A Framework for Performance Optimization of Internet of Things Applications*

Osama Almurshed[1,2], Souham Meshoul[3], Asmail Muftah[4], Ashish Kumar Kaushal[5], Osama Almoghamis[6], Ioan Petri[1], Nitin Auluck[5], and Omer Rana[1]

[1] Cardiff University, United Kingdom
[2] Prince Sattam bin Abdulaziz University, Kingdom of Saudi Arabia
[3] Princess Nourah bint Abdulrahman University, Kingdom of Saudi Arabia
[4] Azzaytuna University, Libya
[5] Indian Institute of Technology Ropar, India
[6] King Saud University, Kingdom of Saudi Arabia

Abstract. A framework to support optimised application placement across the cloud-edge continuum is described, making use of the Optimized-Greedy Nominator Heuristic (EO-GNH). The framework can be employed across a range of different Internet of Things (IoT) applications, such as smart agriculture and healthcare. The framework uses asynchronous MapReduce and parallel meta-heuristics to support the management of IoT applications, focusing on metrics such as execution performance, resource utilization and system resilience. We evaluate EO-GNH using service quality achieved through real-time resource management, across multiple application domains. Performance analysis and optimisation of EO-GNH has also been carried out to demonstrate how it can be configured for use across different IoT usage contexts.

Keywords: Cancer Classification· Edge Computing· Industrial IoT· IoT Management· Meta-heuristics· Precision Agriculture· System Resilience

1 Introduction

An increasing demand for Internet of Things (IoT) technologies across various sectors, including agriculture, healthcare and industry, has heightened the necessity for effective data communication and processing. While cloud computing provides significant benefits, challenges persist in ensuring low latency and data privacy in IoT applications, particularly where bandwidth and power/energy are restricted. Edge computing provides a potential solution, extending cloud capabilities to embedded hardware systems such as single-board computers or user-owned (computational) acceleration devices.

As the IoT ecosystem evolves, there is an increasing demand for platforms that can handle complex applications, particularly those incorporating artificial intelligence (AI) and machine learning (ML). The integration of AI/ML into IoT

* Correspondence should be addressed to Osama Almurshed
email: almurshedo@cardiff.ac.uk; alternate email: o.almurshed@psau.edu.sa.

applications enhances their potential for optimizing processes utilising data acquired at the edge of a data network. This paper explores the use of an AI-based scheduling approach for managing intelligent IoT applications. An optimisation algorithm is used to reduce execution time and improve resource utilisation, referred to as the Enhanced Greedy Nominator Heuristic (EO-GNH). Integrated within the proposed IoT application management framework, EO-GNH aims to provide non-dominant solutions across delay, cost and risk factors. This strategy focuses on advancing one objective without compromising others.

Is EO-GNH adaptable to cross-domain characteristics and infrastructure configurations of IoT applications? We investigate this question within an IoT application landscape, forming the key contribution of this work. This paper is organized as follows: Section 2 discusses related work, providing context for our research. Section 3 describes our proposed Adaptive Platform for Edge-Cloud infrastructure. Section 4 explores the use of this framework within specific applications. Section 5 presents an evaluation of our proposed solution with concluding comments in Section 6.

2 Greedy Nominator Heuristic and Extensions

The Greedy Nominator Heuristic (GNH) is an optimization algorithm that addresses infrastructure deployments involving IoT, fog, and cloud computing environments [2]. GNH consists of several key components: a similarity function, max-heap, mappers, reducers, system controllers, and workers (described below). The similarity function, derived from TOPSIS [10], utilizes a context-dependent distance measure for comparing solutions derived from various optimization algorithms.

In GNH, mappers are assigned to specific nodes (labeled as *locations*), that process workflow functions and generate decision variables. When a placement request is made, mappers nominate potential nodes for deployment. A reducer then evaluates these nominations and selects the best nodes. This selection is repeated until the entire workflow is deployed. A greedy approach is applied in both the mapper and reducer stages, where the Euclidean distance similarity function is used as a heuristic to compare solutions to the ideal one. GNH adjusts to a variety of similarity functions, including cosine similarity [15] and fuzzy measures [17], so long as they can quantify the similarity between the ideal and explored solutions. To manage these solutions, a max-heap is employed. The max-heap is advantageous and allows quicker, more streamlined retrieval and removal of the solutions stored within it.

Max-heap is a binary tree structure which stores results from mappers and the reducer, ensuring that the maximum value is always at the root. Mappers calculate the similarity to an ideal solution for all locations using the norm-2 (euclidean distance) measure. The reducers then consolidate these results to select optimal deployment locations. Both mappers and reducers loop through the search space and update the max-heap.

The GNH system comprises a controller, which functions as the reducer, and workers that act as mappers, monitoring network performance and available computing resources at various locations, while the controller selects deployment locations from these options. This is implemented using Parsl [5], an asynchronous parallel programming library in Python, which can support execution on both high performance computing and edge resources.

The GNH has been used in intelligent IoT applications across diverse environments. For instance, in smart city applications, GNH demonstrated its capacity by autonomously deploying virtual functions across edge and cloud environments. This resulted in optimized resource usage, superior execution performance and significant cost reduction [3]. Further extending its use, GNH was successfully implemented in federated learning within a rural environment (with limited network connectivity). It was used to efficiently balance resource efficiency and performance across diverse IoT applications [1]. However, it became evident that GNH could not guarantee resource availability, underlining the need for further refinement in the optimization algorithm.

The EO-GNH framework is an enhanced GNH variant – each module of this framework is described in Almurshed et al. [2]. EO-GNH also integrates asynchronous parallel computing and machine learning models for meta-heuristic selection. Simulation-based evaluation has been used to develop EO-GNH, allowing assessment of system performance under various IoT conditions and failure scenarios [1], [2], [3].

Despite the merits of GNH [2], it has limited efficiency in supporting Pareto-optimal solutions. EO-GNH was created to address this challenge, and uses asynchronous MapReduce and parallel meta-heuristics to reduce execution time of the optimisation algorithm, avoiding local optima and ensuring service availability. It modifies the jMetalPy framework [7] using Parsl [5] for more efficient optimization, making it suitable for real-time IoT applications.

An important characteristic of EO-GNH is its ability to produce non-dominant solutions from the Pareto front, a crucial aspect of multi-objective optimization. It also simplifies the scalarization process [20] for a more dynamic and faster optimization.

3 Adaptive Framework for Edge-Cloud

This section describes the system components of our optimisation framework that makes use of EO-GNH, including properties of this algorithm. A description of software libraries used to realise EO-GNH is provided, identifying its use across a range of different application use cases.

3.1 System Components

The framework makes use of: Parsl, a Python library for parallel programming [5]; the Integer Linear Programming (ILP) model for adaptive decision-making [2]; and heuristic and meta-heuristic scheduling via jMetalPy [7]. Parsl

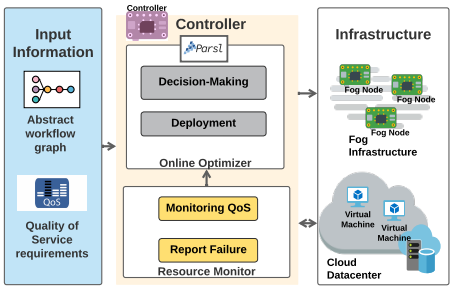

Fig. 1. Proposed adaptive controller architecture

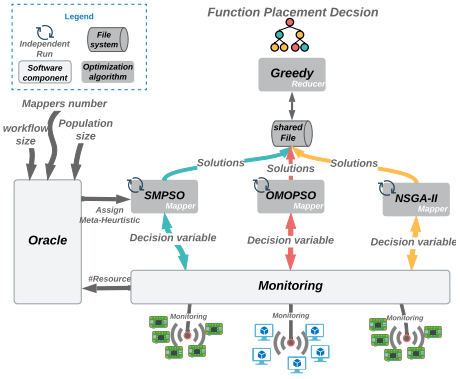

Fig. 2. Asynchronous MapReduce performs EO-GNH, initiated by the Oracle.

provides efficient service function execution, while the ILP model aids in optimal decision-making. As illustrated in Figure 1, the *controller*'s adaptive components consist of a *Resource Monitor* and an *Online Optimizer*. The Resource Monitor measures QoS and reports failures for use by the Online Optimizer. The decision-making process of the Online Optimizer uses ILP modeling to optimize the placement plan, employing heuristics such as GNH and EO-GNH. Furthermore, the Online Optimizer utilizes the Parsl library to accelerate decision-making and perform function deployments.

3.2 Enhanced-Optimized Greedy Nominator Heuristic (EO-GNH)

EO-GNH refines the GNH algorithm by incorporating Asynchronous MapReduce and meta-heuristics to determine optimal locations for redundant deployments. Figure 2 shows the workflow used in the EO-GNH framework. Mappers employ meta-heuristics to explore decision variables, while the Reducer, acting as a control fog node, chooses the most suitable locations. In contrast to GNH, EO-GNH utilizes meta-heuristics to generate a Pareto-front of solutions, giving the Reducer more flexibility when selecting resources, based on the similarity function. EO-GNH operates asynchronously, not requiring the Reducer to wait for all Mappers to complete their meta-heuristic iterations before making placement decisions. Each Mapper maintains a file of current optimal solutions accessible to the Reducer, facilitating instant decision-making, regardless of the solution's quality. The optimization algorithm employs nature-inspired meta-heuristics sourced from the jMetalPy library [7]. These meta-heuristics provide multiple solutions during runtime, each being an approximation of the best discovered Pareto-front.

The Oracle, a software module in EO-GNH, determines the compatibility of meta-heuristics with the application or infrastructure setup. Using decision tree models, it conducts preference-based sorting and assigns meta-heuristics to

Mappers. Our approach dynamically selects one of jMetalPy's meta-heuristic algorithms for optimization. Depending on specific application requirements, it chooses among the available Particle Swarm Optimization algorithms such as OMOPSO or SMPSO, or Genetic Algorithms such as GDE3, HYPE, IBEA, MOCell and NSGAII. This dynamic algorithm selection provides a unique perspective on optimisation not available in approaches.

Solution encoding in EO-GNH involves the transformation of meta-heuristic data into a processable form. The solution, initially represented as an array of integers in the jMetalPy framework, is converted into a data model compatible for the Reducer, ensuring the efficiency of the EO-GNH workflow.

EO-GNH relies on an asynchronous MapReduce mechanism initiated by the Oracle. Each Mapper runs a meta-heuristic chosen by the Oracle, based on attributes learned during the training phase. The Reducer's heuristic is selected manually as greedy.

4 Applications

We developed three IoT applications to inspect the EO-GNH across them. The workflow description highlights RPi 4B execution times and shows real-world functionality.

4.1 Federated Learning (FL) Application

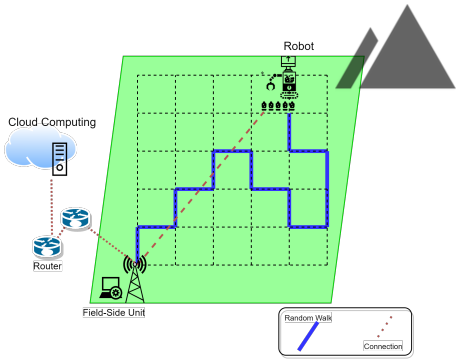

Fig. 3. Robot performs a random walk, affecting the connection to a field-side unit.

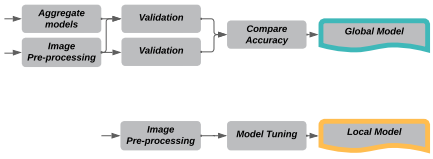

Fig. 4. Workflows for federated learning online training

Precision farming employs data-centric technologies and holds immense potential to improve farming productivity and outcomes [6]. Among its varied applications, automated weed control is a commonly discussed use case. By precisely identifying and handling weeds, this application can enhance farm yield,

reduce labour expenses and decrease pesticide use, thereby promoting efficient and sustainable farming practices. With the intent of supporting farm autonomy, this application leverages federated learning as a secure alternative to conventional machine learning approaches [19]. Federated learning enables model creation with locally sourced data, eliminating the need for centralised data storage, ensuring each farm is able to manage its own data, whilst still sharing good practice with nearby farms. Rural areas, due to their limited network infrastructure, present unique challenges for implementing precision farming. Conventional machine learning algorithms might face difficulties under these conditions, potentially affecting network reliability and service availability. Hence, this scenario necessitates a solution that can operate effectively within this constrained network environment, ensuring consistent and effective precision farming practices. In this context, the scenario involves deploying Federated Learning-enabled mobile robots to mitigate these challenges. Serving as edge devices, these robots enhance field coverage and data collection, contributing to a global model without sharing raw data [1] The path of a robot are guided by a truncated random walk method to ensure efficient field coverage and task accuracy [14]. Significantly, the robot's distance from computing resources symbolises network latency in mobile edge devices, which can affect the quality of data communication (as shown in Figure 3).

Workflow Figure 4 shows the federated learning workflow used in precision farming. The descriptions for each function utilized in this workflows are provided below: *(1) Image pre-processing:* This task makes color mode alterations, image resizing, data formatting and pixel value scaling to prepare images for machine learning models. The processed images and their associated labels are then stored. *(2) Model tuning:* the model weights are adjusted on a new dataset to improve the performance of an existing neural network. The updated weights are saved separately for future use. *(3) Model aggregation:* parameters (weights) of several trained models are combined (e.g. averaged) to create one aggregated model. *(4) Validation:* the trained machine learning model's performance is tested on new data, using loss and accuracy metrics – these metrics are also returned along with the model. *(5) Accuracy comparison:* accuracy/ loss function across different models are compared to determine the most effective model – returning the best performing model. The average execution on RPi's for the following 5 functions are: *0.33s, 178.16s, 22.33s, 37.16s, and 0.10s* respectively.

4.2 Recurrent Neural Network (RNN) Inference Application

Food production, marked by intricate processes and strict food safety regulations, calls for innovative solutions to maintain high standards [4]. Notably, climate-controlled storage units play a crucial role in preserving food batches. IoT-based temperature monitoring systems (Figure 5) offer real-time tracking of temperature, ensuring optimal storage conditions, and regulatory adherence. A food processing facility employing a smart energy cluster for improved energy

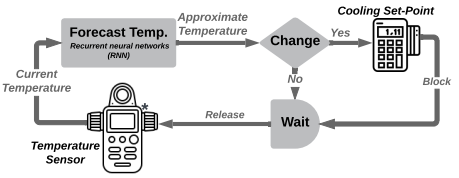

Fig. 5. Forecasting temperature control via sensors, providing data for prediction and setpoint updates.

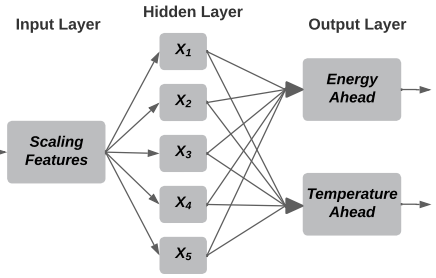

Fig. 6. Workflow of neural network for energy-saving applications

efficiency is used as another case study. The facility integrates a Recurrent Neural Network (RNN) specifically designed for energy conservation. By forecasting room temperature and energy consumption, adjusting temperature setpoints automatically, the RNN facilitates precise energy and temperature forecasts. The RNN implementation is divided into several service functions, depicted in Figure 6. This workflow management approach optimizes resource utilization. The RNN architecture comprises a pre-processing layer for data preparation, a hidden layer for prediction and output layers to yield predicted energy use and temperature values.

Workflow Figure 6 depicts the workflow of the neural network for energy-saving applications. The functions used in the workflow include: *(1) Scaling features:* involving normalization of the input data, such as current readings of the chamber's settings, power, capacity and the current season. The data is transformed into a standard scale compatible with the RNN, to support subsequent model-based predictions. *(2) Neurons X_1 to X_5:* These are the middle-layer neurons in the RNN, which apply weights to inputs and process them through a Hyperbolic Tangent (tanh) activation function. The output of the Scaling Features function is adjusted for combining the weight value passed through the activation output. *(3) Energy ahead and temperature ahead:* these functions predict energy and temperature respectively. They consist of an output layer, unscaling layer and a bounding layer. The output layer calculates the sum of the outputs from the X_j layers, adjusted by the output layer weights. The result is then scaled back to original units (kWh for energy and degrees Celsius for temperature) in the unscaled layer. The average execution time on RPi's for these 3 functions are: *1.29s, 0.44s, and 0.07s, 0.06s* respectively.

4.3 Cancer Diagnosis

Prostate Cancer (PCa) is the second most prevalent cancer among men worldwide, with 1.4 million new cases detected in 2020 [18]. Precise disease classification, assisted by AI, is critical for optimal treatment and risk reduction [13].

Cancer research has made remarkable progress over the last century, leading to innovative diagnostic and treatment methods, especially for PCa [9]. This progress has led to a large amount of cancer-related data. Nonetheless, precise cancer detection still remains a difficult challenge. Currently, machine learning techniques are being utilized, demonstrating profound effectiveness in deciphering complex patterns and predicting cancer types [11]. The current ProstateX dataset, a derivative of the Cancer Imaging Archive (TCIA) dataset, includes retrospective prostate MR studies [12]. It addresses limitations within the TCIA dataset by providing lesion masks and information, facilitating research in medical image analysis and computer-aided diagnosis for prostate cancer [8].

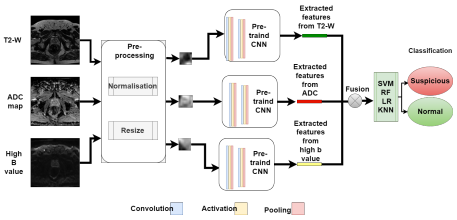

Fig. 7. Intelligent decision support system for prostate cancer diagnosis

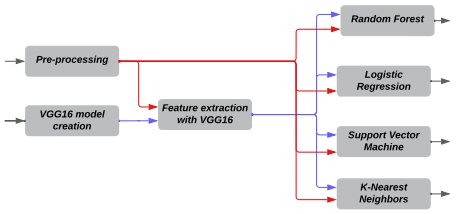

Fig. 8. Workflow for prostate cancer classification using Machine Learning

Workflow Figure 8 shows the training workflow for machine learning in prostate cancer classification. The workflow comprises the following functions: *(1) Preprocessing Data:* involves fetching data from a specified location and performing initial cleaning and formatting. *(2) Creating a VGG16 Model:* involves the application of transfer learning via the use of VGGNet, a broadly recognized 16-layer architecture. The model is pre-trained on ImageNet's database, an extensive repository containing more than 10M natural images across 1000 object categories [16]. *(3) Extracting Features from Magnetic Resonance Imaging (MRI) modalities with VGG16:* involves passing the loaded data through the VGG16 model, to extract the features, the first 2 blocks of the VGG model are used. We extract the features from each MRI modality alone, and then fuse the features. *(4) Creating, Training, and Testing Machine Learning Models:* This task sets up, trains, and evaluates Random Forest, Logistic Regression, Support Vector Machine, and K-Nearest Neighbors models from the Scikit-learn library on the RPi. The models are trained using extracted features and labels, and their performance is evaluated with unseen data. The average execution on RPi's for these 4 functions are: *0.47s, 1.53s, 10.23s, and 2.99s, 28.09s, 39.98s, 7.65s* respectively.

5 Evaluation

5.1 Experimental Setup

Evaluation was carried out through simulation, as outlined in Section 2, by dynamically modifying simulation parameters across the three scenarios: precision agriculture, the intelligence cooling system, and machine learning for Prostate Cancer classification. For these simulations, we utilized RPi benchmark data from section 4. An important feature of this evaluation was the assessment of the EO-GNH under various mapper configurations, from EO-GNH-1 to EO-GNH-4 (representing 1 to 4 mappers respectively). The experiment also incorporated results from a basic greedy approach without replicas, as well as the original GNH approach. Each scenario involved using a unique set of parameters. The number of Raspberry Pis (RPis) implemented was a key parameter, with most scenarios employing 1000 RPis, except the agriculture scenario which utilized 100 RPi resources. Simulation was chosen for its affordability, scalability, and failure control.

The Mean Time to Repair (MTTR) measures the average repair duration: 20-100 seconds for the Agriculture and the Cooling System scenarios; 5-15 seconds for the Prostate Cancer Classification scenario. The Mean Time to Failure (MTTF) measures the average duration between failures: 250-500 seconds in the Agricultural and Cooling System scenarios, and 50-100 seconds in the Prostate Cancer Classification scenario. By adjusting these parameters for each scenario within the simulated environment, the experiments offered detailed insights into the system's performance and resilience under differing conditions. This enables comparison across multiple scenarios, such as greedy approach without replicas and the original GNH approach.

5.2 Results

The evaluation of the EO-GNH framework on the above three scenarios considers the following metrics: *(1) Success rate:* this represents the likelihood of completing a task within the deadline. Together with risk (the probability of not meeting the deadline), it forms a proactive mechanism to minimize service disruptions. *(2) Makespan:* this refers to the total execution time of a workflow across distributed resources. The scheduler's primary objective is to optimize this, aiming for the shortest makespan possible. *(3) Utilized location (Cost):* this is defined by the number of resources used in a workflow. The goal is to balance resource use, minimize network congestion, and manage the risk-cost trade-off of redundant deployments.

As shown in Figure 9 (RNN forecasting, 100 RPis) and Figure 10 (RNN forecasting, 1000 RPis), the EO-GNH configurations employ MapReduce with a varying number of mappers, consistently outperforming other algorithms across all measured metrics: success rate, makespan, and utilized locations. Specifically, EO-GNH-3 and EO-GNH-4 yield the best results for 100 RPi and 1000 RPi configurations, respectively.

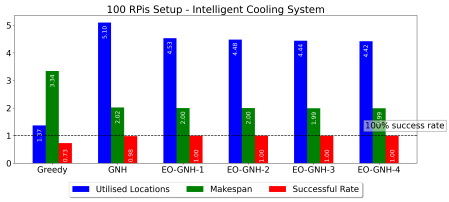

Fig. 9. RNN Forecasting with 100 RPis (average values)

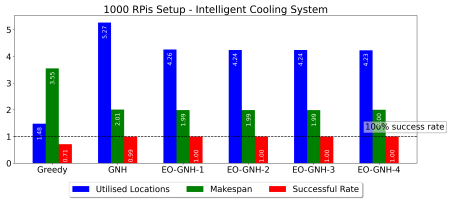

Fig. 10. RNN Forecasting with 1000 RPis (average values)

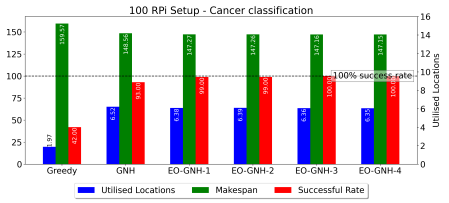

Fig. 11. ML Pipeline with 100 RPis (average values)

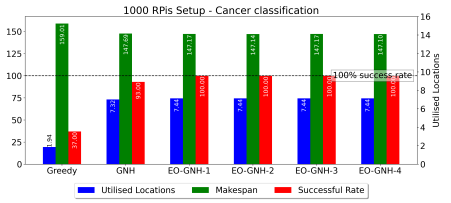

Fig. 12. ML Pipeline with 1000 RPis (average values)

The ML pipeline results for prostate cancer classification is shown in Figures 11 (100 RPis) and 12 (1000 RPis). Results show that EO-GNH configurations demonstrate superior efficiency across all metrics, with EO-GNH-4 being the top performer in both configurations. The longer makespan when transitioning from 100 to 1000 RPi setup indicates that the current task is more computationally intensive than RNN forecasting. In spite of this, EO-GNH configurations maintains a 100 percent success rate, indicating robust performance across a variety of distributed computing scenarios.

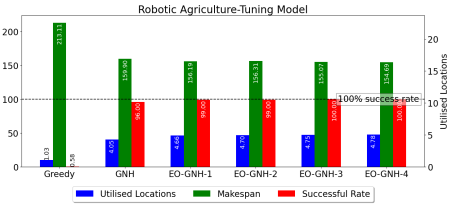

Fig. 13. Federated Learning - Model Tuning (average values)

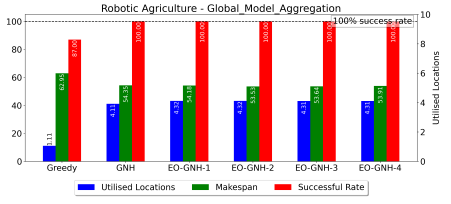

Fig. 14. Federated Learning - Global Model Aggregation (average values)

Examining the results of the tuning model for robotic agriculture using a federated learning approach (Figure 13) reveals a notable difference from previous patterns. While EO-GNH configurations continue to perform well, the

makespan disparities between them become more pronounced. This result may indicate that the number of mappers utilized in the MapReduce implementation is more crucial in this particular context. The GNH algorithm also exhibits significant performance enhancement in this configuration, although it does not surpass results of EO-GNH configurations.

The global model aggregation results (Figure 14) confirm previous findings, with all EO-GNH configurations achieving a 100 percent success rate. The variation in makespan and utilized locations between EO-GNH configurations suggest that adjusting the number of mappers can affect execution speed and resource utilization.

6 Conclusion

We describe the Enhanced Optimized-Greedy Nominator Heuristic (EO-GNH), outlining how it can be used across various IoT applications. Using asynchronous MapReduce and parallel metaheuristics, EO-GNH can be used to support dynamic resource allocation, by adjusting the number of mappers in the MapReduce component of this algorithm.

EO-GNH's adaptability and inherent hierarchical meta-heuristics approach, which is by definition problem-independent, allows it to overcome slow convergence by exploring multiple Pareto front approximations. This opens up its potential use beyond IoT task placements, including feature selection and hyperparameter optimization in machine learning, thereby improving accuracy, reducing overfitting and training time, and simplifying large-scale search problems. In deep learning, EO-GNH could be used to search for the optimal architecture, potentially enhancing Neural Architecture Search methods. All these areas are potential venues to be explored in the future. Additional investigation of potential EO-GNH research areas holds promise for future projects.

In conclusion, we note that EO-GNH effectively manages AI-driven IoT applications and its hierarchical meta-heuristics approach promises a wealth of possibilities for future research. By applying this algorithm to different domains, we may continue to uncover new strategies for optimization in a variety of contexts.

References

1. Almurshed, O., Patros, P., Huang, V., Mayo, M., Ooi, M., Chard, R., Chard, K., Rana, O., Nagra, H., Baughman, M., et al.: Adaptive edge-cloud environments for rural ai. In: 2022 IEEE International Conference on Services Computing (SCC). pp. 74–83. IEEE (2022)
2. Almurshed, O., Rana, O., Chard, K.: Greedy nominator heuristic: Virtual function placement on fog resources. Concurrency and Computation: Practice and Experience **34**(6), e6765 (2022)
3. Almurshed, O., Rana, O., Li, Y., Ranjan, R., Jha, D.N., Patel, P., Jayaraman, P.P., Dustdar, S.: A fault tolerant workflow composition and deployment automation iot framework in a multi cloud edge environment. IEEE Internet Computing (2021)

4. Alzahrani, A., Petri, I., Rezgui, Y.: Analysis and simulation of smart energy clusters and energy value chain for fish processing industries. Energy Reports **6**, 534–540 (2020)
5. Babuji, Y.N., Chard, K., Foster, I.T., Katz, D.S., Wilde, M., Woodard, A., Wozniak, J.M.: Parsl: Scalable parallel scripting in python. In: IWSG (2018)
6. Balducci, F., Impedovo, D., Pirlo, G.: Machine learning applications on agricultural datasets for smart farm enhancement. Machines **6**(3), 38 (2018)
7. Benitez-Hidalgo, A., Nebro, A.J., Garcia-Nieto, J., Oregi, I., Del Ser, J.: jmetalpy: A python framework for multi-objective optimization with metaheuristics. Swarm and Evolutionary Computation **51**, 100598 (2019)
8. Cuocolo, R., Stanzione, A., Castaldo, A., De Lucia, D.R., Imbriaco, M.: Quality control and whole-gland, zonal and lesion annotations for the prostatex challenge public dataset. European Journal of Radiology **138**, 109647 (2021)
9. Hanahan, D., Weinberg, R.A.: Hallmarks of cancer: the next generation. cell **144**(5), 646–674 (2011)
10. Hwang, F., Chen, S.J., Hwang, C.L.: Fuzzy multiple attribute decision making: Methods and applications. Springer Berlin/Heidelberg (1992)
11. Kourou, K., Exarchos, T.P., Exarchos, K.P., Karamouzis, M.V., Fotiadis, D.I.: Machine learning applications in cancer prognosis and prediction. Computational and structural biotechnology journal **13**, 8–17 (2015)
12. Litjens, G., Debats, O., Barentsz, J., Karssemeijer, N., Huisman, H.: Prostatex challenge data. The Cancer Imaging Archive (2017). https://doi.org/10.7937/K9TCIA.2017.MURS5CL
13. Noguerol, T.M., Paulano-Godino, F., Martín-Valdivia, M.T., Menias, C.O., Luna, A.: Strengths, weaknesses, opportunities, and threats analysis of artificial intelligence and machine learning applications in radiology. Journal of the American College of Radiology **16**(9), 1239–1247 (2019)
14. Pang, B., Song, Y., Zhang, C., Yang, R.: Effect of random walk methods on searching efficiency in swarm robots for area exploration. Applied Intelligence **51**(7), 5189–5199 (2021)
15. Radouche, S., Leghris, C.: Network selection based on cosine similarity and combination of subjective and objective weighting. In: 2020 International Conference on Intelligent Systems and Computer Vision (ISCV). pp. 1–7. IEEE (2020)
16. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115**(3), 211–252 (2015)
17. Samriya, J.K., Kumar, N.: An optimal sla based task scheduling aid of hybrid fuzzy topsis-pso algorithm in cloud environment. Materials Today: Proceedings (2020)
18. Sung, H., Ferlay, J., Siegel, R.L., Laversanne, M., Soerjomataram, I., Jemal, A., Bray, F.: Global cancer statistics 2020: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries. CA: a cancer journal for clinicians **71**(3), 209–249 (2021)
19. Yang, Q., Liu, Y., Cheng, Y., Kang, Y., Chen, T., Yu, H.: Federated learning. Synthesis Lectures on Artificial Intelligence and Machine Learning **13**(3), 1–207 (2019)
20. Zeleny, M.: Compromise programming. Multiple criteria decision making (1973)