National College of Ireland

Evaluating Distance Based Pareto Genetic Algorithm for Task-Offloading in Edge-Fog-Cloud Systems

MSc Research Project
Cloud Computing

Muralikrishnan Vijayan Nambiar
Student ID: 21195757

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Muralikrishnan Vijayan Nambiar
Student ID:	21195757
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Vikas Sahni
Submission Due Date:	14/12/2023
Project Title:	Evaluating Distance Based Pareto Genetic Algorithm for Task-Offloading in Edge-Fog-Cloud Systems
Word Count:	4492
Page Count:	19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	14th December 20203

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Evaluating Distance Based Pareto Genetic Algorithm for Task-Offloading in Edge-Fog-Cloud Systems

Muralikrishnan Vijayan Nambiar
21195757

Abstract

In the ever-changing world of the Internet of Things (IoT), effectively handling resource-constrained Edge-Fog-Cloud (EFC) systems is critical. This research investigates the use of the Distance-Based Pareto Genetic Algorithm (DBPGA) for task-offloading in such situations, tackling a difficult NP-complete problem that has largely remained unaddressed in existing literature. The significance of this study comes in its investigation of a novel technique to improve task-offloading efficiency, a vital feature in the performance of IoT systems. The research results show that the DBPGA outperforms the traditional Genetic Algorithm (GA) in optimising task offloading in smaller datasets across many workflows. The discovery is essential because it presents an effective technique for controlling IoT applications in fog computing, which is particularly valuable in enhancing task-offloading efficiency. However, the research reveals challenges with processing larger datasets, indicating the critical need for further improvements in the DBPGA's scalability and adaptability. This research not only contributes to the field by addressing a previously unexplored area, but it also establishes a new avenue for future research to improve computing efficiency in varied EFC systems, an important issue in the IoT domain.

1 Introduction

The expansion of the Internet of Things (IoT) in the modern digital world has ushered in a new era of data-driven devices, profoundly embedding them in our daily lives Martinez et al. (2021). This rise in data collection and transmission, while invaluable, creates new obstacles, particularly when real-time processing and low latency become critical. Fog computing emerges as a possible solution, bridging the gap between IoT devices and remote cloud data centres by relocating computation closer to the data source Bonomi et al. (2012).

1.1 Background and Motivation

The fast growth of Internet of Things devices has resulted in an exponential growth in data generation. While traditional cloud computing models are effective for large-scale data processing, they frequently suffer from latency concerns, especially when real-time answers are required Karamoozian et al. (2019). This is where fog computing comes in, with its decentralised computing structure that puts computation closer to the data source, lowering latency Yousefpour et al. (2019).

However, integrating fog computing into the IoT ecosystem is not without challenges. One of the most serious concerns is task scheduling, which entails deciding the order, timing, and location for completing tasks in a dynamic environment Hosseinzadeh et al. (2023). Task scheduling is a challenging issue to resolve because of the unexpected nature of IoT devices, the variety of processing demands of jobs, and the dynamic availability of fog nodes Taneja and Davy (2017).

The fundamental impetus for this research is the complexity of the problem and the pressing need for efficient solutions. A promising solution to these problems is provided by genetic algorithms, which are motivated by the ideas of natural selection Madhura et al. (2021). Their adaptability and capacity to identify optimal or near-optimal solutions in difficult contexts make them particularly suitable to the dynamic world of IoT and fog computing. While a variety of algorithms have been proposed to address these issues, the potential of distance-based Pareto genetic algorithms (DBPGA) remains largely unexplored for task scheduling in EFC Guerrero et al. (2022).

1.2 Research Question

The central research question of this study is:

"How effectively does the Distance-Based Pareto Genetic Algorithm optimize task offloading in resource-constrained Edge-Fog-Cloud environments?"

1.3 Document Structure

This research is organized systematically to ensure a thorough comprehension of the subject. The literature review in **Section 2:** Related Work highlights the present state of research in the Edge-Fog-Cloud paradigm. **Section 3:** Methodology describes the research methods and experimental design. **Section 4:** Design Specification describes the structure of the algorithm, whereas **Section 5:** Implementation addresses its practical implementation. **Section 6:** Evaluation analyses the findings and their consequences, and **Section 7:** offers concluding thoughts and prospective options for further investigation.

2 Related Work

In the present digital era, the IoT has integrated itself into every aspect of our everyday lives, resulting in a new growth of data-generating and data-transmitting devices. While this large sea of data is invaluable, it also presents new obstacles, particularly when real-time processing and minimal latency are required. Fog computing is a concept that aims to connect IoT devices to remote cloud data centres by moving computation to the edge and closer to the source of the data.

The survey by Martinez et al. (2021) provides a fascinating examination of the complexities of resource design in the context of the IoT ecosystem. While it's easy to dismiss resource allocation as a logistical issue, the study emphasises its strategic significance. In the dynamic and ever-changing world of IoT, where devices can be as unpredictable as they are many, having a robust and flexible resource allocation strategy is critical.

Fog Computing, as discussed in Bonomi et al. (2012), isn't just a trendy term, it's a radical new way of thinking about data processing in a future where IoT dominates. Fog Computing solves the pressing issues of latency and real-time processing by acting as an

interconnect between IoT devices and cloud data centres. But it comes with its own set of difficulties, just like any paradigm change.

The FOGPLAN framework by Yousefpour et al. (2019) is proof that creative solutions are being created in the industry. Allocating resources is important, but it's also important to do so in a way that simultaneously balances user needs with system constraints. This balance is essential, especially in light of the many and frequently conflicting requirements of various IoT applications.

However, there are several challenges involved in integrating fog computing into IoT. As pointed out by Karamoozian et al. (2019), while the potential benefits are numerous, including improved system performance and greater user experiences, the challenges are equally daunting. The road to complete integration is filled with challenges, from establishing seamless communication between devices to maintaining data integrity.

Even though it sounds simple, resource allocation is the key to solving these problems. Taneja and Davy (2017) examines this topic in-depth and emphasises the need for a resource-aware strategy. In a world where resources are limited but needs are unlimited, techniques that prioritise both system restrictions and user expectations are necessary. The Priced Timed Petri Nets (PTPNs) approach by Ni et al. (2017) offers a novel viewpoint on this problem. It emphasises a basic fact in the field of Fog Computing by allowing users to autonomously select resources based on a variety of characteristics such as cost, time, and credibility.

The following sections will explore more into task offloading, recognising its significance and the difficulties it creates in the EFC.

2.1 Task Scheduling in the Edge-Fog-Cloud Continuum

The complex environment of task scheduling in fog computing becomes clear as one builds upon the fundamental knowledge of resource allocation. The vast complexity and amount of jobs in the IoT ecosystem need an in-depth approach to scheduling. It involves carefully choosing the order, time, and location of job execution in a dynamic and unpredictable environment, in addition to allocating resources. The task scheduling issue in fog computing is intrinsically complex because of the varied computational requirements of IoT tasks and the dynamic availability of fog nodes, as mentioned in Hosseinzadeh et al. (2023).

Traditional scheduling systems, while effective to some level, frequently struggle to address these diverse difficulties. While priority-based mechanisms, as discussed in Aladwani (2019), have been useful in scenarios where tasks have varying levels of urgency, the dynamic nature of IoT applications frequently renders static priority assignments insufficient. Xu et al. (2021) on the other hand, introduces ways that focus on the capabilities and availability of fog nodes, providing adaptability but potentially confined by the limited processing resources at the edge.

Another critical component is balancing cost and performance. Because fog computing is frequently used in conjunction with cloud resources, algorithms that can skillfully combine the reduced latency given by edge processing with the computational prowess of the cloud are in high demand. Movahedi et al. (2021) research explores hybrid techniques, aiming for a harmonious task distribution that harnesses the capabilities of both the fog and cloud paradigms.

The limitations of conventional heuristic approaches become clear. Although systematic, these solutions frequently lack the adaptability required to effectively deal with the

ever-changing and unexpected landscape of IoT and fog computing. Madhura et al. (2021) emphasises this realisation and ushers in the investigation of meta-heuristic techniques, most notably genetic algorithms. Adaptability and evolution are introduced to the work scheduling problem by genetic algorithms, which are motivated by natural selection and genetic principles. They can evolve to discover optimal or nearly optimal solutions in complicated, dynamic environments because they are not constrained by the rigidity of conventional algorithms.

2.2 Task Scheduling with Genetic Algorithms

The survey paper by Guerrero et al. (2022) highlights the potential of GA's in tackling the issues provided by the dynamic nature of IoT systems. GA's iterative and adaptive nature enables them to navigate the challenges of the IoT ecosystem, where jobs have varying processing requirements and fog nodes have dynamic availability.

Recent research has focused on improving service quality in fog computing. For example, Akintoye and Bagula (2019) to improve the quality of service in fog computing, evolutionary algorithms were used to strike a compromise between real-time processing demands and fog node constraints. But this balance isn't just about effectiveness. As the complexities of fog computing environments expand, so do the cost consequences of task scheduling. Nikoui et al. (2020) looked into a cost-aware approach to work scheduling and emphasised the ability of genetic algorithms to manage the trade-offs between performance and economic viability.

This domain has a wide range of issues. Given the unpredictable behaviour of IoT devices, as well as the complex nature of jobs, novel solutions are required. These difficulties are explored Nguyen et al. (2019), which offers an evolutionary approach to task scheduling. The research highlights the potential of genetic algorithms in tackling the dynamic nature of IoT systems, emphasising the necessity for a scheduling mechanism that can adjust in real time to the changing demands of IoT devices and workloads.

Building on this, Skarlat et al. (2017) focuses on the optimised scheduling of IoT tasks under fog computing. The research demonstrates the ability of evolutionary algorithms to provide solutions adapted to the unique challenges posed by IoT contexts. It presents a framework that not only ensures effective task execution but also dynamically adjusts to changing computing requirements and resource availability in fog nodes.

The versatility of evolutionary algorithms is further demonstrated by considering their implementation in various computing contexts. Genetic algorithms were crucial in identifying the best task-offloading solution in the setting of bus networks, where Ye et al. (2016) developed a scalable fog computing paradigm. Vorobyev (2019) investigated the combination of mathematical models and evolutionary algorithms, providing a robust mechanism to optimise distributed systems based on cloud and fog technologies.

While genetic algorithms provide a strong mechanism for optimising job allocation, their emphasis on a single goal can sometimes be a constraint. This is especially apparent while handling the multifaceted problems of modern computing configurations. Such difficulties highlight the approaching need for a more holistic approach, one that can address various, often conflicting, goals.

2.3 Multi-objective Genetic Algorithms for Enhanced Task Scheduling

The need for algorithms that can simultaneously optimise numerous objectives becomes more and more obvious as the complexities of task scheduling in the IoT and fog computing landscape continue to develop. Multi-objective genetic algorithms (MOGAs) stand out as a viable answer to this complicated problem.

The hybrid heuristic technique by Hussain and Begh (2022) exhibits MOGA adaptability. The study proposes a revised strategy to optimise service composition in cloud environments by integrating a local search technique with a MOGA. This strategy, which takes into account several objectives such as cost, time, and reliability, demonstrates MOGAs' capacity to create Pareto-optimal solutions that respond to diverse criteria while maintaining efficient service delivery.

The "MOTORS" system, described byShukla and Pandey (2023), uses MOGA to optimise both risk and time constraints in the context of mobile edge computing. This dual-objective strategy not only ensures that offloading decisions are made on time but also reduces the chance of task execution failure, addressing the dynamic and unpredictable nature of mobile edge computing.

Mokni et al. (2023) emphasises the adaptability of MOGAs in cloud computing even further. The study highlights the capacity of MOGAs to find a wide collection of Pareto-optimal solutions while adjusting to the dynamic demands of cloud settings by focusing on minimising both makespan and total cost.

When it comes to real-time systems, Agarwal et al. (2023) emphasises the problem of minimising task-blocking time. The research adds a new dimension to the discussion by introducing two unique multiprocessor real-time locking methods. However, the flexibility of these protocols to change task priorities and workloads requires more investigation.

Aoudia et al. (2020) deftly explores the complex link between QoS and Fog-IoT computing. The research proposes a sophisticated approach to service composition by combining a 5-layered fog computing architecture with a multi-population genetic algorithm. However, the real-world applicability of this 5-layered architecture and the effectiveness of the genetic algorithm across varied IoT contexts remain subjects of interest.

While the works described have shed light on MOGAs' capabilities in task scheduling, one important path remains unexplored: distance-based Pareto genetic algorithms (DBPGA). DBPGA provides a more diversified and flexible optimization strategy as introduced in Ghosh and Dehuri (2004) by prioritising solutions based on their physical proximity in the solution space, which is especially important in the unpredictable settings of IoT and fog computing. The current literature's limited examination of DBPGA highlights the necessity for this research, which aims to fill this gap and provide a more detailed view of the issues of task offloading in EFC.

3 Methodology

3.1 Simulation Environment

The simulation environment was created with the help of the FogWorkflowSim toolkit Liu et al. (2019), which is well-known for its ability to simulate and analyse workflow performance in the fog computing ecosystem. This toolkit represents fog computing behaviour effectively by allowing the configuration of diverse computational resources

across the edge, fog, and cloud levels. The resources of each layer were thoroughly established, with specific processing and storage capacities to effectively imitate real-world conditions. The environment was calibrated for studies to mimic a variety of operational situations, such as changing network latencies and resource availability, mirroring the difficulties encountered in true fog computing deployments. The DBPGA was integrated within this virtualized environment, allowing for careful comparison of its performance to that of the GA under equivalent and repeatable settings, ensuring the dependability of the comparison study.

3.2 Development and Implementation of DBPGA

The DBPGA algorithm has been developed in Java and implemented in the FogWorkflowSim environment for comparison with the traditional GA. The goal of the comparative research was to evaluate the efficiency of DBPGA in a controlled simulated environment while providing consistent settings to validate the algorithms' performance.

3.3 Workflow Datasets

The simulation used five standardised scientific workflows to test the systems' resource allocation capabilities Shukla and Pandey (2023):

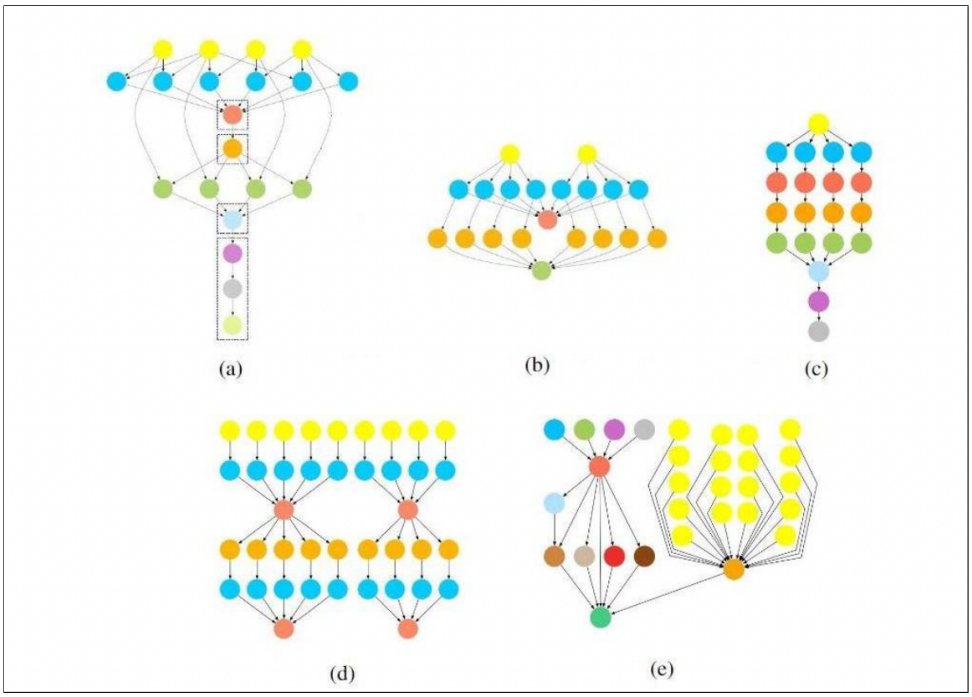

Figure 1: Structure of scientific workflows: (a) Montage, (b) CyberShake, (c) Epigenomics, (d) Inspiral, (e) SIPHT

- **Montage:** An astronomical image mosaic workflow, varying from 20 to 300 tasks.
- **CyberShake:** Earthquake hazard modelling workflow, ranging from 30 to 1000 tasks.

- **Epigenomics:** Genome sequence analysis workflow, with task counts from 24 to 997.
- **Inspiral:** Gravitational wave detection workflow, involving 30 to 1000 tasks.
- **Sipht:** RNA sequence annotation workflow, consisting of 29 to 968 tasks.

These workflows were chosen to cover a wide range of computing and data processing needs, from light to intensive.

3.4 QoS Metrics and Performance Evaluation

The performance of the DBPGA was thoroughly evaluated in a fog computing environment by applying a QoS metrics model that captures multiple aspects of workflow execution. These criteria were used to guarantee that the algorithm achieves a balance between computing efficiency and resource management, both of which are important in resource-constrained systems as mentioned by Shukla and Pandey (2023).

3.4.1 Makespan

Makespan is defined as the amount of time required to complete all tasks in a workflow. It is an important indicator of a scheduling algorithm's temporal efficiency. Makespan's mathematical model for a workflow G is given by:

$$M = \max(DF(T_i))$$

where T_i denotes the i^{th} task in the workflow and $DF(T_i)$ represents the execution finish time of task T_i.

3.4.2 Cost

The total cost measure reflects the whole cost of completing a workflow. This covers both the computational cost and the cost of data transfer between machines. The cost C is determined as follows:

$$C = \sum_{i=0}^{n}\sum_{j=0}^{n}(DF(T_i)\cdot U_j) + \sum_{i=1}^{n}\sum_{j=1}^{n}(C_{wij}\cdot CTR_{ij})$$

Here, $DF(T_i)$ is the execution completion time of task T_i, U_j is the cost per unit time of the VM j processing task T_i, C_{wij} is the connection weight between tasks T_i and T_j, and CTR_{ij} is the cost of data transfer between machines where T_i and T_j are mapped.

3.4.3 Energy Consumption

The energy consumption is made up of active and idle energy components, which are indicated by E_{active} and E_{idle}, respectively. The active energy component, which refers to the energy expended while executing a task, is represented by:

$$E_{active} = \sum_{i=1}^{n}\alpha f_i^2(v_i - v_{min_i})$$

where α is a constant, f_i represents the frequency, and v_i and v_{min_i} represent the supply voltage for the resource on which task i is being performed. The idle energy component, which accounts for energy consumed by idle resources, is calculated using:

$$E_{idle} = \sum_{j=1}^{n} \sum_{k \in IDLE_{jk}} \alpha f_{min_j}^2 L_{jk}$$

with $IDLE_{jk}$ being the set of all idle slots of resource j, f_{min_j} and v_{min_j} as the lowest supply voltage and frequency of resource j, and L_{jk} as the amount of idle time for $idle_{jk}$. The total energy consumption (E) during the execution of tasks is the sum of active and idle energy components:

$$E = E_{active} + E_{idle}$$

3.4.4 Resource Utilization Factor

RUF is a metric that measures how effectively a scheduling strategy utilises virtual machines (VMs) by giving the maximum workload to each while remaining within the resource utilisation threshold. It is defined as follows:

$$RUF = \frac{\text{Workload}}{\text{Resource Utilization Threshold}}$$

The most effective scheduling technique employs the fewest amount of VMs, increasing the RUF value and enhancing the system's energy efficiency.

4 Design Specification

4.1 Algorithm Overview

Over the EFC continuum, DBPGA is designed to optimise task offloading solutions with an emphasis on three key performance metrics: cost, energy, and time. DBPGA was introduced by Ghosh and Dehuri (2004).

4.2 Algorithmic Steps

1. **Initialization:**
    - Generate an initial population of solutions randomly or based on heuristic information.
    - Each solution encodes a potential task offloading strategy within the EFC system.
2. **Fitness Evaluation:**
    - Evaluate each solution using the FogWorkflowSim's integrated metrics for time, energy, and cost.
    - Assign a fitness score based on how well each solution meets the objectives, with lower scores indicating better performance.

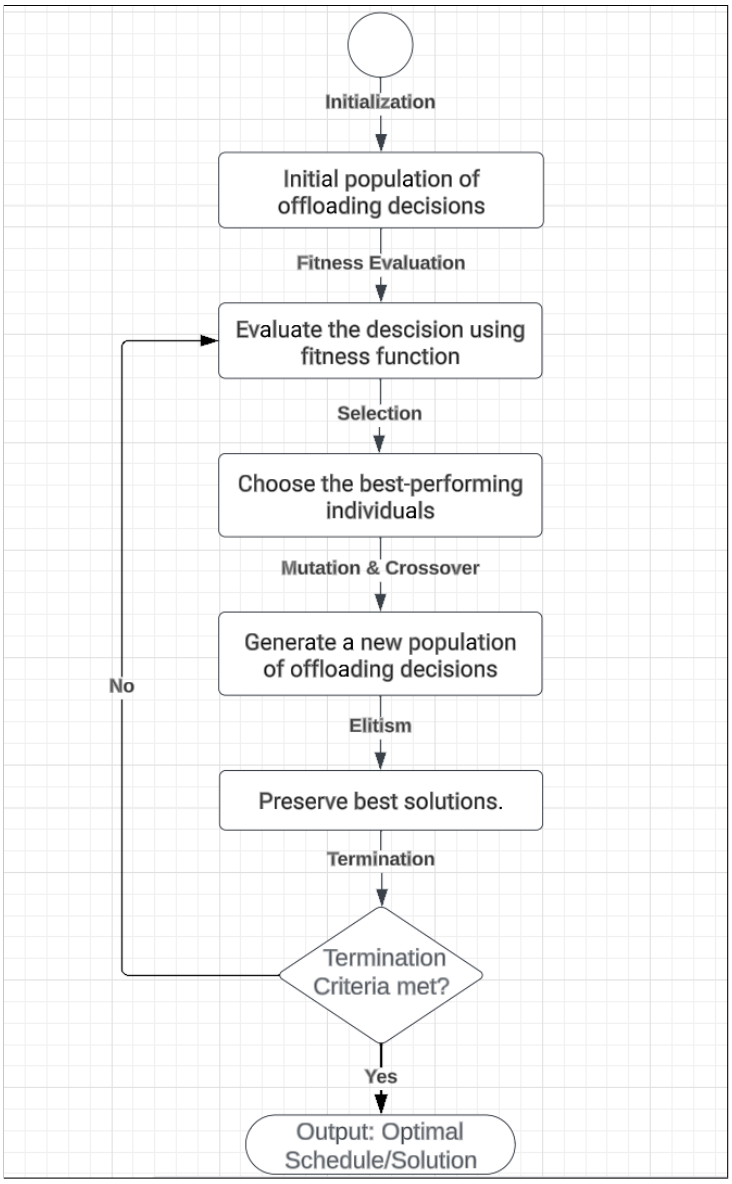

Figure 2: Flowchart of Distance Based Pareto Genetic Algorithm (DBPGA)

3. **Distance-Based Pareto Sorting:**

   - Sort the population into different levels of Pareto fronts. The first front contains non-dominated solutions, while subsequent fronts are formed by removing the previous fronts and repeating the process.
   - Calculate the distance between solutions within the same front to maintain diversity, typically using the Euclidean distance in the objective space.

4. **Selection:**

   - Select solutions for reproduction based on their ranking and distance. Solutions that are non-dominated and have greater distances from others are preferred.

5. **Crossover:**

   - Pair selected solutions and perform crossover to produce offspring. The crossover can be single-point, multi-point, or uniform, depending on the chosen strategy.
   - Offspring inherit characteristics from both parents, which promotes the combination of effective strategies from different solutions.

6. **Mutation:**

   - Apply mutation to the offspring at a predetermined probability rate. This introduces random changes and helps explore new areas of the solution space.
   - Mutation can adjust individual task offloading decisions within a solution, allowing the algorithm to escape local optima.

7. **Termination:**

   - Check for the termination condition, which could be a set number of generations, a convergence criterion, or a satisfactory fitness level.
   - Once the termination condition is met, the algorithm concludes, presenting the final set of Pareto-optimal solutions.

4.3 Algorithm Pseudocode

Input:

- A workflow of tasks $T = \{t_1, t_2, \ldots, t_n\}$ represented as a Directed Acyclic Graph (DAG)
- A set of resources $R = \{r_1, r_2, \ldots, r_m\}$ in the EFC system
- The number of generations G
- The population size P

Output:

- A set of Pareto-optimal offloading decisions

1. **Initialization:**

   - For each task t_i in T, assign it to the resource $r_{\min}$ that minimizes the finish time of the task:
     - For each resource r_j in R, calculate the finish time FT_{ij} of task t_i if it is assigned to resource r_j:

$$FT_{ij} = \max(FT_{\text{last}}(r_j), ST_i) + ET_{ij}$$

     - Find the resource $r_{\min}$ that minimizes the finish time of task t_i:

$$r_{\min} = \arg\min(FT_{ij}) \text{ for all } r_j \in R$$

– Assign task t_i to resource $r_{\min}$:

$$d_i = r_{\min}$$

- Repeat the above process P times to create an initial population of offloading decisions $D = \{D_1, D_2, \ldots, D_P\}$

2. **Fitness Evaluation:**

- For each decision D_i in D, calculate its fitness F_i based on the objectives:

$$F_i = w_1 \cdot MS_{\text{norm}}(D_i) + w_2 \cdot TC_{\text{norm}}(D_i) + w_3 \cdot TE_{\text{norm}}(D_i)$$

- Determine the set of elite decisions E from D that are non-dominated
- Calculate the maximum fitness value $F_{\max}$ among the elite decisions:

$$F_{\max} = \max(F_i \text{ for all } D_i \in E)$$

3. **Selection, Crossover, and Mutation:**

- Perform selection, crossover, and mutation operations on the population to generate a new population:
  - **Selection:** Use the crowded tournament selection operator to select decisions for the next generation
  - **Crossover:** Combine two decisions to generate two new decisions
  - **Mutation:** Alter one or more elements of a decision to generate a new decision

4. **Elitism:**

- Assign the maximum fitness value $F_{\max}$ to all elite decisions:

$$F_i = F_{\max} \text{ for all } D_i \in E$$

5. **Termination:**

- If the maximum number of generations G is reached, stop the algorithm and return the set of elite decisions E as the set of Pareto-optimal offloading decisions
- Otherwise, go back to the fitness evaluation step

5 Implementation

5.1 Integration of DBPGA into FogWorkflowSim

The successful integration of the DBPGA into the FogWorkflowSim simulation tools marked the start of the implementation phase. The DBPGA, written in Java, was designed to expand FogWorkflowSim's existing capabilities by allowing a detailed comparison with its in-built GA under a range of task-offloading strategies.

5.2 Core Functionalities

The basic capabilities of the DBPGA were managing a set of elite decisions, each capturing the best-found solutions in terms of time, energy, and cost, as well as an associated fitness value. These elite decisions are critical in leading the algorithm to optimal offloading solutions across various workflows and scenarios.

5.3 Adaptive Optimization Process

The programme contains an adaptive optimisation approach that alters the fitness evaluation of possible solutions dynamically based on their proximity to previously selected elite decisions. This mechanism enables a continual learning approach, which improves the algorithm's capacity to navigate the complicated optimisation of fog computing.

5.4 Outputs of the DBPGA

The implementation of the DBPGA produced numerous key outputs required for performance evaluation. These outputs include workflow makespan, the total energy utilised by the data centres, and the overall cost of performing the operation. The programme also produced a breakdown of task offloading distribution, indicating the number of tasks offloaded to the cloud, fog, and mobile devices. These results were critical in determining the DBPGA's multi-objective optimisation capabilities.

5.5 Performance Metrics

- **Workflow Makespan:** The total time taken to complete the entire set of tasks within the workflow.
- **Energy Consumption:** The total energy consumed by the computing resources, particularly within the data centers during task execution.
- **Cost Efficiency:** The total cost incurred from executing the workflow, which includes computational and data transmission expenses.

5.6 Development Environment

The DBPGA was developed in Java in an integrated development environment fogworkflowsim that allowed for thorough testing and iterative refining. The environment allowed for the modelling of multiple task offloading schemes, allowing the DBPGA to be compared to specified performance benchmarks.

6 Evaluation

6.1 Overview

The evaluation phase compared the performance of the DBPGA to the standard GA across diverse scientific workflows. The focus was on comparing average algorithm execution time while maintaining consistency in other parameters such as Workflow Makespan,

Energy Consumed, and Total Cost. The insights gained from this analysis will be critical for determining the DBPGA's efficacy in optimising task execution in fog computing environments.

The comparative analysis used statistical methods to determine the significance of the differences discovered between the DBPGA and GA. This evaluation ranged from 20 to 1000 rows, representing varied complexities within the Montage, Cybershake, Epigenomics, Inspiral, and Sipht workflows.

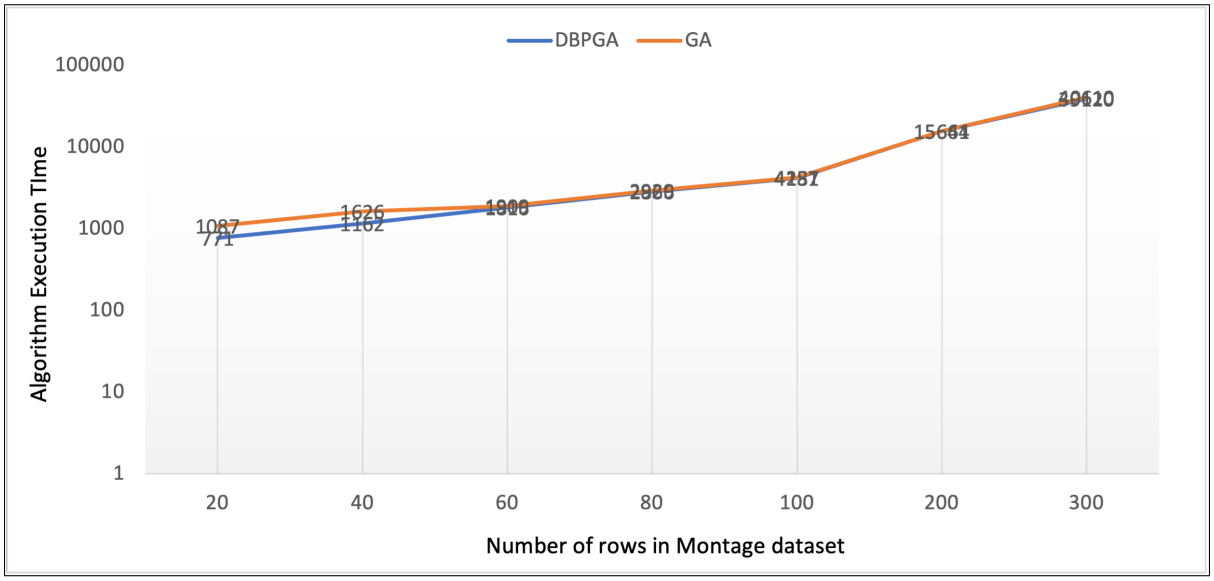

Figure 3: DBPGA vs GA for Montage

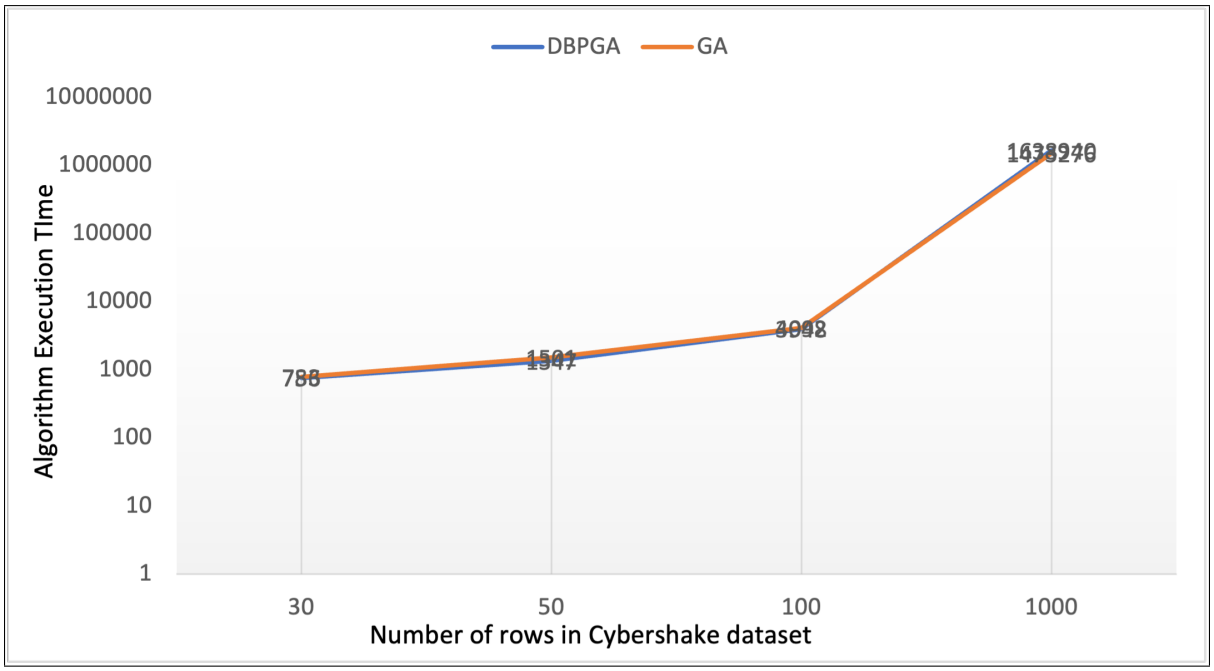

Figure 4: DBPGA vs GA for Cybershake

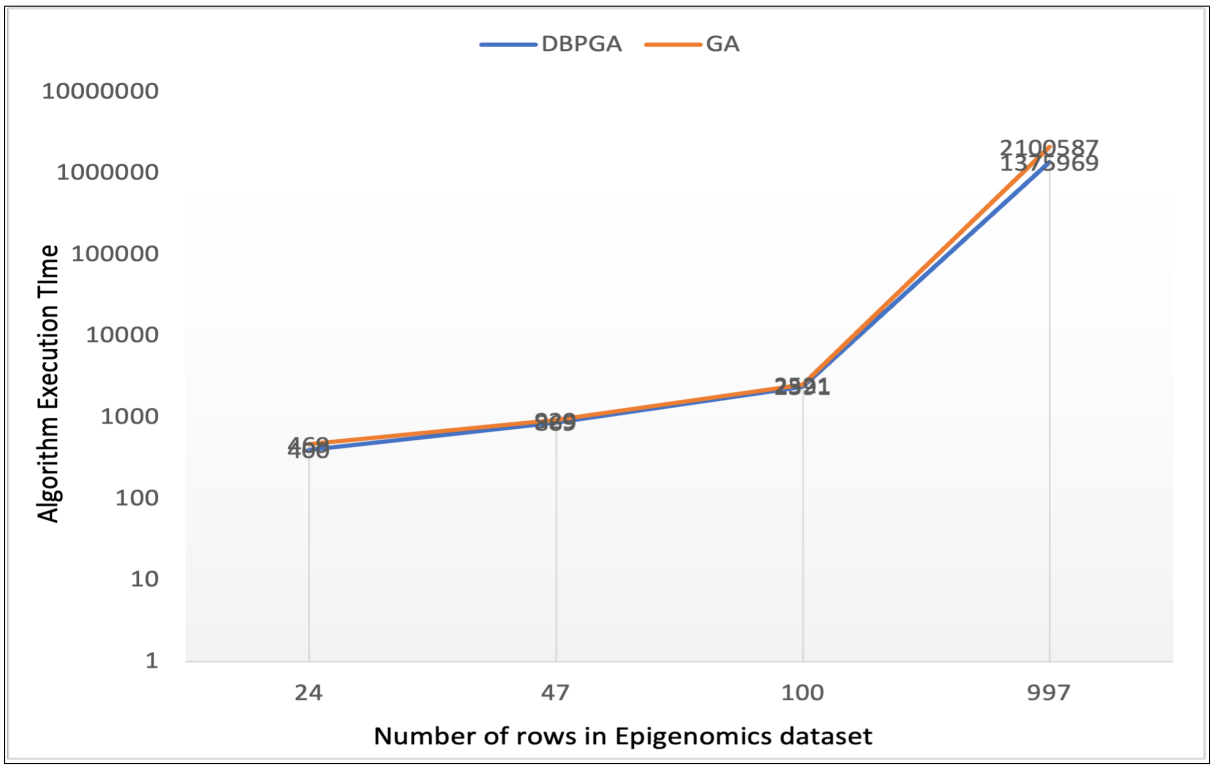

Figure 5: DBPGA vs GA for Epigenomics

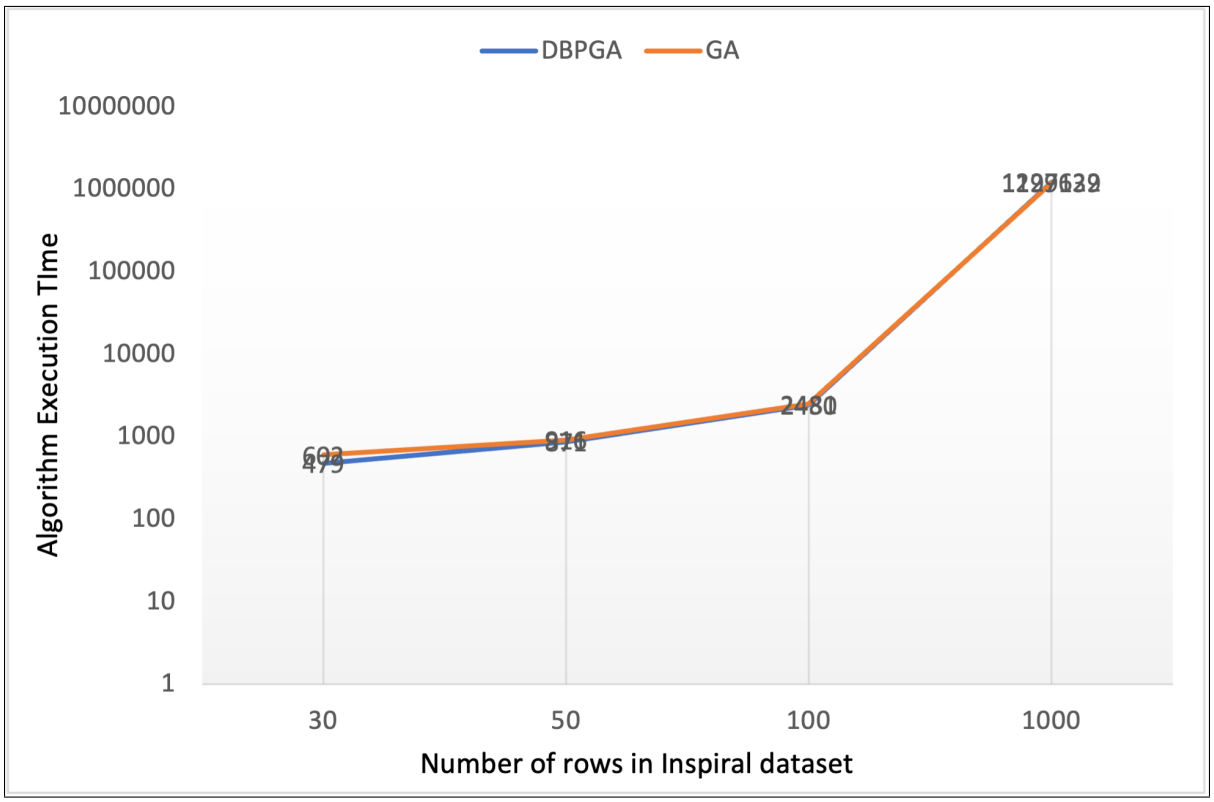

Figure 6: DBPGA vs GA for Inspiral

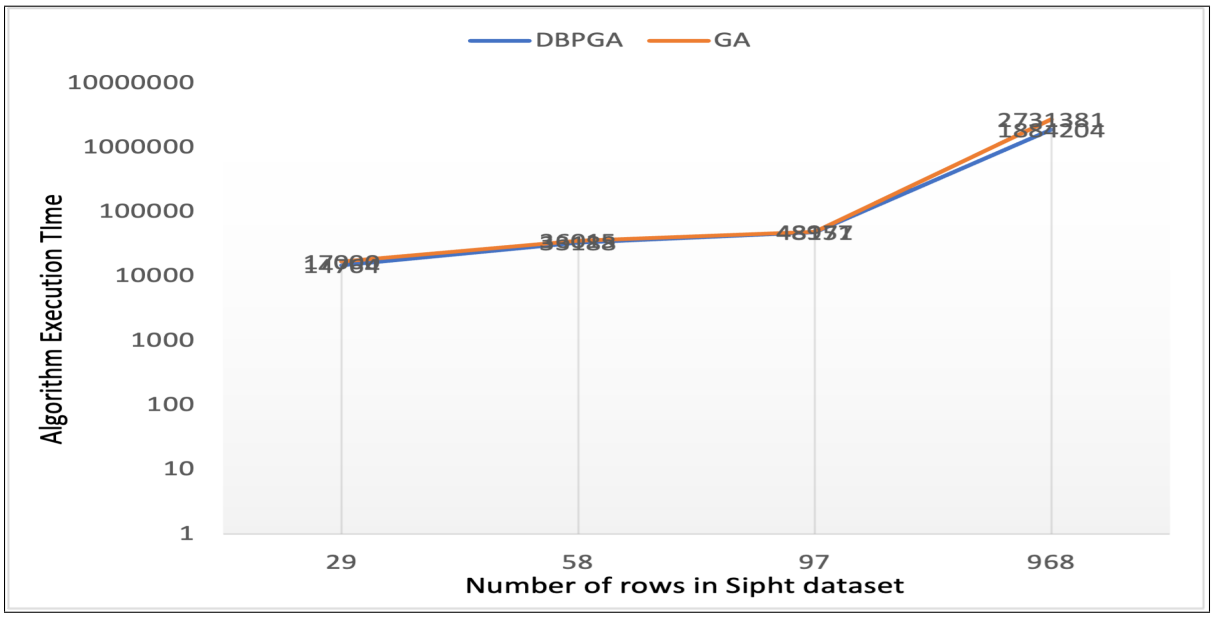

Figure 7: DBPGA vs GA for Sipht

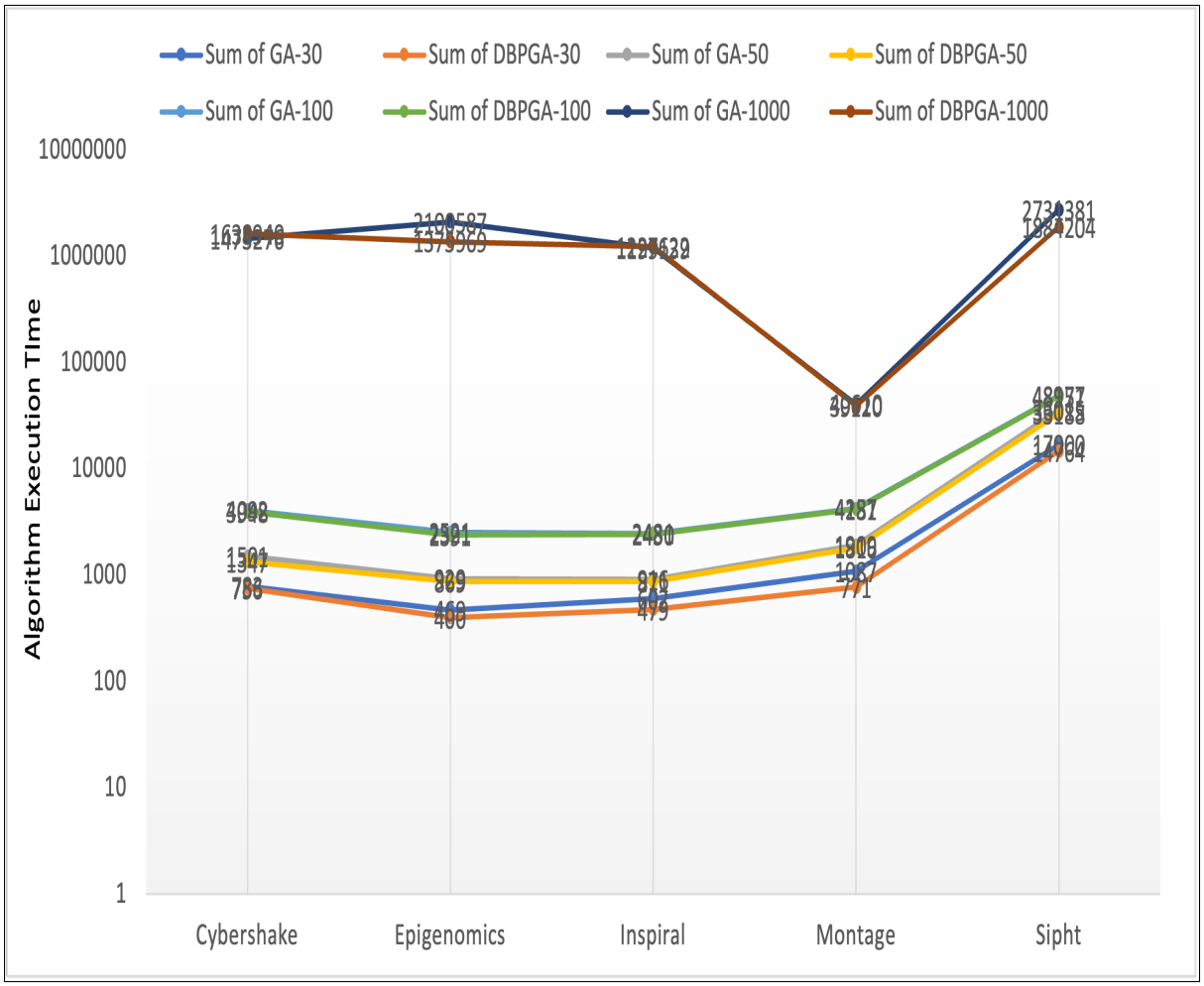

Figure 8: DBPGA vs GA plotted for all datasets

6.2 Montage Workflow

In the Montage workflow, the DBPGA significantly reduced the average algorithm execution time when compared to the GA. This efficiency is especially notable for smaller datasets, where the DBPGA outscored the GA by up to 29%, implying an improved optimisation process for less complex task executions. Figure 3 shows the number of dataset rows to the algorithm execution time graph.

6.3 Cybershake Workflow

For the Cybershake workflow, the DBPGA provided improved time efficiency, but less noticeable as the dataset size increased as seen in figure 4. This pattern could reflect the DBPGA's ability to manage energy-intensive operations, which is important for fog computing applications that prioritise energy conservation.

6.4 Epigenomics Workflow

In the context of the Epigenomics workflow, the DBPGA consistently outperformed the GA, with the difference in execution time increasing as the dataset size increased as seen in figure 5. This signifies the DBPGA's better scalability and capacity to handle large-scale genomic data processing.

6.5 Inspiral Workflow

As per figure 6 the findings were similar for the Inspiral workflow, with the DBPGA exhibiting small gains in execution time. This implies that, while the DBPGA is optimised for time-critical processes, its advantages are nuanced and vary depending on the computational demands of the activities.

6.6 Sipht Workflow

The Sipht workflow, which is distinguished by complex data dependencies, highlighted the DBPGA's time efficiency. Despite the complexities required, the DBPGA successfully reduced execution time, demonstrating its robustness in addressing complicated data processing tasks which is evident in figure 7.

6.7 Discussion

The experimental research, the DBPGA, demonstrated varying performance across diverse processes, offering insights into its efficiency and scalability. The DBPGA performed well in the Montage workflow, particularly in smaller datasets (20 and 40 rows), where it displayed a considerable reduction in average algorithm execution time. This efficiency demonstrates the DBPGA's ability to handle jobs with low computational complexity. However, as the complexity of larger datasets, such as the 300-row Montage, the performance benefits became less noticeable, implying a possible loss in efficiency as complexity increased.

A similar pattern was observed in the Cybershake workflow, where the DBPGA improved in smaller datasets but lagged behind the GA in the 1000-row dataset. This raises

concerns about the DBPGA's ability to scale efficiently for large and computationally intensive tasks. In the Epigenomics workflow, on the other hand, the DBPGA consistently outperformed the GA across all dataset sizes, including the largest dataset of 997 rows. This consistency indicates that the DBPGA is well-suited for data-intensive procedures like those encountered in genomic studies.

The Inspiral and Sipht workflow, on the other hand, provided a mixed picture. While the DBPGA's performance advantage reduced as dataset sizes grew larger, it still outperformed the GA in certain large-scale cases, most notably the Sipht workflow. These findings imply that the efficiency of the DBPGA may vary depending on the type and complexity of the workflow. Figure 8 shows the plot for DBPGA and GA for all the datasets.

In short, the DBPGA excels at optimising execution time for less complex jobs but struggles to maintain this efficiency when dealing with larger, more complex datasets. Its performance in genomic data processing is a particular strength, although its scalability in handling very big datasets in certain workflows needs additional investigation and improvement.

7 Conclusion and Future Work

The research effectively showed that the DBPGA may be used to optimise task scheduling in EFC systems. In smaller datasets across diverse workflows, the DBPGA consistently outperformed the traditional GA, demonstrating its potential to improve computational efficiency in less complex environments. Its performance in these situations proves its importance and use in fog computing, particularly for applications requiring speedy and efficient processing. While the DBPGA's performance advantage was less apparent in larger datasets, it nonetheless retained a competitive advantage, indicating its adaptability and potential for optimisation in a wide range of computational workloads. These findings validate the DBPGA's position as a helpful tool in task scheduling, providing significant improvements in execution time efficiency.

Future research will focus on improving the adaptability and scalability of the DBPGA, particularly for larger and more complicated workflows. This entails improving the algorithm so that it can dynamically modify its strategies based on the size and type of the work, assuring optimal performance over a wider range of scenarios. Efforts can be made to incorporate complex heuristic algorithms or adaptive learning processes, which could allow the DBPGA to manage the diverse needs of large-scale EFC environments more effectively. The DBPGA's potential in this work establishes a solid platform for its development and implementation in more challenging computing environments.

References

Agarwal, G., Gupta, S., Ahuja, R. and Rai, A. K. (2023). Multiprocessor task scheduling using multi-objective hybrid genetic algorithm in fog–cloud computing, *Knowledge-Based Systems* **272**: 110563.

Akintoye, S. B. and Bagula, A. (2019). Improving quality-of-service in cloud/fog computing through efficient resource allocation, *Sensors* **19**(6): 1267.

Aladwani, T. (2019). Scheduling iot healthcare tasks in fog computing based on their importance, *Procedia Computer Science* **163**: 560–569. 16th Learning and Technology Conference 2019Artificial Intelligence and Machine Learning: Embedding the Intelligence.
URL: *https://www.sciencedirect.com/science/article/pii/S1877050919321775*

Aoudia, I., Benharzallah, S., Kahloul, L. and Kazar, O. (2020). Qos-aware service composition in fog-iot computing using multi-population genetic algorithm, *2020 21st International Arab Conference on Information Technology (ACIT)*, IEEE, pp. 1–9.

Bonomi, F., Milito, R., Zhu, J. and Addepalli, S. (2012). Fog computing and its role in the internet of things, *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16.

Ghosh, A. and Dehuri, S. (2004). Evolutionary algorithms for multi-criterion optimization: A survey, *International Journal of Computing and Information Sciences* **2**.

Guerrero, C., Lera, I. and Juiz, C. (2022). Genetic-based optimization in fog computing: Current trends and research opportunities, *Swarm and Evolutionary Computation* **72**: 101094.

Hosseinzadeh, M., Azhir, E., Lansky, J., Mildeova, S., Ahmed, O. H., Malik, M. H. and Khan, F. (2023). Task scheduling mechanisms for fog computing: A systematic survey, *IEEE Access* **11**: 50994–51017.

Hussain, S. M. and Begh, G. R. (2022). Hybrid heuristic algorithm for cost-efficient qos aware task scheduling in fog–cloud environment, *Journal of Computational Science* **64**: 101828.

Karamoozian, A., Hafid, A. and Aboulhamid, E. M. (2019). On the fog-cloud cooperation: How fog computing can address latency concerns of iot applications, *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 166–172.

Liu, X., Fan, L., Xu, J., Li, X., Gong, L., Grundy, J. and Yang, Y. (2019). Fogworkflowsim: An automated simulation toolkit for workflow performance evaluation in fog computing, *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 1114–1117.

Madhura, R., Elizabeth, B. L. and Uthariaraj, V. R. (2021). An improved list-based task scheduling algorithm for fog computing environment, *Computing* **103**: 1353–1389.

Martinez, I., Hafid, A. S. and Jarray, A. (2021). Design, resource management, and evaluation of fog computing systems: A survey, *IEEE Internet of Things Journal* **8**(4): 2494–2516.

Mokni, M., Yassa, S., Hajlaoui, J. E., Omri, M. N. and Chelouah, R. (2023). Multi-objective fuzzy approach to scheduling and offloading workflow tasks in fog–cloud computing, *Simulation Modelling Practice and Theory* **123**: 102687.

Movahedi, Z., Defude, B. et al. (2021). An efficient population-based multi-objective task scheduling approach in fog computing systems, *Journal of Cloud Computing* **10**(1): 1–31.

Nguyen, B. M., Thi Thanh Binh, H., The Anh, T. and Bao Son, D. (2019). Evolutionary algorithms to optimize task scheduling problem for the iot based bag-of-tasks application in cloud–fog computing environment, *Applied Sciences* **9**(9).
URL: *https://www.mdpi.com/2076-3417/9/9/1730*

Ni, L., Zhang, J., Jiang, C., Yan, C. and Yu, K. (2017). Resource allocation strategy in fog computing based on priced timed petri nets, *IEEE Internet of Things Journal* **4**(5): 1216–1228.

Nikoui, T. S., Balador, A., Rahmani, A. M. and Bakhshi, Z. (2020). Cost-aware task scheduling in fog-cloud environment, *2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*, pp. 1–8.

Shukla, P. and Pandey, S. (2023). Motors: Multi-objective task offloading and resource scheduling algorithm for heterogeneous fog-cloud computing scenario.

Skarlat, O., Nardelli, M., Schulte, S., Borkowski, M. and Leitner, P. (2017). Optimized iot service placement in the fog, *Serv. Oriented Comput. Appl.* **11**(4): 427–443.
URL: *https://doi.org/10.1007/s11761-017-0219-8*

Taneja, M. and Davy, A. (2017). Resource aware placement of iot application modules in fog-cloud computing paradigm, *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 1222–1228.

Vorobyev, S. (2019). Mathematical model of the architecture of a distributed information-measuring system based on cloud and fog technologies, *Journal of Physics: Conference Series* **1352**: 012059.

Xu, F., Yin, Z., Gu, A., Li, Y., Yu, H. and Zhang, F. (2021). Adaptive scheduling strategy of fog computing tasks with different priority for intelligent production lines, *Procedia Computer Science* **183**: 311–317.

Ye, D., Wu, M., Tang, S. and Yu, R. (2016). Scalable fog computing with service offloading in bus networks, *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, pp. 247–251.

Yousefpour, A., Patil, A., Ishigaki, G., Kim, I., Wang, X., Cankaya, H. C., Zhang, Q., Xie, W. and Jue, J. P. (2019). Fogplan: A lightweight qos-aware dynamic fog service provisioning framework, *IEEE Internet of Things Journal* **6**(3): 5080–5096.