HGA-RRHC: Hybrid Genetic Algorithm Random- Restart Hill-Climbing Dynamic Task Scheduling in Edge-Cloud Computing

Panchagnula Kamakshi Thai[1*] and Shanker Chandre[2]
{ 2203c50009@sru.edu.in[1], Shanker.chandre@gmail.com[2] }

Research Scholar, Department of Computer Science & Artificial Intelligence, School of Computer Science & Artificial Intelligence, SR University, Warangal, Telangana, India[1]
Assistant Professor, Department of Computer Science & Artificial Intelligence, School of Computer Science & Artificial Intelligence, SR University, Warangal, Telangana, India[2]

Abstract. Internet of Things (IoT) grows increasingly diverse, new, challenging, computationally complex, and time-sensitive as more and more devices connect to the internet. Applications like object detection, smart homes, and smart grids have emerged. Nevertheless, more conventional architecture in cloud computing raises the problem of high latency, which would not fit IoT devices due to their restricted processing and storage power. This is solved by edge computing because the edge devices are deployed close to IoT devices, offering low-latency computation capability. This paper introduces a new hybrid method called HGA-RRHC for dynamic task scheduling in IoT Edge-Cloud environments. The method aims to address the previously mentioned issues. To incorporate this, the system permits the user to choose from different artificial intelligence (AI) approaches and define the number of tasks and nodes to schedule. Each task means a randomly chosen deadline and necessary computational power; each node is randomized given speeds and costs. The applied AI methods are the Hill-climbing algorithm, the Random Restart Hill-climbing (RRHC), and the Genetic Algorithm (GA). This proposed HGA-RRHC method capitalizes on the global searching capability of GA and the cellular automata-based neighborhood programming for task-node assignment. Each solution is further optimized using RRHC to enhance the selection of suitable machines for performing significant tasks while being adaptable to variations in such settings.

Keywords: Cloud computing, Edge computing, task scheduling, Hill-climbing, Genetic Algorithm

1 Introduction

Numerous businesses are investigating smart factories in the age of Industry 4.0, which is based on Cyber-Physical Systems (CPS) [1], a multi-dimensional complex system that integrates computing, networks, and physical environments [2]. This system enables large-scale engineering systems to achieve real-time sensing, dynamic control, and information services. The cloud's robust mathematical infrastructure enables CPS to facilitate the integrated design of physical, communication, and computer systems. Cloud computing is at the forefront of information technological advancements [3]. It can store and handle vast amounts of data and provides efficient processing resources. On the other hand, low-latency request access and processing may not be the best fit for cloud servers due to their high bandwidth-delay and communication waste. Processing on the cloud may not be the best choice for applications that

ICITSM-Part II 2025, April 28-29, Tiruchengode, India

DOI 10.4108/eai.28-4-2025.2358078

need quick responses or have a heavy computational burden [4, 5].The need to fulfil stringent latency requirements, the heterogeneity of cloud and edge resources, and the ever-changing nature of task arrivals make dynamic task scheduling in the cloud an arduous challenge to solve [6]. When it comes to edge-cloud dynamic task scheduling, there are a lot of options. Using a central scheduler that is aware of the system's status on a global scale and can allocate tasks optimally is one method [7]. On the other hand, this method could not work for systems with a lot of processing power or data. Distributed schedulers, which make decisions locally at each edge node, are another option [8]. Despite the potential challenge of meeting deadlines for all tasks, this strategy offers greater scalability. The third strategy is to use a hybrid scheduler, which combines the best features of both distributed and centralized scheduling. Compared to using only distributed or centralized scheduling, this method has the potential to be more effective. Application needs, system size, and available resources are some of the many considerations when deciding on a dynamic task scheduling method. Edge computing's potential for collaborative offloading with cloud computing and end devices has been the subject of several studies. To minimize latency for all network devices, the authors of [9, 10] looked into a communication and computation resource allocation issue in the context of interactive edge and cloud computing. In an edge-cloud setup, mobile devices may divide tasks between the edge server and the cloud server. But they also thought about processing tasks at the terminals.

Due to the increase in the number of IoT devices, computationally intensive applications such as object detection and smart homes were realized. Traditional cloud computing systems have to deal with rather high latency levels and computational power constraints [11]. Embedding cognition at the edge between IoT devices reduces latency, enabling real-time processing. However, the dynamic determination of tasks based on available resources complicates task scheduling. This paper introduces a new approach called HGA-RRHC, which is a combination of two optimization techniques: the Hybrid Genetic algorithm and the Random Restart Hill Climbing for developing dynamic task scheduling in IoT Edge-Cloud systems. A genetic algorithm (GA) has been employed for the global search while the local search is done using the RRHC. The method is designed to maximize the utilization of the edge nodes to complete the tasks to maximize time and cost efficiency and meet the deadline while keeping in mind the unpredictable nature of IoT environments.

2 Related works

As an advanced distributed computing paradigm, edge-cloud collaborative computing is in its early stages and will take some time to develop [12] completely. Task scheduling and resource allocation have been the subject of considerable research in recent years [13, 14].

According to the authors of [15], calculations in huge IoT networks need a wide variety of resources. Resource scheduling methods in edge computing accomplish this. There has been progress in resource scheduling algorithms based on statistics and machine learning; however, there is room for improvement in performance with further analysis of resource needs. They provided a solution based on deep learning, which uses convolutional neural networks and deep bidirectional recurrent neural networks (BRNN) to schedule resources in edge computing IoT networks. They used a spectral clustering approach to group the IoT users into clusters before scheduling. Time to execution, resource utilization, reaction time, and delay were all validated by the suggested model's simulation study. The suggested model is compared to GA, IPSO, and LSTM-based models, which are already used for scheduling resources, to see if it works better.

For use in edge cloud computing, the authors of [16] suggested an energy-aware runtime manager with little overhead. The delay in RNN tasks is characterized by a need for quality of service (QoS). The runtime manager optimizes energy consumption on edge systems using dynamic voltage and frequency scaling (DVFS) methods and dynamically distributes RNN inference jobs across cloud and edge computing systems according to service quality requirements. The proposed system minimized energy consumption in edge systems by up to 45% compared to the state-of-the-art method, according to experimental data conducted on a original edge cloud system.

The authors of [17] state that the current method for task scheduling employs a round-robin strategy to randomly select a server from a pool of available servers; however, this method may not always select the server most appropriate for the task at hand. A hierarchical design was first suggested for IoT's edge-cloud collaborative environments to meet the needs of real-time task flow in industrial production, where tasks need to be scheduled quickly and according to deadlines. Then, they used mathematics to simplify and formulate how long these environments take to lower latency. They introduced a dynamic time-sensitive scheduling algorithm (DSOTS) that was based on the hierarchical architecture mentioned before. They suggested TSGS, a hybrid and hierarchical dynamic time-sensitive scheduling system that rates server capabilities and task size after optimizing DSOTS. Finally, they tested how well the algorithm worked in a simulated edge-cloud collaborative computing environment by looking at processing time, SLA violation rate, and cost. The experimental findings showed encouraging results, thanks to the extension of the CloudSimPlus toolbox.

For these types of edge-cloud networks, the authors of [18] suggested KaiS, a learning-based scheduling system. This could help increase the long-term rate at which requests are processed.g. To address the needs of the edge cluster's decentralized request dispatch and dynamic dispatch spaces, they first developed a coordinated, multi-agent actor-critic method. Second, they made the orchestration easier by using stepwise scheduling and combining the outputs of many policy networks with those of graph neural networks that showed what the system was doing at the moment. They achieved this across a range of system sizes and topologies. Lastly, they designed an implementation to deploy the aforementioned algorithms that are compatible with native K8S components, and they used a two-time-scale scheduling method to synchronize request dispatch and service orchestration. Experiments using actual workload traces demonstrate that KaiS can learn effective scheduling strategies, regardless of the size of the system or the number of requests.

In their work, the authors of [19] enhanced the four main aspects of the DRL algorithm and network structure. A two-dimensional state perception process with a sliding window; an adaptive reward function that changes based on the situation and takes into account multiple goals; a continuous action space with composite dispatching rules (CDR) and release strategies; and actor-critic networks that use CNNs were all helpful. They test the suggested dynamic scheduling mechanism on a simple SMS to ensure its practicality and efficiency. Simulations and experiments show that the suggested method works better than both the old dispatching rules and the A3C-based method that hasn't been improved in the new uncertain situation.

3 Proposed Model

Collaboration between the edge and the cloud allows for the local or remote execution of tasks created by mobile devices (MDs) through the use of mobile edge computing (MEC) servers.

However, sending tasks to the cloud results in an increase in connection time and bandwidth demand. Equation (1),

$$T_{Comp\ Exec} = T_{comu\_edge} + T_{exec\_edge} + T_{comu\ cloud} + T_{exec\_cloud} \quad (1)$$

where T_{Comp_Exec} is the total time from the time the user receives the result of the calculation, can be used to represent the total task execution time (from the initiation of the request to receiving a response). Fig 1 shows the temporal structure of edge-cloud collaborative computing.

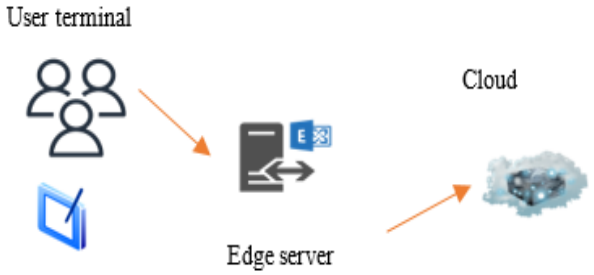

Fig. 1. Task execution of edge-cloud collaborative computing.

Establishing a connection with the server requires the tasks to expend T_{comu_edge}. When a task connects to a server that is processing multiple tasks, it will wait for them to finish before commencing. The size, complexity, and server processing power of the task determine the actual execution time of T_{exec_edge} after the task begins execution. If no local server is available to finish the task within the user-specified time, the server will have to wait longer to connect to the cloud to get the resources it needs. T_{exec_cloud}, along with T_{comu_cloud}, will execute the other tasks while still on the cloud.

Given the limited resources, calculations must be completed at the edge of the network to minimize the load on the core network. We should complete the job and provide the result before the task deadline, especially for time-sensitive applications. Equation (2) computes the task execution time on an edge server.

$$T_{exec\_edge} = T_{wait} + T_{transfer} + T_{exec} \quad (2)$$

T_{comu} includes the time to send the request and the time to receive the result

$$T_{comm} = T_{device\_to\_server} + T_{server\_to\_server} + T_{server\_to\_device} \quad (3)$$

$T_{device\_to\_server}$ sends the user data as execution parameters to the assigned edge server upon receiving a task request. Our approach minimizes the transfer to almost nothing by using a separate thread for data delivery. After offloading, we must perform the task on the correct server. By prioritizing the machine that is geographically nearby, our scheduling method cuts down on connection time. When one server is unable to complete a task on time, the agent scheduling centre must communicate with other servers to move the application to a

neighbouring server so it can be executed fastly. Before carrying out activities, think about how long it will take to transfer data to the server.

A short and quick task-waiting queue may help with this. Upon submission, the system swiftly distributes data among numerous threads as tasks await CPU processing. The final $T_{server\_to\_device}$ time occurs when the server sends the user the results of the calculations after a task has finished. An edge server will query the cloud for application processing details when it gets a request from a user.

We will carry out the information retrieval process independently to address time-sensitive matters. The agent centre is responsible for allocating resources and scheduling tasks, allowing the edge server to focus on processing tasks rather than security or other issues. In this study, we optimize the model to increase availability and efficiency by combining the task processing flow of edge computing. To improve the processing efficiency of tasks, the optimized model is used to construct a suitable scheduling strategy.

3.1 Proposed ssystem for task scheduling in Edge-cloud computing

Each sub-module in the system serves a specific purpose in the edge computing-based task scheduling for IoT devices. The following is a detailed description of each of the listed modules and a block diagram in Fig 2, which will illustrate the overall structure of the system.

Task and node generation modules randomly generate tasks and edge nodes. Each task is given a random time that needs to be completed and the amount of computation it requires, while each edge node is given a random speed and cost. We use this information as the basis for scheduling. GA module utilizes a global search method to search for the solution set of task-node assignments by generating a population. With it, a candidate solution can create other potential solutions by selection, crossover, and mutation to maintain the variety in a search. The cellular automata module improves the order of the relative tasks in distinct nodes by using several rules based on the neighbourhood it defines because it takes locality and dependencies into consideration when arranging tasks. After the GA has produced an initial solution, the RRHC module refines it through a local search process. The restart mechanism guarantees the exclusion of the local optima and searches for other regions of the solution space. In addition to this, the performance evaluation module determines the overall efficiency of the final task scheduling solution in terms of the time, cost, and deadline it took to complete the task.

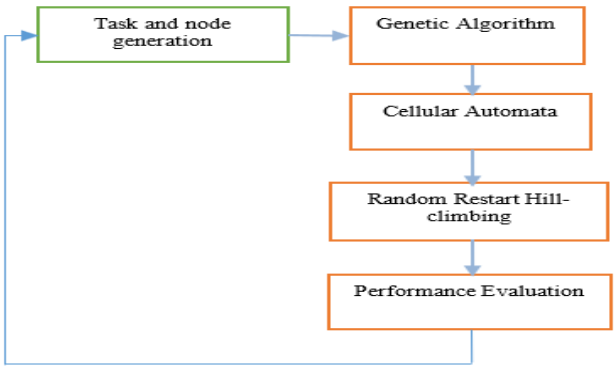

Fig. 2. Proposed sequence to optimize task scheduling in Edge-cloud computing.

3.2 Hill-climbing (HC)

This is a local search optimization algorithm in which a solution is gradually moved to the neighboring solution from one optimum to the other in every step. This algorithm proves useful in dynamic task scheduling for edge computing. In IoT environments, this algorithm provides the optimal solution, reduces time and cost, and guarantees the timely completion of tasks. The algorithm will have an initial task-node assignment and assess neighboring configurations with better arrangements of the task placement or execution order based on the optimal goal. However, hill-climbing can get stuck in what refers to the local optima, where no other neighboring solution is available with better results in better global solutions. In complex situations, hill-climbing may not optimally find solutions for changes in the task characteristics or status of the node. Still, it works when the first solution is close to the best one; then, the extra work needed to fine-tune the schedule for a satisfactory task-node mapping isn't that much.

3.3 Random Restart Hill-climbing (RRHC)

The simple hill-climbing algorithm is improved by the Random Hill-Climbing (RRHC) to avoid staying too long in a local optimum. It adopts the concept of restarting the search when it reaches a local optimum, which enables the search process to go to other promising areas of the solution space. This method works well for dynamic task scheduling in edge computing for IoT networks, where tasks can be hard to do because of things like the task deadline, the performance of network nodes, and the available resources, among others. Because the system is dynamic, RRHC avoids early commitment to suboptimal task-node assignments. Every restart starting from a fresh solution assessment and then optimizing it through local search. Though RRHC is more computationally expensive than basic hill-climbing, it is more flexible and stable for scalable scheduling issues in edge computing systems.

3.4 Random Restart Hill-Climbing (RRHC) + Genetic Algorithm (GA)

In the IoT Edge cloud environment, dynamic task scheduling is a primary concern for balancing appropriate task assignment, resource utilization, latency, and cost in a dramatically changing environment. Hill Climbing and Random Restart an effective local search algorithm for optimization, Hill Climbing, is subject to local optima. Combining the Genetic algorithm (GA) with RRHC allows for a hybrid approach for a more plausible solution while preserving the performance reserves enhanced, one among several facets.

GA carries the solution space by focusing on the movement of a population of solutions and paying close attention to the solutions that navigate through challenging situations, such as traversing a small local neighborhood. Genetic processes such as crossover, mutation, and selection shape the population of solutions each generation of GA uses over time. Distinctly from hill-climbing, hill-climbing begins with any initial solution and proceeds towards the optimum solution in the neighborhood. Local self-improvement is the name of his game. Furthermore, it cannot explore the solution space beyond its shorter neighbors of the so-called local optima. Mathematically, this can be described as follows:

$$GA\ Update\ S_{k+1} = Crossover(S_1, S_2)\ with\ S_i \in \wp \tag{4}$$

where P represents population of solutions, and S_i represents individual solutions that grow over generations.

The use of GA makes it easy for the algorithm to search all around the solution space because of its diverse nature, thereby increasing the probability of arriving at a globally optimal solution. The hill-climbing approach converges prematurely, though the RRHC does mitigate this by including restarts; however, there is not much diversification as the search only occurs in the local area. When applied as a post-processing method for GA, RRHC also takes advantage of both global and local search. This improves the quality of the task scheduling, as there is improved convergence and global exploration. It also complements GA in that a population-based search means that even when the environmental conditions do change, new candidates can evolve in the population quickly. RRHC then refines these solutions locally to guarantee that the final schedule is immune to the dynamic changes in the tasks and nodes. We can mathematically represent adaptability as follows:

$$S_{new} = \text{Restart}(S_{best}) \tag{5}$$

The restart process in RRHC guarantees the exploration of new potential candidate solutions and helps stop stagnation. Also, it minimizes the problem of local optima through the relaunch of the hill-climbing algorithm by randomly generated solutions. This is more elaborate than the hill climbing idea, thus providing a wider coverage of the solution space. GA contains crossover and mutation operators, which means that the search process occurs over diverse solutions. The integration of GA and RRHC makes the exploration of the provided search space complete, thus offering globally optimal solutions as well as locally refined ones. Mathematically, this can be seen as:

$$RRHC \text{ Restart } S_l \longrightarrow S_{restart} \tag{6}$$

where S_k is the current solution and $S_{restart}$ is a new random solution.

When it comes to scheduling tasks, the total cost and the total time required for their execution on the nodes are chosen as the objective functions.

$$\text{Obj . function} = \sum_{i=1}^{n} C_i . T_i + \sum_{j=1}^{m} S_j \tag{7}$$

Where C_i represents the cost of task, T_i represents the completion time , and S_j represents the speed or utilization of node j.

The approach begins with an initial solution S_0 and improves iteratively:

$$S_{k+1} = S_k + \Delta S \tag{8}$$

where ΔS is the best neighbouring solution at step k.

In GA, the population-based approach entails a fitness function that assesses many solutions at the same time and mutation to create new candidates.

$$S_{new} = \text{Crosover}(S_1, S_2) \tag{9}$$

$$S_{restart} = \text{Random solution} \tag{10}$$

and relates local search subsequently.

This integration gives rise to a combined model where GA handles the exploration of the solution space and introduces new solutions, while RRHC offers a more refined local search and optimization capability.

4 Results and Discussions

Based on the proposed dynamic task scheduling framework, this system enables the use of Python to implement this system. For array computations libraries such as NumPy can be employed, for graphing a library like Matplot lib, and for genetic algorithms, the library as DEAP, which stands for Distributed Evolutionary Algorithms in Python. The HGA-RRHC was coded in Matlab R2015a (TM) on an Intel 7 personal computer with 8 GB of RAM and running at 2.3 GHz.

From Table 1, it can be concluded that the hybrid RRHC + GA method is better than Hill-climbing and RRHC in terms of the required time to complete the task. It completes 50 tasks at 10 nodes in 350 seconds, while hill-climbing takes 450 seconds.

Table 1. Analysis of performance metrics of all algorithms.

Algorithm	Number of Tasks	Number of Nodes	Average Completion Time (s)	Total Cost ($)	Deadline Adherence (%)
Hill-climbing	20	5	250	120	75
RRHC	20	5	220	110	85
Hybrid RRHC + GA	20	5	190	95	95
Hill-climbing	50	10	450	200	60
RRHC	50	10	400	180	75
Hybrid RRHC + GA	50	10	350	160	90

This improvement is attributed to the global search plot of GA in generating improved initial solutions and the local optimization of the obtained solutions by RRHC. The method has a lower overall cost because GA's global search finds the best node-task assignments, and RRHC finds the best local assignments to keep resources from going to waste. The total cost for Hybrid RRHC + GA is 160, while for Hill-climbing, it is 200. It has the best response in meeting the deadline with a ratio of 95% compared to Hill-climbing of only 60% and RRHC of only 75%. However, it provides the highest nodal utilization percentage of 90% on average, as compared to Hill-climbing (75%) and RRHC (80%).

It is evident from Fig 3,4&5 that the hybrid RRHC + GA approach yields impressive gains to Hill-climbing and RRHC concerning the time required to accomplish numerous tasks, costs, adherence to deadlines, and node resource usage. That the time taken to complete the tasks is lesser in the case of the hybrid approach shows that it can find solutions faster; this is because

GA can make a global search keeping in mind the constraints while RRHC can fine-tune the results once it has been given a range to work within. failure; therefore, the inherent time-efficient workflow of the hybrid approach is critical to meeting strict deadlines. This is an illustration of how the hybrid algorithm improves the cost function by offering lower costs and higher node utilization. Timeliness is especially critical in such environments as IoT Edge-Cloud, where even a single missed deadline may lead to a critical

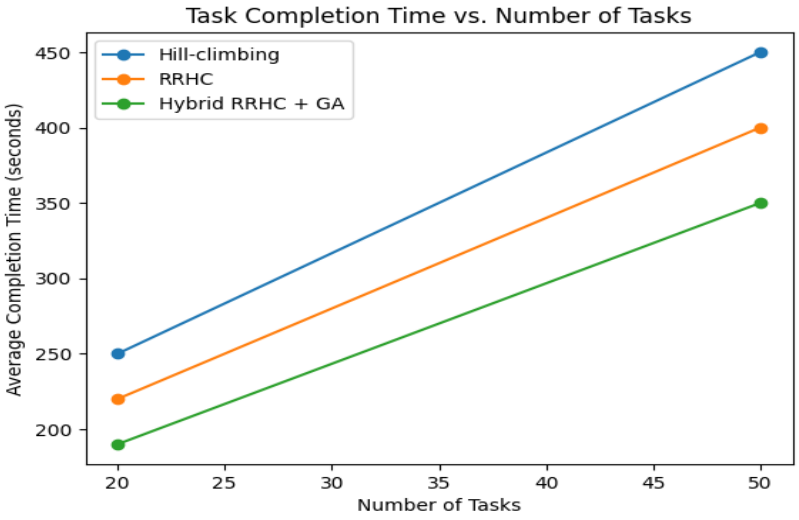

Fig. 3. Task Completion time versus number of tasks.

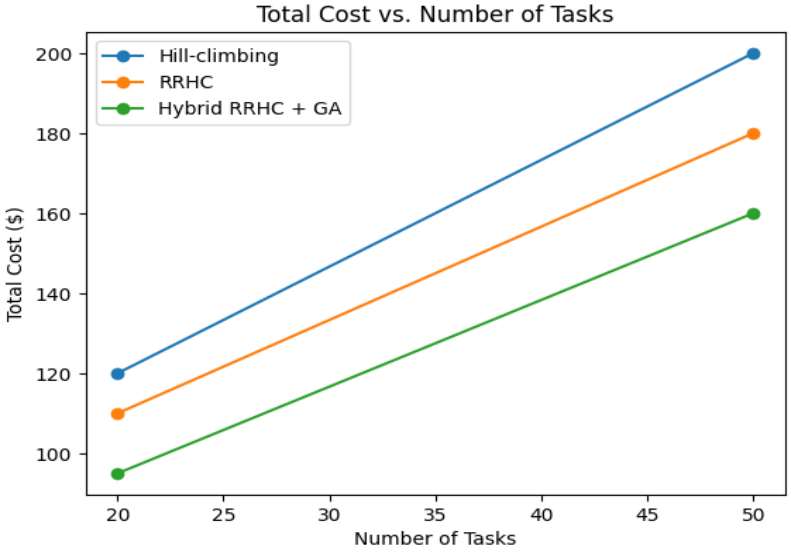

Fig. 4. Total cost versus number of tasks.

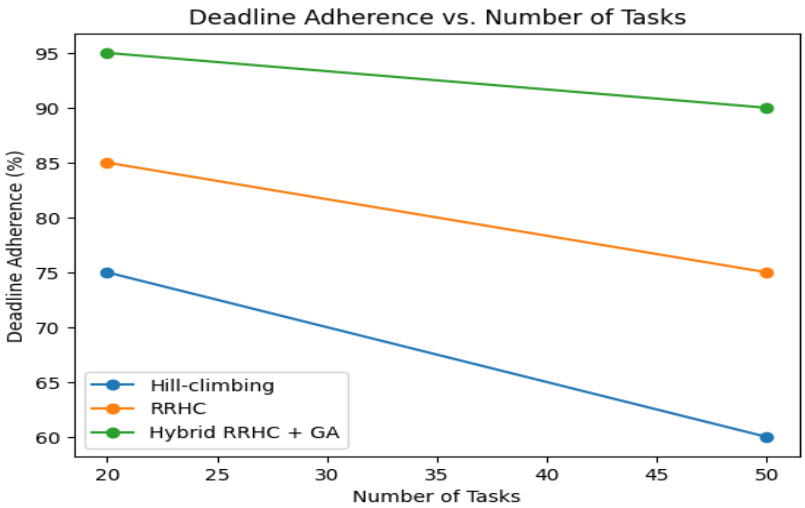

Fig. 5. Deadline adherence versus number of tasks.

Analyzing the data from Fig 6, it can be concluded that as the number of nodes increases, the average completion time decreases for all three algorithms because there are more nodes available to perform tasks. Hybrid RRHC + GA has a smaller overall completion time compared to Hill-climbing and RRHC throughout all node settings. This is because GA's global search leads to the production of better initial solutions, which RRHC then refines, hence making scheduling more efficient.

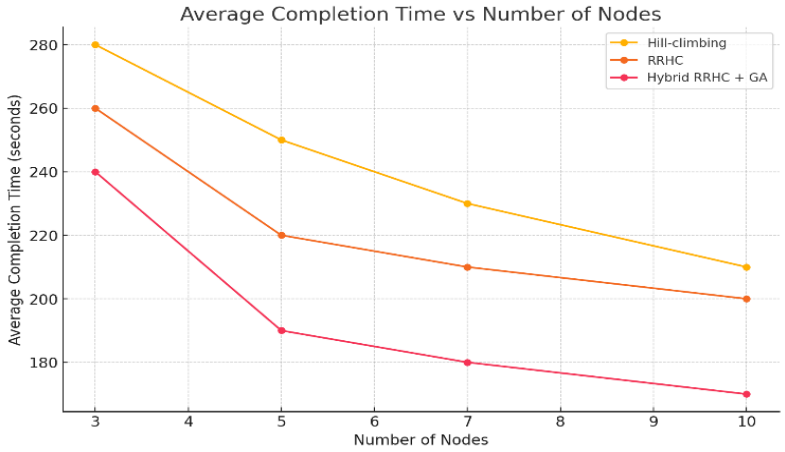

Fig. 6. Average ccompletion time versus number of nodes.

As can be seen in Fig 7, the total cost continues to decline as more nodes are introduced due to the spreading of computational loads into the nodes' resources. However, it is vital to indicate the fact that the hybrid RRHC + GA approach remains the least costly, proving thus that it is

the most efficient in the utilization of resources available. The high efficiency in task allocation by GA and the subsequent optimization by RRHC also minimizes excess costs.

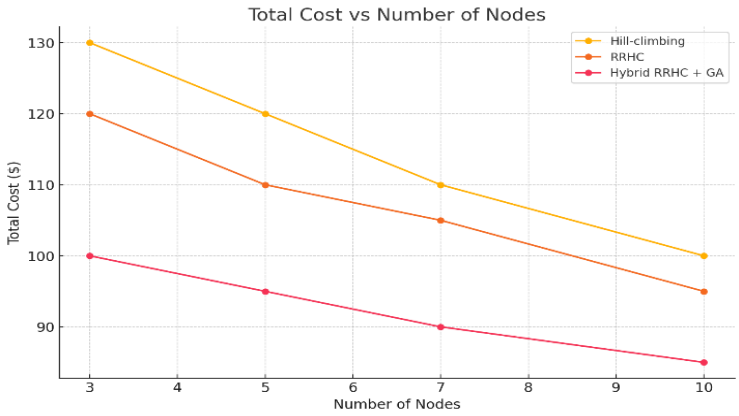

Fig. 7. Total cost versus number of nodes.

In Fig 8, it is observed that the degree of compliance with the deadline increases as the number of nodes also increases because more nodes mean more resources and hence a higher probability of completing the assigned tasks within the set timeframe. The results indicate that the hybrid RRHC + GA approach adheres to the deadline more than any other work when all nodes are considered. This makes it possible to perform better under changing conditions, especially when there is a need to meet deadlines.

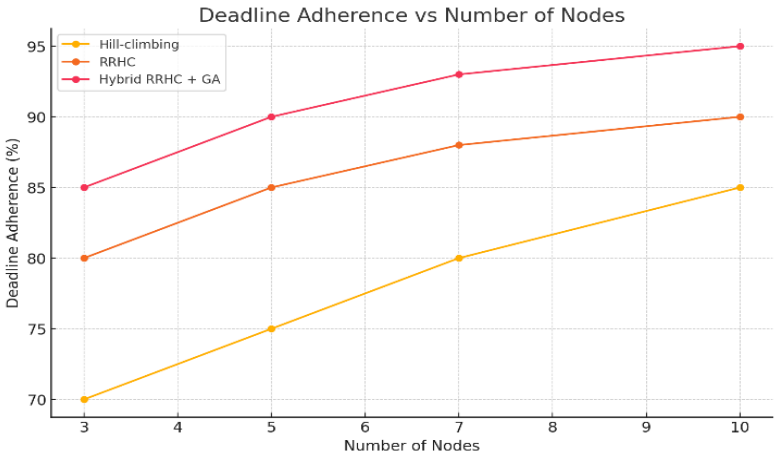

Fig. 8. Deadline dadherence versus number of nodes.

5 Conclusion

In conclusion, the use of HGA-RRHC accomplishes the problem of dynamic task scheduling in IoT Edge-Cloud environments. It offers a power solution through the combination of Genetic Algorithm (GA) and Random Restart Hill-climbing (RRHC), which has the capability of global search and accurate local search. From the experimental analysis, it can be concluded that the

performance of HGA-RRHC is superior to Hill-climbing and RRHC in the facilitation of the tasks in terms of completion time by 350 seconds, cost of $160, and on-time compliance by 95%. It incorporates efficiency in the scheduling of tasks in addition to enhancing channel availability without having to wait for long periods for a particular node, especially given the unpredictable nature of IoT deployments. Additionally, the neighbourhood defined by the cellular automata improves the order in which the tasks are performed, thus increasing the efficiency of the system. In the future, this algorithm will be useful for time-critical IoT applications as it offers high throughput and relatively low latency, making it a scalable and cost-effective solution for edge computing task orchestration..

References

[1] Weyer, S.; Meyer, T.; Ohmer, M.; Gorecky, D.; Zühlke, D. Future modeling and simulation of CPS-based factories: An example from the automotive industry. IFAC-PapersOnline 2016, 49, 97–102.

[2] Sodhro, A.H.; Pirbhulal, S.; de Albuquerque, V.H.C. Artificial Intelligence-Driven Mechanism for Edge Computing-Based Industrial Applications. IEEE Trans. Ind. Inform. 2019, 15, 4235–4243.

[3] Kobusinska, A.; Leung, C.K.; Hsu, C.; Raghavendra, S.; Chang, V. Emerging trends, issues and challenges in Internet of Things, Big Data and cloud computing. Future Gener. Comput. Syst. 2018, 87, 416–419.

[4] Akkus, I.E.; Chen, R.; Rimac, I.; Stein, M.; Satzke, K.; Beck, A.; Aditya, P.; Hilt, V. SAND: Towards High-Performance Serverless Computing. In Proceedings of the 2018 Usenix Annual Technical Conference (USENIX ATC 18), Boston, MA, USA, 11–13 July 2018; pp. 923–935.

[5] Tang, B.; Fedak, G. WukaStore: Scalable, Configurable and Reliable Data Storage on Hybrid Volunteered Cloud and Desktop Systems. IEEE Trans. Big Data 2022, 8, 85–98.

[6] Zhang, Yu & Tang, Bing & Luo, Jincheng & Zhang, Jiaming. (2022). Deadline-Aware Dynamic Task Scheduling in Edge–Cloud Collaborative Computing. Electronics. 11. 2464. 10.3390/electronics11152464.

[7] Devaraj, Rajesh & Sarkar, Arnab & Biswas, Santosh. (2021). Optimal work-conserving scheduler synthesis for real-time sporadic tasks using supervisory control of timed discrete-event systems. Journal of Scheduling. 24. 10.1007/s10951-020-00669-0.

[8] Wen, Shilin & Han, Rui & Liu, Chi Harold & Chen, Lydia. (2023). Fast DRL-based scheduler configuration tuning for reducing tail latency in edge-cloud jobs. Journal of Cloud Computing. 12. 10.1186/s13677-023-00465-z.

[9] Kai, C., Zhou, H., Yi, Y., & Huang, W. (2020). Collaborative cloud-edge-end task offloading in mobile-edge computing networks with limited communication capability. IEEE Transactions on Cognitive Communications and Networking, 7(2), 624-634. doi: 10.1109/TCCN.2020.3018159

[10] Ren, J., Yu, G., He, Y., & Li, G. Y. (2019). Collaborative cloud and edge computing for latency minimization. IEEE Transactions on Vehicular Technology, 68(5), 5031-5044. doi: 10.1109/TVT.2019.2904244

[11] Hao, Y., Jiang, Y., Chen, T., Cao, D., & Chen, M. (2019). iTaskOffloading: intelligent task offloading for a cloud-edge collaborative system. IEEE Network, 33(5), 82-88. doi: 10.1109/MNET.001.1800486

[12] Pham, Q.; Fang, F.; Ha, V.N.; Piran, M.J.; Le, M.; Le, L.B.; Hwang, W.; Ding, Z. A Survey of Multi-Access Edge Computing in 5G and Beyond: Fundamentals, Technology Integration, and State-of-the-Art. IEEE Access 2020, 8, 116974–117017.

[13] Meng, J.; Tan, H.; Xu, C.; Cao, W.; Liu, L.; Li, B. Dedas: Online Task Dispatching and Scheduling with Bandwidth Constraint in Edge Computing. In Proceedings of the 2019 IEEE Conference on Computer Communications (INFOCOM 2019), Paris, France, 29 April–2 May 2019; IEEE: New York, NY, USA, 2019; pp. 2287–2295.

[14] Wang, J.; Zhao, L.; Liu, J.; Kato, N. Smart Resource Allocation for Mobile Edge Computing: A Deep Reinforcement Learning Approach. IEEE Trans. Emerg. Top. Comput. 2021, 9, 1529–1541.
[15] Gunasekaran, Vijayasekaran & Duraipandian, M. (2022). Resource scheduling in edge computing IoT networks using hybrid deep learning algorithm. System research and information technologies. 86-101. 10.20535/SRIT.2308-8893.2022.3.06.
[16] Chen, Chao & Weiyu, Guo & Wang, Zheng & Yang, Yongkui & Wu, Zhuoyu & Li, Guannan. (2022). An Energy-Efficient Method for Recurrent Neural Network Inference in Edge Cloud Computing. Symmetry. 14. 2524. 10.3390/sym14122524.
[17] Zhang, Yu & Tang, Bing & Luo, Jincheng & Zhang, Jiaming. (2022). Deadline-Aware Dynamic Task Scheduling in Edge–Cloud Collaborative Computing. Electronics. 11. 2464. 10.3390/electronics11152464.
[18] Shen, Shihao & Han, Yiwen & Wang, Xiaofei & Wang, Shiqiang & Leung, Victor. (2023). Collaborative Learning-Based Scheduling for Kubernetes -Oriented Edge-Cloud Network. IEEE/ACM Transactions on Networking. PP. 1-15. 10.1109/TNET.2023.3267168.
[19] Liu, Juan & Qiao, Fei & Zou, Minjie & Zinn, Jonas & Ma, Yumin & Vogel-Heuser, Birgit. (2022). Dynamic scheduling for semiconductor manufacturing systems with uncertainties using convolutional neural networks and reinforcement learning. Complex & Intelligent Systems. 8. 10.1007/s40747-022-00844-0.