essential to address the problem of effectively allocating the resources in cloud computing and allocating the resources based on user needs.

This research looked at intelligent reconfiguring and planning for a smart industrial system with advanced technologies. The goal is to reduce the entire cost of all tasks related to tardiness. The research enables real-time optimization and intelligent rearrangement judgment. Q-learning, the most prominent algorithm in reinforcement learning, calculates reinforcement for states and actions. Q-results learning is affected by both the initial state and the subsequent actions taken. Where there is a finite set of states and actions for a reinforcement learning problem, Q-learning is employed.

Schedules can be made in a variety of ways, and the genetic algorithm is among them. By calculating fitness costs, this method can evaluate different schedules and choose the one that provides the best overall outcome. Then, people can find more effective routines by employing crossover, mutation, and elitism selections.
The main contributions are listed below:

- An optimal scheduling model founded on deep learning is utilized to assess and process different tasks to reduce energy costs and end-to-end delay.
- A regional exploratory search strategy enhances the optimization algorithm’s ability to use data and avoid becoming stuck in local optimization.
- Accurate trace data and the CloudSim tools were used to compare DLA-BDTSS to other popular task allocation methods. DLA-BDTSS’s rapid convergence made it an excellent choice for scheduling tasks involving big data. The task is scheduled and processed by means of Q-learning, which can be developed as the effective scheduling agent for all potential users to complete every task that needs to be fulfilled. From the collected task, the GA encodes the task and processes the task to obtain a better result. With a 34s execution time and a fitness value evaluation of 76.8 percent, DLA-BDTSS achieved an 8.43% improvement in results.

In this study, we propose a deep learning algorithm for scheduling big data tasks in the cloud and the Internet of Things. The results of the study showed that DLA-BDTSS was superior to the most common competing methods. It is helpful for scheduling big data tasks because it achieves higher percentage improvement in the outcomes and converges more quickly than competing methods.

The later part of the research is listed below: In Section 2, we will cover the history and results of the task scheduling models. In Section 3, we develop, test, and deploy the Deep Learning Algorithm for Big Data Task Scheduling System (DLA-BDTSS) that we propose. In Section 4, we show the results of our analysis of the proposed system’s software. Section 5 collects the report’s final thoughts and findings from the system.

2. Background Study

There are two types of standard task scheduling techniques: exact scheduling techniques and approximate scheduling techniques. By searching for the complete optimal solution, the optimal solution on a global scale is found by the same methods. The high computational complexity makes precise practices ineffective for dealing with large-scale planning issues. Instead of searching every possible solution, approximation scheduling methods explore a large number of possible paths based on a small number of predetermined strategies. Therefore, approximate procedures are preferable in dealing with complex scheduling issues because they generate practical solutions more quickly and with less computing complexity. Estimation techniques, however, cannot promise the best answer for a specific scheduling problem.

Some academics have suggested reinforcement learning techniques for work planning in a service-oriented setting. A reinforcement learning technique was presented [6] to address the scheduling issue in distributed systems while considering the heterogeneity and placement of units within the grids to achieve a faster completion time. A multi-agent paradigm for optimization utilizing metaheuristics is suggested. All individuals share data and cooperate to allow the agent to change their actions depending on experiences obtained through engaging with other agencies and their surroundings [7]. The number of process apps and virtualization software were employed as state variables [8] in multi-agent deep-Q networks, along with the maximum completion time and cost, to help schedule multiple workflows in the public cloud.

Wang et al. [9] expanded the deep-Q system to handle the task scheduling choices of numerous network edges to monitor the development of multi-workflows over infrastructure-as-a-service clouds. The issue of scheduling algorithms in cloud technology has drawn attention for some time. However, fog computing has lately been proposed, and studies on different cloud computing features have been published. Still, only a few tasks are scheduled. Yang et al. [10] present Real-Time Dynamic Max min (RTDM) dynamic task scheduling, which accounts for makespan, load balancing, and overall waiting time. The main goal of the Expense-Makespan Aware Scheduling Heuristic Model proposed by [11]
is to distinguish between the benefits of employing cloud services and infrastructure and the efficiency of application code.

Fellir et al. [12] presented a system that uses a decision tree to handle tasks while taking into account changing environments. Every sensor’s decision tree algorithm categorizes seasonal chores and determines which ones should be used in which order. The investigation of enhanced heuristic algorithm-based hierarchical capacity scheduling for IoT technology is thorough. Many scientists have previously investigated ways to strengthen task scheduling weaknesses through the use of intelligent algorithms for example genetic algorithms (GA) and particle swarm optimizations (PSO) [13,14], which have produced excellent results on a variety of levels (such as makespan and planning time).

A new planning technique was also introduced in [15], founded on the Q-learning process, which is among the most effective reinforcement learning techniques. An adaptive Cauchy mutation-based cuckoo search approach is utilized in [16]. Designing an optimization strategy is based on the life of a cuckoo. Therefore, this guideline for optimization strategies helps to find the best way to carry out the task with reliable and beneficial results. To create a system-performance-sensitive planner [17], considering the framework of the IoT, one examines and analyzes K-means, principal component analysis, and fuzzy C-means clustering (FCM) methods. Sefati et al. [18] proposed a dynamic scheduling technique for cloud computing that prioritized requests from the IoT while considering homogenous and heterogeneous servers.

Hasan et al. [19] propose a scheduling algorithm strategy for time-sensitive data streaming systems on homogeneous clouds that reschedules such tasks and provides enough resources to reduce their waiting time. Pal S. [20] created the IoT-SCOM model and optimise it to select the optimal deployment option with the shortest guaranteed delay. The experimental results show that the IoT-SCOM approach outperforms established methods and the stochastic optimization algorithm in terms of reliability and performance for the difficulty of data-intensive delivery component installation in the edge-cloud environment.

After carefully examining the studies mentioned above, it was discovered that they have advantages and disadvantages in several areas: (1) Numerous studies, such as [21–23], contributed to the study of optimization techniques and provided adequate exposure to fog computing theory and platforms. (2) Previous research, such as [24–26], considered optimizing the duration and power usage caused by cloud computing. Nonetheless, they failed to consider the issue of device loading and latency in terms of activities and resource types.

Pal. S. et al. [27] discussed a system design for the optimal sequence of the Johnson Scheduling algorithm. This sequence represents service times. A multi-server queueing model with finite capacity diminishes waiting time and channel capacity, improving job scheduling.

Mukherjee, D et al. [28] created the Adaptive Scheduling Algorithm Based Task Loading (ASA-TL) algorithm, which is used to store tasks from cloud data centers on digital devices. According to the results of the experiment, ASA-TL appears to have the best scores for response time, data center processing time, and overall cost.

M. Goudarzi et al. [29] organized recent research on IoT system scheduling in fog computing. The current literature is examined, research gaps are identified, and future directions are described in light of their new categorizations. Saha, S. et al. [30] describe a Task Scheduling algorithm based on a genetic algorithm that employs a queueing model to reduce the amount of time people must wait and the length of the queue.

Sing, R. et al. [31] presented a multi-objective optimization technique that concurrently minimizes execution time, cost, and power usage by enhancing their respective parameters for optimal performance using evolutionary techniques. Better performance can be obtained by balancing cloud and fog nodes, as demonstrated by simulation results using various cloud and fog nodes.

In order to evaluate the efficacy of the proposed framework, Shresthamali, S. et al. [32] focused on designs to simulate both single-task and dual-task systems. The results show
that their Multi-Objective Reinforcement Learning (MORL) algorithms can learn superior policies while incurring lower learning costs and effectively balancing competing goals during execution. Pal. S [20] and Mukherjee D. [33] have also discussed optimization techniques for better results in cloud-IoT-edge environments. Table 1 deals with the contributions and limitations of this background work in tabular format.

Table 1. Summary of previous contributions and limitations.

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Paper Details</th>
<th>Contributions</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Wang. Y. et al. [9]</td>
<td>This paper optimizes makespan and cost by training process models and heterogeneous VMs with different resource configurations. Their scheduling algorithm does not require human input or expert knowledge.</td>
<td>It lacks QoS metrics such as reliability, security, and loadbalancing. Real-time data collection is too costly and time-consuming.</td>
</tr>
<tr>
<td>2</td>
<td>Yang, S et al. [10]</td>
<td>In this paper, a reconfigurable flow line (RFL) with deep reinforcement learning (DRL) is used for intelligent dynamic scheduling and reconfiguration decision-making. In this paper, a mathematical model is used to minimize total tardiness cost.</td>
<td>The work’s limitation is its inability to automatically create system features and actions through learning or to design states and actions that are more efficient.</td>
</tr>
<tr>
<td>3</td>
<td>Ding, D et al. [11]</td>
<td>The authors of this paper developed the QEEC framework, which addresses power usage, task scheduling, and task assignment in cloud data centers. Their experimental findings show that the M/M/S queuing model performs better.</td>
<td>Their paper lacks when using large-scale cloud environments with hundreds of servicing nodes. This paper may address other heuristics that may improve Q-learning for cloud task scheduling.</td>
</tr>
<tr>
<td>4</td>
<td>Fellir, F et al. [12]</td>
<td>A multi-agent-based model that leads to better resource utilization and improves performance has been proposed in this paper. Considerations include the task’s priority, wait time, status, and the resources needed.</td>
<td>A small number of IoT gadgets may need specialized memory. There needs to be a wider range of resource scheduling algorithms compared to the proposed work.</td>
</tr>
<tr>
<td>5</td>
<td>Gao, Q et al. [13]</td>
<td>In this paper, various internal, external, and management risks are addressed. The work of the SCRM contributed to the development and improvement of supply chain information.</td>
<td>The work does not address how to reduce time consumption based on algorithm performance.</td>
</tr>
<tr>
<td>6</td>
<td>Ben Alla et al. [14]</td>
<td>TSDQ-FLPSO and TSDQ-SAPSO algorithms are proposed in this paper for high-dimensional problems, and they show advantages in terms of queue length, makespan, cost, resource consumption, degree of imbalance, and bandwidth allocation.</td>
<td>The algorithm’s robustness needs to be increased. The QoS parameters, such as task priorities and the ability for tasks to move between queues, are not covered.</td>
</tr>
<tr>
<td>7</td>
<td>Hosseinzadeh, M et al. [15]</td>
<td>The metaheuristic multi-objective optimization is the main topic of this paper. Future research directions are presented along with a comparison of various methodologies and algorithms.</td>
<td>The suggested multi-objective scheduling plans ought to pay more attention to fresh settings and improve the workflow, task scheduling procedure, etc.</td>
</tr>
<tr>
<td>8</td>
<td>Mostafavi, S et al. [16]</td>
<td>A method of online reinforcement learning is suggested for allocating tasks in the future. The results demonstrate that it improves resource efficiency while also reducing response times and turnaround times for tasks submitted.</td>
<td>A number of existing works must be compared to the proposed system.</td>
</tr>
</tbody>
</table>
Table 1. Cont.

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Paper Details</th>
<th>Contributions</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Pal. S. et al. [27]</td>
<td>This paper includes a system design for the Johnson Scheduling algorithm’s optimal sequence. This sequence gives service times. Queuing model with multi-server and finite capacity reduces waiting time and queue length, improving job scheduling.</td>
<td>Their approach lacks performance analysis based on cost per service and wait time due to increased server count.</td>
</tr>
<tr>
<td>10</td>
<td>Mukherjee, D et al. [28]</td>
<td>This paper is about an algorithm called Adaptive Scheduling Algorithm Based Task Loading (ASA-TL), which is used to store tasks from cloud data centres on digital devices. Based on the results of the experiment, it seems that ASA-TL gets the best scores for response time, processing time in the data centre, and overall cost.</td>
<td>Their plan is not complete because it does not use virtual software resources on a cloud server. There may be a need for more parameters to find the best performance.</td>
</tr>
<tr>
<td>11</td>
<td>Goudarzi, M. et al. [29]</td>
<td>This paper organizes recent research on scheduling IoT systems in fog computing. Current literature is analyzed, research gaps are identified, and directions for future are described based on their new categorizations.</td>
<td>Micro-services-based applications, systems for ML, and privacy-aware and adaptive decision-making engines are all missing from this paper.</td>
</tr>
<tr>
<td>12</td>
<td>Saha, S. et al. [30]</td>
<td>This paper describes a Task Scheduling algorithm that is based on a genetic algorithm and uses a queuing model to reduce the amount of time people have to wait and the length of the queue.</td>
<td>This algorithm does not have batch processing, which could help make good scheduling choices.</td>
</tr>
<tr>
<td>13</td>
<td>Sing, R. et al. [31]</td>
<td>In this paper, a multi-objective optimization algorithm is presented that simultaneously reduces execution time, cost, and energy consumption by optimizing their respective parameters via evolutionary techniques for optimal performance. A better performance can be achieved by striking a balance between cloud and fog nodes, as shown by simulation results using different cloud and fog nodes.</td>
<td>In order to resolve scheduling issues, the authors may decide to implement additional meta-heuristic algorithms. Additional areas of investigation could include the weight of urgency, the impact of time limits, and the use of virtual machines in scheduling.</td>
</tr>
<tr>
<td>14</td>
<td>Shresthamali, S. et al. [32]</td>
<td>This paper focuses on simulating both single-task and dual-task systems in order to assess the effectiveness of the proposed framework. The outcomes prove that their Multi-Objective Reinforcement Learning (MORL) algorithms are able to learn superior policies with reduced learning costs and effectively balance competing goals during execution.</td>
<td>The MORL solution proposed by the authors still needs refinement before it can be used in extremely resource-constrained IoT systems. Alternatives to DDPG include tabular approaches, linear function approximation, and distributed learning, all of which can theoretically be implemented using their framework.</td>
</tr>
</tbody>
</table>

Gap Analysis

The literature review above indicates that meta-heuristics and deep learning have been thoroughly researched for production planning. Additionally, collaborative scheduling and reconfiguration optimization have become increasingly popular in recent years. Research has not yet addressed smart planning and reconfiguration that take dynamic work arrival into account. Methods for creating task schedules are driven by one or more goals;
thus, it is important to compare them to gauge how effective they are. In this sense, the following critical metrics can be taken into account in scheduling techniques in distributed applications, such as cloud and fog environments:

- **Makespan**: The overall amount of time required to process a group of tasks through to completion. However, in scheduling algorithms, makespan is useless because it is hard to anticipate when a collection of tasks is completed because a new task stream is always being added.
- **Efficiency**: Measured as the ratio of runtime to overall makespan.
- **Throughput**: Throughput is the number of units of information that a system can process in a given amount of time.
- **Waiting period**: The interval between task submissions and implementation, as well as the interval during which a task waits for occurrences or other supplies.
- **Execution time**: The length of time a process runs and uses its resources.
- **Response time**: This is the amount of time between when a task is submitted and when it is finished. It can also be computed as the total of the waiting period and the processing time.
- **Cost**: The overall amount paid for the use of resources as well as supplemental costs such as energy costs.

The primary motivation for the creation of scheduling algorithms was the desire to both maximize efficiency and ensure fair distribution of available resources. Scheduling is the process of allocating available resources to the requests that have been received. Schedulers are a general class of algorithms.

In this paper, we propose a Deep Learning Algorithm for Big Data Task Scheduling System (DLA-BDTSS) that can be implemented in scheduling systems that operate in the cloud or over the Internet of Things. The research concluded that DLA-BDTSS is superior to several widely used alternatives. When compared to other methods, it converged faster and produced better results, making it a good choice for scheduling big data tasks.

3. Deep Learning Algorithm for Big Data Task Scheduling System

To simulate task planning and reconfiguration systems, this section designed reward, react, and state elements for planning and reconfiguring agents. When one of the two agents needs to take action, the agent creates that action based on aspects of the current environment. A reward is given to the agency to change its parameters when the action is completed.

In Figure 1, we see the overall layout of the DLA-BDTSS system. Elements of the architecture include the Internet of Things units, cloud nodes, and fog nodes. The system’s generated tasks are queued using the deep learning model in accordance with the advised scheduling strategies. Emulating a network required the development of both reconfiguration and programming agents. A few reasons why autonomous design is a good idea are listed below. Firstly, it can ease training burdens by reducing the dimensionality of state features and action domains. The cloud, a fog node, and the final users are the foundation of Figure 1. Fog nodes receive cloud data, which are then sent to the devices in the field. It is easy to see the information flowing between the cloud and the fog node. There can be both an actual queue and a scheduled queue for tasks to be completed in a fog node. The information is provided in its final form for consumers.

Additionally, planning and reconfiguring decision-making durations are similar if a unified agent is built. The agents must go through several epochs to understand how to decrease the frequency of reconfiguring. Additionally, the separate designation is scalable and can include other decision-makers, including a logistics provider.

3.1. System Architecture

Processing occurs in an information hub on a connected home, a smart router, port, converter, host system, etc., in the cloud environment. The connected home in the information hub represents the connection from the smart router to the host system in the cloud.
environment. The connected home in the cloud environment is the connection between the router and the host system. All the devices in the cloud environment are given to the connected home. The power and the smog nodes are given to the connected home. The complete connection in the information hub is dealt with by the connected hub. Due to the inadequate processing power, some fog nodes collaborate locally, and simultaneously link to cloud endpoints recognized as virtual cloud servers to meet mobile users’ demands. They assume that a fog brokerage, node, and fog nodes are all included in the network. Fog nodes talk to phone devices immediately. The fog brokers, who are in charge of analyzing, estimating, and then scheduling all activities to be carried out in the cloud service, receive all demands from phone devices right away. Since fog brokers are located near fog devices, data transfer between them takes time and is eliminated. Figure 2A presents the DLA-BDTSS flow diagram for your viewing pleasure. Data analytics and related services are part of the cloud services offered from the beginning stages onward. The actual queue and the scheduled queue in the fog node work together to ensure that the task is placed in the appropriate queue. The final users are provided with the planned task in the manner of a smartphone.

![Figure 1. The architecture of the DLA-BDTSS system.](image-url)
Data analytics and related services are part of the cloud services offered from the beginning. Figure 2A presents the DLA-BDTSS flow diagram for your viewing pleasure.

Out in the cloud service, receive all demands from phone devices right away. Since fog who are in charge of analyzing, estimating, and then scheduling all activities to be carried home. The complete connection in the information hub is dealt with by the connected home. The power and the smog nodes are given to the connected home. The connection form represents the connection from the smart router to the host system in the cloud environment. The connected home in the cloud environment is the connection given to the connected buying nodes, host system, etc., in the cloud environment. The connected home in the cloud environment is connected to when the outcome of the task is merged (step 9) after all tasks have been age (step 8). The response is given to the smartphone network via the fog layer the user is provided with the planned task in the manner of a smartphone.

Additional, the separate designation is scalable to secure the study’s validity. First, a message is sent by a smartphone network (step 1), and the connected fog node responds. The fog broker immediately receives this request (step 2). Step 3 divides each task into a group of activities that will be handled in a distributed manner (step 1), and the connected fog node responds. The fog broker immediately receives this request (step 2). Step 3 divides each task into a group of activities that will be handled in a distributed manner (step 2). Step 4 calculates the expected number of commands and resource usage. The fog broker manages all project and node data and uses a scheduling scheme (step 5) to determine a suitable task allocation. After the output, tasks are sent to the appropriate cloud networks and systems (step 6). Every node is in charge of completing all tasks allocated to it (step 7), after which it delivers the results directly to the fog broker.

As shown in Figure 2B, the DLA-BDTSS pipeline has been broken down and its workflow of the proposed scheduling scheme.

Step-wise DLA-BDTSS Algorithm: The DLA-BDTSS method, built into fog brokers, seeks to create an ideal task execution schedule that maximizes time and money efficiency to secure the study’s validity. First, a message is sent by a smartphone network (step 1), and the connected fog node responds. The fog broker immediately receives this request (step 2). Step 3 divides each task into a group of activities that will be handled in a distributed network. Step 4 calculates the expected number of commands and resource usage. The fog broker manages all project and node data and uses a scheduling scheme (step 5) to determine a suitable task allocation. After the output, tasks are sent to the appropriate cloud networks and systems (step 6). Every node is in charge of completing all tasks allocated to it (step 7), after which it delivers the results directly to the fog broker.

Figure 2. (A) The flow diagram of DLA-BDTSS. (B) The workflow of the proposed scheduling scheme.
networks and systems (step 6). Every node is in charge of completing all tasks allocated to it (step 7), after which it delivers the results directly to the fog brokerage (step 8). The response is given to the smartphone network via the fog layer the user is connected to when the outcome of the task is merged (step 9) after all tasks have been executed (step 10).

As shown in Figure 2B, the DLA-BDTSS pipeline has been broken down and its process mapped out. This system includes fog nodes, cloud nodes, base stations, and mobile user devices. The following is a description of the scheduler’s process for the proposed plan: (1) The data from the mobile are sent as a request to the fog node. (2) This forms the fog node, and the task is forwarded to the base station. (3) The task from the base station is estimated, scheduled, and given to the cloud nodes. (4) The cloud node executes the task and then sends the result in a particular time. The process is initiated by users, and contact is maintained between cloud and fog nodes. The base station collects the forwarded task from the fog node. The base station collects the task, decomposes the task, estimates, and sends the obtained task in the form of a combined result. The base station combines results by decomposing the task, and schedules and sends the task. The combined result in the base station is implemented in particular time. The final combined result is calculated and distributed to users. The DLA-BDTSS architecture with the smartphone network sends the request to the fog node. The task is routed to the base station from the fog node. The task is transmitted to and communicated with via the base station. The cloud node receives the decomposed task from the base station. The task is carried out in the cloud node, and the results are executed. The architecture stage is followed by the reconfiguration agent.

3.2. Reconfiguration Agent

When the reconfiguring judgment unit chooses to rearrange, the reconfiguring agent (RA) generates a reconfiguring activity based on recent state characteristics and receives the benefit of the previous reconfiguring activity as well as the benefits of earlier transactions at each T step. The reconfiguration agent is used to rearrange the reconfiguring activity. In order to rearrange the reconfiguring operation, the reconfiguration agent is utilized. The creation of reward, training, and state elements is described below.

3.2.1. Reward

Remember that the goal of the investigated problem is to lower the overall tardiness cost for all tasks in the network. Every reconfiguration measure should contribute to lowering the overall cost of delay. In other words, the increase in the overall cost of timeliness throughout this configuration stage must be minimized for every reconfiguration process. A deep learning agent aims to maximize the cumulative reward for an episode. As a result, every reconfiguration phase’s compensation is specified as the opposite of the tardy cost that was just implemented per second at that step.

The timeliness cost is produced during a phase using buffer tasks and completed tasks. Let each finished task be converted during a reconfiguration process. When a present reconfiguration phase is finished, the function is reset. \( r' \) and \( r'' \) are the starting and end duration of the present reconfiguration stage, respectively. Consequently, the time step \([r', r'']\) reconfigure reward is shown in Equation (1).

\[
R_R = \frac{1}{r} - \frac{1}{r'} \left[ \sum_{x=0}^{O} \sum_{y=0}^{O} \frac{C_y}{\alpha_y} (r' - \max(r', l_y)) \right] + \left[ \sum_{y=0}^{N} \frac{\alpha_y}{C_y} (r' - \max(r', l_y)) \right] \tag{1}
\]

The initial and final reconfiguration stages are expressed as \( r' \) and \( r'' \). The cumulative rewards in each episode are rewarded and given for the reconfiguration judgement. The reconfiguration judgement is described in the section below.

3.2.2. Reconfiguration Judgment

When the first device completes a task, the reconfiguration judgment is triggered to determine whether to rearrange. The RA makes a reconfiguration decision only when it
chooses to do so. The frequency of human reconfiguration is decreased by the reconfiguration judgment system. If not, it will take many events for the RA to learn how to reduce the frequency of reconfiguration.

Let $t' = 0, 1, 2, \cdots, N$ represent the existing production mode and $W_{t'}$ represent its buffer. The present system time is indicated by $t'$, $t'_c$. The $\beta_x$ stands for the work $x$’s current area tardy cost, where current cost refers to the moment in time $r'$. The current cost $\beta_x$ is expressed in Equation (2).

$$\beta_x = \begin{cases} \alpha_x & r' > l_y \\ \frac{g_y}{C_y} & r' > l_y > r'' \\ 0 & \text{else} \end{cases}$$  \hspace{1cm} (2)$$

At the three main circumstances, the reconfiguration judgment unit makes a decision to reconfigure. The current time is denoted as $r'_c$, and the length of the queue is denoted as $l_y$. The lower and higher cost function is denoted as $r'$ and $r''$.

The cost function’s pictorial representation is expressed in Figure 3. The final result is obtained from the current computation cost of the task and the initial and final task values. The cost function is expressed below:

![Figure 3. The pictorial view of the cost function.](image)

Case 1: when $W_{t'}$ is blank.

Case 2: When $\beta_y$ is smaller than the $x$th% of and there are no outstanding tasks in $W_{t'}$. Here, $\beta$ is a list and $\beta_y$ is the mean of the present unit delay cost of $y$. The mean delay is expressed in Equation (3).

$$\beta_y = \frac{1}{N} \sum_{x=0}^{N} \beta_x$$  \hspace{1cm} (3)$$
The mean cost is expressed as $\beta_x$, and the total number of samples available is denoted as $N$.

Case 3: When $N$ more tasks have been handled in $W_t$ and $\beta_q$ is below the $y\%$ of $\beta$. Every state characteristic is an array, so take note of it. The maximum, lowest, mean, and standard error are computed for every state attribute to reflect the properties of the underlying data. As a result, the state vector has a total dimension of $4 \times 4 = 16$. For state characteristics, min-max normalizing is employed. State feature $F_t'$'s counterpart, $F_t''$, is expressed in Equation (4).

$$F_t'' = \frac{F_t' - \max(F_t')}{\max(F_t') - \min(F_t')}$$

The minimum and maximum functions are represented as min(,) and max(,). The group of all $F_t'$ without resampling during the first ten periods is represented by $F_t''$ in this case. For the resource $F_t''$, the max-min algorithm gives higher priority to the larger task $F_t''$ than to the smaller tasks. In max-min, several smaller tasks can run parallel to the larger ones. The overall timeframe is determined by the completion of the longest job here. RA is a reward earned for an episode that reduces the cost delay. Human configuration is reduced as a result of the reconfiguration judgment. After the device reconfiguration stage is completed, deep learning uses an efficient scheduling agent to schedule the task. The goal of the scheduling agent is to choose, for each new task, the most suitable one from the pool of available options for resources and services.

3.3. Deep Learning Method

To create a satisfying real-time timetable for all upcoming tasks over the period, it is sometimes necessary to identify one or more good planning rules to employ throughout the entire plan-making process. Shortest Processing Time (SPT), Longest Processing Time (LPT), Shortest Queue Time (SQT), and Longest Queue Time (LQT) are examples of standard scheduling criteria. This study develops an effective scheduling agent to select an appropriate service among all potential resources for every task that arrives. The scheduling strategy is implemented using the deep learning system [18]. Deep learning is a popular approach to machine learning that was developed with the purpose of teaching computers to perform tasks that are second nature to humans. The methods of deep learning are incorporated into the scheduling agent in order to partition the $Q$-values into a variety of activities and services that are determined by the task. Specifically, the deep $Q$-learning approach is used to resolve the problem under consideration. The $Q$-learning method is used to observe the $Q$-value with the particular action and services. The actions are given as the tasks which are given as the reward to serve the function. The scheduling agent totally depends on the $Q$-value and the actions that focus on the services. The observed data are completely focused on the $Q$-value and the action. The complete stages of $Q$-value, action, service function, and the reward are called the scheduling agent.

The deep-learning-based scheduling model is depicted in Figure 4. The system is used to analyze and process different tasks and produce an optimized task management model to lower the energy cost and end-to-end delay.

Scheduling Agent

It expresses the agency as an equation that can be optimized by representing it coherently and educating it. The loss function measures how far expectations and actual objectives diverge. A model might predict that delivering a task to its prospective services provides excellent value, even if providing the task to a different service yields higher rewards. The difference among the projected value and the desired value should be close. Equation (5) shows the cost function.

$$LF = \frac{Q}{[r + \beta \cdot \max(F_t', F_t'')]^2}$$

(5)
The initial and the current cost function are denoted as $F_t$ and $F_{t'}$. The $Q$ value is expressed as $Q$. The cost function is denoted as $\beta$, and the number of data is represented as $r$.

![Diagram](image)

**Figure 4.** The deep-learning-based scheduling model.

First, it involves selecting a provider for the actual project and adding it to that service’s backlog. The prize and the new state of the scheme are then visible. It determines the maximum goal $Q$-value derived from the findings and then reduces it. Finally, it calculates the goal value by multiplying the present reward by the reduced target $Q$-value.

There is a significant discrepancy between the specific value and the forecast value, so it is risky to use the same system to generate both. Therefore, it employs the framework comprising the target system and prediction system for instructing throughout the learning phase. With some values locked in place, the architecture of the predictive network is transplanted into the target environment. Consistent learning is achieved by periodically copying variables from the forecast network into the target network. The scheduling agent makes it possible to monitor the state of operations and services in relation to assigned jobs. In the serve function, the $Q$-value is rewarded. A target and prediction system are types of systems encountered during the learning phase. The task-scheduling mechanism is then provided with the normative values.

### 3.4. Task Scheduling Mechanism

The configuration of the edge server includes several virtual machines (VMs), indicated by $M = (m_1, m_2, \ldots, m_N)$. The computational capability of these VMs, shown by $CC = (c_1, c_2, \ldots, c_N)$, is diverse. The scheduling algorithm chooses how tasks should be scheduled, including which VM to allocate each task to. The first task kept in the queue is placed into the awaiting slot that was left when a task was planned, which causes it to leave the awaiting slot. It is anticipated that just one VM would handle every task and that all of the VM’s computational resources would be used. Each VM’s anticipated processing period is unknown until it is executed.

A task’s system response, when assigned to a VM, comprises both the VM completion time and the waiting period in the awaiting slot. One can calculate task $x$’s completion time in VM $y$ as follows in Equation (6).

$$ET_{xy} = \frac{D_x}{M_y}$$  \hspace{1cm} (6)
The launch time \( (D_x) \) of the actual project is the arrival date if the VM is not currently performing any tasks \( (M_y) \); otherwise, the task starts when the VM becomes available, which means all previous activities are done. Let \( I_{xy} \) and \( F_{xy} \) stand for the opening time and end time of task \( x \) on VM \( y \), respectively. As a result, the beginning time is dependent on the completion times of all previous tasks as shown: \( ST_{xy} = \text{maximum} \ (I_{xy}, F_{xy}) \), and the task \( x \) completion time is given by Equation (7).

\[
I_{xy} = ST_{xy} + PT_{xy}
\]

where \( PT_{xy} \) is how long it takes VM \( y \) to complete task \( x \), and the starting time is denoted as \( ST_{xy} \). The waiting period and the completion time make up the reaction duration of task \( x \) on VM \( y \), and are given in Equation (8).

\[
P_{xy} = B_{xy} + PT_{xy}
\]

where \( B_{xy} \) is the duration of task \( x \)’s waiting period on VM \( y \), the task completion time is denoted as \( PT_{xy} \). If the mission is finished instantly, there is no time delay; if not, the waiting period is defined as the interval between the task’s initial onset and its completion. The waiting time is denoted in Equation (9).

\[
B_{xy} = ST_{xy} - \alpha_{xy}
\]

The task’s quality of experience is assessed using internet speed. The task starting time is expressed as \( ST_{xy} \), and the delay time is expressed as \( \alpha_{xy} \). The work satisfaction level, which is calculated as the proportion of the predicted delay and the reaction time, is the quality of experience (QoE) definition for every task. Equation (10) is used to indicate the task’s overall satisfaction of the work and the cost computation was carried out on VM \( y \).

\[
\phi_{xy} = \frac{l_x}{1 + P_{xy}}
\]

where \( l_x \) is anticipated delay, \( y \) denotes the computational cost. It is evident that the ratio increases as work satisfaction increases. The task starting point and the ending point calculate the QoE. The edge server with the VM measures the computational capability and schedules the task for each VM. After task scheduling to VM, each task is given to an evolutionary algorithm called a genetic algorithm.

3.5. Evolutionary Algorithm

The DLA-BDTSS, built on an evolutionary process called genetic algorithm (GA), is suggested in order to address work scheduling issues in cloud-fog processing environments as follows.

3.5.1. Encoding of Chromosomes

In evolutionary algorithms, a person defined by a chromosome stands in for a task-scheduling system. An n-dimensional array matching y chromosomes is being used to encode chromosomally. Every gene has a meaning of a numeral \( x \) in the band \([1; k]\) with \( k \) being the set of nodes, indicating that the associated task is given to the cluster designated \( x \). The sequence identity of a gene is also the sequence identity of a task. For instance, if a system of three nodes in the cloud-fog executes a series of 10 tasks, one scheduling algorithm option is as follows:

- Node tasks are denoted as \( J_1^1, J_2^2, J_3^3, J_4^4, J_5^5, J_6^6, J_7^7, J_8^8, J_9^9, J_{10}^{10} \). The chromosome that expresses the above solution will be \( C = \{2, 1, 2, 3, 1, 3, 2, 1, 2\} \). Node 1 performs the collection of tasks \( \{2,5,9\} \), node 2 is tasked with processing the collection of tasks \( \{1,3,8,10\} \), and node 3 is in charge of the collection of tasks \( \{4,6,7\} \).

This method of chromosomal encoding was chosen because it allows for flexible genetic processes such as crossovers and mutations to produce new organisms that explore
the space of potential solutions while inheriting good gene sections from their parents. Because goals, such as makespan and overall cost, are unaffected by tasks performed in a different sequence on a single machine, the order in which tasks are executed is ignored. The GA encodes the chromosomes, and each task is scheduled based on the processing of the collected task. The population is initialized after scheduling.

3.5.2. Initialize the Population

The starting population is the collection of all individuals used in a GA to determine the best course of action. Considering an N-person population, these N people are completely random in order to find many locations in the solution space and to guarantee the variety of the community in the first iteration. The next generation is formed by selecting some people from the starting population and performing some procedures on them. The upcoming generation is initialized, and the fitness evaluation is implemented in the next section of the fitness function.

3.5.3. Fitness Function

A person’s supremacy in the population is determined by their fitness function. A high-fitness person stands in for a high-quality solution. The utility function F estimates each person’s fitness value. The ability of a person to survive or perish through every generation depends on their fitness level. After the fitness function is initialized, the crossover operators, selection strategy, and mutation operator are initialized.

3.5.4. Genetic Operators

- Crossover operator

To produce offspring that inherit healthy genetics from their parents, two crossover operations are used with chromosomally encoded integers. In the process, two crossing nodes are randomly chosen, the first parent trades the middle genetic segment with the step-parent, and the other genes are left unaltered, resulting in the creation of a new person.

Figure 5 depicts the analysis of the DLA-BDTSS crossover model. The two parents are taken into account, and the crossover offspring chromosome is depicted above. The parental choice has a notable impact on the algorithm’s effectiveness in the population-wide crossover procedure. Every individual has a crossover rate. Each member of the community is assigned a probability of becoming the first parent, and the next parent in the community is chosen to participate in the crossover procedure using the roulette wheel method. High-quality participants with higher fitness values are much more probable to be select as parents in this type of selection, increasing the likelihood that beneficial gene segments will be passed down to the next generation.

- Selection strategy

The selection method is needed to choose individuals who will make up a community for the following generation. Spontaneous selection begins just after the crossing process. The fitness worth of the child is determined and compared to that of the mother; if the child is superior, the individual is retained to create the community of the following generation. Otherwise, the parent is kept in the following generation and the child is destroyed.

Whenever the genetic mutations of the less capable people are not removed as quickly and they still donate to exploring new territories on the search window, this selection retains the uniqueness of the populace over centuries. At the same time, great gene sections are not found in large numbers of people in a population.

- Mutation operator

A new population of N people is formed as a result of the processes of crossover and spontaneous choice. At a certain frequency of N people, everyone participates in one-point alteration. The position of the gene mutation was chosen at random and replaced with a new number to assign the task to a separate network.
Mutations improve the shortcomings of crossover and mutation by identifying the best solution when people avoid the extremes or flee them to investigate other regions of the candidate solutions. The array is matched with the chromosome after the matching process in encoding chromosomes. The population is created for the first time following the matching process. The fitness function determines a person’s authority. The genetic operators employ the crossover operator, the selection strategy, and the mutation operator. After initialization and processing, evaluation and validation take place.

![Crossover model of the DLA-BDTSS](image)

**Figure 5.** Crossover model of the DLA-BDTSS.

### 3.6. Evaluation and Validation

The method for evaluating the tasks and validating the solutions is discussed below.

#### 3.6.1. Applicative Results

It first builds a practical scenario to assess the suggested scheduler. It created a cloud layer of two fog networks to meet user demands. In the scenario, the fog node was set up to develop schedules for task devices, five of which were carried out in a public cloud. Applying the proposed approach in the timeframe, slots $t_1$ and $t_2$ produced the planned queuing results.

Each agent is assigned important tasks to deliver the critical ones and create the scheduled queue. Important information is used to show how important the activities are. This includes task arrival priorities, dependencies, progress, and the resources needed to finish.

$A_1, A_2, A_3, A_4,$ and $A_5$ are the tasks that have arrived. The number of the agencies is denoted as $n_1, n_2, n_3, n_4,$ and $n_5$. The asset “types” in the instance are Type A, Type B, Type C, and Type D on our fog-cloud system, and the number of components available for
each type. The suggested model’s planned queue and the worldwide queue of employees
predicated on the first-come first-served method are shown as follows:

- Tasks $A_4, A_5, A_2, A_3,$ and $A_1$ are in the universal queue depending on FCFS (applying
wait period).
- The queue that was planned at time $t_1$.

Only $A_4$ and $A_5$ are available in the first time frame since their condition is Yes, and in
our suggested model that equals Yes = 1 and No = 0. All tasks are prepared to be completed
during the next time slot because their state is Yes, which in the suggested model equals
Yes = 1. The queue was planned based on the proposed model for the period.

As seen in the practical scenario, the most critical work was completed first due to
the application of the suggested model, which takes into account the task’s importance,
duration, data processing, condition, and resource requirements. This results in better
resource utilization. This review investigates the possibility of deep learning to provide in-
formed recommendations regarding working groups and reconfiguration when scheduling
activities with dynamical task delivery. To improve simulation results, the DLA-BDTSS
system’s IoT nodes and big data analytics module with edge and fog layers employ an
innovative task scheduling system with optimization models. The system is made up of fog
nodes, cloud nodes, base stations, and mobile user devices. The user initiates the process,
and interaction occurs between cloud and fog nodes. The results are totaled and reported
to customers.

3.6.2. Task Scheduling

The following is the scheduling algorithm procedure:

- Set up settings, such as metrics for techniques, virtualization software, and tasks.
- The integration of resources and the classification of tasks into three groups.
- Start with N explosions; each one symbolizes a solution.
- Choose the resources needed to optimize.
- Determine the fitness feature value for fireworks and the number of sparkles and
  explosion ranges.
- Develop a technique to detect explosion radii and calculate the present cost.
- Choose an optimization parameter to undergo a random evolution for task scheduling.
- Select N iterations to represent the future testing population.

Step 4 (mentioned in Section 3.1. System Architecture, Step-wise DLA-BDTSS Algorithm)
continues if the maximal amount of repetitions is not achieved; otherwise, it ends.

In this section, the DLA-BDTSS system is designed with a deep learning and schedul-
ing model. The DLA-BDTSS system’s results are theoretically evaluated in this section,
and the findings are shown in the following section. Each stage of the DLA-BDTSS system
is carried out in stages, such as initialization and task scheduling, and the evaluation is
carried out in the proposed section.

4. Simulation Outcomes and Comparisons

We ran a series of numerical simulations in the trial report’s cloud-fog architecture
to evaluate the performance of the DLA-BDTSS system in scheduling a collection of tasks.
The costs of processing energy and resource utilization differ between fog and edge devices.
Every node has a distinct processing capability, which is indicated by its operating rate
and distinguished by its central processing unit (CPU), storage, and bandwidth utilization
costs. In the cloud, edge devices such as routers, bridges, and workstations have limited
processing capabilities.

In the cloud layer, tasks are handled by servers or virtualization software in high-speed
data centers. As a result, cloud nodes interpret things significantly more quickly than fog
nodes. However, using virtualized resources is more costly than using them in the cloud.
Dollars (USD, $), a unit of cash used in simulations to represent actual money, are used to
determine these expenditures.
The fitness level evaluation of the DLA-BDTSS system is represented in Figure 6. The simulation evaluations of the DLA-BDTSS system are performed by varying the iteration size from minimum to maximum, and the outcomes are evaluated. As the iteration count increases, the individual fitness level of the system also increases. The existing PSO and GA exhibit poor performance, and the DLA-BDTSS system with deep learning and big data for task scheduling produces higher simulation outcomes.

Figure 6. Fitness level evaluation of the DLA-BDTSS system.

Figure 7 shows the results of the cost study for the DLA-BDTSS system. After retrieving the number of tasks from the big data models, the overall cost of the computation may be determined. The energy cost rises proportionally with the number of operations performed. Overall system performance is enhanced by the DLA-BDTSS system's combination of deep learning, hybrid cloud, fog, and IoT modules with big data analytics modules. With varying job sizes, the system's efficiency improves while the cost does not.

The execution time analysis of the DLA-BDTSS is analyzed, and the results are tabulated in Table 2. The simulation outcomes are measured by the variation in the tasks, which is varied from 10 to 100 at the same time. The proposed DLA-BDTSS with a deep learning algorithm and big data analytics module enhances task scheduling and hence produces a lower execution time than the other existing models. The cost function is directly linked to execution time, and the big data module with deep learning reduces the computation time and hence the execution time.

The software outcome evaluations of the DLA-BDTSS system are analyzed and plotted in Figure 8. The balancing function is based on the cloud and fog nodes. The queries and the task are implemented based functions implemented. The scheduling of tasks is focused on the balancing function. The balancing function is computed using the edge and fog layers. The total energy cost and the makespan of the system are calculated based on the number of tasks. Depending on the number of tasks, the total cost and the life span of the proposed method are given. The balancing function is plotted against the total cost and the life span. The big data analytics module helps generate enough samples for the simulation evaluations. As the number of functions increases, the total energy cost of the system also
increases, and the makespan of the system decreases. The makespan time should be kept to a minimum to generate faster results.

![Figure 6. Fitness level evaluation of the DLA-BDTSS system.](image)

Figure 6. Fitness level evaluation of the DLA-BDTSS system.

Figure 7 shows the results of the cost study for the DLA-BDTSS system. After retrieving the number of tasks from the big data models, the overall cost of the computation may be determined. The energy cost rises proportionally with the number of operations performed. Overall system performance is enhanced by the DLA-BDTSS system’s combination of deep learning, hybrid cloud, fog, and IoT modules with big data analytics modules. With varying job sizes, the system’s efficiency improves while the cost does not.

Figure 7. Cost computation analysis.

Table 2. Execution time analysis.

<table>
<thead>
<tr>
<th>Number of Tasks</th>
<th>PSO (s)</th>
<th>GA (s)</th>
<th>DLA-BDTSS (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>13</td>
<td>32</td>
<td>9</td>
</tr>
<tr>
<td>20</td>
<td>15</td>
<td>36</td>
<td>12</td>
</tr>
<tr>
<td>30</td>
<td>17</td>
<td>43</td>
<td>15</td>
</tr>
<tr>
<td>40</td>
<td>32</td>
<td>48</td>
<td>13</td>
</tr>
<tr>
<td>50</td>
<td>46</td>
<td>54</td>
<td>17</td>
</tr>
<tr>
<td>60</td>
<td>52</td>
<td>47</td>
<td>21</td>
</tr>
<tr>
<td>70</td>
<td>57</td>
<td>52</td>
<td>25</td>
</tr>
<tr>
<td>80</td>
<td>63</td>
<td>49</td>
<td>27</td>
</tr>
<tr>
<td>90</td>
<td>67</td>
<td>56</td>
<td>31</td>
</tr>
<tr>
<td>100</td>
<td>72</td>
<td>61</td>
<td>34</td>
</tr>
</tbody>
</table>

Figure 9 depicts the execution time analysis computed and plotted. With a step count of ten tasks, the number of tasks for the simulation outcomes varied from low to high. The execution time is calculated as the difference between the time it takes to generate a task and the time it obtains to complete it in the cloud and at edge nodes. The DLA-BDTSS system, which includes big data analytics and an optimized model, improves task scheduling and management. In terms of execution time, the hybrid edge and fog modules with IoT nodes produce better simulation results.

The fitness value evaluation of the DLA-BDTSS system is represented in Table 3. The different iterations are considered for the simulation analysis, the outcomes are evaluated for the DLA-BDTSS and the results are compared with the existing models, such as PSO and GA. As the number of iterations increases, so does the fitness value of the DLA-BDTSS system. The fitness function is directly linked to the cost function and indirectly related to the execution time. The DLA-BDTSS exhibits higher outcomes with the hybrid cloud system with IoT and a big data analytics module.
produces a lower execution time than the other existing models. The cost function is directly linked to execution time, and the big data module with deep learning reduces the computation time and hence the execution time.

Table 2. Execution time analysis.

<table>
<thead>
<tr>
<th>Number of Tasks</th>
<th>PSO (s)</th>
<th>GA (s)</th>
<th>DLA-BDTSS (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>13</td>
<td>32</td>
<td>9</td>
</tr>
<tr>
<td>20</td>
<td>15</td>
<td>36</td>
<td>12</td>
</tr>
<tr>
<td>30</td>
<td>17</td>
<td>43</td>
<td>15</td>
</tr>
<tr>
<td>40</td>
<td>32</td>
<td>48</td>
<td>13</td>
</tr>
<tr>
<td>50</td>
<td>46</td>
<td>54</td>
<td>17</td>
</tr>
<tr>
<td>60</td>
<td>52</td>
<td>47</td>
<td>21</td>
</tr>
<tr>
<td>70</td>
<td>57</td>
<td>52</td>
<td>25</td>
</tr>
<tr>
<td>80</td>
<td>63</td>
<td>49</td>
<td>27</td>
</tr>
<tr>
<td>90</td>
<td>67</td>
<td>56</td>
<td>31</td>
</tr>
<tr>
<td>100</td>
<td>72</td>
<td>61</td>
<td>34</td>
</tr>
</tbody>
</table>

The software outcome evaluations of the DLA-BDTSS system are analyzed and plotted in Figure 8. The balancing function is based on the cloud and fog nodes. The queries and the task are implemented based functions implemented. The scheduling of tasks is focused on the balancing function. The balancing function is computed using the edge and fog layers. The total energy cost and the makespan of the system are calculated based on the number of tasks. Depending on the number of tasks, the total cost and the life span of the proposed method are given. The balancing function is plotted against the total cost and the life span. The big data analytics module helps generate enough samples for the simulation evaluations. As the number of functions increases, the total energy cost of the system also increases, and the makespan of the system decreases. The makespan time should be kept to a minimum to generate faster results.

Figure 8. Software outcome evaluations.

Figure 9. Execution time analysis.
Table 3. Fitness value evaluation of the DLA-BDTSS.

<table>
<thead>
<tr>
<th>Iterations</th>
<th>PSO (%)</th>
<th>GA (%)</th>
<th>DLA-BDTSS (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>47</td>
<td>39</td>
<td>65</td>
</tr>
<tr>
<td>100</td>
<td>49</td>
<td>42</td>
<td>69</td>
</tr>
<tr>
<td>150</td>
<td>52</td>
<td>47</td>
<td>72</td>
</tr>
<tr>
<td>200</td>
<td>56</td>
<td>52</td>
<td>74</td>
</tr>
<tr>
<td>250</td>
<td>59</td>
<td>54</td>
<td>75</td>
</tr>
<tr>
<td>300</td>
<td>61</td>
<td>57</td>
<td>76</td>
</tr>
<tr>
<td>350</td>
<td>61</td>
<td>59</td>
<td>76.2</td>
</tr>
<tr>
<td>400</td>
<td>62.1</td>
<td>63</td>
<td>76.4</td>
</tr>
<tr>
<td>450</td>
<td>62.5</td>
<td>65</td>
<td>76.6</td>
</tr>
<tr>
<td>500</td>
<td>62.8</td>
<td>65.7</td>
<td>76.8</td>
</tr>
</tbody>
</table>

The DLA-BDTSS system is designed and evaluated in this section. The DLA-BDTSS system with IoT nodes and a big data analytics module with an edge and fog layer enhances the simulation outcomes with a higher task scheduling system with optimization models. The performance measurement of DLA-BDTSS is based on the fitness value, cost computation, execution time, and software outcome evaluation.

5. Conclusions and Findings

This paper investigated intelligent planning and realignment for task scheduling with dynamical task delivery, utilizing deep learning to offer informed decisions for task scenarios and reconfiguring. System architecture for thoughtful planning and reconfiguring in a smart factory is provided as a solution to the issue. To reduce the overall timeliness cost for all tasks, a quantitative model is developed. Additionally, a deep learning system is suggested with a schedule and reconfigurable agents. The two services are the target audience for the agency’s activities, features, and incentives.

A deep learning model is also used to propose solutions to dynamic sequencing and reconfiguration with new work delivery. We can bridge the gap between traditional classroom instruction and digital learning environments by combining well-designed deep learning, instant decision making after training, and extensive use of performance data. These advantages highlight the value of applying extensive knowledge to address careful planning and reconfiguration issues in a real-world production environment. The findings of the study can be used to create a real-time, self-optimizing, decentralized platform for task scheduling in industrial automation systems.

The main bottleneck for cloud computing systems is task scheduling. A dependable task-scheduling method is required to improve system performance. CPU memory, execution time, and execution cost are all major concerns for today’s task scheduling algorithms. As a solution to the aforementioned issues, the Deep Learning Algorithm for Big Data Task Scheduling System (DLA-BDTSS) is proposed. DLA-BDTSS improved outcomes by 8.43% with an execution time of 34 s and a fitness value evaluation of 76.8%.

The outcomes of DLA-BDTSS are improved to 8.43% with an execution time of 34 s and fitness value evaluation of 76.8%. More research is needed to determine the optimal production parameters, for example the optimal number of terminals, vehicles, and workers. In order to create more efficient states and activities, or to supply the network with characteristics and actions based on the results of training and testing conducted with big data analytics, more study of the system’s architecture is required. In the future of the Internet of Things and cloud computing, an optimal scheduling algorithm should be designed to optimize CPU utilization, throughput, latency, and energy efficiency. It is standard practice in IoT applications to have numerous sensor nodes conduct the same operation repeatedly, which can increase the sensing cost and lower the network lifetime.
Author Contributions: The concept analysis Intelligent Task Scheduling Model [DLA-BDTSS] was created and designed by S.P. and D.A.; N.Z.J. and A.S.A. Abdulbaqi created the theory and ran the computations in Cloud-IoT systems using the Task Scheduling Model. S.P. has validated the analytical methods for determining the DLA-cost BDTSS’s computation analysis, software outcome evaluations, execution time analysis, and fitness value evaluation in the edge-cloud environment. The data and analysis tools were gathered by P.S.A. and A.A.A.; S.P. conducted the analysis and drafted the paper. D.A. and N.Z.J. have gone over the results and analysis section. They went over how Deep Learning Algorithm for Big Data Task Scheduling System (DLA-BDTSS) can be used in Internet of Things (IoT) and cloud computing applications again. The method employs a multi-objective strategy to shorten the makespan and maximize resource consumption. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Phuyal, S.; Bista, D.; Bista, R. Challenges, opportunities and future directions of smart manufacturing: Est outcomes a state of art review. Sustain. Futures 2020, 2, 100023. [CrossRef]


22. Qi, Q.; Tao, F. A smart manufacturing service system based on edge computing, fog computing, and cloud computing. *IEEE Access* **2019**, *7*, 86769–86777. [CrossRef]


**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.